mpi$_{a}rrayDocumentation$

Release 0.1.0dev1-202-g6a6c707

Shane J. Latham

Nov 16, 2017

Read Me

Release 0.1.0dev1-202-g6a6c707

Version 0.1.0

Date Nov 16, 2017

CHAPTER 1

Contents

1.1 Introduction

The mpi_array python package provides a numpy.ndarray API to a Partitioned Global Address Space array which utilizes MPI (via mpi4py) for parallelism.

1.2 Quick Start Example

The following `quickstart.py` script creates a zero-initialised array and performs some element assignments:

```
import mpi_array as mpia

# creates zero-initialized PGAS (distributed) array
dary = mpia.zeros((1000, 1000, 1000), dtype="uint16")

# Add one to all elements of array
dary += 1

# Assign to slice
dary[250:750, :, 250:750] = 8

# ufuncs
dary[...] = mpia.power(dary, 1.0/3.0)
```

The `quickstart.py` script can be executed serially (single process) as:

```
python quickstart.py
```

or in parallel (using 8 processes) as:

```
mpirun -n 8 python quickstart.py
```

1.3 Related Work

Related distributed array (PGAS) implementations with python API:

- Global Arrays (ga at github) with ga4py python API and GAiN numpy API
- DistArray
- pnumpy
- Spartan (at github here)
- caput.mpiarray
- dask.distributed
- bolt
- distnumpy and distnumpy at github

Related parallel array implementations (not PGAS) with python API:

- Bohrium
- Intel Distribution for Python

1.4 Installation

Using `pip` from latest github source:

```
pip install --user git+git://github.com/mpi-array/mpi_array.
git#egg=mpi_array
```

1.5 Requirements

Requires:

- python-2 version >= *2.7* or python-3 version >= *3.3*,
- array_split version >= *0.4.0*,
- psutil >= *4.3.0*
- numpy version >= *1.13* (require __array_ufunc__ API),
- an MPI implementation which supports at least MPI-3 (such as OpenMPI, MPICH or Intel MPI)
- mpi4py version >= *2.0*.

1.6 Testing

Run tests (unittest test-cases and doctest docstring test-cases) using:

```
python -m mpi_array.tests
```

or, from the source tree, run:

```
python setup.py test
```

Run tests with parallelism:

```
mpirun -n 8 python -m mpi_array.tests
```

Travis CI at:

https://travis-ci.org/mpi-array/mpi_array/

1.7 Documentation

Latest sphinx generated documentation at readthedocs.org:

http://mpi-array.readthedocs.io/en/latest

and at github *gh-pages*:

https://mpi-array.github.io/mpi_array/

Sphinx documentation can be built from the source:

```
python setup.py build_sphinx
```

with the HTML generated in `docs/_build/html`.

1.8 Latest source code

Source at github:

https://github.com/mpi-array/mpi_array

clone with:

```
git clone https://github.com/mpi-array/mpi_array.git
```

1.9 License information

See the file LICENSE.txt for terms & conditions, for usage and a DISCLAIMER OF ALL WARRANTIES.

1.10 Terminology

1.11 Distributing arrays amongst *nodes*

1.12 The `mpi_array` Package

Python package for multi-dimensional distributed arrays (Partitioned Global Address Space) with `numpy.ndarray`-like API.

1.12.1 Classes and Functions

—

1.12.2 Attributes

1.13 The `mpi_array.benchmarks` Package

Runtime benchmarking.

1.13.1 Modules

benchmark	Manages finding, running and recoding benchmark results.
bench_creation	Benchmarks for array creation.
bench_ufunc	Benchmarks for ufuncs.
core	Core utilities for benchmark implementations.
utils	Miscellaneous benchmark utilities.

mpi_array.benchmarks.benchmark

Manages finding, running and recoding benchmark results.

This module has shamelessly borrows from the airspeed velocity (asv) file benchmark.py. See the airspeed velocity (asv) LICENSE.

Functions

create_runner_argument_parser()	
disc_benchmarks(root[, package])	Discover all benchmarks in a given directory tree, yielding Benchmark
disc_files(root[, package])	Iterate over all .py files in a given directory tree.
get_process_time_timer()	The best timer we can use is `time.process_time()`, but it is not available in the Python stdlib until Python 3.3.
get_setup_cache_key(func)	
get_source_code(items)	Extract source code of given items, and concatenate and dedent it.
import_module(name[, package])	Import a module.
root_and_package_from_name(module_name)	Returns root filename for the package/module named by *module_name*.
run_main(argv)	Runs the benchmarks.
sha256	Returns a sha256 hash object; optionally initialized with a string

mpi_array.benchmarks.benchmark.create_runner_argument_parser

mpi_array.benchmarks.benchmark.**create_runner_argument_parser**()

mpi_array.benchmarks.benchmark.disc_benchmarks

mpi_array.benchmarks.benchmark.**disc_benchmarks**(*root*, *package=None*)
Discover all benchmarks in a given directory tree, yielding Benchmark objects

For each class definition, looks for any methods with a special name.

For each free function, yields all functions with a special name.

mpi_array.benchmarks.benchmark.disc_files

mpi_array.benchmarks.benchmark.**disc_files**(*root*, *package=''*)
Iterate over all .py files in a given directory tree.

mpi_array.benchmarks.benchmark.get_process_time_timer

mpi_array.benchmarks.benchmark.**get_process_time_timer**()
The best timer we can use is `time.process_time()`, but it is not available in the Python stdlib until Python 3.3. This is a `ctypes` backport for Pythons that don't have it.

Return type `function`

Returns The `time.process_time()` function, if available, otherwise, if possible, an equivalent created using `ctypes`, otherwise `timeit.default_timer()`.

mpi_array.benchmarks.benchmark.get_setup_cache_key

mpi_array.benchmarks.benchmark.**get_setup_cache_key**(*func*)

mpi_array.benchmarks.benchmark.get_source_code

mpi_array.benchmarks.benchmark.**get_source_code**(*items*)
Extract source code of given items, and concatenate and dedent it.

mpi_array.benchmarks.benchmark.import_module

mpi_array.benchmarks.benchmark.**import_module**(*name*, *package=None*)
Import a module.

The 'package' argument is required when performing a relative import. It specifies the package to use as the anchor point from which to resolve the relative import to an absolute import.

mpi_array.benchmarks.benchmark.root_and_package_from_name

mpi_array.benchmarks.benchmark.**root_and_package_from_name**(*module_name*)
Returns root filename for the package/module named by *module_name*.

mpi_array.benchmarks.benchmark.run_main

mpi_array.benchmarks.benchmark.**run_main**(*argv*)
Runs the benchmarks.

Parameters **argv** (list of str) – The command line arguments (e.g. sys.argv).

See also:

BenchmarkRunner

mpi_array.benchmarks.benchmark.sha256

mpi_array.benchmarks.benchmark.**sha256**()
Returns a sha256 hash object; optionally initialized with a string

Classes

Benchmark(name, func, attr_sources)	Represents a single benchmark.
BenchmarkRunner([argv])	Discovers and runs benchmarks.
TimeBenchmark(name, func, attr_sources)	Represents a single benchmark for timing.

mpi_array.benchmarks.benchmark.Benchmark

class mpi_array.benchmarks.benchmark.**Benchmark**(*name*, *func*, *attr_sources*)
Bases: object

Represents a single benchmark.

Methods

__init__(name, func, attr_sources)	
barrier()	Barrier.
bcast(value)	Broadcast value from *root_rank* to all ranks of *comm*.
do_profile_run()	
do_run()	
do_setup()	
do_setup_cache()	
do_teardown()	
initialise()	
insert_param(param)	Insert a parameter at the front of the parameter list.
redo_setup()	
set_param_idx(param_idx)	Set the parameter combo via the index *param_idx*.

mpi_array.benchmarks.benchmark.Benchmark.__init__

Benchmark.**__init__**(*name*, *func*, *attr_sources*)

mpi_array.benchmarks.benchmark.Benchmark.barrier

Benchmark.**barrier**()
Barrier.

mpi_array.benchmarks.benchmark.Benchmark.bcast

Benchmark.**bcast**(*value*)
Broadcast value from *root_rank* to all ranks of *comm*.

Return type object

Returns value on rank *root_rank* rank process.

mpi_array.benchmarks.benchmark.Benchmark.do_profile_run

Benchmark.**do_profile_run**()

mpi_array.benchmarks.benchmark.Benchmark.do_run

Benchmark.**do_run**()

mpi_array.benchmarks.benchmark.Benchmark.do_setup

Benchmark.**do_setup**()

mpi_array.benchmarks.benchmark.Benchmark.do_setup_cache

Benchmark.**do_setup_cache**()

mpi_array.benchmarks.benchmark.Benchmark.do_teardown

Benchmark.**do_teardown**()

mpi_array.benchmarks.benchmark.Benchmark.initialise

Benchmark.**initialise**()

mpi_array.benchmarks.benchmark.Benchmark.insert_param

Benchmark.**insert_param**(*param*)
Insert a parameter at the front of the parameter list.

mpi_array.benchmarks.benchmark.Benchmark.redo_setup

Benchmark.**redo_setup**()

mpi_array.benchmarks.benchmark.Benchmark.set_param_idx

Benchmark.**set_param_idx**(*param_idx*)
Set the parameter combo via the index *param_idx*.

Raises **ValueError** – if param_idx is out of range.

Attributes

comm	The mpi4pi.MPI.Comm used for synchronization.
current_params	The current set of parameters, set via *set_param_idx()*.
name_regex	
params	The list of benchmark parameters.
root_rank	An int indicating the *root* rank process of *comm*.
setup_error	The error which occured during *do_setup()*, None if no error occurred.

mpi_array.benchmarks.benchmark.Benchmark.comm

Benchmark.**comm**
The mpi4pi.MPI.Comm used for synchronization.

mpi_array.benchmarks.benchmark.Benchmark.current_params

Benchmark.**current_params**
The current set of parameters, set via *set_param_idx()*.

mpi_array.benchmarks.benchmark.Benchmark.name_regex

Benchmark.**name_regex = re.compile('^$')**

mpi_array.benchmarks.benchmark.Benchmark.params

Benchmark.**params**
The list of benchmark parameters.

mpi_array.benchmarks.benchmark.Benchmark.root_rank

Benchmark.**root_rank**
An int indicating the *root* rank process of *comm*.

mpi_array.benchmarks.benchmark.Benchmark.setup_error

Benchmark.**setup_error**
The error which occured during *do_setup()*, None if no error occurred.

mpi_array.benchmarks.benchmark.BenchmarkRunner

class mpi_array.benchmarks.benchmark.**BenchmarkRunner**(*argv=None*)
Bases: object

Discovers and runs benchmarks.

Methods

__init__([argv])	Initialise.
create_argument_parser()	Creates argparse.ArgumentParser for handling command line.
create_profile_stats(profile_file_name)	Factory function for creating a pstats.Stats instance.
discover_benchmarks()	Find benchmarks, store *Benchmark* objects in *benchmarks*.
filter(b)	Returns True if the *Benchmark* *b* passes the filtering criterion.
handle_profile(profiler)	
log_profile_stats(profile_stats[, sort_keys])	Logs the output from pstats.Stats.print_stats().
run()	Discover and run benchmarks.
run_and_write_results()	Discovers, runs and records benchmark results in files.
run_benchmarks()	Runs the benchmarks, results are stored in *bench_results*.
write_bench_results(bench_results, ...)	Writes the benchmark results *bench_results* as json string to file named *bench_results_file_name*.
write_benchmarks(benchmarks, ...)	Writes individual *Benchmark* elements of *benchmarks* as json string to file named *benchmarks_file_name*.
write_profile_stats(profile_stats, ...)	Dump the profile stats from the pstats.Stats object to file.

mpi_array.benchmarks.benchmark.BenchmarkRunner.__init__

BenchmarkRunner.**__init__**(*argv=None*)
Initialise.

Parameters **argv** (list of str) – Command line arguments, parsed with the argparse.ArgumentParser instance returned by *create_argument_parser()*.

mpi_array.benchmarks.benchmark.BenchmarkRunner.create_argument_parser

BenchmarkRunner.**create_argument_parser**()
Creates argparse.ArgumentParser for handling command line.

Return type argparse.ArgumentParser

Returns A argparse.ArgumentParser for parsing command line.

See also:

create_runner_argument_parser()

mpi_array.benchmarks.benchmark.BenchmarkRunner.create_profile_stats

BenchmarkRunner.**create_profile_stats**(*profile_file_name*)
Factory function for creating a pstats.Stats instance.

Parameters **profile_file_name** (str) – Name of file which contains profile data.

Return type pstats.Stats

Returns Stats object initialised with data from *profile_file_name*.

mpi_array.benchmarks.benchmark.BenchmarkRunner.discover_benchmarks

BenchmarkRunner.**discover_benchmarks**()
Find benchmarks, store *Benchmark* objects in *benchmarks*.

mpi_array.benchmarks.benchmark.BenchmarkRunner.filter

BenchmarkRunner.**filter**(*b*)
Returns True if the *Benchmark* *b* passes the filtering criterion.

mpi_array.benchmarks.benchmark.BenchmarkRunner.handle_profile

BenchmarkRunner.**handle_profile**(*profiler*)

mpi_array.benchmarks.benchmark.BenchmarkRunner.log_profile_stats

BenchmarkRunner.**log_profile_stats**(*profile_stats*, *sort_keys=('cumtime',)*)
Logs the output from pstats.Stats.print_stats().

Parameters **profile_stats** (pstat.Stats) – The stats to dump to file.

mpi_array.benchmarks.benchmark.BenchmarkRunner.run

BenchmarkRunner.**run**()
Discover and run benchmarks.

mpi_array.benchmarks.benchmark.BenchmarkRunner.run_and_write_results

BenchmarkRunner.**run_and_write_results**()
Discovers, runs and records benchmark results in files.

mpi_array.benchmarks.benchmark.BenchmarkRunner.run_benchmarks

BenchmarkRunner.**run_benchmarks**()
Runs the benchmarks, results are stored in *bench_results*. Assumes benchmarks have already been *discovered*.

mpi_array.benchmarks.benchmark.BenchmarkRunner.write_bench_results

BenchmarkRunner.**write_bench_results**(*bench_results*, *bench_results_file_name*)
Writes the benchmark results *bench_results* as json string to file named *bench_results_file_name*.

mpi_array.benchmarks.benchmark.BenchmarkRunner.write_benchmarks

BenchmarkRunner.**write_benchmarks**(*benchmarks*, *benchmarks_file_name*)
Writes individual *Benchmark* elements of *benchmarks* as json string to file named *benchmarks_file_name*.

mpi_array.benchmarks.benchmark.BenchmarkRunner.write_profile_stats

BenchmarkRunner.**write_profile_stats**(*profile_stats*, *profile_file_name*)
Dump the profile stats from the pstats.Stats object to file.

Parameters

- **profile_stats** (pstat.Stats) – The stats to dump to file.
- **profile_file_name** (str) – The name of the file where stats are dumped (overwritten if it exists).

Attributes

bench_results	A list of benchmark results.
bench_results_file_name	A str, name of file where benchmark results are written.
benchmark_module_names	The list of module names which are searched to discover benchmarks.
benchmarks	The list of *Benchmark* objects.
benchmarks_file_name	A str, name of file where benchmark results are written.
comm	A mpi4py.MPI.Comm object, typically mpi4py.MPI.COMM_WORLD.
default_timer	An callable used to measure benchmark duration, e.g.
discover_only	A bool, if True only *discover* tests and do not run them.
do_profile	A bool, if True, performs a *profile* run in addition to the run.
do_quick_run	A bool, if True, performs a *quick* run.

Continued on next page

Table 1.8 – continued from previous page

filter_name_regex	The `re.RegularExpression` used to filter the benchmarks by name.
is_root_rank	A `bool`, if `True` this is running on the *root_rank* MPI rank.
profile_file_name	A `str` indicating the file name in which profile info is saved.
profile_stats	A `pstats.Stats` object in which profile info is accumulated.
rank_logger	A `logging.Logger` object for logging all rank process messages.
root_logger	A `logging.Logger` object for logging root-rank process messages.
root_rank	An `int` indicating the rank of the master MPI (root) process.

mpi_array.benchmarks.benchmark.BenchmarkRunner.bench_results

BenchmarkRunner.**bench_results**
A `list` of benchmark results.

mpi_array.benchmarks.benchmark.BenchmarkRunner.bench_results_file_name

BenchmarkRunner.**bench_results_file_name**
A `str`, name of file where benchmark results are written.

mpi_array.benchmarks.benchmark.BenchmarkRunner.benchmark_module_names

BenchmarkRunner.**benchmark_module_names**
The `list` of module names which are searched to discover benchmarks.

mpi_array.benchmarks.benchmark.BenchmarkRunner.benchmarks

BenchmarkRunner.**benchmarks**
The `list` of *Benchmark* objects.

mpi_array.benchmarks.benchmark.BenchmarkRunner.benchmarks_file_name

BenchmarkRunner.**benchmarks_file_name**
A `str`, name of file where benchmark results are written.

mpi_array.benchmarks.benchmark.BenchmarkRunner.comm

BenchmarkRunner.**comm**
A `mpi4py.MPI.Comm` object, typically `mpi4py.MPI.COMM_WORLD`.

mpi_array.benchmarks.benchmark.BenchmarkRunner.default_timer

BenchmarkRunner.**default_timer**
An `callable` used to measure benchmark duration, e.g. `time.process_time()`.

mpi_array.benchmarks.benchmark.BenchmarkRunner.discover_only

BenchmarkRunner.**discover_only**
A `bool`, if `True` only *discover* tests and do not run them.

mpi_array.benchmarks.benchmark.BenchmarkRunner.do_profile

BenchmarkRunner.**do_profile**
A `bool`, if `True`, performs a *profile* run in addition to the run.

mpi_array.benchmarks.benchmark.BenchmarkRunner.do_quick_run

BenchmarkRunner.**do_quick_run**
A `bool`, if `True`, performs a *quick* run. Each benchmark is only executed once.

mpi_array.benchmarks.benchmark.BenchmarkRunner.filter_name_regex

BenchmarkRunner.**filter_name_regex**
The `re.RegularExpression` used to filter the benchmarks by name.

mpi_array.benchmarks.benchmark.BenchmarkRunner.is_root_rank

BenchmarkRunner.**is_root_rank**
A `bool`, if `True` this is running on the *root_rank* MPI rank.

mpi_array.benchmarks.benchmark.BenchmarkRunner.profile_file_name

BenchmarkRunner.**profile_file_name**
A `str` indicating the file name in which profile info is saved.

mpi_array.benchmarks.benchmark.BenchmarkRunner.profile_stats

BenchmarkRunner.**profile_stats**
A `pstats.Stats` object in which profile info is accumulated.

mpi_array.benchmarks.benchmark.BenchmarkRunner.rank_logger

BenchmarkRunner.**rank_logger**
A `logging.Logger` object for logging all rank process messages.

mpi_array.benchmarks.benchmark.BenchmarkRunner.root_logger

BenchmarkRunner.**root_logger**
A logging.Logger object for logging root-rank process messages.

mpi_array.benchmarks.benchmark.BenchmarkRunner.root_rank

BenchmarkRunner.**root_rank**
An int indicating the rank of the master MPI (root) process.

mpi_array.benchmarks.benchmark.TimeBenchmark

class mpi_array.benchmarks.benchmark.**TimeBenchmark**(*name*, *func*, *attr_sources*)
Bases: *mpi_array.benchmarks.benchmark.Benchmark*

Represents a single benchmark for timing.

Methods

__init__(name, func, attr_sources)	
barrier()	Barrier.
bcast(value)	Broadcast value from *root_rank* to all ranks of *comm*.
benchmark_timing(timer, repeat, warmup_time)	
do_profile_run()	
do_run()	
do_setup()	
do_setup_cache()	
do_teardown()	
initialise()	
insert_param(param)	Insert a parameter at the front of the parameter list.
redo_setup()	
run(*param)	
set_param_idx(param_idx)	Set the parameter combo via the index *param_idx*.
wall_time()	Return *current* time in seconds.

mpi_array.benchmarks.benchmark.TimeBenchmark.__init__

TimeBenchmark.**__init__**(*name*, *func*, *attr_sources*)

mpi_array.benchmarks.benchmark.TimeBenchmark.barrier

TimeBenchmark.**barrier**()
Barrier.

mpi_array.benchmarks.benchmark.TimeBenchmark.bcast

TimeBenchmark.**bcast**(*value*)
Broadcast value from *root_rank* to all ranks of *comm*.

Return type object

Returns value on rank *root_rank* rank process.

mpi_array.benchmarks.benchmark.TimeBenchmark.benchmark_timing

TimeBenchmark.**benchmark_timing**(*timer*, *repeat*, *warmup_time*, *number=0*)

mpi_array.benchmarks.benchmark.TimeBenchmark.do_profile_run

TimeBenchmark.**do_profile_run**()

mpi_array.benchmarks.benchmark.TimeBenchmark.do_run

TimeBenchmark.**do_run**()

mpi_array.benchmarks.benchmark.TimeBenchmark.do_setup

TimeBenchmark.**do_setup**()

mpi_array.benchmarks.benchmark.TimeBenchmark.do_setup_cache

TimeBenchmark.**do_setup_cache**()

mpi_array.benchmarks.benchmark.TimeBenchmark.do_teardown

TimeBenchmark.**do_teardown**()

mpi_array.benchmarks.benchmark.TimeBenchmark.initialise

TimeBenchmark.**initialise**()

mpi_array.benchmarks.benchmark.TimeBenchmark.insert_param

TimeBenchmark.**insert_param**(*param*)
Insert a parameter at the front of the parameter list.

mpi_array.benchmarks.benchmark.TimeBenchmark.redo_setup

TimeBenchmark.**redo_setup**()

mpi_array.benchmarks.benchmark.TimeBenchmark.run

TimeBenchmark.**run**(**param*)

mpi_array.benchmarks.benchmark.TimeBenchmark.set_param_idx

TimeBenchmark.**set_param_idx**(*param_idx*)

Set the parameter combo via the index *param_idx*.

Raises **ValueError** – if param_idx is out of range.

mpi_array.benchmarks.benchmark.TimeBenchmark.wall_time

TimeBenchmark.**wall_time**()

Return *current* time in seconds.

Attributes

comm	The mpi4pi.MPI.Comm used for synchronization.
current_params	The current set of parameters, set via *set_param_idx()*.
default_timer	An callable used to measure benchmark duration, e.g.
goal_time	
name_regex	
number	
params	The list of benchmark parameters.
repeat	
root_rank	An int indicating the *root* rank process of *comm*.
setup_error	The error which occured during *do_setup()*, None if no error occurred.
timer	
wall_timer	
warmup_time	

mpi_array.benchmarks.benchmark.TimeBenchmark.comm

TimeBenchmark.**comm**

The mpi4pi.MPI.Comm used for synchronization.

mpi_array.benchmarks.benchmark.TimeBenchmark.current_params

TimeBenchmark.**current_params**

The current set of parameters, set via *set_param_idx()*.

mpi_array.benchmarks.benchmark.TimeBenchmark.default_timer

TimeBenchmark.**default_timer**
An `callable` used to measure benchmark duration, e.g. `time.process_time()`.

mpi_array.benchmarks.benchmark.TimeBenchmark.goal_time

TimeBenchmark.**goal_time**

mpi_array.benchmarks.benchmark.TimeBenchmark.name_regex

TimeBenchmark.**name_regex = re.compile('^(Time[A-Z_].+)|(time_.+)$')**

mpi_array.benchmarks.benchmark.TimeBenchmark.number

TimeBenchmark.**number**

mpi_array.benchmarks.benchmark.TimeBenchmark.params

TimeBenchmark.**params**
The list of benchmark parameters.

mpi_array.benchmarks.benchmark.TimeBenchmark.repeat

TimeBenchmark.**repeat**

mpi_array.benchmarks.benchmark.TimeBenchmark.root_rank

TimeBenchmark.**root_rank**
An `int` indicating the *root* rank process of *comm*.

mpi_array.benchmarks.benchmark.TimeBenchmark.setup_error

TimeBenchmark.**setup_error**
The error which occured during *do_setup()*, `None` if no error occurred.

mpi_array.benchmarks.benchmark.TimeBenchmark.timer

TimeBenchmark.**timer**

mpi_array.benchmarks.benchmark.TimeBenchmark.wall_timer

TimeBenchmark.**wall_timer**

mpi_array.benchmarks.benchmark.TimeBenchmark.warmup_time

TimeBenchmark.**warmup_time**

mpi_array.benchmarks.bench_creation

Benchmarks for array creation.

Classes

BlockPartitionCreateBench	Benchmarks for *mpi_array.distribution.BlockPartition* construction.
CommsAllocBench()	Benchmarks for *mpi_array.comms.LocaleComms.alloc_locale_buffer()*.
CommsCreateBench()	Benchmarks for *mpi_array.comms.LocaleComms* and *mpi_array.comms.CartLocaleComms* construction.
CreateBench()	Base class for array creation benchmarks.
MangoCreateBench()	Benchmarks for `mango.empty()` and `mango.zeros()` (mango tomography software).
MpiArrayCreateBench()	Benchmarks for `mpi_array.empty()`, `mpi_array.zeros()` etc.
MpiArrayCreateLikeBench()	Benchmarks for `mpi_array.empty_like()`, `mpi_array.zeros_like()` etc.
NumpyCreateBench()	Comparison benchmarks for `numpy.empty()`, `numpy.zeros()` etc.

mpi_array.benchmarks.bench_creation.BlockPartitionCreateBench

class mpi_array.benchmarks.bench_creation.**BlockPartitionCreateBench**

Bases: object

Benchmarks for *mpi_array.distribution.BlockPartition* construction.

Methods

setup(num_locales)	Import *mpi_array* module and assign to `self.module`.
time_block_partition(num_locales)	Time construction of *mpi_array.distribution.BlockPartition*.

mpi_array.benchmarks.bench_creation.BlockPartitionCreateBench.setup

BlockPartitionCreateBench.**setup**(*num_locales*)

Import *mpi_array* module and assign to `self.module`.

mpi_array.benchmarks.bench_creation.BlockPartitionCreateBench.time_block_partition

BlockPartitionCreateBench.**time_block_partition**(*num_locales*)
Time construction of *mpi_array.distribution.BlockPartition*.

Attributes

goal_time	
param_names	
params	Number of locales.
repeat	
warmup_time	

mpi_array.benchmarks.bench_creation.BlockPartitionCreateBench.goal_time

BlockPartitionCreateBench.**goal_time** = 0.25

mpi_array.benchmarks.bench_creation.BlockPartitionCreateBench.param_names

BlockPartitionCreateBench.**param_names** = ['num_locales']

mpi_array.benchmarks.bench_creation.BlockPartitionCreateBench.params

BlockPartitionCreateBench.**params** = [64, 128, 256]
Number of locales.

mpi_array.benchmarks.bench_creation.BlockPartitionCreateBench.repeat

BlockPartitionCreateBench.**repeat** = 8

mpi_array.benchmarks.bench_creation.BlockPartitionCreateBench.warmup_time

BlockPartitionCreateBench.**warmup_time** = 0.15

mpi_array.benchmarks.bench_creation.CommsAllocBench

class mpi_array.benchmarks.bench_creation.**CommsAllocBench**
Bases: *mpi_array.benchmarks.bench_creation.CreateBench*

Benchmarks for *mpi_array.comms.LocaleComms.alloc_locale_buffer()*.

Methods

__init__()	Initialise, set *module* to None.

Continued on next page

Table 1.14 – continued from previous page

free(a)	See *free_mpi_array_obj()*.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
setup(shape)	Import *mpi_array* module and assign to `self.module`.
teardown(shape)	Free *locale_comms* communicators.
time_alloc_locale_buffer(shape)	Time call of *mpi_array.comms.LocaleComms.alloc_locale_buffer()*.

mpi_array.benchmarks.bench_creation.CommsAllocBench.__init__

CommsAllocBench.**__init__**()
Initialise, set *module* to `None`.

mpi_array.benchmarks.bench_creation.CommsAllocBench.free

CommsAllocBench.**free**(*a*)
See *free_mpi_array_obj()*.

mpi_array.benchmarks.bench_creation.CommsAllocBench.free_mpi_array_obj

CommsAllocBench.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_creation.CommsAllocBench.get_globale_shape

CommsAllocBench.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of `int`) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_creation.CommsAllocBench.setup

CommsAllocBench.**setup**(*shape*)
Import *mpi_array* module and assign to `self.module`. Also initialise *locale_comms* with a *mpi_array.comms.CartLocaleComms* instance.

mpi_array.benchmarks.bench_creation.CommsAllocBench.teardown

CommsAllocBench.**teardown**(*shape*)
Free *locale_comms* communicators.

mpi_array.benchmarks.bench_creation.CommsAllocBench.time_alloc_locale_buffer

CommsAllocBench.**time_alloc_locale_buffer**(*shape*)
Time call of *mpi_array.comms.LocaleComms.alloc_locale_buffer()*.

Attributes

cart_comm_dims	
goal_time	
locale_comms	A *mpi_array.comms.CartLocaleComms* instance used to allocate memory.
module	The *module* used to create array instances.
param_names	
params	The set of array-shape parameters.
repeat	
warmup_time	

mpi_array.benchmarks.bench_creation.CommsAllocBench.cart_comm_dims

CommsAllocBench.**cart_comm_dims = None**

mpi_array.benchmarks.bench_creation.CommsAllocBench.goal_time

CommsAllocBench.**goal_time = 5.0**

mpi_array.benchmarks.bench_creation.CommsAllocBench.locale_comms

CommsAllocBench.**locale_comms**
A *mpi_array.comms.CartLocaleComms* instance used to allocate memory.

mpi_array.benchmarks.bench_creation.CommsAllocBench.module

CommsAllocBench.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_creation.CommsAllocBench.param_names

CommsAllocBench.**param_names = ['shape']**

mpi_array.benchmarks.bench_creation.CommsAllocBench.params

CommsAllocBench.**params = [[(4, 1024, 1024), (64, 1024, 1024), (1024, 1024, 1024)]]**
The set of array-shape parameters.

mpi_array.benchmarks.bench_creation.CommsAllocBench.repeat

CommsAllocBench.**repeat = 10**

mpi_array.benchmarks.bench_creation.CommsAllocBench.warmup_time

CommsAllocBench.**warmup_time = 2.0**

mpi_array.benchmarks.bench_creation.CommsCreateBench

class mpi_array.benchmarks.bench_creation.**CommsCreateBench**
Bases: *mpi_array.benchmarks.bench_creation.CreateBench*

Benchmarks for *mpi_array.comms.LocaleComms* and *mpi_array.comms.CartLocaleComms* construction.

Methods

__init__()	Initialise, set *module* to None.
free(a)	See *free_mpi_array_obj()*.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
setup()	Import *mpi_array* module and assign to self.module.
time_cart_locale_comms()	Time construction of *mpi_array.comms.CartLocaleComms*.
time_locale_comms()	Time construction of *mpi_array.comms.LocaleComms*.

mpi_array.benchmarks.bench_creation.CommsCreateBench.__init__

CommsCreateBench.**__init__**()
Initialise, set *module* to None.

mpi_array.benchmarks.bench_creation.CommsCreateBench.free

CommsCreateBench.**free**(*a*)
See *free_mpi_array_obj()*.

mpi_array.benchmarks.bench_creation.CommsCreateBench.free_mpi_array_obj

CommsCreateBench.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_creation.CommsCreateBench.get_globale_shape

CommsCreateBench.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_creation.CommsCreateBench.setup

CommsCreateBench.**setup**()
Import *mpi_array* module and assign to self.module.

mpi_array.benchmarks.bench_creation.CommsCreateBench.time_cart_locale_comms

CommsCreateBench.**time_cart_locale_comms**()
Time construction of *mpi_array.comms.CartLocaleComms*.

mpi_array.benchmarks.bench_creation.CommsCreateBench.time_locale_comms

CommsCreateBench.**time_locale_comms**()
Time construction of *mpi_array.comms.LocaleComms*.

Attributes

cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	No param, is None.
repeat	
warmup_time	

mpi_array.benchmarks.bench_creation.CommsCreateBench.cart_comm_dims

CommsCreateBench.**cart_comm_dims = None**

mpi_array.benchmarks.bench_creation.CommsCreateBench.goal_time

CommsCreateBench.**goal_time = 5.0**

mpi_array.benchmarks.bench_creation.CommsCreateBench.module

CommsCreateBench.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_creation.CommsCreateBench.param_names

CommsCreateBench.**param_names = ['shape']**

mpi_array.benchmarks.bench_creation.CommsCreateBench.params

CommsCreateBench.**params = None**
No param, is None.

mpi_array.benchmarks.bench_creation.CommsCreateBench.repeat

CommsCreateBench.**repeat = 10**

mpi_array.benchmarks.bench_creation.CommsCreateBench.warmup_time

CommsCreateBench.**warmup_time = 2.0**

mpi_array.benchmarks.bench_creation.CreateBench

class mpi_array.benchmarks.bench_creation.**CreateBench**
Bases: *mpi_array.benchmarks.core.Bench*

Base class for array creation benchmarks.

Methods

__init__()	Initialise, set *module* to None.
free(a)	Clean up array resources, over-ridden in sub-classes.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
setup(shape)	Should be over-ridden in sub-classes.

mpi_array.benchmarks.bench_creation.CreateBench.__init__

CreateBench.**__init__**()
Initialise, set *module* to None.

mpi_array.benchmarks.bench_creation.CreateBench.free

CreateBench.**free**(*a*)
Clean up array resources, over-ridden in sub-classes.

mpi_array.benchmarks.bench_creation.CreateBench.free_mpi_array_obj

CreateBench.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_creation.CreateBench.get_globale_shape

CreateBench.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of `int`) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_creation.CreateBench.setup

CreateBench.**setup**(*shape*)
Should be over-ridden in sub-classes.

Attributes

cart_comm_dims	Inter-locale cartesian communicator dims
goal_time	
module	The *module* used to create array instances.
param_names	The name of the array-shape parameters.
params	The set of array-shape parameters.
repeat	Number of repetitions to run each benchmark
warmup_time	

mpi_array.benchmarks.bench_creation.CreateBench.cart_comm_dims

CreateBench.**cart_comm_dims** = None
Inter-locale cartesian communicator dims

mpi_array.benchmarks.bench_creation.CreateBench.goal_time

CreateBench.**goal_time** = 5.0

mpi_array.benchmarks.bench_creation.CreateBench.module

CreateBench.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_creation.CreateBench.param_names

CreateBench.**param_names = ['shape']**
The name of the array-shape parameters.

mpi_array.benchmarks.bench_creation.CreateBench.params

CreateBench.**params = [[(100, 100, 100), (1000, 100, 100), (1024, 1024, 1024)]]**
The set of array-shape parameters.

mpi_array.benchmarks.bench_creation.CreateBench.repeat

CreateBench.**repeat = 10**
Number of repetitions to run each benchmark

mpi_array.benchmarks.bench_creation.CreateBench.warmup_time

CreateBench.**warmup_time = 2.0**

mpi_array.benchmarks.bench_creation.MangoCreateBench

class mpi_array.benchmarks.bench_creation.**MangoCreateBench**
Bases: *mpi_array.benchmarks.bench_creation.NumpyCreateBench*

Benchmarks for `mango.empty()` and `mango.zeros()` (mango tomography software).

Methods

__init__()	Initialise, set *module* to None.
free(a)	Clean up array resources, over-ridden in sub-classes.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
setup(shape)	Import `mango` module and assign to `self.module`.
time_empty(shape)	Time uninitialised array creation.
time_full(shape)	No `full` function in `mango`.
time_ones(shape)	Time one-initialised array creation.
time_zeros(shape)	Time zero-initialised array creation.

mpi_array.benchmarks.bench_creation.MangoCreateBench.__init__

MangoCreateBench.**__init__**()
Initialise, set *module* to None.

mpi_array.benchmarks.bench_creation.MangoCreateBench.free

MangoCreateBench.**free**(*a*)
Clean up array resources, over-ridden in sub-classes.

mpi_array.benchmarks.bench_creation.MangoCreateBench.free_mpi_array_obj

MangoCreateBench.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_creation.MangoCreateBench.get_globale_shape

MangoCreateBench.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of `int`) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_creation.MangoCreateBench.setup

MangoCreateBench.**setup**(*shape*)
Import `mango` module and assign to `self.module`.

mpi_array.benchmarks.bench_creation.MangoCreateBench.time_empty

MangoCreateBench.**time_empty**(*shape*)
Time uninitialised array creation.

mpi_array.benchmarks.bench_creation.MangoCreateBench.time_full

MangoCreateBench.**time_full**(*shape*)
No `full` function in `mango`.

mpi_array.benchmarks.bench_creation.MangoCreateBench.time_ones

MangoCreateBench.**time_ones**(*shape*)
Time one-initialised array creation.

mpi_array.benchmarks.bench_creation.MangoCreateBench.time_zeros

MangoCreateBench.**time_zeros**(*shape*)
Time zero-initialised array creation.

Attributes

cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
repeat	
warmup_time	

mpi_array.benchmarks.bench_creation.MangoCreateBench.cart_comm_dims

MangoCreateBench.**cart_comm_dims = None**

mpi_array.benchmarks.bench_creation.MangoCreateBench.goal_time

MangoCreateBench.**goal_time = 5.0**

mpi_array.benchmarks.bench_creation.MangoCreateBench.module

MangoCreateBench.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_creation.MangoCreateBench.param_names

MangoCreateBench.**param_names = ['shape']**

mpi_array.benchmarks.bench_creation.MangoCreateBench.params

MangoCreateBench.**params = [[(100, 100, 100), (1000, 100, 100), (1024, 1024, 1024)]]**

mpi_array.benchmarks.bench_creation.MangoCreateBench.repeat

MangoCreateBench.**repeat = 10**

mpi_array.benchmarks.bench_creation.MangoCreateBench.warmup_time

MangoCreateBench.**warmup_time = 2.0**

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench

class mpi_array.benchmarks.bench_creation.**MpiArrayCreateBench**
Bases: *mpi_array.benchmarks.bench_creation.NumpyCreateBench*

Benchmarks for `mpi_array.empty()`, `mpi_array.zeros()` etc.

Methods

__init__()	Initialise, set module to None.
free(a)	See free_mpi_array_obj().
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
setup(shape)	Import mpi_array module and assign to self.module.
time_empty(shape)	Time uninitialised array creation.
time_full(shape)	Time value-initialised array creation.
time_ones(shape)	Time one-initialised array creation.
time_zeros(shape)	Time zero-initialised array creation.

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.__init__

MpiArrayCreateBench.**__init__**()
Initialise, set *module* to None.

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.free

MpiArrayCreateBench.**free**(*a*)
See *free_mpi_array_obj()*.

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.free_mpi_array_obj

MpiArrayCreateBench.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.get_globale_shape

MpiArrayCreateBench.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.setup

MpiArrayCreateBench.**setup**(*shape*)
Import *mpi_array* module and assign to self.module.

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.time_empty

MpiArrayCreateBench.**time_empty**(*shape*)
Time uninitialised array creation.

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.time_full

MpiArrayCreateBench.**time_full**(*shape*)
Time value-initialised array creation.

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.time_ones

MpiArrayCreateBench.**time_ones**(*shape*)
Time one-initialised array creation.

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.time_zeros

MpiArrayCreateBench.**time_zeros**(*shape*)
Time zero-initialised array creation.

Attributes

cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
repeat	
warmup_time	

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.cart_comm_dims

MpiArrayCreateBench.**cart_comm_dims** = None

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.goal_time

MpiArrayCreateBench.**goal_time** = 5.0

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.module

MpiArrayCreateBench.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.param_names

MpiArrayCreateBench.**param_names = ['shape']**

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.params

MpiArrayCreateBench.**params = [[(100, 100, 100), (1000, 100, 100), (1024, 1024, 1024)]]**

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.repeat

MpiArrayCreateBench.**repeat = 10**

mpi_array.benchmarks.bench_creation.MpiArrayCreateBench.warmup_time

MpiArrayCreateBench.**warmup_time = 2.0**

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench

class mpi_array.benchmarks.bench_creation.**MpiArrayCreateLikeBench**
Bases: *mpi_array.benchmarks.bench_creation.CreateBench*

Benchmarks for `mpi_array.empty_like()`, `mpi_array.zeros_like()` etc.

Methods

__init__()	Initialise, set *module* to None.
free(a)	See *free_mpi_array_obj()*.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
setup(shape)	Import *mpi_array* module and assign to `self.module`.
teardown(shape)	Free the *like* array.
time_empty_like(shape)	Test creation of *uninitialised* array using another array as template.
time_ones_like(shape)	Test creation of *one initialised* array using another array as template.

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.__init__

MpiArrayCreateLikeBench.**__init__**()
Initialise, set *module* to None.

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.free

MpiArrayCreateLikeBench.**free**(*a*)
See *free_mpi_array_obj()*.

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.free_mpi_array_obj

MpiArrayCreateLikeBench.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.get_globale_shape

MpiArrayCreateLikeBench.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of `int`) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.setup

MpiArrayCreateLikeBench.**setup**(*shape*)
Import *mpi_array* module and assign to `self.module`.

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.teardown

MpiArrayCreateLikeBench.**teardown**(*shape*)
Free the *like* array.

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.time_empty_like

MpiArrayCreateLikeBench.**time_empty_like**(*shape*)
Test creation of *uninitialised* array using another array as template.

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.time_ones_like

MpiArrayCreateLikeBench.**time_ones_like**(*shape*)
Test creation of *one initialised* array using another array as template.

Attributes

cart_comm_dims	
goal_time	
module	The *module* used to create array instances.

Continued on next page

Table 1.25 – continued from previous page

param_names
params
repeat
warmup_time

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.cart_comm_dims

MpiArrayCreateLikeBench.**cart_comm_dims = None**

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.goal_time

MpiArrayCreateLikeBench.**goal_time = 5.0**

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.module

MpiArrayCreateLikeBench.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.param_names

MpiArrayCreateLikeBench.**param_names = ['shape']**

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.params

MpiArrayCreateLikeBench.**params = [[(100, 100, 100), (1000, 100, 100), (1024, 1024, 1024)]]**

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.repeat

MpiArrayCreateLikeBench.**repeat = 10**

mpi_array.benchmarks.bench_creation.MpiArrayCreateLikeBench.warmup_time

MpiArrayCreateLikeBench.**warmup_time = 2.0**

mpi_array.benchmarks.bench_creation.NumpyCreateBench

class mpi_array.benchmarks.bench_creation.**NumpyCreateBench**
Bases: *mpi_array.benchmarks.bench_creation.CreateBench*

Comparison benchmarks for `numpy.empty()`, `numpy.zeros()` etc.

Methods

__init__()	Initialise, set *module* to None.
free(a)	Clean up array resources, over-ridden in sub-classes.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
setup(shape)	Import numpy module and assign to *module*.
time_empty(shape)	Time uninitialised array creation.
time_full(shape)	Time value-initialised array creation.
time_ones(shape)	Time one-initialised array creation.
time_zeros(shape)	Time zero-initialised array creation.

mpi_array.benchmarks.bench_creation.NumpyCreateBench.__init__

NumpyCreateBench.**__init__**()
Initialise, set *module* to None.

mpi_array.benchmarks.bench_creation.NumpyCreateBench.free

NumpyCreateBench.**free**(*a*)
Clean up array resources, over-ridden in sub-classes.

mpi_array.benchmarks.bench_creation.NumpyCreateBench.free_mpi_array_obj

NumpyCreateBench.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_creation.NumpyCreateBench.get_globale_shape

NumpyCreateBench.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_creation.NumpyCreateBench.setup

NumpyCreateBench.**setup**(*shape*)
Import numpy module and assign to *module*.

mpi_array.benchmarks.bench_creation.NumpyCreateBench.time_empty

NumpyCreateBench.**time_empty**(*shape*)
Time uninitialised array creation.

mpi_array.benchmarks.bench_creation.NumpyCreateBench.time_full

NumpyCreateBench.**time_full**(*shape*)
Time value-initialised array creation.

mpi_array.benchmarks.bench_creation.NumpyCreateBench.time_ones

NumpyCreateBench.**time_ones**(*shape*)
Time one-initialised array creation.

mpi_array.benchmarks.bench_creation.NumpyCreateBench.time_zeros

NumpyCreateBench.**time_zeros**(*shape*)
Time zero-initialised array creation.

Attributes

cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
repeat	
warmup_time	

mpi_array.benchmarks.bench_creation.NumpyCreateBench.cart_comm_dims

NumpyCreateBench.**cart_comm_dims** = None

mpi_array.benchmarks.bench_creation.NumpyCreateBench.goal_time

NumpyCreateBench.**goal_time** = 5.0

mpi_array.benchmarks.bench_creation.NumpyCreateBench.module

NumpyCreateBench.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_creation.NumpyCreateBench.param_names

NumpyCreateBench.**param_names** = ['shape']

mpi_array.benchmarks.bench_creation.NumpyCreateBench.params

NumpyCreateBench.**params** = [[(100, 100, 100), (1000, 100, 100), (1024, 1024, 1024)]]

mpi_array.benchmarks.bench_creation.NumpyCreateBench.repeat

NumpyCreateBench.**repeat = 10**

mpi_array.benchmarks.bench_creation.NumpyCreateBench.warmup_time

NumpyCreateBench.**warmup_time = 2.0**

mpi_array.benchmarks.bench_ufunc

Benchmarks for ufuncs.

Functions

create_ufunc_bench(module_name, ufunc_name, ...)	Creates a new benchmark type for the ufunc `getattr(module_name, ufunc_name)`.
create_ufunc_benchmarks(ufunc_names, ...[, ...])	Creates a new benchmark type for each ufunc name in `ufunc_names` and for each module name in *module_names*.
find_scipy_ufuncs()	Imports the `scipy.special.erf()` ufunc and sets it as an attribute of the `numpy` and *mpi_array* modules.

mpi_array.benchmarks.bench_ufunc.create_ufunc_bench

mpi_array.benchmarks.bench_ufunc.**create_ufunc_bench**(*module_name*, *ufunc_name*, *method_dict*)

Creates a new benchmark type for the ufunc `getattr(module_name, ufunc_name)`.

mpi_array.benchmarks.bench_ufunc.create_ufunc_benchmarks

mpi_array.benchmarks.bench_ufunc.**create_ufunc_benchmarks**(*ufunc_names*, *module_names*, *module=None*)

Creates a new benchmark type for each ufunc name in `ufunc_names` and for each module name in *module_names*. Sets created types as attributes of the module *module*.

mpi_array.benchmarks.bench_ufunc.find_scipy_ufuncs

mpi_array.benchmarks.bench_ufunc.**find_scipy_ufuncs**()

Imports the `scipy.special.erf()` ufunc and sets it as an attribute of the `numpy` and *mpi_array* modules.

Classes

Bench

Continued on next page

Table 1.29 – continued from previous page

MpiArrayUfuncBench([ufunc_name])	Benchmarks for *mpi_array* ufuncs..
MpiArrayUfuncBench_add()	Benchmark for numpy.add with *mpi_array.globale.gndarray* array inputs.
MpiArrayUfuncBench_invsqrt()	Benchmarks for mpi_array.power(ary, -0.5) ufunc for *mpi_array.globale.gndarray* input.
MpiArrayUfuncBench_log()	Benchmark for numpy.log with *mpi_array.globale.gndarray* array inputs.
MpiArrayUfuncBench_log10()	Benchmark for numpy.log10 with *mpi_array.globale.gndarray* array inputs.
MpiArrayUfuncBench_multiply()	Benchmark for numpy.multiply with *mpi_array.globale.gndarray* array inputs.
MpiArrayUfuncBench_subtract()	Benchmark for numpy.subtract with *mpi_array.globale.gndarray* array inputs.
MpiArrayUfuncBench_true_divide()	Benchmark for numpy.true_divide with *mpi_array.globale.gndarray* array inputs.
NumpyUfuncBench([ufunc_name])	Comparison benchmarks for numpy.ufunc instances.
NumpyUfuncBench_add()	Benchmark for numpy.add with numpy.ndarray array inputs.
NumpyUfuncBench_invsqrt()	Benchmarks for numpy.power(ary, -0.5) ufunc for numpy.ndarray input.
NumpyUfuncBench_log()	Benchmark for numpy.log with numpy.ndarray array inputs.
NumpyUfuncBench_log10()	Benchmark for numpy.log10 with numpy.ndarray array inputs.
NumpyUfuncBench_multiply()	Benchmark for numpy.multiply with numpy.ndarray array inputs.
NumpyUfuncBench_subtract()	Benchmark for numpy.subtract with numpy.ndarray array inputs.
NumpyUfuncBench_true_divide()	Benchmark for numpy.true_divide with numpy.ndarray array inputs.
UfuncBench([ufunc_name])	Base class for array ufunc benchmarks.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench

class mpi_array.benchmarks.bench_ufunc.**MpiArrayUfuncBench**(*ufunc_name=None*)

Bases: *mpi_array.benchmarks.bench_ufunc.UfuncBench*

Benchmarks for *mpi_array* ufuncs..

Methods

__init__([ufunc_name])	Initialise.
free(a)	See *free_mpi_array_obj()*.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.

Continued on next page

Table 1.30 – continued from previous page

initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.
setup(shape[, dtype])	Import *mpi_array* module and assign to self.module.
teardown(shape)	Free arrays allocated during *setup()*.
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.__init__

MpiArrayUfuncBench.**__init__**(*ufunc_name=None*)

Initialise.

Parameters **ufunc_name** (str) – Name of the ufunc, should be the name of an attribute of *module*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.free

MpiArrayUfuncBench.**free**(*a*)

See *free_mpi_array_obj()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.free_mpi_array_obj

MpiArrayUfuncBench.**free_mpi_array_obj**(*a*)

Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.get_globale_shape

MpiArrayUfuncBench.**get_globale_shape**(*locale_shape*)

Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.initialise

MpiArrayUfuncBench.**initialise**(*shape*, *dtype*, *module_name*)

Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.initialise_arrays

MpiArrayUfuncBench.**initialise_arrays**(*shape*)

Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of int) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.setup

MpiArrayUfuncBench.**setup**(*shape*, *dtype='float64'*)

Import *mpi_array* module and assign to self.module.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.teardown

MpiArrayUfuncBench.**teardown**(*shape*)

Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.try_import_for_setup

MpiArrayUfuncBench.**try_import_for_setup**(*module_name*)

Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (str) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an ImportError.

See also:

mpi_array.benchmarks.utils.try_import_for_setup().

Attributes

a_ary	First input array.
a_ary_range	A (low, high) tuple indicating the uniform random range of scalars for the *a_ary* array elements.
b_ary	Second input array.
b_ary_range	A (low, high) tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
MpiArrayUfuncBench.cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
random_state	A numpy.random.RandomState instance, used to generate random values in arrays.
repeat	
ufunc	The numpy.ufunc for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.a_ary

MpiArrayUfuncBench.**a_ary**
First input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.a_ary_range

MpiArrayUfuncBench.**a_ary_range**
A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.b_ary

MpiArrayUfuncBench.**b_ary**
Second input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.b_ary_range

MpiArrayUfuncBench.**b_ary_range**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.b_scalar

MpiArrayUfuncBench.**b_scalar**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.c_ary

MpiArrayUfuncBench.**c_ary**
Output array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.goal_time

MpiArrayUfuncBench.**goal_time = 1.0**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.module

MpiArrayUfuncBench.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.param_names

MpiArrayUfuncBench.**param_names = ['shape']**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.params

MpiArrayUfuncBench.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.random_state

MpiArrayUfuncBench.**random_state**
A numpy.random.RandomState instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.repeat

MpiArrayUfuncBench.**repeat = 8**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.ufunc

MpiArrayUfuncBench.**ufunc**
The numpy.ufunc for this benchmark.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench.warmup_time

MpiArrayUfuncBench.**warmup_time = 1.0**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add

class mpi_array.benchmarks.bench_ufunc.**MpiArrayUfuncBench_add**
Bases: *mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench*

Benchmark for numpy.add with *mpi_array.globale.gndarray* array inputs.

Methods

__init__()	
free(a)	See *free_mpi_array_obj()*.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.
setup(shape[, dtype])	Import *mpi_array* module and assign to self.module.
teardown(shape)	Free arrays allocated during *setup()*.
time_array_array_op(shape[, dtype])	Timing for two input ufuncs (two array inputs).

Continued on next page

Table 1.32 – continued from previous page

time_array_scalar_op(shape[, dtype])	Timing for two input ufuncs (one array input, one scalar input).
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.__init__

MpiArrayUfuncBench_add.**__init__**()

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.free

MpiArrayUfuncBench_add.**free**(*a*)

See *free_mpi_array_obj()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.free_mpi_array_obj

MpiArrayUfuncBench_add.**free_mpi_array_obj**(*a*)

Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.get_globale_shape

MpiArrayUfuncBench_add.**get_globale_shape**(*locale_shape*)

Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of `int`) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.initialise

MpiArrayUfuncBench_add.**initialise**(*shape*, *dtype*, *module_name*)

Sets the *module*, `dtype` and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.initialise_arrays

MpiArrayUfuncBench_add.**initialise_arrays**(*shape*)

Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of `int`) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.setup

MpiArrayUfuncBench_add.**setup**(*shape*, *dtype='float64'*)

Import *mpi_array* module and assign to `self.module`.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.teardown

MpiArrayUfuncBench_add.**teardown**(*shape*)
Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.time_array_array_op

MpiArrayUfuncBench_add.**time_array_array_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (two array inputs).

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.time_array_scalar_op

MpiArrayUfuncBench_add.**time_array_scalar_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (one array input, one scalar input).

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.try_import_for_setup

MpiArrayUfuncBench_add.**try_import_for_setup**(*module_name*)
Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (str) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an ImportError.

See also:

mpi_array.benchmarks.utils.try_import_for_setup().

Attributes

a_ary	First input array.
a_ary_range	A (low, high) tuple indicating the uniform random range of scalars for the *a_ary* array elements.
b_ary	Second input array.
b_ary_range	A (low, high) tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
MpiArrayUfuncBench_add.cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
random_state	A numpy.random.RandomState instance, used to generate random values in arrays.
repeat	

Continued on next page

Table 1.33 – continued from previous page

ufunc	The `numpy.ufunc` for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.a_ary

MpiArrayUfuncBench_add.**a_ary**
First input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.a_ary_range

MpiArrayUfuncBench_add.**a_ary_range**
A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.b_ary

MpiArrayUfuncBench_add.**b_ary**
Second input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.b_ary_range

MpiArrayUfuncBench_add.**b_ary_range**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.b_scalar

MpiArrayUfuncBench_add.**b_scalar**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.c_ary

MpiArrayUfuncBench_add.**c_ary**
Output array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.goal_time

MpiArrayUfuncBench_add.**goal_time = 1.0**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.module

MpiArrayUfuncBench_add.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.param_names

MpiArrayUfuncBench_add.**param_names = ['shape']**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.params

MpiArrayUfuncBench_add.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.random_state

MpiArrayUfuncBench_add.**random_state**
A `numpy.random.RandomState` instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.repeat

MpiArrayUfuncBench_add.**repeat = 8**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.ufunc

MpiArrayUfuncBench_add.**ufunc**
The `numpy.ufunc` for this benchmark.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_add.warmup_time

MpiArrayUfuncBench_add.**warmup_time = 1.0**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt

class mpi_array.benchmarks.bench_ufunc.**MpiArrayUfuncBench_invsqrt**
Bases: *mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench*

Benchmarks for `mpi_array.power(ary, -0.5)` ufunc for *mpi_array.globale.gndarray* input.

Methods

__init__()	
free(a)	See *free_mpi_array_obj()*.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, `dtype` and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.

Continued on next page

Table 1.34 – continued from previous page

setup(shape[, dtype])	Import *mpi_array* module and assign to `self.module`.
teardown(shape)	Free arrays allocated during *setup()*.
time_array_op(shape[, dtype])	Timing for single argument ufuncs.
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.__init__

MpiArrayUfuncBench_invsqrt.**__init__**()

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.free

MpiArrayUfuncBench_invsqrt.**free**(*a*)

See *free_mpi_array_obj()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.free_mpi_array_obj

MpiArrayUfuncBench_invsqrt.**free_mpi_array_obj**(*a*)

Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.get_globale_shape

MpiArrayUfuncBench_invsqrt.**get_globale_shape**(*locale_shape*)

Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of `int`) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.initialise

MpiArrayUfuncBench_invsqrt.**initialise**(*shape*, *dtype*, *module_name*)

Sets the *module*, `dtype` and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.initialise_arrays

MpiArrayUfuncBench_invsqrt.**initialise_arrays**(*shape*)

Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of `int`) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.setup

MpiArrayUfuncBench_invsqrt.**setup**(*shape*, *dtype='float64'*)
Import *mpi_array* module and assign to `self.module`.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.teardown

MpiArrayUfuncBench_invsqrt.**teardown**(*shape*)
Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.time_array_op

MpiArrayUfuncBench_invsqrt.**time_array_op**(*shape*, *dtype='float64'*)
Timing for single argument ufuncs.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.try_import_for_setup

MpiArrayUfuncBench_invsqrt.**try_import_for_setup**(*module_name*)
Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (`str`) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an `ImportError`.

See also:

`mpi_array.benchmarks.utils.try_import_for_setup()`.

Attributes

a_ary	First input array.
a_ary_range	A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.
b_ary	Second input array.
b_ary_range	A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
MpiArrayUfuncBench_invsqrt.cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
random_state	A `numpy.random.RandomState` instance, used to generate random values in arrays.

Continued on next page

Table 1.35 – continued from previous page

repeat	
ufunc	The `numpy.ufunc` for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.a_ary

`MpiArrayUfuncBench_invsqrt.`**`a_ary`**
First input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.a_ary_range

`MpiArrayUfuncBench_invsqrt.`**`a_ary_range`**
A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.b_ary

`MpiArrayUfuncBench_invsqrt.`**`b_ary`**
Second input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.b_ary_range

`MpiArrayUfuncBench_invsqrt.`**`b_ary_range`**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.b_scalar

`MpiArrayUfuncBench_invsqrt.`**`b_scalar`**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.c_ary

`MpiArrayUfuncBench_invsqrt.`**`c_ary`**
Output array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.goal_time

`MpiArrayUfuncBench_invsqrt.`**`goal_time = 1.0`**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.module

`MpiArrayUfuncBench_invsqrt.`**`module`**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.param_names

MpiArrayUfuncBench_invsqrt.**param_names = ['shape']**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.params

MpiArrayUfuncBench_invsqrt.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.random_state

MpiArrayUfuncBench_invsqrt.**random_state**
A numpy.random.RandomState instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.repeat

MpiArrayUfuncBench_invsqrt.**repeat = 8**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.ufunc

MpiArrayUfuncBench_invsqrt.**ufunc**
The numpy.ufunc for this benchmark.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_invsqrt.warmup_time

MpiArrayUfuncBench_invsqrt.**warmup_time = 1.0**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log

class mpi_array.benchmarks.bench_ufunc.**MpiArrayUfuncBench_log**
Bases: *mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench*

Benchmark for numpy.log with *mpi_array.globale.gndarray* array inputs.

Methods

__init__()	
free(a)	See *free_mpi_array_obj()*.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.

Continued on next page

Table 1.36 – continued from previous page

setup(shape[, dtype])	Import *mpi_array* module and assign to `self.module`.
teardown(shape)	Free arrays allocated during *setup()*.
time_array_op(shape[, dtype])	Timing for single input ufuncs (single array input).
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.__init__

`MpiArrayUfuncBench_log.`**`__init__`**`()`

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.free

`MpiArrayUfuncBench_log.`**`free`**`(`*a*`)`

See *free_mpi_array_obj()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.free_mpi_array_obj

`MpiArrayUfuncBench_log.`**`free_mpi_array_obj`**`(`*a*`)`

Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.get_globale_shape

`MpiArrayUfuncBench_log.`**`get_globale_shape`**`(`*locale_shape*`)`

Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of `int`) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.initialise

`MpiArrayUfuncBench_log.`**`initialise`**`(`*shape*, *dtype*, *module_name*`)`

Sets the *module*, `dtype` and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.initialise_arrays

`MpiArrayUfuncBench_log.`**`initialise_arrays`**`(`*shape*`)`

Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of `int`) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.setup

MpiArrayUfuncBench_log.**setup**(*shape*, *dtype='float64'*)
Import *mpi_array* module and assign to `self.module`.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.teardown

MpiArrayUfuncBench_log.**teardown**(*shape*)
Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.time_array_op

MpiArrayUfuncBench_log.**time_array_op**(*shape*, *dtype='float64'*)
Timing for single input ufuncs (single array input).

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.try_import_for_setup

MpiArrayUfuncBench_log.**try_import_for_setup**(*module_name*)
Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (`str`) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an `ImportError`.

See also:

`mpi_array.benchmarks.utils.try_import_for_setup()`.

Attributes

a_ary	First input array.
a_ary_range	A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.
b_ary	Second input array.
b_ary_range	A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
MpiArrayUfuncBench_log.cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
random_state	A `numpy.random.RandomState` instance, used to generate random values in arrays.
repeat	

Continued on next page

Table 1.37 – continued from previous page

ufunc	The `numpy.ufunc` for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.a_ary

MpiArrayUfuncBench_log.**a_ary**
First input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.a_ary_range

MpiArrayUfuncBench_log.**a_ary_range**
A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.b_ary

MpiArrayUfuncBench_log.**b_ary**
Second input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.b_ary_range

MpiArrayUfuncBench_log.**b_ary_range**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.b_scalar

MpiArrayUfuncBench_log.**b_scalar**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.c_ary

MpiArrayUfuncBench_log.**c_ary**
Output array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.goal_time

MpiArrayUfuncBench_log.**goal_time = 1.0**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.module

MpiArrayUfuncBench_log.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.param_names

MpiArrayUfuncBench_log.**param_names** = ['shape']

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.params

MpiArrayUfuncBench_log.**params** = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.random_state

MpiArrayUfuncBench_log.**random_state**
A numpy.random.RandomState instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.repeat

MpiArrayUfuncBench_log.**repeat** = 8

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.ufunc

MpiArrayUfuncBench_log.**ufunc**
The numpy.ufunc for this benchmark.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log.warmup_time

MpiArrayUfuncBench_log.**warmup_time** = 1.0

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10

class mpi_array.benchmarks.bench_ufunc.**MpiArrayUfuncBench_log10**
Bases: *mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench*

Benchmark for numpy.log10 with *mpi_array.globale.gndarray* array inputs.

Methods

__init__()	
free(a)	See *free_mpi_array_obj()*.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.

Continued on next page

Table 1.38 – continued from previous page

setup(shape[, dtype])	Import *mpi_array* module and assign to `self.module`.
teardown(shape)	Free arrays allocated during *setup()*.
time_array_op(shape[, dtype])	Timing for single input ufuncs (single array input).
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.__init__

`MpiArrayUfuncBench_log10.`**`__init__`**`()`

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.free

`MpiArrayUfuncBench_log10.`**`free`**`(`*a*`)`
See *free_mpi_array_obj()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.free_mpi_array_obj

`MpiArrayUfuncBench_log10.`**`free_mpi_array_obj`**`(`*a*`)`
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.get_globale_shape

`MpiArrayUfuncBench_log10.`**`get_globale_shape`**`(`*locale_shape*`)`
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of `int`) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.initialise

`MpiArrayUfuncBench_log10.`**`initialise`**`(`*shape*, *dtype*, *module_name*`)`
Sets the *module*, `dtype` and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.initialise_arrays

`MpiArrayUfuncBench_log10.`**`initialise_arrays`**`(`*shape*`)`
Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of `int`) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.setup

MpiArrayUfuncBench_log10.**setup**(*shape*, *dtype='float64'*)
Import *mpi_array* module and assign to `self.module`.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.teardown

MpiArrayUfuncBench_log10.**teardown**(*shape*)
Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.time_array_op

MpiArrayUfuncBench_log10.**time_array_op**(*shape*, *dtype='float64'*)
Timing for single input ufuncs (single array input).

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.try_import_for_setup

MpiArrayUfuncBench_log10.**try_import_for_setup**(*module_name*)
Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (`str`) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an `ImportError`.

See also:

`mpi_array.benchmarks.utils.try_import_for_setup()`.

Attributes

a_ary	First input array.
a_ary_range	A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.
b_ary	Second input array.
b_ary_range	A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
MpiArrayUfuncBench_log10.cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
random_state	A `numpy.random.RandomState` instance, used to generate random values in arrays.

Continued on next page

Table 1.39 – continued from previous page

repeat	
ufunc	The `numpy.ufunc` for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.a_ary

`MpiArrayUfuncBench_log10.`**`a_ary`**
First input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.a_ary_range

`MpiArrayUfuncBench_log10.`**`a_ary_range`**
A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.b_ary

`MpiArrayUfuncBench_log10.`**`b_ary`**
Second input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.b_ary_range

`MpiArrayUfuncBench_log10.`**`b_ary_range`**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.b_scalar

`MpiArrayUfuncBench_log10.`**`b_scalar`**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.c_ary

`MpiArrayUfuncBench_log10.`**`c_ary`**
Output array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.goal_time

`MpiArrayUfuncBench_log10.`**`goal_time = 1.0`**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.module

`MpiArrayUfuncBench_log10.`**`module`**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.param_names

MpiArrayUfuncBench_log10.**param_names = ['shape']**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.params

MpiArrayUfuncBench_log10.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.random_state

MpiArrayUfuncBench_log10.**random_state**
A numpy.random.RandomState instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.repeat

MpiArrayUfuncBench_log10.**repeat = 8**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.ufunc

MpiArrayUfuncBench_log10.**ufunc**
The numpy.ufunc for this benchmark.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_log10.warmup_time

MpiArrayUfuncBench_log10.**warmup_time = 1.0**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply

class mpi_array.benchmarks.bench_ufunc.**MpiArrayUfuncBench_multiply**
Bases: *mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench*

Benchmark for numpy.multiply with *mpi_array.globale.gndarray* array inputs.

Methods

__init__()	
free(a)	See *free_mpi_array_obj()*.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.

Continued on next page

Table 1.40 – continued from previous page

setup(shape[, dtype])	Import *mpi_array* module and assign to `self.module`.
teardown(shape)	Free arrays allocated during *setup()*.
time_array_array_op(shape[, dtype])	Timing for two input ufuncs (two array inputs).
time_array_scalar_op(shape[, dtype])	Timing for two input ufuncs (one array input, one scalar input).
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.__init__

MpiArrayUfuncBench_multiply.**__init__**()

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.free

MpiArrayUfuncBench_multiply.**free**(*a*)
See *free_mpi_array_obj()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.free_mpi_array_obj

MpiArrayUfuncBench_multiply.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.get_globale_shape

MpiArrayUfuncBench_multiply.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of `int`) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.initialise

MpiArrayUfuncBench_multiply.**initialise**(*shape*, *dtype*, *module_name*)
Sets the *module*, `dtype` and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.initialise_arrays

MpiArrayUfuncBench_multiply.**initialise_arrays**(*shape*)
Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of `int`) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.setup

MpiArrayUfuncBench_multiply.**setup**(*shape*, *dtype='float64'*)
Import *mpi_array* module and assign to `self.module`.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.teardown

MpiArrayUfuncBench_multiply.**teardown**(*shape*)
Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.time_array_array_op

MpiArrayUfuncBench_multiply.**time_array_array_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (two array inputs).

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.time_array_scalar_op

MpiArrayUfuncBench_multiply.**time_array_scalar_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (one array input, one scalar input).

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.try_import_for_setup

MpiArrayUfuncBench_multiply.**try_import_for_setup**(*module_name*)
Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (`str`) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an `ImportError`.

See also:

`mpi_array.benchmarks.utils.try_import_for_setup()`.

Attributes

a_ary	First input array.
a_ary_range	A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.
b_ary	Second input array.
b_ary_range	A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
MpiArrayUfuncBench_multiply.cart_comm_dims	
goal_time	

Continued on next page

Table 1.41 – continued from previous page

module	The *module* used to create array instances.
param_names	
params	
random_state	A `numpy.random.RandomState` instance, used to generate random values in arrays.
repeat	
ufunc	The `numpy.ufunc` for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.a_ary

MpiArrayUfuncBench_multiply.**a_ary**
First input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.a_ary_range

MpiArrayUfuncBench_multiply.**a_ary_range**
A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.b_ary

MpiArrayUfuncBench_multiply.**b_ary**
Second input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.b_ary_range

MpiArrayUfuncBench_multiply.**b_ary_range**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.b_scalar

MpiArrayUfuncBench_multiply.**b_scalar**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.c_ary

MpiArrayUfuncBench_multiply.**c_ary**
Output array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.goal_time

MpiArrayUfuncBench_multiply.**goal_time = 1.0**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.module

MpiArrayUfuncBench_multiply.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.param_names

MpiArrayUfuncBench_multiply.**param_names = ['shape']**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.params

MpiArrayUfuncBench_multiply.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.random_state

MpiArrayUfuncBench_multiply.**random_state**
A numpy.random.RandomState instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.repeat

MpiArrayUfuncBench_multiply.**repeat = 8**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.ufunc

MpiArrayUfuncBench_multiply.**ufunc**
The numpy.ufunc for this benchmark.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_multiply.warmup_time

MpiArrayUfuncBench_multiply.**warmup_time = 1.0**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract

class mpi_array.benchmarks.bench_ufunc.**MpiArrayUfuncBench_subtract**
Bases: *mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench*

Benchmark for numpy.subtract with *mpi_array.globale.gndarray* array inputs.

Methods

__init__()	
free(a)	See *free_mpi_array_obj()*.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.

Continued on next page

Table 1.42 – continued from previous page

get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.
setup(shape[, dtype])	Import *mpi_array* module and assign to self.module.
teardown(shape)	Free arrays allocated during *setup()*.
time_array_array_op(shape[, dtype])	Timing for two input ufuncs (two array inputs).
time_array_scalar_op(shape[, dtype])	Timing for two input ufuncs (one array input, one scalar input).
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.__init__

MpiArrayUfuncBench_subtract.**__init__**()

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.free

MpiArrayUfuncBench_subtract.**free**(*a*)
See *free_mpi_array_obj()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.free_mpi_array_obj

MpiArrayUfuncBench_subtract.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.get_globale_shape

MpiArrayUfuncBench_subtract.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.initialise

MpiArrayUfuncBench_subtract.**initialise**(*shape*, *dtype*, *module_name*)
Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.initialise_arrays

MpiArrayUfuncBench_subtract.**initialise_arrays**(*shape*)
Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of int) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.setup

MpiArrayUfuncBench_subtract.**setup**(*shape*, *dtype='float64'*)
Import *mpi_array* module and assign to self.module.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.teardown

MpiArrayUfuncBench_subtract.**teardown**(*shape*)
Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.time_array_array_op

MpiArrayUfuncBench_subtract.**time_array_array_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (two array inputs).

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.time_array_scalar_op

MpiArrayUfuncBench_subtract.**time_array_scalar_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (one array input, one scalar input).

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.try_import_for_setup

MpiArrayUfuncBench_subtract.**try_import_for_setup**(*module_name*)
Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (str) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an ImportError.

See also:

mpi_array.benchmarks.utils.try_import_for_setup().

Attributes

a_ary	First input array.
a_ary_range	A (low, high) tuple indicating the uniform random range of scalars for the *a_ary* array elements.

Continued on next page

Table 1.43 – continued from previous page

b_ary	Second input array.
b_ary_range	A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
`MpiArrayUfuncBench_subtract.cart_comm_dims`	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
random_state	A `numpy.random.RandomState` instance, used to generate random values in arrays.
repeat	
ufunc	The `numpy.ufunc` for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.a_ary

`MpiArrayUfuncBench_subtract.`**`a_ary`**
First input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.a_ary_range

`MpiArrayUfuncBench_subtract.`**`a_ary_range`**
A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.b_ary

`MpiArrayUfuncBench_subtract.`**`b_ary`**
Second input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.b_ary_range

`MpiArrayUfuncBench_subtract.`**`b_ary_range`**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.b_scalar

`MpiArrayUfuncBench_subtract.`**`b_scalar`**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.c_ary

`MpiArrayUfuncBench_subtract.`**`c_ary`**
Output array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.goal_time

MpiArrayUfuncBench_subtract.**goal_time = 1.0**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.module

MpiArrayUfuncBench_subtract.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.param_names

MpiArrayUfuncBench_subtract.**param_names = ['shape']**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.params

MpiArrayUfuncBench_subtract.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.random_state

MpiArrayUfuncBench_subtract.**random_state**
A numpy.random.RandomState instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.repeat

MpiArrayUfuncBench_subtract.**repeat = 8**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.ufunc

MpiArrayUfuncBench_subtract.**ufunc**
The numpy.ufunc for this benchmark.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_subtract.warmup_time

MpiArrayUfuncBench_subtract.**warmup_time = 1.0**

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide

class mpi_array.benchmarks.bench_ufunc.**MpiArrayUfuncBench_true_divide**
Bases: *mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench*

Benchmark for numpy.true_divide with *mpi_array.globale.gndarray* array inputs.

Methods

__init__()	
free(a)	See *free_mpi_array_obj()*.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.
setup(shape[, dtype])	Import *mpi_array* module and assign to self.module.
teardown(shape)	Free arrays allocated during *setup()*.
time_array_array_op(shape[, dtype])	Timing for two input ufuncs (two array inputs).
time_array_scalar_op(shape[, dtype])	Timing for two input ufuncs (one array input, one scalar input).
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.__init__

MpiArrayUfuncBench_true_divide.**__init__**()

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.free

MpiArrayUfuncBench_true_divide.**free**(*a*)
See *free_mpi_array_obj()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.free_mpi_array_obj

MpiArrayUfuncBench_true_divide.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.get_globale_shape

MpiArrayUfuncBench_true_divide.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.initialise

MpiArrayUfuncBench_true_divide.**initialise**(*shape*, *dtype*, *module_name*)
Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.initialise_arrays

MpiArrayUfuncBench_true_divide.**initialise_arrays**(*shape*)
Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of int) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.setup

MpiArrayUfuncBench_true_divide.**setup**(*shape*, *dtype='float64'*)
Import *mpi_array* module and assign to self.module.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.teardown

MpiArrayUfuncBench_true_divide.**teardown**(*shape*)
Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.time_array_array_op

MpiArrayUfuncBench_true_divide.**time_array_array_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (two array inputs).

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.time_array_scalar_op

MpiArrayUfuncBench_true_divide.**time_array_scalar_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (one array input, one scalar input).

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.try_import_for_setup

MpiArrayUfuncBench_true_divide.**try_import_for_setup**(*module_name*)
Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (str) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an ImportError.

See also:

mpi_array.benchmarks.utils.try_import_for_setup().

Attributes

a_ary	First input array.
a_ary_range	A (low, high) tuple indicating the uniform random range of scalars for the *a_ary* array elements.

Continued on next page

Table 1.45 – continued from previous page

b_ary	Second input array.
b_ary_range	A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
`MpiArrayUfuncBench_true_divide.cart_comm_dims`	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
random_state	A `numpy.random.RandomState` instance, used to generate random values in arrays.
repeat	
ufunc	The `numpy.ufunc` for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.a_ary

`MpiArrayUfuncBench_true_divide.`**`a_ary`**
First input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.a_ary_range

`MpiArrayUfuncBench_true_divide.`**`a_ary_range`**
A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.b_ary

`MpiArrayUfuncBench_true_divide.`**`b_ary`**
Second input array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.b_ary_range

`MpiArrayUfuncBench_true_divide.`**`b_ary_range`**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.b_scalar

`MpiArrayUfuncBench_true_divide.`**`b_scalar`**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.c_ary

`MpiArrayUfuncBench_true_divide.`**`c_ary`**
Output array.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.goal_time

MpiArrayUfuncBench_true_divide.**goal_time** = 1.0

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.module

MpiArrayUfuncBench_true_divide.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.param_names

MpiArrayUfuncBench_true_divide.**param_names** = ['shape']

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.params

MpiArrayUfuncBench_true_divide.**params** = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.random_state

MpiArrayUfuncBench_true_divide.**random_state**
A numpy.random.RandomState instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.repeat

MpiArrayUfuncBench_true_divide.**repeat** = 8

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.ufunc

MpiArrayUfuncBench_true_divide.**ufunc**
The numpy.ufunc for this benchmark.

mpi_array.benchmarks.bench_ufunc.MpiArrayUfuncBench_true_divide.warmup_time

MpiArrayUfuncBench_true_divide.**warmup_time** = 1.0

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench

class mpi_array.benchmarks.bench_ufunc.**NumpyUfuncBench**(*ufunc_name=None*)
Bases: *mpi_array.benchmarks.bench_ufunc.UfuncBench*

Comparison benchmarks for numpy.ufunc instances.

Methods

__init__([ufunc_name])	Initialise.
free(a)	Clean up array resources, over-ridden in sub-classes.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.
setup(shape[, dtype])	Import numpy module and assign to *module*.
teardown(shape)	Free arrays allocated during *setup()*.
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.__init__

NumpyUfuncBench.**__init__**(*ufunc_name=None*)

Initialise.

Parameters **ufunc_name** (str) – Name of the ufunc, should be the name of an attribute of *module*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.free

NumpyUfuncBench.**free**(*a*)

Clean up array resources, over-ridden in sub-classes.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.free_mpi_array_obj

NumpyUfuncBench.**free_mpi_array_obj**(*a*)

Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.get_globale_shape

NumpyUfuncBench.**get_globale_shape**(*locale_shape*)

Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.initialise

NumpyUfuncBench.**initialise**(*shape*, *dtype*, *module_name*)

Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.initialise_arrays

NumpyUfuncBench.**initialise_arrays**(*shape*)
Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of int) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.setup

NumpyUfuncBench.**setup**(*shape*, *dtype='float64'*)
Import numpy module and assign to *module*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.teardown

NumpyUfuncBench.**teardown**(*shape*)
Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.try_import_for_setup

NumpyUfuncBench.**try_import_for_setup**(*module_name*)
Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (str) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an ImportError.

See also:

mpi_array.benchmarks.utils.try_import_for_setup().

Attributes

a_ary	First input array.
a_ary_range	A (low, high) tuple indicating the uniform random range of scalars for the *a_ary* array elements.
b_ary	Second input array.
b_ary_range	A (low, high) tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
NumpyUfuncBench.cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
random_state	A numpy.random.RandomState instance, used to generate random values in arrays.

Continued on next page

Table 1.47 – continued from previous page

repeat	
ufunc	The `numpy.ufunc` for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.a_ary

`NumpyUfuncBench.`**`a_ary`**
First input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.a_ary_range

`NumpyUfuncBench.`**`a_ary_range`**
A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.b_ary

`NumpyUfuncBench.`**`b_ary`**
Second input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.b_ary_range

`NumpyUfuncBench.`**`b_ary_range`**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.b_scalar

`NumpyUfuncBench.`**`b_scalar`**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.c_ary

`NumpyUfuncBench.`**`c_ary`**
Output array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.goal_time

`NumpyUfuncBench.`**`goal_time = 1.0`**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.module

`NumpyUfuncBench.`**`module`**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.param_names

NumpyUfuncBench.**param_names = ['shape']**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.params

NumpyUfuncBench.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.random_state

NumpyUfuncBench.**random_state**
A `numpy.random.RandomState` instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.repeat

NumpyUfuncBench.**repeat = 8**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.ufunc

NumpyUfuncBench.**ufunc**
The `numpy.ufunc` for this benchmark.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench.warmup_time

NumpyUfuncBench.**warmup_time = 1.0**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add

class mpi_array.benchmarks.bench_ufunc.**NumpyUfuncBench_add**
Bases: *mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench*

Benchmark for `numpy.add` with `numpy.ndarray` array inputs.

Methods

__init__()	
free(a)	Clean up array resources, over-ridden in sub-classes.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, `dtype` and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.

Continued on next page

Table 1.48 – continued from previous page

setup(shape[, dtype])	Import numpy module and assign to *module*.
teardown(shape)	Free arrays allocated during *setup()*.
time_array_array_op(shape[, dtype])	Timing for two input ufuncs (two array inputs).
time_array_scalar_op(shape[, dtype])	Timing for two input ufuncs (one array input, one scalar input).
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.__init__

NumpyUfuncBench_add.**__init__**()

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.free

NumpyUfuncBench_add.**free**(*a*)

Clean up array resources, over-ridden in sub-classes.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.free_mpi_array_obj

NumpyUfuncBench_add.**free_mpi_array_obj**(*a*)

Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.get_globale_shape

NumpyUfuncBench_add.**get_globale_shape**(*locale_shape*)

Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.initialise

NumpyUfuncBench_add.**initialise**(*shape*, *dtype*, *module_name*)

Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.initialise_arrays

NumpyUfuncBench_add.**initialise_arrays**(*shape*)

Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of int) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.setup

NumpyUfuncBench_add.**setup**(*shape*, *dtype='float64'*)
Import `numpy` module and assign to *`module`*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.teardown

NumpyUfuncBench_add.**teardown**(*shape*)
Free arrays allocated during *`setup()`*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.time_array_array_op

NumpyUfuncBench_add.**time_array_array_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (two array inputs).

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.time_array_scalar_op

NumpyUfuncBench_add.**time_array_scalar_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (one array input, one scalar input).

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.try_import_for_setup

NumpyUfuncBench_add.**try_import_for_setup**(*module_name*)
Attempt to import module named *`module_name`*, return the module if it exists and is importable.

Parameters **module_name** (`str`) – Attempt to import this module.

Return type *`module`*

Returns Module named *`module_name`*.

Raises **NotImplementedError** – if there is an `ImportError`.

See also:

`mpi_array.benchmarks.utils.try_import_for_setup()`.

Attributes

`a_ary`	First input array.
`a_ary_range`	A `(low, high)` tuple indicating the uniform random range of scalars for the *`a_ary`* array elements.
`b_ary`	Second input array.
`b_ary_range`	A `(low, high)` tuple inficating the uniform random range of scalars for the *`b_ary`* array elements.
`b_scalar`	Second scalar input.
`c_ary`	Output array.
NumpyUfuncBench_add.cart_comm_dims	
`goal_time`	
`module`	The *`module`* used to create array instances.

Continued on next page

Table 1.49 – continued from previous page

param_names	
params	
random_state	A `numpy.random.RandomState` instance, used to generate random values in arrays.
repeat	
ufunc	The `numpy.ufunc` for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.a_ary

NumpyUfuncBench_add.**a_ary**
First input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.a_ary_range

NumpyUfuncBench_add.**a_ary_range**
A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.b_ary

NumpyUfuncBench_add.**b_ary**
Second input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.b_ary_range

NumpyUfuncBench_add.**b_ary_range**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.b_scalar

NumpyUfuncBench_add.**b_scalar**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.c_ary

NumpyUfuncBench_add.**c_ary**
Output array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.goal_time

NumpyUfuncBench_add.**goal_time** = 1.0

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.module

NumpyUfuncBench_add.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.param_names

NumpyUfuncBench_add.**param_names = ['shape']**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.params

NumpyUfuncBench_add.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.random_state

NumpyUfuncBench_add.**random_state**
A numpy.random.RandomState instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.repeat

NumpyUfuncBench_add.**repeat = 8**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.ufunc

NumpyUfuncBench_add.**ufunc**
The numpy.ufunc for this benchmark.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_add.warmup_time

NumpyUfuncBench_add.**warmup_time = 1.0**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt

class mpi_array.benchmarks.bench_ufunc.**NumpyUfuncBench_invsqrt**
Bases: *mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench*

Benchmarks for numpy.power(ary, -0.5) ufunc for numpy.ndarray input.

Methods

__init__()	
free(a)	Clean up array resources, over-ridden in sub-classes.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.

Continued on next page

Table 1.50 – continued from previous page

get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.
setup(shape[, dtype])	Import numpy module and assign to *module*.
teardown(shape)	Free arrays allocated during *setup()*.
time_array_op(shape[, dtype])	Timing for single argument ufuncs.
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.__init__

NumpyUfuncBench_invsqrt.**__init__**()

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.free

NumpyUfuncBench_invsqrt.**free**(*a*)

Clean up array resources, over-ridden in sub-classes.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.free_mpi_array_obj

NumpyUfuncBench_invsqrt.**free_mpi_array_obj**(*a*)

Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.get_globale_shape

NumpyUfuncBench_invsqrt.**get_globale_shape**(*locale_shape*)

Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.initialise

NumpyUfuncBench_invsqrt.**initialise**(*shape*, *dtype*, *module_name*)

Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.initialise_arrays

NumpyUfuncBench_invsqrt.**initialise_arrays**(*shape*)

Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of int) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.setup

NumpyUfuncBench_invsqrt.**setup**(*shape*, *dtype='float64'*)
Import `numpy` module and assign to *module*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.teardown

NumpyUfuncBench_invsqrt.**teardown**(*shape*)
Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.time_array_op

NumpyUfuncBench_invsqrt.**time_array_op**(*shape*, *dtype='float64'*)
Timing for single argument ufuncs.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.try_import_for_setup

NumpyUfuncBench_invsqrt.**try_import_for_setup**(*module_name*)
Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (`str`) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an `ImportError`.

See also:

`mpi_array.benchmarks.utils.try_import_for_setup()`.

Attributes

a_ary	First input array.
a_ary_range	A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.
b_ary	Second input array.
b_ary_range	A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
NumpyUfuncBench_invsqrt.cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
random_state	A `numpy.random.RandomState` instance, used to generate random values in arrays.

Continued on next page

Table 1.51 – continued from previous page

repeat	
ufunc	The `numpy.ufunc` for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.a_ary

`NumpyUfuncBench_invsqrt.`**`a_ary`**
First input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.a_ary_range

`NumpyUfuncBench_invsqrt.`**`a_ary_range`**
A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.b_ary

`NumpyUfuncBench_invsqrt.`**`b_ary`**
Second input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.b_ary_range

`NumpyUfuncBench_invsqrt.`**`b_ary_range`**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.b_scalar

`NumpyUfuncBench_invsqrt.`**`b_scalar`**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.c_ary

`NumpyUfuncBench_invsqrt.`**`c_ary`**
Output array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.goal_time

`NumpyUfuncBench_invsqrt.`**`goal_time = 1.0`**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.module

`NumpyUfuncBench_invsqrt.`**`module`**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.param_names

NumpyUfuncBench_invsqrt.**param_names = ['shape']**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.params

NumpyUfuncBench_invsqrt.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.random_state

NumpyUfuncBench_invsqrt.**random_state**
A numpy.random.RandomState instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.repeat

NumpyUfuncBench_invsqrt.**repeat = 8**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.ufunc

NumpyUfuncBench_invsqrt.**ufunc**
The numpy.ufunc for this benchmark.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_invsqrt.warmup_time

NumpyUfuncBench_invsqrt.**warmup_time = 1.0**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log

class mpi_array.benchmarks.bench_ufunc.**NumpyUfuncBench_log**
Bases: *mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench*

Benchmark for numpy.log with numpy.ndarray array inputs.

Methods

__init__()	
free(a)	Clean up array resources, over-ridden in sub-classes.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.

Continued on next page

Table 1.52 – continued from previous page

setup(shape[, dtype])	Import numpy module and assign to *module*.
teardown(shape)	Free arrays allocated during *setup()*.
time_array_op(shape[, dtype])	Timing for single input ufuncs (single array input).
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.__init__

NumpyUfuncBench_log.**__init__**()

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.free

NumpyUfuncBench_log.**free**(*a*)

Clean up array resources, over-ridden in sub-classes.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.free_mpi_array_obj

NumpyUfuncBench_log.**free_mpi_array_obj**(*a*)

Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.get_globale_shape

NumpyUfuncBench_log.**get_globale_shape**(*locale_shape*)

Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.initialise

NumpyUfuncBench_log.**initialise**(*shape*, *dtype*, *module_name*)

Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.initialise_arrays

NumpyUfuncBench_log.**initialise_arrays**(*shape*)

Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of int) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.setup

NumpyUfuncBench_log.**setup**(*shape*, *dtype='float64'*)

Import numpy module and assign to *module*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.teardown

NumpyUfuncBench_log.**teardown**(*shape*)
Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.time_array_op

NumpyUfuncBench_log.**time_array_op**(*shape*, *dtype='float64'*)
Timing for single input ufuncs (single array input).

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.try_import_for_setup

NumpyUfuncBench_log.**try_import_for_setup**(*module_name*)
Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (str) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an ImportError.

See also:

mpi_array.benchmarks.utils.try_import_for_setup().

Attributes

a_ary	First input array.
a_ary_range	A (low, high) tuple indicating the uniform random range of scalars for the *a_ary* array elements.
b_ary	Second input array.
b_ary_range	A (low, high) tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
NumpyUfuncBench_log.cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
random_state	A numpy.random.RandomState instance, used to generate random values in arrays.
repeat	
ufunc	The numpy.ufunc for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.a_ary

NumpyUfuncBench_log.**a_ary**
First input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.a_ary_range

NumpyUfuncBench_log.**a_ary_range**
A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.b_ary

NumpyUfuncBench_log.**b_ary**
Second input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.b_ary_range

NumpyUfuncBench_log.**b_ary_range**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.b_scalar

NumpyUfuncBench_log.**b_scalar**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.c_ary

NumpyUfuncBench_log.**c_ary**
Output array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.goal_time

NumpyUfuncBench_log.**goal_time = 1.0**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.module

NumpyUfuncBench_log.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.param_names

NumpyUfuncBench_log.**param_names = ['shape']**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.params

NumpyUfuncBench_log.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.random_state

NumpyUfuncBench_log.**random_state**
A numpy.random.RandomState instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.repeat

NumpyUfuncBench_log.**repeat = 8**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.ufunc

NumpyUfuncBench_log.**ufunc**
The numpy.ufunc for this benchmark.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log.warmup_time

NumpyUfuncBench_log.**warmup_time = 1.0**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10

class mpi_array.benchmarks.bench_ufunc.**NumpyUfuncBench_log10**
Bases: *mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench*

Benchmark for numpy.log10 with numpy.ndarray array inputs.

Methods

__init__()	
free(a)	Clean up array resources, over-ridden in sub-classes.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.
setup(shape[, dtype])	Import numpy module and assign to *module*.
teardown(shape)	Free arrays allocated during *setup()*.
time_array_op(shape[, dtype])	Timing for single input ufuncs (single array input).
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.__init__

NumpyUfuncBench_log10.**__init__**()

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.free

NumpyUfuncBench_log10.**free**(*a*)
Clean up array resources, over-ridden in sub-classes.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.free_mpi_array_obj

NumpyUfuncBench_log10.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.get_globale_shape

NumpyUfuncBench_log10.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of `int`) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.initialise

NumpyUfuncBench_log10.**initialise**(*shape*, *dtype*, *module_name*)
Sets the *module*, `dtype` and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.initialise_arrays

NumpyUfuncBench_log10.**initialise_arrays**(*shape*)
Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of `int`) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.setup

NumpyUfuncBench_log10.**setup**(*shape*, *dtype='float64'*)
Import `numpy` module and assign to *module*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.teardown

NumpyUfuncBench_log10.**teardown**(*shape*)
Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.time_array_op

NumpyUfuncBench_log10.**time_array_op**(*shape*, *dtype='float64'*)
Timing for single input ufuncs (single array input).

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.try_import_for_setup

NumpyUfuncBench_log10.**try_import_for_setup**(*module_name*)

Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (str) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an ImportError.

See also:

mpi_array.benchmarks.utils.try_import_for_setup().

Attributes

a_ary	First input array.
a_ary_range	A (low, high) tuple indicating the uniform random range of scalars for the *a_ary* array elements.
b_ary	Second input array.
b_ary_range	A (low, high) tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
NumpyUfuncBench_log10.cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
random_state	A numpy.random.RandomState instance, used to generate random values in arrays.
repeat	
ufunc	The numpy.ufunc for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.a_ary

NumpyUfuncBench_log10.**a_ary**

First input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.a_ary_range

NumpyUfuncBench_log10.**a_ary_range**

A (low, high) tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.b_ary

NumpyUfuncBench_log10.**b_ary**

Second input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.b_ary_range

NumpyUfuncBench_log10.**b_ary_range**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.b_scalar

NumpyUfuncBench_log10.**b_scalar**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.c_ary

NumpyUfuncBench_log10.**c_ary**
Output array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.goal_time

NumpyUfuncBench_log10.**goal_time = 1.0**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.module

NumpyUfuncBench_log10.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.param_names

NumpyUfuncBench_log10.**param_names = ['shape']**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.params

NumpyUfuncBench_log10.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.random_state

NumpyUfuncBench_log10.**random_state**
A `numpy.random.RandomState` instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.repeat

NumpyUfuncBench_log10.**repeat = 8**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.ufunc

NumpyUfuncBench_log10.**ufunc**
The `numpy.ufunc` for this benchmark.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_log10.warmup_time

NumpyUfuncBench_log10.**warmup_time** = 1.0

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply

class mpi_array.benchmarks.bench_ufunc.**NumpyUfuncBench_multiply**
Bases: *mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench*

Benchmark for numpy.multiply with numpy.ndarray array inputs.

Methods

__init__()	
free(a)	Clean up array resources, over-ridden in sub-classes.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.
setup(shape[, dtype])	Import numpy module and assign to *module*.
teardown(shape)	Free arrays allocated during *setup()*.
time_array_array_op(shape[, dtype])	Timing for two input ufuncs (two array inputs).
time_array_scalar_op(shape[, dtype])	Timing for two input ufuncs (one array input, one scalar input).
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.__init__

NumpyUfuncBench_multiply.**__init__**()

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.free

NumpyUfuncBench_multiply.**free**(*a*)
Clean up array resources, over-ridden in sub-classes.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.free_mpi_array_obj

NumpyUfuncBench_multiply.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.get_globale_shape

NumpyUfuncBench_multiply.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.initialise

NumpyUfuncBench_multiply.**initialise**(*shape*, *dtype*, *module_name*)
Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.initialise_arrays

NumpyUfuncBench_multiply.**initialise_arrays**(*shape*)
Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of int) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.setup

NumpyUfuncBench_multiply.**setup**(*shape*, *dtype='float64'*)
Import numpy module and assign to *module*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.teardown

NumpyUfuncBench_multiply.**teardown**(*shape*)
Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.time_array_array_op

NumpyUfuncBench_multiply.**time_array_array_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (two array inputs).

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.time_array_scalar_op

NumpyUfuncBench_multiply.**time_array_scalar_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (one array input, one scalar input).

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.try_import_for_setup

NumpyUfuncBench_multiply.**try_import_for_setup**(*module_name*)
Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (str) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an ImportError.

See also:

mpi_array.benchmarks.utils.try_import_for_setup().

Attributes

a_ary	First input array.
a_ary_range	A (low, high) tuple indicating the uniform random range of scalars for the *a_ary* array elements.
b_ary	Second input array.
b_ary_range	A (low, high) tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
NumpyUfuncBench_multiply.cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
random_state	A numpy.random.RandomState instance, used to generate random values in arrays.
repeat	
ufunc	The numpy.ufunc for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.a_ary

NumpyUfuncBench_multiply.**a_ary**
First input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.a_ary_range

NumpyUfuncBench_multiply.**a_ary_range**
A (low, high) tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.b_ary

NumpyUfuncBench_multiply.**b_ary**
Second input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.b_ary_range

NumpyUfuncBench_multiply.**b_ary_range**
A (low, high) tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.b_scalar

NumpyUfuncBench_multiply.**b_scalar**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.c_ary

NumpyUfuncBench_multiply.**c_ary**
Output array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.goal_time

NumpyUfuncBench_multiply.**goal_time = 1.0**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.module

NumpyUfuncBench_multiply.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.param_names

NumpyUfuncBench_multiply.**param_names = ['shape']**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.params

NumpyUfuncBench_multiply.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.random_state

NumpyUfuncBench_multiply.**random_state**
A numpy.random.RandomState instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.repeat

NumpyUfuncBench_multiply.**repeat = 8**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.ufunc

NumpyUfuncBench_multiply.**ufunc**
The numpy.ufunc for this benchmark.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_multiply.warmup_time

NumpyUfuncBench_multiply.**warmup_time = 1.0**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract

class mpi_array.benchmarks.bench_ufunc.**NumpyUfuncBench_subtract**
Bases: *mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench*

Benchmark for numpy.subtract with numpy.ndarray array inputs.

Methods

__init__()	
free(a)	Clean up array resources, over-ridden in sub-classes.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.
setup(shape[, dtype])	Import numpy module and assign to *module*.
teardown(shape)	Free arrays allocated during *setup()*.
time_array_array_op(shape[, dtype])	Timing for two input ufuncs (two array inputs).
time_array_scalar_op(shape[, dtype])	Timing for two input ufuncs (one array input, one scalar input).
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.__init__

NumpyUfuncBench_subtract.**__init__**()

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.free

NumpyUfuncBench_subtract.**free**(*a*)
Clean up array resources, over-ridden in sub-classes.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.free_mpi_array_obj

NumpyUfuncBench_subtract.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.get_globale_shape

NumpyUfuncBench_subtract.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.initialise

NumpyUfuncBench_subtract.**initialise**(*shape*, *dtype*, *module_name*)
Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.initialise_arrays

NumpyUfuncBench_subtract.**initialise_arrays**(*shape*)
Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of int) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.setup

NumpyUfuncBench_subtract.**setup**(*shape*, *dtype='float64'*)
Import numpy module and assign to *module*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.teardown

NumpyUfuncBench_subtract.**teardown**(*shape*)
Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.time_array_array_op

NumpyUfuncBench_subtract.**time_array_array_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (two array inputs).

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.time_array_scalar_op

NumpyUfuncBench_subtract.**time_array_scalar_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (one array input, one scalar input).

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.try_import_for_setup

NumpyUfuncBench_subtract.**try_import_for_setup**(*module_name*)
Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (str) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an ImportError.

See also:

`mpi_array.benchmarks.utils.try_import_for_setup()`.

Attributes

a_ary	First input array.
a_ary_range	A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.
b_ary	Second input array.
b_ary_range	A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
`NumpyUfuncBench_subtract.cart_comm_dims`	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
random_state	A `numpy.random.RandomState` instance, used to generate random values in arrays.
repeat	
ufunc	The `numpy.ufunc` for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.a_ary

`NumpyUfuncBench_subtract.`**`a_ary`**
First input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.a_ary_range

`NumpyUfuncBench_subtract.`**`a_ary_range`**
A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.b_ary

`NumpyUfuncBench_subtract.`**`b_ary`**
Second input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.b_ary_range

`NumpyUfuncBench_subtract.`**`b_ary_range`**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.b_scalar

NumpyUfuncBench_subtract.**b_scalar**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.c_ary

NumpyUfuncBench_subtract.**c_ary**
Output array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.goal_time

NumpyUfuncBench_subtract.**goal_time = 1.0**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.module

NumpyUfuncBench_subtract.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.param_names

NumpyUfuncBench_subtract.**param_names = ['shape']**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.params

NumpyUfuncBench_subtract.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.random_state

NumpyUfuncBench_subtract.**random_state**
A numpy.random.RandomState instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.repeat

NumpyUfuncBench_subtract.**repeat = 8**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.ufunc

NumpyUfuncBench_subtract.**ufunc**
The numpy.ufunc for this benchmark.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_subtract.warmup_time

NumpyUfuncBench_subtract.**warmup_time = 1.0**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide

class mpi_array.benchmarks.bench_ufunc.**NumpyUfuncBench_true_divide**
Bases: *mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench*

Benchmark for numpy.true_divide with numpy.ndarray array inputs.

Methods

__init__()	
free(a)	Clean up array resources, over-ridden in sub-classes.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.
setup(shape[, dtype])	Import numpy module and assign to *module*.
teardown(shape)	Free arrays allocated during *setup()*.
time_array_array_op(shape[, dtype])	Timing for two input ufuncs (two array inputs).
time_array_scalar_op(shape[, dtype])	Timing for two input ufuncs (one array input, one scalar input).
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.__init__

NumpyUfuncBench_true_divide.**__init__**()

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.free

NumpyUfuncBench_true_divide.**free**(*a*)
Clean up array resources, over-ridden in sub-classes.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.free_mpi_array_obj

NumpyUfuncBench_true_divide.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.get_globale_shape

NumpyUfuncBench_true_divide.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.initialise

NumpyUfuncBench_true_divide.**initialise**(*shape*, *dtype*, *module_name*)
Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.initialise_arrays

NumpyUfuncBench_true_divide.**initialise_arrays**(*shape*)
Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of int) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.setup

NumpyUfuncBench_true_divide.**setup**(*shape*, *dtype='float64'*)
Import numpy module and assign to *module*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.teardown

NumpyUfuncBench_true_divide.**teardown**(*shape*)
Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.time_array_array_op

NumpyUfuncBench_true_divide.**time_array_array_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (two array inputs).

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.time_array_scalar_op

NumpyUfuncBench_true_divide.**time_array_scalar_op**(*shape*, *dtype='float64'*)
Timing for two input ufuncs (one array input, one scalar input).

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.try_import_for_setup

NumpyUfuncBench_true_divide.**try_import_for_setup**(*module_name*)
Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (str) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an ImportError.

See also:

`mpi_array.benchmarks.utils.try_import_for_setup()`.

Attributes

a_ary	First input array.
a_ary_range	A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.
b_ary	Second input array.
b_ary_range	A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
NumpyUfuncBench_true_divide.cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	
params	
random_state	A `numpy.random.RandomState` instance, used to generate random values in arrays.
repeat	
ufunc	The `numpy.ufunc` for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.a_ary

NumpyUfuncBench_true_divide.**a_ary**
First input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.a_ary_range

NumpyUfuncBench_true_divide.**a_ary_range**
A `(low, high)` tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.b_ary

NumpyUfuncBench_true_divide.**b_ary**
Second input array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.b_ary_range

NumpyUfuncBench_true_divide.**b_ary_range**
A `(low, high)` tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.b_scalar

NumpyUfuncBench_true_divide.**b_scalar**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.c_ary

NumpyUfuncBench_true_divide.**c_ary**
Output array.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.goal_time

NumpyUfuncBench_true_divide.**goal_time = 1.0**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.module

NumpyUfuncBench_true_divide.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.param_names

NumpyUfuncBench_true_divide.**param_names = ['shape']**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.params

NumpyUfuncBench_true_divide.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.random_state

NumpyUfuncBench_true_divide.**random_state**
A numpy.random.RandomState instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.repeat

NumpyUfuncBench_true_divide.**repeat = 8**

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.ufunc

NumpyUfuncBench_true_divide.**ufunc**
The numpy.ufunc for this benchmark.

mpi_array.benchmarks.bench_ufunc.NumpyUfuncBench_true_divide.warmup_time

NumpyUfuncBench_true_divide.**warmup_time = 1.0**

mpi_array.benchmarks.bench_ufunc.UfuncBench

class mpi_array.benchmarks.bench_ufunc.**UfuncBench**(*ufunc_name=None*)

Bases: *mpi_array.benchmarks.core.Bench*

Base class for array ufunc benchmarks. Run benchmarks for calls of the form:

```
numpy.exp(self.a_ary, out=self.c_ary)
numpy.add(self.a_ary, self.b_ary, out=self.c_ary)
```

where the *a_ary*, *b_ary* and *c_ary* arrays are initialised (with uniform random scalars) during *setup()*.

Methods

__init__([ufunc_name])	Initialise.
free(a)	Clean up array resources, over-ridden in sub-classes.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
initialise(shape, dtype, module_name)	Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.
initialise_arrays(shape)	Initialise arrays/scalars passed to ufunc instances.
setup(shape[, dtype])	Should be over-ridden in sub-classes.
teardown(shape)	Free arrays allocated during *setup()*.
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.bench_ufunc.UfuncBench.__init__

UfuncBench.**__init__**(*ufunc_name=None*)

Initialise.

Parameters **ufunc_name** (str) – Name of the ufunc, should be the name of an attribute of *module*.

mpi_array.benchmarks.bench_ufunc.UfuncBench.free

UfuncBench.**free**(*a*)

Clean up array resources, over-ridden in sub-classes.

mpi_array.benchmarks.bench_ufunc.UfuncBench.free_mpi_array_obj

UfuncBench.**free_mpi_array_obj**(*a*)

Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.bench_ufunc.UfuncBench.get_globale_shape

UfuncBench.**get_globale_shape**(*locale_shape*)

Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.bench_ufunc.UfuncBench.initialise

UfuncBench.**initialise**(*shape*, *dtype*, *module_name*)

Sets the *module*, dtype and *ufunc* attributes and initialises the *a_ary*, *b_ary*, *b_scalar* and *c_ary* arrays.

mpi_array.benchmarks.bench_ufunc.UfuncBench.initialise_arrays

UfuncBench.**initialise_arrays**(*shape*)

Initialise arrays/scalars passed to ufunc instances.

Parameters **shape** (sequence of int) – Shape of the arrays to be passed to ufuncs.

mpi_array.benchmarks.bench_ufunc.UfuncBench.setup

UfuncBench.**setup**(*shape*, *dtype='float64'*)

Should be over-ridden in sub-classes.

mpi_array.benchmarks.bench_ufunc.UfuncBench.teardown

UfuncBench.**teardown**(*shape*)

Free arrays allocated during *setup()*.

mpi_array.benchmarks.bench_ufunc.UfuncBench.try_import_for_setup

UfuncBench.**try_import_for_setup**(*module_name*)

Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (str) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an ImportError.

See also:

mpi_array.benchmarks.utils.try_import_for_setup().

Attributes

a_ary	First input array.
a_ary_range	A (low, high) tuple indicating the uniform random range of scalars for the *a_ary* array elements.
b_ary	Second input array.
b_ary_range	A (low, high) tuple inficating the uniform random range of scalars for the *b_ary* array elements.
b_scalar	Second scalar input.
c_ary	Output array.
UfuncBench.cart_comm_dims	
goal_time	
module	The *module* used to create array instances.
param_names	The name of the array-shape parameters.
params	The set of array-shape parameters.
random_state	A numpy.random.RandomState instance, used to generate random values in arrays.
repeat	Number of repetitions to run each benchmark
ufunc	The numpy.ufunc for this benchmark.
warmup_time	

mpi_array.benchmarks.bench_ufunc.UfuncBench.a_ary

UfuncBench.**a_ary**
First input array.

mpi_array.benchmarks.bench_ufunc.UfuncBench.a_ary_range

UfuncBench.**a_ary_range**
A (low, high) tuple indicating the uniform random range of scalars for the *a_ary* array elements.

mpi_array.benchmarks.bench_ufunc.UfuncBench.b_ary

UfuncBench.**b_ary**
Second input array.

mpi_array.benchmarks.bench_ufunc.UfuncBench.b_ary_range

UfuncBench.**b_ary_range**
A (low, high) tuple inficating the uniform random range of scalars for the *b_ary* array elements.

mpi_array.benchmarks.bench_ufunc.UfuncBench.b_scalar

UfuncBench.**b_scalar**
Second scalar input.

mpi_array.benchmarks.bench_ufunc.UfuncBench.c_ary

UfuncBench.**c_ary**
Output array.

mpi_array.benchmarks.bench_ufunc.UfuncBench.goal_time

UfuncBench.**goal_time = 1.0**

mpi_array.benchmarks.bench_ufunc.UfuncBench.module

UfuncBench.**module**
The *module* used to create array instances.

mpi_array.benchmarks.bench_ufunc.UfuncBench.param_names

UfuncBench.**param_names = ['shape']**
The name of the array-shape parameters.

mpi_array.benchmarks.bench_ufunc.UfuncBench.params

UfuncBench.**params = [[(131072,), (1000, 100, 100), (128, 1024, 1024)]]**
The set of array-shape parameters.

mpi_array.benchmarks.bench_ufunc.UfuncBench.random_state

UfuncBench.**random_state**
A `numpy.random.RandomState` instance, used to generate random values in arrays.

mpi_array.benchmarks.bench_ufunc.UfuncBench.repeat

UfuncBench.**repeat = 8**
Number of repetitions to run each benchmark

mpi_array.benchmarks.bench_ufunc.UfuncBench.ufunc

UfuncBench.**ufunc**
The `numpy.ufunc` for this benchmark.

mpi_array.benchmarks.bench_ufunc.UfuncBench.warmup_time

UfuncBench.**warmup_time = 1.0**

mpi_array.benchmarks.core

Core utilities for benchmark implementations.

Classes

Bench()	Base class for benchmarks.

mpi_array.benchmarks.core.Bench

class mpi_array.benchmarks.core.**Bench**
Bases: object

Base class for benchmarks.

Methods

__init__()	Initialise, set *module* to None.
free_mpi_array_obj(a)	Free MPI communicators and MPI windows for the given object.
get_globale_shape(locale_shape)	Returns a *globale* array shape for the given shape of the *locale* array.
setup(shape)	Should be over-ridden in sub-classes.
try_import_for_setup(module_name)	Attempt to import module named *module_name*, return the module if it exists and is importable.

mpi_array.benchmarks.core.Bench.__init__

Bench.**__init__**()
Initialise, set *module* to None.

mpi_array.benchmarks.core.Bench.free_mpi_array_obj

Bench.**free_mpi_array_obj**(*a*)
Free MPI communicators and MPI windows for the given object.

Parameters **a** (*free-able*) – A *mpi_array.globale.gndarray* instance or a *mpi_array.comms.LocaleComms* instance.

mpi_array.benchmarks.core.Bench.get_globale_shape

Bench.**get_globale_shape**(*locale_shape*)
Returns a *globale* array shape for the given shape of the *locale* array.

Parameters **locale_shape** (sequence of int) – The shape of the array to be allocated on each *locale*.

mpi_array.benchmarks.core.Bench.setup

Bench.**setup**(*shape*)
Should be over-ridden in sub-classes.

mpi_array.benchmarks.core.Bench.try_import_for_setup

Bench.**try_import_for_setup**(*module_name*)
Attempt to import module named *module_name*, return the module if it exists and is importable.

Parameters **module_name** (str) – Attempt to import this module.

Return type *module*

Returns Module named *module_name*.

Raises **NotImplementedError** – if there is an ImportError.

See also:

mpi_array.benchmarks.utils.try_import_for_setup().

Attributes

Bench.cart_comm_dims	
module	The *module* used to create array instances.

mpi_array.benchmarks.core.Bench.module

Bench.**module**
The *module* used to create array instances.

mpi_array.benchmarks.utils

Miscellaneous benchmark utilities.

Modules

misc	Benchmark utilities.
wlm	Generate *workload-mangager* (e.g.
wlm_test	Module defining *mpi_array.benchmarks.utils.wlm* unit-tests.

mpi_array.benchmarks.utils.misc

Benchmark utilities.

Parts of this source borrows from the airspeed velocity (asv) file benchmark.py.

See the LICENSE.

Functions

get_process_time_timer()	The best timer we can use is `time.process_time()`, but it is not available in the Python stdlib until Python 3.3.
try_import_for_setup(module_name)	Returns the imported module named *module_name*.
update_sys_path(root)	Inserts a *SpecificImporter* instance into the `sys.meta_path`.

mpi_array.benchmarks.utils.misc.get_process_time_timer

mpi_array.benchmarks.utils.misc.**get_process_time_timer**()

The best timer we can use is `time.process_time()`, but it is not available in the Python stdlib until Python 3.3. This is a `ctypes` backport for Pythons that don't have it.

Return type `function`

Returns The `time.process_time()` function, if available, otherwise, if possible, an equivalent created using `ctypes`, otherwise `timeit.default_timer()`.

mpi_array.benchmarks.utils.misc.try_import_for_setup

mpi_array.benchmarks.utils.misc.**try_import_for_setup**(*module_name*)

Returns the imported module named *module_name*. Attempts to import the module named *module_name* and translates any raised `ImportError` into a `NotImplementedError`. This is useful in benchmark `setup`, so that a failed import results in the benchmark getting *skipped*.

Parameters **module_name** (`str`) – Attempt to import this module.

Return type `module`

Returns The imported module named *module_name*.

Raises **NotImplementedError** – if there is an `ImportError`.

mpi_array.benchmarks.utils.misc.update_sys_path

mpi_array.benchmarks.utils.misc.**update_sys_path**(*root*)

Inserts a *SpecificImporter* instance into the `sys.meta_path`.

Parameters **root** (`str`) – The :obj'SpecificImported' path inserted at the head of `sys.meta_path`.

Classes

SpecificImporter(name, root)	Module importer that only allows loading a given module from the given path.

mpi_array.benchmarks.utils.misc.SpecificImporter

class mpi_array.benchmarks.utils.misc.**SpecificImporter**(*name*, *root*)

Bases: `object`

Module importer that only allows loading a given module from the given path.

Using this enables importing the asv benchmark suite without adding its parent directory to sys.path. The parent directory can in principle contain anything, including some version of the project module (common situation if asv.conf.json is on project repository top level).

Methods

__init__(name, root)	
find_module(fullname[, path])	
load_module(fullname)	

mpi_array.benchmarks.utils.misc.SpecificImporter.__init__

SpecificImporter.**__init__**(*name*, *root*)

mpi_array.benchmarks.utils.misc.SpecificImporter.find_module

SpecificImporter.**find_module**(*fullname*, *path=None*)

mpi_array.benchmarks.utils.misc.SpecificImporter.load_module

SpecificImporter.**load_module**(*fullname*)

mpi_array.benchmarks.utils.wlm

Generate *workload-mangager* (e.g. PBS Pro, Slurm, Torque) benchmark batch scripts.

Functions

adjust_list_to_length(obj, desired_length)	Returns a `list` of length *desired_length*, by extending or truncating the `obj` `list`.

mpi_array.benchmarks.utils.wlm.adjust_list_to_length

mpi_array.benchmarks.utils.wlm.**adjust_list_to_length**(*obj*, *desired_length*)

Returns a `list` of length *desired_length*, by extending or truncating the `obj` `list`. The sequence is extended by repeating the last element of *obj*.

Parameters

- **obj** (sequence or `str`) – The sequence to be extended/truncated. If *obj* is a string, it is converted to a list using *obj = [obj,]*.
- **desired_length** (`int`) – The length of the returned `list`.

Return type `list`

Returns A `list` of length *desired_length* which contains the elements of the :samp:`obj` sequence.

Classes

WlmScriptGenerator(subst_dict, script_template)	Generates multiple script files from a template.

mpi_array.benchmarks.utils.wlm.WlmScriptGenerator

class mpi_array.benchmarks.utils.wlm.**WlmScriptGenerator**(*subst_dict*, *script_template*, *script_base_name='mpia_%(max_num_cpus)04d_ben*, *script_ext='.sh'*, *script_dir='.'*)

Bases: `object`

Generates multiple script files from a template.

Methods

__init__(subst_dict, script_template[, ...])	Initialise.
generate_script_file_name(subst_dict)	Returns the name of the script file.
generate_script_files()	Generates the script files.
generate_script_string(subst_dict)	Returns the text which is to be written to a script file.
update_dict_max_num_cpus(subst_dict)	Add an entry for the "max_num_cpus" key in *subst_dict* if it does not already exist.
update_dict_num_cpus_string(subst_dict)	Replaces "num_cpus" list of values in *subst_dict* with a space separated string.

mpi_array.benchmarks.utils.wlm.WlmScriptGenerator.__init__

WlmScriptGenerator.**__init__**(*subst_dict*, *script_template*, *script_base_name='mpia_%(max_num_cpus)04d_bench'*, *script_ext='.sh'*, *script_dir='.'*)

Initialise.

mpi_array.benchmarks.utils.wlm.WlmScriptGenerator.generate_script_file_name

WlmScriptGenerator.**generate_script_file_name**(*subst_dict*)

Returns the name of the script file.

Parameters **subst_dict** (`dict`) – Substitution dictionary, substitutions made in *script_base_name*.

Return type `str`

Returns Name of script file.

mpi_array.benchmarks.utils.wlm.WlmScriptGenerator.generate_script_files

WlmScriptGenerator.**generate_script_files**()
Generates the script files.

mpi_array.benchmarks.utils.wlm.WlmScriptGenerator.generate_script_string

WlmScriptGenerator.**generate_script_string**(*subst_dict*)
Returns the text which is to be written to a script file.

Parameters **subst_dict** (dict) – Substitution dictionary, substitutions made in *script_template*.

Return type str

Returns Text to be written to script file.

mpi_array.benchmarks.utils.wlm.WlmScriptGenerator.update_dict_max_num_cpus

WlmScriptGenerator.**update_dict_max_num_cpus**(*subst_dict*)
Add an entry for the "max_num_cpus" key in *subst_dict* if it does not already exist.

Parameters **subst_dict** (dict) – Substitution dictionary.

mpi_array.benchmarks.utils.wlm.WlmScriptGenerator.update_dict_num_cpus_string

WlmScriptGenerator.**update_dict_num_cpus_string**(*subst_dict*)
Replaces "num_cpus" list of values in *subst_dict* with a space separated string.

Parameters **subst_dict** (dict) – Substitution dictionary.

Attributes

script_base_name	A str with the *base name* part of the script file name.
script_dir	A str indicating the directory where script files are written.
script_ext	A str with the *extension* part of the script file name.
script_template	A str which gets written to file after substitutions from *subst_dict*.
subst_dict	A dict with the string substitutions for the *script_template* template.

mpi_array.benchmarks.utils.wlm.WlmScriptGenerator.script_base_name

WlmScriptGenerator.**script_base_name**
A str with the *base name* part of the script file name.

mpi_array.benchmarks.utils.wlm.WlmScriptGenerator.script_dir

WlmScriptGenerator.**script_dir**
A str indicating the directory where script files are written.

mpi_array.benchmarks.utils.wlm.WlmScriptGenerator.script_ext

WlmScriptGenerator.**script_ext**
A str with the *extension* part of the script file name.

mpi_array.benchmarks.utils.wlm.WlmScriptGenerator.script_template

WlmScriptGenerator.**script_template**
A str which gets written to file after substitutions from *subst_dict*.

mpi_array.benchmarks.utils.wlm.WlmScriptGenerator.subst_dict

WlmScriptGenerator.**subst_dict**
A dict with the string substitutions for the *script_template* template.

mpi_array.benchmarks.utils.wlm_test

Module defining *mpi_array.benchmarks.utils.wlm* unit-tests. Execute as:

```
python -m mpi_array.benchmarks.utils.wlm_test
```

Classes

WlmScriptGeneratorTest([methodName])	unittest.TestCase for *mpi_array.benchmarks.utils.wlm.WlmScriptGenerator*.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest

class mpi_array.benchmarks.utils.wlm_test.**WlmScriptGeneratorTest**(*methodName='runTest'*)
Bases: *mpi_array.unittest.TestCase*

unittest.TestCase for *mpi_array.benchmarks.utils.wlm.WlmScriptGenerator*.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.

Continued on next page

Table 1.76 – continued from previous page

addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertAlmostEqual(first, second[, places, ...])	Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.
assertAlmostEquals(*args, **kwargs)	
assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
assertCountEqual(first, second[, msg])	An unordered sequence comparison asserting that the same elements, regardless of order.
assertDictContainsSubset(subset, dictionary)	Checks whether dictionary is a superset of subset.
assertDictEqual(d1, d2[, msg])	
assertEqual(first, second[, msg])	Fail if the two objects are unequal as determined by the '==' operator.
assertEquals(*args, **kwargs)	
assertFalse(expr[, msg])	Check that the expression is false.
assertGreater(a, b[, msg])	Just like self.assertTrue(a > b), but with a nicer default message.
assertGreaterEqual(a, b[, msg])	Just like self.assertTrue(a >= b), but with a nicer default message.
assertIn(member, container[, msg])	Just like self.assertTrue(a in b), but with a nicer default message.
assertIs(expr1, expr2[, msg])	Just like self.assertTrue(a is b), but with a nicer default message.
assertIsInstance(obj, cls[, msg])	Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.
assertIsNone(obj[, msg])	Same as self.assertTrue(obj is None), with a nicer default message.
assertIsNot(expr1, expr2[, msg])	Just like self.assertTrue(a is not b), but with a nicer default message.
assertIsNotNone(obj[, msg])	Included for symmetry with assertIsNone.
assertItemsEqual(*args, **kwargs)	See unittest.TestCase.assertItemsEqual.
assertLess(a, b[, msg])	Just like self.assertTrue(a < b), but with a nicer default message.
assertLessEqual(a, b[, msg])	Just like self.assertTrue(a <= b), but with a nicer default message.
assertListEqual(*args, **kwargs)	See unittest.TestCase.assertListEqual.
assertLogs([logger, level])	Fail unless a log message of level *level* or higher is emitted on *logger_name* or its children.
assertMultiLineEqual(first, second[, msg])	Assert that two multi-line strings are equal.
assertNotAlmostEqual(first, second[, ...])	Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.
assertNotAlmostEquals(*args, **kwargs)	

Continued on next page

Table 1.76 – continued from previous page

assertNotEqual(first, second[, msg])	Fail if the two objects are equal as determined by the '!=' operator.
assertNotEquals(*args, **kwargs)	
assertNotIn(member, container[, msg])	Just like self.assertTrue(a not in b), but with a nicer default message.
assertNotIsInstance(obj, cls[, msg])	Included for symmetry with assertIsInstance.
assertNotRegex(text, unexpected_regex[, msg])	Fail the test if the text matches the regular expression.
assertNotRegexpMatches(*args, **kwargs)	
assertRaises(expected_exception, *args, **kwargs)	Fail unless an exception of class expected_exception is raised by the callable when invoked with specified positional and keyword arguments.
assertRaisesRegex(*args, **kwargs)	See unittest.TestCase.assertRaisesRegex.
assertRaisesRegexp(*args, **kwargs)	See unittest.TestCase.assertRaisesRegexp.
assertRegex(text, expected_regex[, msg])	Fail the test unless the text matches the regular expression.
assertRegexpMatches(*args, **kwargs)	
assertSequenceEqual(*args, **kwargs)	See unittest.TestCase.assertSequenceEqual.
assertSetEqual(*args, **kwargs)	See unittest.TestCase.assertSetEqual.
assertTrue(expr[, msg])	Check that the expression is true.
assertTupleEqual(*args, **kwargs)	See unittest.TestCase.assertTupleEqual.
assertWarns(expected_warning, *args, **kwargs)	Fail unless a warning of class warnClass is triggered by the callable when invoked with specified positional and keyword arguments.
assertWarnsRegex(*args, **kwargs)	See unittest.TestCase.assertWarnsRegex.
assert_(*args, **kwargs)	
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
fail([msg])	Fail immediately, with the given message.
failIf(*args, **kwargs)	
failIfAlmostEqual(*args, **kwargs)	
failIfEqual(*args, **kwargs)	
failUnless(*args, **kwargs)	
failUnlessAlmostEqual(*args, **kwargs)	
failUnlessEqual(*args, **kwargs)	
failUnlessRaises(*args, **kwargs)	
id()	
run([result])	
setUp()	
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.

Continued on next page

Table 1.76 – continued from previous page

subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_adjust_list_to_length()	Test *mpi_array.benchmarks.utils.wlm.adjust_list_to_length()*.
test_generate_scripts()	Test *mpi_array.benchmarks.utils.wlm.WlmScriptGenerator.generate_script_files()*.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.__init__

WlmScriptGeneratorTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.addCleanup

WlmScriptGeneratorTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.addTypeEqualityFunc

WlmScriptGeneratorTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertAlmostEqual

WlmScriptGeneratorTest.**assertAlmostEqual**(*first*, *second*, *places=None*, *msg=None*, *delta=None*)

Fail if the two objects are unequal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is more than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most signficant digit).

If the two objects compare equal then they will automatically compare almost equal.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertAlmostEquals

WlmScriptGeneratorTest.**assertAlmostEquals**(**args*, ***kwargs*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertArraySplitEqual

WlmScriptGeneratorTest.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertCountEqual

WlmScriptGeneratorTest.**assertCountEqual**(*first*, *second*, *msg=None*)

An unordered sequence comparison asserting that the same elements, regardless of order. If the same element occurs more than once, it verifies that the elements occur the same number of times.

self.assertEqual(Counter(list(first)), Counter(list(second)))

Example:

- [0, 1, 1] and [1, 0, 1] compare equal.
- [0, 0, 1] and [0, 1] compare unequal.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertDictContainsSubset

WlmScriptGeneratorTest.**assertDictContainsSubset**(*subset*, *dictionary*, *msg=None*)

Checks whether dictionary is a superset of subset.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertDictEqual

WlmScriptGeneratorTest.**assertDictEqual**(*d1*, *d2*, *msg=None*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertEqual

WlmScriptGeneratorTest.**assertEqual**(*first*, *second*, *msg=None*)

Fail if the two objects are unequal as determined by the '==' operator.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertEquals

WlmScriptGeneratorTest.**assertEquals**(**args*, ***kwargs*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertFalse

WlmScriptGeneratorTest.**assertFalse**(*expr*, *msg=None*)
Check that the expression is false.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertGreater

WlmScriptGeneratorTest.**assertGreater**(*a*, *b*, *msg=None*)
Just like self.assertTrue(a > b), but with a nicer default message.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertGreaterEqual

WlmScriptGeneratorTest.**assertGreaterEqual**(*a*, *b*, *msg=None*)
Just like self.assertTrue(a >= b), but with a nicer default message.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertIn

WlmScriptGeneratorTest.**assertIn**(*member*, *container*, *msg=None*)
Just like self.assertTrue(a in b), but with a nicer default message.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertIs

WlmScriptGeneratorTest.**assertIs**(*expr1*, *expr2*, *msg=None*)
Just like self.assertTrue(a is b), but with a nicer default message.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertIsInstance

WlmScriptGeneratorTest.**assertIsInstance**(*obj*, *cls*, *msg=None*)
Same as self.assertTrue(isinstance(obj, cls)), with a nicer default message.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertIsNone

WlmScriptGeneratorTest.**assertIsNone**(*obj*, *msg=None*)
Same as self.assertTrue(obj is None), with a nicer default message.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertIsNot

WlmScriptGeneratorTest.**assertIsNot**(*expr1*, *expr2*, *msg=None*)
Just like self.assertTrue(a is not b), but with a nicer default message.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertIsNotNone

WlmScriptGeneratorTest.**assertIsNotNone**(*obj*, *msg=None*)
Included for symmetry with assertIsNone.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertItemsEqual

WlmScriptGeneratorTest.**assertItemsEqual**(**args*, ***kwargs*)
See `unittest.TestCase.assertItemsEqual`.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertLess

WlmScriptGeneratorTest.**assertLess**(*a*, *b*, *msg=None*)
Just like self.assertTrue(a < b), but with a nicer default message.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertLessEqual

WlmScriptGeneratorTest.**assertLessEqual**(*a*, *b*, *msg=None*)
Just like self.assertTrue(a <= b), but with a nicer default message.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertListEqual

WlmScriptGeneratorTest.**assertListEqual**(**args*, ***kwargs*)
See `unittest.TestCase.assertListEqual`.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertLogs

WlmScriptGeneratorTest.**assertLogs**(*logger=None*, *level=None*)
Fail unless a log message of level *level* or higher is emitted on *logger_name* or its children. If omitted, *level* defaults to INFO and *logger* defaults to the root logger.

This method must be used as a context manager, and will yield a recording object with two attributes: *output* and *records*. At the end of the context manager, the *output* attribute will be a list of the matching formatted log messages and the *records* attribute will be a list of the corresponding LogRecord objects.

Example:

```
with self.assertLogs('foo', level='INFO') as cm:
    logging.getLogger('foo').info('first message')
    logging.getLogger('foo.bar').error('second message')
self.assertEqual(cm.output, ['INFO:foo:first message',
                             'ERROR:foo.bar:second message'])
```

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertMultiLineEqual

WlmScriptGeneratorTest.**assertMultiLineEqual**(*first*, *second*, *msg=None*)
Assert that two multi-line strings are equal.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertNotAlmostEqual

WlmScriptGeneratorTest.**assertNotAlmostEqual**(*first*, *second*, *places=None*, *msg=None*, *delta=None*)

Fail if the two objects are equal as determined by their difference rounded to the given number of decimal places (default 7) and comparing to zero, or by comparing that the between the two objects is less than the given delta.

Note that decimal places (from zero) are usually not the same as significant digits (measured from the most signficant digit).

Objects that are equal automatically fail.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertNotAlmostEquals

WlmScriptGeneratorTest.**assertNotAlmostEquals**(**args*, ***kwargs*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertNotEqual

WlmScriptGeneratorTest.**assertNotEqual**(*first*, *second*, *msg=None*)

Fail if the two objects are equal as determined by the '!=' operator.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertNotEquals

WlmScriptGeneratorTest.**assertNotEquals**(**args*, ***kwargs*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertNotIn

WlmScriptGeneratorTest.**assertNotIn**(*member*, *container*, *msg=None*)

Just like self.assertTrue(a not in b), but with a nicer default message.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertNotIsInstance

WlmScriptGeneratorTest.**assertNotIsInstance**(*obj*, *cls*, *msg=None*)

Included for symmetry with assertIsInstance.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertNotRegex

WlmScriptGeneratorTest.**assertNotRegex**(*text*, *unexpected_regex*, *msg=None*)

Fail the test if the text matches the regular expression.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertNotRegexpMatches

WlmScriptGeneratorTest.**assertNotRegexpMatches**(**args*, ***kwargs*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertRaises

WlmScriptGeneratorTest.**assertRaises**(*expected_exception*, **args*, ***kwargs*)

Fail unless an exception of class expected_exception is raised by the callable when invoked with specified positional and keyword arguments. If a different type of exception is raised, it will not be caught, and the test case will be deemed to have suffered an error, exactly as for an unexpected exception.

If called with the callable and arguments omitted, will return a context object used like this:

```
with self.assertRaises(SomeException):
    do_something()
```

An optional keyword argument 'msg' can be provided when assertRaises is used as a context object.

The context manager keeps a reference to the exception as the 'exception' attribute. This allows you to inspect the exception after the assertion:

```
with self.assertRaises(SomeException) as cm:
    do_something()
the_exception = cm.exception
self.assertEqual(the_exception.error_code, 3)
```

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertRaisesRegex

WlmScriptGeneratorTest.**assertRaisesRegex**(**args*, ***kwargs*)

See unittest.TestCase.assertRaisesRegex.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertRaisesRegexp

WlmScriptGeneratorTest.**assertRaisesRegexp**(**args*, ***kwargs*)

See unittest.TestCase.assertRaisesRegexp.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertRegex

WlmScriptGeneratorTest.**assertRegex**(*text*, *expected_regex*, *msg=None*)

Fail the test unless the text matches the regular expression.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertRegexpMatches

WlmScriptGeneratorTest.**assertRegexpMatches**(**args*, ***kwargs*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertSequenceEqual

WlmScriptGeneratorTest.**assertSequenceEqual**(**args*, ***kwargs*)

See unittest.TestCase.assertSequenceEqual.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertSetEqual

WlmScriptGeneratorTest.**assertSetEqual**(**args*, ***kwargs*)
See unittest.TestCase.assertSetEqual.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertTrue

WlmScriptGeneratorTest.**assertTrue**(*expr*, *msg=None*)
Check that the expression is true.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertTupleEqual

WlmScriptGeneratorTest.**assertTupleEqual**(**args*, ***kwargs*)
See unittest.TestCase.assertTupleEqual.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertWarns

WlmScriptGeneratorTest.**assertWarns**(*expected_warning*, **args*, ***kwargs*)
Fail unless a warning of class warnClass is triggered by the callable when invoked with specified positional and keyword arguments. If a different type of warning is triggered, it will not be handled: depending on the other warning filtering rules in effect, it might be silenced, printed out, or raised as an exception.

If called with the callable and arguments omitted, will return a context object used like this:

```
with self.assertWarns(SomeWarning):
    do_something()
```

An optional keyword argument 'msg' can be provided when assertWarns is used as a context object.

The context manager keeps a reference to the first matching warning as the 'warning' attribute; similarly, the 'filename' and 'lineno' attributes give you information about the line of Python code from which the warning was triggered. This allows you to inspect the warning after the assertion:

```
with self.assertWarns(SomeWarning) as cm:
    do_something()
the_warning = cm.warning
self.assertEqual(the_warning.some_attribute, 147)
```

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assertWarnsRegex

WlmScriptGeneratorTest.**assertWarnsRegex**(**args*, ***kwargs*)
See unittest.TestCase.assertWarnsRegex.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.assert_

WlmScriptGeneratorTest.**assert_**(**args*, ***kwargs*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.countTestCases

WlmScriptGeneratorTest.**countTestCases**()

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.debug

WlmScriptGeneratorTest.**debug**()
Run the test without collecting errors in a TestResult

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.defaultTestResult

WlmScriptGeneratorTest.**defaultTestResult**()

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.doCleanups

WlmScriptGeneratorTest.**doCleanups**()
Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.fail

WlmScriptGeneratorTest.**fail**(*msg=None*)
Fail immediately, with the given message.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.failIf

WlmScriptGeneratorTest.**failIf**(**args*, ***kwargs*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.failIfAlmostEqual

WlmScriptGeneratorTest.**failIfAlmostEqual**(**args*, ***kwargs*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.failIfEqual

WlmScriptGeneratorTest.**failIfEqual**(**args*, ***kwargs*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.failUnless

WlmScriptGeneratorTest.**failUnless**(**args*, ***kwargs*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.failUnlessAlmostEqual

WlmScriptGeneratorTest.**failUnlessAlmostEqual**(**args*, ***kwargs*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.failUnlessEqual

WlmScriptGeneratorTest.**failUnlessEqual**(**args*, ***kwargs*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.failUnlessRaises

WlmScriptGeneratorTest.**failUnlessRaises**(**args*, ***kwargs*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.id

WlmScriptGeneratorTest.**id**()

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.run

WlmScriptGeneratorTest.**run**(*result=None*)

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.setUp

WlmScriptGeneratorTest.**setUp**()

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.setUpClass

WlmScriptGeneratorTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.shortDescription

WlmScriptGeneratorTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.skipTest

WlmScriptGeneratorTest.**skipTest**(*reason*)
Skip this test.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.subTest

WlmScriptGeneratorTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.tearDown

WlmScriptGeneratorTest.**tearDown**()

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.tearDownClass

WlmScriptGeneratorTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.test_adjust_list_to_length

WlmScriptGeneratorTest.**test_adjust_list_to_length**()
Test *mpi_array.benchmarks.utils.wlm.adjust_list_to_length()*.

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.test_generate_scripts

WlmScriptGeneratorTest.**test_generate_scripts**()
Test *mpi_array.benchmarks.utils.wlm.WlmScriptGenerator.generate_script_files()*.

Attributes

longMessage
maxDiff

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.longMessage

WlmScriptGeneratorTest.**longMessage** = True

mpi_array.benchmarks.utils.wlm_test.WlmScriptGeneratorTest.maxDiff

WlmScriptGeneratorTest.**maxDiff** = 640

1.14 The mpi_array.comms Module

MPI communicators and windows for locales.

1.14.1 Classes

LocaleCommsInfo	Communicators associated with a locale.
LocaleComms([peer_comm, intra_locale_comm, ...])	MPI communicators for inter and intra locale data exchange.

Continued on next page

Table 1.78 – continued from previous page

CartLocaleCommsInfo	Communicators associated with a cartesian topology locales.
CartLocaleComms([ndims, dims, peer_comm, ...])	Defines cartesian communication topology for locales.
CommsAndDistribution	A 3 element tuple `(locale_comms, distribution, this_locale)` describing the apportionment of array elements over MPI processes.
ThisLocaleInfo	Pair of communicator rank values `(inter_locale_rank, peer_rank)` which indicates that the rank `inter_locale_rank` of the `inter_locale_comm` communicator corresponds to the `peer_rank` rank of the `peer_comm` communicator.
RmaWindowBuffer(is_shared, shape, dtype, ...)	Details of the buffer allocated on a locale.

mpi_array.comms.LocaleCommsInfo

class mpi_array.comms.**LocaleCommsInfo**
Bases: tuple

Communicators associated with a locale.

Methods

count(...)	
index((value, [start, ...)	Raises ValueError if the value is not present.

mpi_array.comms.LocaleCommsInfo.count

LocaleCommsInfo.**count**(*value*) → integer – return number of occurrences of value

mpi_array.comms.LocaleCommsInfo.index

LocaleCommsInfo.**index**(*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

Attributes

inter_locale_comm	Alias for field number 2
intra_locale_comm	Alias for field number 1
num_locales	Alias for field number 3
peer_comm	Alias for field number 0
peer_ranks_per_locale	Alias for field number 4
rank_logger	Alias for field number 5
root_logger	Alias for field number 6

mpi_array.comms.LocaleCommsInfo.inter_locale_comm

LocaleCommsInfo.**inter_locale_comm**
Alias for field number 2

mpi_array.comms.LocaleCommsInfo.intra_locale_comm

LocaleCommsInfo.**intra_locale_comm**
Alias for field number 1

mpi_array.comms.LocaleCommsInfo.num_locales

LocaleCommsInfo.**num_locales**
Alias for field number 3

mpi_array.comms.LocaleCommsInfo.peer_comm

LocaleCommsInfo.**peer_comm**
Alias for field number 0

mpi_array.comms.LocaleCommsInfo.peer_ranks_per_locale

LocaleCommsInfo.**peer_ranks_per_locale**
Alias for field number 4

mpi_array.comms.LocaleCommsInfo.rank_logger

LocaleCommsInfo.**rank_logger**
Alias for field number 5

mpi_array.comms.LocaleCommsInfo.root_logger

LocaleCommsInfo.**root_logger**
Alias for field number 6

mpi_array.comms.LocaleComms

class mpi_array.comms.**LocaleComms**(*peer_comm=None*, *intra_locale_comm=None*, *inter_locale_comm=None*, *comms_info=None*)

Bases: object

MPI communicators for inter and intra locale data exchange. There are three communicators:

peer_comm Typically this is mpi4py.MPI.COMM_WORLD. It is the group of processes which operate on (perform computations on portions of) a globale array.

intra_locale_comm Can be `mpi4py.MPI.COMM_SELF`, but is more typically the communicator returned by `self.peer_comm.Split_type(mpi4py.MPI.COMM_TYPE_SHARED, key=self.peer_comm.rank)`. It is the communicator passed to the `mpi4py.MPI.Win.Allocate_shared()` function which allocates shared-memory and creates a shared-memory `mpi4py.MPI.Win` window.

inter_locale_comm Typically this communicator is formed by selecting a single process from each locale. This communicator (and associated *mpi4py.MPI.Win* window) is used to exchange data between locales.

Methods

__init__([peer_comm, intra_locale_comm, ...])	Construct, this is a collective call over the *peer_comm* communcator.
alloc_locale_buffer(shape, dtype)	Allocates a buffer using `mpi4py.MPI.Win.Allocate_shared()` which provides storage for the elements of the locale multi-dimensional array.
free()	

mpi_array.comms.LocaleComms.__init__

`LocaleComms.`**`__init__`**(*peer_comm=None*, *intra_locale_comm=None*, *inter_locale_comm=None*, *comms_info=None*)

Construct, this is a collective call over the *peer_comm* communcator.

Parameters

- **peer_comm** (`mpi4py.MPI.Comm`) – Communicator which is split according to shared memory allocation (uses `mpi4py.MPI.Comm.Split_type()`). If `None`, uses `mpi4py.MPI.COMM_WORLD`.
- **intra_locale_comm** (`mpi4py.MPI.Comm`) – Intra-locale communicator. Should be a subset of processes returned by *peer_comm*`.Split_type(mpi4py.MPI.COMM_TYPE_SHARED)`. If `None`, *peer_comm* is *split* into groups which can use a MPI window to allocate shared memory (i.e. in this case locale is a (possibly NUMA) node). Can also specify as `mpi4py.MPI.COMM_SELF`, in which case the locale is a single process and regular (non-shared) memory is not allocated in *alloc_locale_buffer()*.
- **inter_locale_comm** (`mpi4py.MPI.Comm`) – Inter-locale communicator used to exchange data between different locales. If `None` then one process (the *intra_locale_comm*`.rank == 0` process) is selected from each locale to form the *inter_locale_comm* communicator group.
- **comms_info** (*LocaleCommsInfo*) – Convenience construction without need for lookup via *get_locale_comms_info()*.

mpi_array.comms.LocaleComms.alloc_locale_buffer

`LocaleComms.`**`alloc_locale_buffer`**(*shape*, *dtype*)

Allocates a buffer using `mpi4py.MPI.Win.Allocate_shared()` which provides storage for the elements of the locale multi-dimensional array.

Parameters

- **shape** (sequence of `int`) – The shape of the locale array for which a buffer is allocated.

- **dtype** (numpy.dtype) – The array element type.

Return type *RmaWindowBuffer*

Returns A collections.namedtuple containing allocated buffer and associated RMA MPI windows.

mpi_array.comms.LocaleComms.free

LocaleComms.**free**()

Attributes

have_valid_inter_locale_comm	Is True if this *peer rank* has *self*.inter_locale_comm which is not None and is not mpi4py.MPI.COMM_NULL.
inter_locale_comm	A mpi4py.MPI.Comm communicator defining the group of processes which exchange data between locales.
inter_locale_rank_to_peer_rank_map	Returns sequence, m say, of int where m[inter_r] is the *peer rank* of self.peer_comm which corresponds to the *inter-locale rank* inter_r of self.inter_locale_comm.
intra_locale_comm	A mpi4py.MPI.Comm object which defines the group of processes which can allocate (and access) MPI window shared memory (allocated via mpi4py.MPI.Win.Allocate_shared() if available).
num_locales	An int indicating the number of *locales* over which an array is distributed.
peer_comm	A mpi4py.MPI.Comm which is superset of the *intra_locale_comm* and *inter_locale_comm* communicators.
peer_ranks_per_locale	A numpy.ndarray of shape (self.num_locales, num_peer_ranks_per_locale) with numpy.int64 elements.
rank_logger	A *peer_comm* logging.Logger.
root_logger	A *peer_comm* logging.Logger.
this_locale_rank_info	A *ThisLocaleInfo* object.

mpi_array.comms.LocaleComms.have_valid_inter_locale_comm

LocaleComms.**have_valid_inter_locale_comm**
Is True if this *peer rank* has *self*.inter_locale_comm which is not None and is not mpi4py.MPI.COMM_NULL.

mpi_array.comms.LocaleComms.inter_locale_comm

LocaleComms.**inter_locale_comm**
A mpi4py.MPI.Comm communicator defining the group of processes which exchange data between

locales.

mpi_array.comms.LocaleComms.inter_locale_rank_to_peer_rank_map

LocaleComms.**inter_locale_rank_to_peer_rank_map**
Returns sequence, `m` say, of `int` where `m[inter_r]` is the *peer rank* of `self.peer_comm` which corresponds to the *inter-locale rank* `inter_r` of `self.inter_locale_comm`.

Return type `None` or sequence of `int`

Returns Sequence of length `self.inter_locale_comm.size` on ranks for which `self.have_valid_inter_locale_comm is True`, `None` otherwise.

mpi_array.comms.LocaleComms.intra_locale_comm

LocaleComms.**intra_locale_comm**
A `mpi4py.MPI.Comm` object which defines the group of processes which can allocate (and access) MPI window shared memory (allocated via `mpi4py.MPI.Win.Allocate_shared()` if available).

mpi_array.comms.LocaleComms.num_locales

LocaleComms.**num_locales**
An `int` indicating the number of *locales* over which an array is distributed.

mpi_array.comms.LocaleComms.peer_comm

LocaleComms.**peer_comm**
A `mpi4py.MPI.Comm` which is super-set of the *intra_locale_comm* and *inter_locale_comm* communicators.

mpi_array.comms.LocaleComms.peer_ranks_per_locale

LocaleComms.**peer_ranks_per_locale**
A `numpy.ndarray` of shape `(self.num_locales, num_peer_ranks_per_locale)` with `numpy.int64` elements. The `self.peer_ranks_per_locale[inter_locale_rank]` sequence of elements are the *peer_comm* ranks which make up locale associated with *inter_locale_comm* rank `inter_locale_rank`.

mpi_array.comms.LocaleComms.rank_logger

LocaleComms.**rank_logger**
A *peer_comm* `logging.Logger`.

mpi_array.comms.LocaleComms.root_logger

LocaleComms.**root_logger**
A *peer_comm* `logging.Logger`.

mpi_array.comms.LocaleComms.this_locale_rank_info

LocaleComms.**this_locale_rank_info**
A *ThisLocaleInfo* object. Indicates the `self.inter_locale_comm.rank` and *self.peer_comm.rank* on processes for which `self.have_valid_inter_locale_comm` is `True`. Is `mpi4py.MPI.UNDEFINED` on processes where `self.have_valid_inter_locale_comm` is `False`.

mpi_array.comms.CartLocaleCommsInfo

class mpi_array.comms.**CartLocaleCommsInfo**
Bases: `tuple`

Communicators associated with a cartesian topology locales.

Methods

count(...)	
index((value, [start, ...)	Raises ValueError if the value is not present.

mpi_array.comms.CartLocaleCommsInfo.count

CartLocaleCommsInfo.**count**(*value*) → integer – return number of occurrences of value

mpi_array.comms.CartLocaleCommsInfo.index

CartLocaleCommsInfo.**index**(*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

Attributes

cart_comm	Alias for field number 8
dims	Alias for field number 7
inter_locale_comm	Alias for field number 2
intra_locale_comm	Alias for field number 1
num_locales	Alias for field number 3
peer_comm	Alias for field number 0
peer_ranks_per_locale	Alias for field number 4
rank_logger	Alias for field number 5
root_logger	Alias for field number 6

mpi_array.comms.CartLocaleCommsInfo.cart_comm

CartLocaleCommsInfo.**cart_comm**
Alias for field number 8

mpi_array.comms.CartLocaleCommsInfo.dims

CartLocaleCommsInfo.**dims**
Alias for field number 7

mpi_array.comms.CartLocaleCommsInfo.inter_locale_comm

CartLocaleCommsInfo.**inter_locale_comm**
Alias for field number 2

mpi_array.comms.CartLocaleCommsInfo.intra_locale_comm

CartLocaleCommsInfo.**intra_locale_comm**
Alias for field number 1

mpi_array.comms.CartLocaleCommsInfo.num_locales

CartLocaleCommsInfo.**num_locales**
Alias for field number 3

mpi_array.comms.CartLocaleCommsInfo.peer_comm

CartLocaleCommsInfo.**peer_comm**
Alias for field number 0

mpi_array.comms.CartLocaleCommsInfo.peer_ranks_per_locale

CartLocaleCommsInfo.**peer_ranks_per_locale**
Alias for field number 4

mpi_array.comms.CartLocaleCommsInfo.rank_logger

CartLocaleCommsInfo.**rank_logger**
Alias for field number 5

mpi_array.comms.CartLocaleCommsInfo.root_logger

CartLocaleCommsInfo.**root_logger**
Alias for field number 6

mpi_array.comms.CartLocaleComms

class mpi_array.comms.**CartLocaleComms**(*ndims=None*, *dims=None*, *peer_comm=None*, *intra_locale_comm=None*, *inter_locale_comm=None*, *cart_comm=None*)
Bases: *mpi_array.comms.LocaleComms*

Defines cartesian communication topology for locales. In addition to the *LocaleComms* communicators, defines:

cart_comm Typically this communicator is created using the call *inter_locale_comm*.`Create_cart(...)`. This communicator (and associated *mpi4py.MPI.Win* window) is used to exchange data between locales. Note that *inter_locale_comm* property over-rides the *LocaleComms.inter_locale_comm* property to return the *cart_comm* communicator.

Methods

__init__([ndims, dims, peer_comm, ...])	Initialises cartesian communicator for inter-locale data exchange.
alloc_locale_buffer(shape, dtype)	Allocates a buffer using `mpi4py.MPI.Win.Allocate_shared()` which provides storage for the elements of the locale multi-dimensional array.
free()	

mpi_array.comms.CartLocaleComms.__init__

`CartLocaleComms.`**`__init__`**(*ndims=None*, *dims=None*, *peer_comm=None*, *intra_locale_comm=None*, *inter_locale_comm=None*, *cart_comm=None*)

Initialises cartesian communicator for inter-locale data exchange. Need to specify at least one of the *ndims* or *dims*. to indicate the dimension of the cartesian partitioning.

Parameters

- **ndims** (`int`) – Dimension of the cartesian partitioning, e.g. 1D, 2D, 3D, etc. If `None`, *ndims*=`len(`*dims*`)`.
- **dims** (sequence of `int`) – The number of partitions along each array axis, zero elements are replaced with positive integers such that `numpy.product(`*dims*`) ==` *peer_comm*`.size`. If `None`, *dims* `= (0,)*`*ndims*.
- **peer_comm** (`mpi4py.MPI.Comm`) – The MPI processes which will have access (via a `mpi4py.MPI.Win` object) to the distributed array. If `None` uses `mpi4py.MPI.COMM_WORLD`.
- **intra_locale_comm** (`mpi4py.MPI.Comm`) – The MPI communicator used to create a window which can be used to allocate shared memory via `mpi4py.MPI.Win.Allocate_shared()`.
- **inter_locale_comm** (`mpi4py.MPI.Comm`) – Inter-locale communicator used to exchange data between different locales.
- **cart_comm** (`mpi4py.MPI.Comm`) – Cartesian topology inter-locale communicator used to exchange data between different locales.

mpi_array.comms.CartLocaleComms.alloc_locale_buffer

`CartLocaleComms.`**`alloc_locale_buffer`**(*shape*, *dtype*)

Allocates a buffer using `mpi4py.MPI.Win.Allocate_shared()` which provides storage for the elements of the locale multi-dimensional array.

Parameters

- **shape** (sequence of int) – The shape of the locale array for which a buffer is allocated.
- **dtype** (numpy.dtype) – The array element type.

Return type *RmaWindowBuffer*

Returns A collections.namedtuple containing allocated buffer and associated RMA MPI windows.

mpi_array.comms.CartLocaleComms.free

CartLocaleComms.**free**()

Attributes

cart_comm	A mpi4py.MPI.CartComm communicator defining a cartesian topology of MPI processes (typically one process per locale) used for inter-locale exchange of array data.
cart_coord_to_cart_rank_map	A dict of tuple cartesian coordinate (mpi4py.MPI.CartComm.Get_coords()) keys which map to the associated *cart_comm* peer_rank.
dims	The number of partitions along each array axis.
have_valid_cart_comm	Is True if this peer_rank has *self*.cart_comm which is not None and is not mpi4py.MPI.COMM_NULL.
have_valid_inter_locale_comm	Is True if this *peer rank* has *self*.inter_locale_comm which is not None and is not mpi4py.MPI.COMM_NULL.
inter_locale_comm	Overrides *LocaleComms.inter_locale_comm* to return *cart_comm*.
inter_locale_rank_to_peer_rank_map	Returns sequence, m say, of int where m[inter_r] is the *peer rank* of self.peer_comm which corresponds to the *inter-locale rank* inter_r of self.inter_locale_comm.
intra_locale_comm	A mpi4py.MPI.Comm object which defines the group of processes which can allocate (and access) MPI window shared memory (allocated via mpi4py.MPI.Win.Allocate_shared() if available).
ndim	An int indicating the dimension of the cartesian topology.
num_locales	An int indicating the number of *locales* over which an array is distributed.
peer_comm	A mpi4py.MPI.Comm which is superset of the *intra_locale_comm* and *inter_locale_comm* communicators.
peer_ranks_per_locale	A numpy.ndarray of shape (self.num_locales, num_peer_ranks_per_locale) with numpy.int64 elements.
rank_logger	A *peer_comm* logging.Logger.

Continued on next page

Table 1.86 – continued from previous page

root_logger	A *peer_comm* `logging.Logger`.
this_locale_rank_info	A *ThisLocaleInfo* object.

mpi_array.comms.CartLocaleComms.cart_comm

`CartLocaleComms.`**`cart_comm`**
A `mpi4py.MPI.CartComm` communicator defining a cartesian topology of MPI processes (typically one process per locale) used for inter-locale exchange of array data.

mpi_array.comms.CartLocaleComms.cart_coord_to_cart_rank_map

`CartLocaleComms.`**`cart_coord_to_cart_rank_map`**
A `dict` of `tuple` cartesian coordinate (`mpi4py.MPI.CartComm.Get_coords()`) keys which map to the associated *cart_comm* peer_rank.

mpi_array.comms.CartLocaleComms.dims

`CartLocaleComms.`**`dims`**
The number of partitions along each array axis. Defines the cartesian topology over which an array is distributed.

mpi_array.comms.CartLocaleComms.have_valid_cart_comm

`CartLocaleComms.`**`have_valid_cart_comm`**
Is `True` if this peer_rank has *self*`.cart_comm` which is not `None` and is not `mpi4py.MPI.COMM_NULL`.

mpi_array.comms.CartLocaleComms.have_valid_inter_locale_comm

`CartLocaleComms.`**`have_valid_inter_locale_comm`**
Is `True` if this *peer rank* has *self*`.inter_locale_comm` which is not `None` and is not `mpi4py.MPI.COMM_NULL`.

mpi_array.comms.CartLocaleComms.inter_locale_comm

`CartLocaleComms.`**`inter_locale_comm`**
Overrides *LocaleComms.inter_locale_comm* to return *cart_comm*.

mpi_array.comms.CartLocaleComms.inter_locale_rank_to_peer_rank_map

`CartLocaleComms.`**`inter_locale_rank_to_peer_rank_map`**
Returns sequence, `m` say, of `int` where `m[inter_r]` is the *peer rank* of `self.peer_comm` which corresponds to the *inter-locale rank* `inter_r` of `self.inter_locale_comm`.

Return type `None` or sequence of `int`

Returns Sequence of length `self.inter_locale_comm.size` on ranks for which `self.have_valid_inter_locale_comm is True`, `None` otherwise.

mpi_array.comms.CartLocaleComms.intra_locale_comm

`CartLocaleComms.`**`intra_locale_comm`**
A `mpi4py.MPI.Comm` object which defines the group of processes which can allocate (and access) MPI window shared memory (allocated via `mpi4py.MPI.Win.Allocate_shared()` if available).

mpi_array.comms.CartLocaleComms.ndim

`CartLocaleComms.`**`ndim`**
An `int` indicating the dimension of the cartesian topology.

mpi_array.comms.CartLocaleComms.num_locales

`CartLocaleComms.`**`num_locales`**
An `int` indicating the number of *locales* over which an array is distributed.

mpi_array.comms.CartLocaleComms.peer_comm

`CartLocaleComms.`**`peer_comm`**
A `mpi4py.MPI.Comm` which is super-set of the *intra_locale_comm* and *inter_locale_comm* communicators.

mpi_array.comms.CartLocaleComms.peer_ranks_per_locale

`CartLocaleComms.`**`peer_ranks_per_locale`**
A `numpy.ndarray` of shape `(self.num_locales, num_peer_ranks_per_locale)` with `numpy.int64` elements. The `self.peer_ranks_per_locale[inter_locale_rank]` sequence of elements are the *peer_comm* ranks which make up locale associated with *inter_locale_comm* rank `inter_locale_rank`.

mpi_array.comms.CartLocaleComms.rank_logger

`CartLocaleComms.`**`rank_logger`**
A *peer_comm* `logging.Logger`.

mpi_array.comms.CartLocaleComms.root_logger

`CartLocaleComms.`**`root_logger`**
A *peer_comm* `logging.Logger`.

mpi_array.comms.CartLocaleComms.this_locale_rank_info

CartLocaleComms.**this_locale_rank_info**
A *ThisLocaleInfo* object. Indicates the `self.inter_locale_comm.rank` and *self.peer_comm.rank* on processes for which `self.have_valid_inter_locale_comm` is `True`. Is `mpi4py.MPI.UNDEFINED` on processes where `self.have_valid_inter_locale_comm` is `False`.

mpi_array.comms.CommsAndDistribution

class mpi_array.comms.**CommsAndDistribution**
Bases: `tuple`

A 3 element tuple `(locale_comms, distribution, this_locale)` describing the apportionment of array elements over MPI processes.

Methods

count(...)	
index((value, [start, ...)	Raises ValueError if the value is not present.

mpi_array.comms.CommsAndDistribution.count

CommsAndDistribution.**count**(*value*) → integer – return number of occurrences of value

mpi_array.comms.CommsAndDistribution.index

CommsAndDistribution.**index**(*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

Attributes

distribution	A *mpi_array.distribution.Distribution* object describing the apportionment of array elements over locales.
locale_comms	A *LocaleComms* object containing communicators for exchanging data between locales.
this_locale	A *ThisLocaleInfo* with rank pair pertinent for this locale.

mpi_array.comms.CommsAndDistribution.distribution

CommsAndDistribution.**distribution**
A *mpi_array.distribution.Distribution* object describing the apportionment of array elements over locales.

mpi_array.comms.CommsAndDistribution.locale_comms

CommsAndDistribution.**locale_comms**
A *LocaleComms* object containing communicators for exchanging data between locales.

mpi_array.comms.CommsAndDistribution.this_locale

CommsAndDistribution.**this_locale**
A *ThisLocaleInfo* with rank pair pertinent for this locale.

mpi_array.comms.ThisLocaleInfo

class mpi_array.comms.**ThisLocaleInfo**
Bases: tuple

Pair of communicator rank values `(inter_locale_rank, peer_rank)` which indicates that the rank `inter_locale_rank` of the `inter_locale_comm` communicator corresponds to the `peer_rank` rank of the `peer_comm` communicator.

Methods

count(...)	
index((value, [start, ...)	Raises ValueError if the value is not present.

mpi_array.comms.ThisLocaleInfo.count

ThisLocaleInfo.**count**(*value*) → integer – return number of occurrences of value

mpi_array.comms.ThisLocaleInfo.index

ThisLocaleInfo.**index**(*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

Attributes

inter_locale_rank	A int indicating the rank of inter_locale_comm communicator.
peer_rank	A int indicating the rank of peer_comm communicator.

mpi_array.comms.ThisLocaleInfo.inter_locale_rank

ThisLocaleInfo.**inter_locale_rank**
A int indicating the rank of inter_locale_comm communicator.

mpi_array.comms.ThisLocaleInfo.peer_rank

ThisLocaleInfo.**peer_rank**
A int indicating the rank of peer_comm communicator.

mpi_array.comms.RmaWindowBuffer

class mpi_array.comms.**RmaWindowBuffer**(*is_shared*, *shape*, *dtype*, *itemsize*, *intra_locale_win*, *peer_comm=None*, *inter_locale_comm=None*, *peer_win=None*, *inter_locale_win=None*, *root_logger=None*, *rank_logger=None*)

Bases: object

Details of the buffer allocated on a locale.

Methods

__init__(is_shared, shape, dtype, itemsize, ...)	
create_inter_locale_win()	Creates a mpi4py.MPI.Win window (over the *inter_locale_comm* communicator) exposing the *buffer* memory.
create_peer_win()	Creates a mpi4py.MPI.Win window (over the *peer_win* communicator) exposing the *buffer* memory.
free()	Free MPI windows and associated buffer memory.
initialise_windows()	Creates both of the *peer_win* and *inter_locale_win* mpi4py.MPI.Win objects.

mpi_array.comms.RmaWindowBuffer.__init__

RmaWindowBuffer.**__init__**(*is_shared*, *shape*, *dtype*, *itemsize*, *intra_locale_win*, *peer_comm=None*, *inter_locale_comm=None*, *peer_win=None*, *inter_locale_win=None*, *root_logger=None*, *rank_logger=None*)

mpi_array.comms.RmaWindowBuffer.create_inter_locale_win

RmaWindowBuffer.**create_inter_locale_win**()
Creates a mpi4py.MPI.Win window (over the *inter_locale_comm* communicator) exposing the *buffer* memory.

Return type mpi4py.MPI.Win

Returns Newly created window exposing the *buffer* memory. Returns mpi4py.MPI.WIN_NULL on processes which have *inter_locale_comm* equal to mpi4py.MPI.COMM_NULL.

mpi_array.comms.RmaWindowBuffer.create_peer_win

RmaWindowBuffer.**create_peer_win**()
Creates a mpi4py.MPI.Win window (over the *peer_win* communicator) exposing the *buffer* memory.

Return type mpi4py.MPI.Win

Returns Newly created window exposing the *buffer* memory.

mpi_array.comms.RmaWindowBuffer.free

RmaWindowBuffer.**free**()
Free MPI windows and associated buffer memory.

mpi_array.comms.RmaWindowBuffer.initialise_windows

RmaWindowBuffer.**initialise_windows**()
Creates both of the *peer_win* and *inter_locale_win* mpi4py.MPI.Win objects.

Attributes

buffer	A numpy.ndarray with memory allocated using one of mpi4py.MPI.Win.Allocate() or mpi4py.MPI.Win.Allocate_shared() as the buffer.
dtype	A numpy.dtype indicating the data type of elements stored in the array.
inter_locale_comm	The *inter-locale* mpi4py.MPI.Win MPI communicator.
inter_locale_win	The mpi4py.MPI.Win created from the inter_locale_comm communicator which exposes *buffer* for inter-locale RMA access.
inter_locale_win_initialised	Returns True if *inter-locale* RMA window (*inter_locale_win*) has been created.
intra_locale_win	The mpi4py.MPI.Win created from the intra_locale_comm communicator which exposes *buffer* for intra-locale RMA access.
intra_locale_win_memory	The memory allocated using one of mpi4py.MPI.Win.Allocate() or mpi4py.MPI.Win.Allocate_shared().
is_shared	A bool, if True then buffer memory was allocated using mpi4py.MPI.Win.Allocate_shared(), otherwise memory was allocated using mpi4py.MPI.Win.Allocate().
itemsize	An int indicating the number of bytes per array element (same as numpy.dtype.itemsize).
peer_comm	The *peer* mpi4py.MPI.Comm MPI communicator.
peer_win	The mpi4py.MPI.Win created from the peer_comm communicator which exposes *buffer* for inter-locale RMA access.

Continued on next page

Table 1.92 – continued from previous page

peer_win_initialised	Returns `True` if *peer* RMA window (*peer_win*) has been created.
rank_logger	A `logging.Logger` for logging messages from all rank MPI processes of *peer_comm*.
root_logger	A `logging.Logger` for logging messages from the *root* MPI process of *peer_comm*.
shape	A sequence of `int` indicating the shape of the locale array.
windows_initialised	Returns `True` if both *peer* RMA window (*peer_win*) and *inter-locale* RMA window (*inter_locale_win*) have been created.

mpi_array.comms.RmaWindowBuffer.buffer

RmaWindowBuffer.**buffer**
A `numpy.ndarray` with memory allocated using one of `mpi4py.MPI.Win.Allocate()` or `mpi4py.MPI.Win.Allocate_shared()` as the buffer.

mpi_array.comms.RmaWindowBuffer.dtype

RmaWindowBuffer.**dtype**
A `numpy.dtype` indicating the data type of elements stored in the array.

mpi_array.comms.RmaWindowBuffer.inter_locale_comm

RmaWindowBuffer.**inter_locale_comm**
The *inter-locale* `mpi4py.MPI.Win` MPI communicator.

mpi_array.comms.RmaWindowBuffer.inter_locale_win

RmaWindowBuffer.**inter_locale_win**
The `mpi4py.MPI.Win` created from the `inter_locale_comm` communicator which exposes *buffer* for inter-locale RMA access.

mpi_array.comms.RmaWindowBuffer.inter_locale_win_initialised

RmaWindowBuffer.**inter_locale_win_initialised**
Returns `True` if *inter-locale* RMA window (*inter_locale_win*) has been created. `False` otherwise.

mpi_array.comms.RmaWindowBuffer.intra_locale_win

RmaWindowBuffer.**intra_locale_win**
The `mpi4py.MPI.Win` created from the `intra_locale_comm` communicator which exposes *buffer* for intra-locale RMA access. When *intra_locale_win*`.group.size > 1` then *buffer* was allocated as shared memory (using `mpi4py.MPI.Win.Allocate_shared()`).

mpi_array.comms.RmaWindowBuffer.intra_locale_win_memory

RmaWindowBuffer.**intra_locale_win_memory**
: The memory allocated using one of `mpi4py.MPI.Win.Allocate()` or `mpi4py.MPI.Win.Allocate_shared()`. This memory is used to store elements of the globale array apportioned to a locale.

mpi_array.comms.RmaWindowBuffer.is_shared

RmaWindowBuffer.**is_shared**
: A `bool`, if `True` then buffer memory was allocated using `mpi4py.MPI.Win.Allocate_shared()`, otherwise memory was allocated using `mpi4py.MPI.Win.Allocate()`.

mpi_array.comms.RmaWindowBuffer.itemsize

RmaWindowBuffer.**itemsize**
: An `int` indicating the number of bytes per array element (same as `numpy.dtype.itemsize`).

mpi_array.comms.RmaWindowBuffer.peer_comm

RmaWindowBuffer.**peer_comm**
: The *peer* `mpi4py.MPI.Comm` MPI communicator.

mpi_array.comms.RmaWindowBuffer.peer_win

RmaWindowBuffer.**peer_win**
: The `mpi4py.MPI.Win` created from the `peer_comm` communicator which exposes *buffer* for inter-locale RMA access.

mpi_array.comms.RmaWindowBuffer.peer_win_initialised

RmaWindowBuffer.**peer_win_initialised**
: Returns `True` if *peer* RMA window (*peer_win*) has been created. `False` otherwise.

mpi_array.comms.RmaWindowBuffer.rank_logger

RmaWindowBuffer.**rank_logger**
: A `logging.Logger` for logging messages from all rank MPI processes of *peer_comm*.

mpi_array.comms.RmaWindowBuffer.root_logger

RmaWindowBuffer.**root_logger**
: A `logging.Logger` for logging messages from the *root* MPI process of *peer_comm*.

mpi_array.comms.RmaWindowBuffer.shape

RmaWindowBuffer.**shape**
A sequence of int indicating the shape of the locale array.

mpi_array.comms.RmaWindowBuffer.windows_initialised

RmaWindowBuffer.**windows_initialised**
Returns True if both *peer* RMA window (*peer_win*) and *inter-locale* RMA window (*inter_locale_win*) have been created. False otherwise.

1.14.2 Factory Functions

create_locale_comms_info([peer_comm, ...])	Creates a *LocaleCommsInfo* associated with the specified communicators.
get_locale_comms_info([peer_comm, ...])	Finds or creates a *LocaleCommsInfo* associated with the specified communicators.
create_cart_locale_comms_info([ndims, dims, ...])	Creates a *CartLocaleCommsInfo* associated with the specified communicators.
get_cart_locale_comms_info([ndims, dims, ...])	Finds or creates a *CartLocaleCommsInfo* associated with the specified communicators.
create_locale_comms([locale_type, ...])	Factory function for creating a *LocaleComms* object.
create_block_distribution(shape[, ...])	Factory function for creating mpi_array.distrbution.BlockPartition distribution and associated *CartLocaleComms*.
create_cloned_distribution(shape[, ...])	Factory function for creating mpi_array.distrbution.ClonedDistribution distribution and associated *LocaleComms*.
create_single_locale_distribution(shape[, ...])	Factory function for creating mpi_array.distrbution.SingleLocaleDistribution distribution and associated *LocaleComms*.
create_distribution(shape[, distrib_type, ...])	Factory function for creating *mpi_array.distribution.Distribution* and associated *LocaleComms*.

mpi_array.comms.create_locale_comms_info

mpi_array.comms.**create_locale_comms_info**(*peer_comm=None*, *intra_locale_comm=None*, *inter_locale_comm=None*)
Creates a *LocaleCommsInfo* associated with the specified communicators. Collective over *peer_comm*.

mpi_array.comms.get_locale_comms_info

mpi_array.comms.**get_locale_comms_info**(*peer_comm=None*, *intra_locale_comm=None*, *inter_locale_comm=None*)
Finds or creates a *LocaleCommsInfo* associated with the specified communicators. Collective over *peer_comm*.

mpi_array.comms.create_cart_locale_comms_info

mpi_array.comms.**create_cart_locale_comms_info**(*ndims=None*, *dims=None*, *peer_comm=None*, *intra_locale_comm=None*, *inter_locale_comm=None*, *cart_comm=None*)

Creates a *CartLocaleCommsInfo* associated with the specified communicators. Collective over *peer_comm*.

mpi_array.comms.get_cart_locale_comms_info

mpi_array.comms.**get_cart_locale_comms_info**(*ndims=None*, *dims=None*, *peer_comm=None*, *intra_locale_comm=None*, *inter_locale_comm=None*, *cart_comm=None*)

Finds or creates a *CartLocaleCommsInfo* associated with the specified communicators. Collective over *peer_comm*.

mpi_array.comms.create_locale_comms

mpi_array.comms.**create_locale_comms**(*locale_type=None*, *peer_comm=None*, *intra_locale_comm=None*, *inter_locale_comm=None*)

Factory function for creating a *LocaleComms* object.

Parameters

- **locale_type** (str) – One of mpi_array.comms.DT_PROCESS or mpi_array.comms.DT_NODE.
- **peer_comm** (mpi4py.MPI.Comm) – See *LocaleComms*.
- **intra_locale_comm** (mpi4py.MPI.Comm) – See *LocaleComms*.
- **inter_locale_comm** (mpi4py.MPI.Comm) – See *LocaleComms*.

Return type *LocaleComms*

Returns A *LocaleComms* object.

mpi_array.comms.create_block_distribution

mpi_array.comms.**create_block_distribution**(*shape*, *locale_type=None*, *dims=None*, *halo=0*, *peer_comm=None*, *intra_locale_comm=None*, *inter_locale_comm=None*, *cart_comm=None*)

Factory function for creating mpi_array.distrbution.BlockPartition distribution and associated *CartLocaleComms*.

Parameters

- **shape** (sequence of int) – Shape of the globale array.
- **locale_type** (str) – One of mpi_array.comms.DT_PROCESS or mpi_array.comms.DT_NODE. Defines locales.
- **dims** (sequence of int) – Defines the partitioning of the globale array axes.
- **peer_comm** (mpi4py.MPI.Comm) – See *LocaleComms*.
- **intra_locale_comm** (mpi4py.MPI.Comm) – See *LocaleComms*.

- **inter_locale_comm** (mpi4py.MPI.Comm) – See *LocaleComms*.
- **cart_comm** (mpi4py.MPI.Comm) – See *CartLocaleComms*.

Return type *CommsAndDistribution*

Returns A *CommsAndDistribution* collections.namedtuple.

mpi_array.comms.create_cloned_distribution

mpi_array.comms.**create_cloned_distribution**(*shape*, *locale_type=None*, *halo=0*, *peer_comm=None*, *intra_locale_comm=None*, *inter_locale_comm=None*)

Factory function for creating mpi_array.distrbution.ClonedDistribution distribution and associated *LocaleComms*.

Return type *CommsAndDistribution*

Returns A *CommsAndDistribution* pair.

mpi_array.comms.create_single_locale_distribution

mpi_array.comms.**create_single_locale_distribution**(*shape*, *locale_type=None*, *halo=0*, *inter_locale_rank=0*, *peer_comm=None*, *intra_locale_comm=None*, *inter_locale_comm=None*)

Factory function for creating mpi_array.distrbution.SingleLocaleDistribution distribution and associated *LocaleComms*.

Return type *CommsAndDistribution*

Returns A *CommsAndDistribution* pair.

mpi_array.comms.create_distribution

mpi_array.comms.**create_distribution**(*shape*, *distrib_type=None*, *locale_type='node'*, ***kwargs*)

Factory function for creating *mpi_array.distribution.Distribution* and associated *LocaleComms*.

Parameters

- **shape** (sequence of int) – Shape of the globale array.
- **distrib_type** (str) – One of *mpi_array.comms.DT_BLOCK* or *mpi_array.comms.DT_SLAB* or *mpi_array.comms.DT_CLONED* or *mpi_array.comms.DT_SINGLE_LOCALE*. Defines how the globale array is dstributed over locales. If None defaults to *mpi_array.comms.DT_BLOCK* if numpy.product(shape) > 0 otherwise *mpi_array.comms.DT_CLONED*.
- **locale_type** (str) – One of mpi_array.comms.DT_PROCESS or mpi_array.comms.DT_NODE. Defines locales.
- **dims** (sequence of int) – Only relevant when *distrib_type* == DT_BLOCK. Defines the partitioning of the globale array axes.

- **axis** (int) – Only relevant when *distrib_type* == DT_SLAB. Indicates the single axis of the globale array partitioned into slabs.
- **peer_comm** (mpi4py.MPI.Comm) – See *LocaleComms*.
- **intra_locale_comm** (mpi4py.MPI.Comm) – See *LocaleComms*.
- **inter_locale_comm** (mpi4py.MPI.Comm) – See *LocaleComms*.
- **cart_comm** (mpi4py.MPI.Comm) – Only relevant when *distrib_type* == DT_BLOCK or *distrib_type* == DT_SLAB. See *CartLocaleComms*.

Return type *CommsAndDistribution*

Returns A *CommsAndDistribution* collections.namedtuple.

See also:

create_block_distribution() *create_cloned_distribution()* *create_single_locale_distribution()*

1.14.3 Attributes

LT_PROCESS	Single process locale type
LT_NODE	Node (NUMA) locale type
DT_BLOCK	Hyper-block partition distribution type
DT_SLAB	Hyper-slab partition distribution type
DT_CLONED	Entire array repeated on each locale.
DT_SINGLE_LOCALE	Entire array on single locale, no array elements on other locales.

mpi_array.comms.LT_PROCESS

mpi_array.comms.**LT_PROCESS = 'process'**
Single process locale type

mpi_array.comms.LT_NODE

mpi_array.comms.**LT_NODE = 'node'**
Node (NUMA) locale type

mpi_array.comms.DT_BLOCK

mpi_array.comms.**DT_BLOCK = 'block'**
Hyper-block partition distribution type

mpi_array.comms.DT_SLAB

mpi_array.comms.**DT_SLAB = 'slab'**
Hyper-slab partition distribution type

mpi_array.comms.DT_CLONED

mpi_array.comms.**DT_CLONED** = **'cloned'**
Entire array repeated on each locale.

mpi_array.comms.DT_SINGLE_LOCALE

mpi_array.comms.**DT_SINGLE_LOCALE** = **'single_locale'**
Entire array on single locale, no array elements on other locales.

1.15 The mpi_array.comms_test Module

Module defining *mpi_array.comms* unit-tests. Execute as:

```
python -m mpi_array.comms_test
```

or:

```
mpirun -n 4 python -m mpi_array.comms_test
```

1.15.1 Classes

LocaleCommsTest([methodName])	Tests for *mpi_array.comms.LocaleComms*.
CartLocaleCommsTest([methodName])	unittest.TestCase for *mpi_array.comms.CartLocaleComms*.
CreateDistributionTest([methodName])	Tests for *mpi_array.comms.create_distribution()*.

mpi_array.comms_test.LocaleCommsTest

class mpi_array.comms_test.**LocaleCommsTest**(*methodName='runTest'*)
Bases: *mpi_array.unittest.TestCase*

Tests for *mpi_array.comms.LocaleComms*.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
countTestCases()	

Continued on next page

Table 1.96 – continued from previous page

debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	Hook method for setting up the test fixture before exercising it.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_construct()	Test mpi_array.comms.LocaleComms.__init__()
test_construct_invalid_comms()	Test mpi_array.comms.LocaleComms.__init__()
test_construct_no_shared()	
test_get_shared_mem_usage_percent_string()	Coverage for mpi_array.comms.get_shared_mem_usage_percent_string().

mpi_array.comms_test.LocaleCommsTest.__init__

LocaleCommsTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.comms_test.LocaleCommsTest.addCleanup

LocaleCommsTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.comms_test.LocaleCommsTest.addTypeEqualityFunc

LocaleCommsTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.comms_test.LocaleCommsTest.assertArraySplitEqual

LocaleCommsTest.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.comms_test.LocaleCommsTest.countTestCases

LocaleCommsTest.**countTestCases**()

mpi_array.comms_test.LocaleCommsTest.debug

LocaleCommsTest.**debug**()

Run the test without collecting errors in a TestResult

mpi_array.comms_test.LocaleCommsTest.defaultTestResult

LocaleCommsTest.**defaultTestResult**()

mpi_array.comms_test.LocaleCommsTest.doCleanups

LocaleCommsTest.**doCleanups**()

Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.comms_test.LocaleCommsTest.id

LocaleCommsTest.**id**()

mpi_array.comms_test.LocaleCommsTest.run

LocaleCommsTest.**run**(*result=None*)

mpi_array.comms_test.LocaleCommsTest.setUp

LocaleCommsTest.**setUp**()
Hook method for setting up the test fixture before exercising it.

mpi_array.comms_test.LocaleCommsTest.setUpClass

LocaleCommsTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.comms_test.LocaleCommsTest.shortDescription

LocaleCommsTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.comms_test.LocaleCommsTest.skipTest

LocaleCommsTest.**skipTest**(*reason*)
Skip this test.

mpi_array.comms_test.LocaleCommsTest.subTest

LocaleCommsTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.comms_test.LocaleCommsTest.tearDown

LocaleCommsTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.comms_test.LocaleCommsTest.tearDownClass

LocaleCommsTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.comms_test.LocaleCommsTest.test_construct

LocaleCommsTest.**test_construct**()
Test *mpi_array.comms.LocaleComms.__init__()*

mpi_array.comms_test.LocaleCommsTest.test_construct_invalid_comms

LocaleCommsTest.**test_construct_invalid_comms**()
Test *mpi_array.comms.LocaleComms.__init__()*

mpi_array.comms_test.LocaleCommsTest.test_construct_no_shared

LocaleCommsTest.**test_construct_no_shared**()

mpi_array.comms_test.LocaleCommsTest.test_get_shared_mem_usage_percent_string

LocaleCommsTest.**test_get_shared_mem_usage_percent_string**()
Coverage for mpi_array.comms.get_shared_mem_usage_percent_string().

Attributes

longMessage
maxDiff

mpi_array.comms_test.LocaleCommsTest.longMessage

LocaleCommsTest.**longMessage** = True

mpi_array.comms_test.LocaleCommsTest.maxDiff

LocaleCommsTest.**maxDiff** = 640

mpi_array.comms_test.CartLocaleCommsTest

class mpi_array.comms_test.**CartLocaleCommsTest**(*methodName='runTest'*)
Bases: *mpi_array.unittest.TestCase*

unittest.TestCase for *mpi_array.comms.CartLocaleComms*.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.

Continued on next page

Table 1.98 – continued from previous page

countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	Hook method for setting up the test fixture before exercising it.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_alloc_locale_buffer()	
test_construct_invalid_cart_comm()	
test_construct_invalid_dims()	
test_construct_no_shared()	
test_construct_shared()	

mpi_array.comms_test.CartLocaleCommsTest.__init__

CartLocaleCommsTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.comms_test.CartLocaleCommsTest.addCleanup

CartLocaleCommsTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.comms_test.CartLocaleCommsTest.addTypeEqualityFunc

CartLocaleCommsTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.comms_test.CartLocaleCommsTest.assertArraySplitEqual

CartLocaleCommsTest.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.comms_test.CartLocaleCommsTest.countTestCases

CartLocaleCommsTest.**countTestCases**()

mpi_array.comms_test.CartLocaleCommsTest.debug

CartLocaleCommsTest.**debug**()

Run the test without collecting errors in a TestResult

mpi_array.comms_test.CartLocaleCommsTest.defaultTestResult

CartLocaleCommsTest.**defaultTestResult**()

mpi_array.comms_test.CartLocaleCommsTest.doCleanups

CartLocaleCommsTest.**doCleanups**()

Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.comms_test.CartLocaleCommsTest.id

CartLocaleCommsTest.**id**()

mpi_array.comms_test.CartLocaleCommsTest.run

CartLocaleCommsTest.**run**(*result=None*)

mpi_array.comms_test.CartLocaleCommsTest.setUp

CartLocaleCommsTest.**setUp**()

Hook method for setting up the test fixture before exercising it.

mpi_array.comms_test.CartLocaleCommsTest.setUpClass

CartLocaleCommsTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.comms_test.CartLocaleCommsTest.shortDescription

CartLocaleCommsTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.comms_test.CartLocaleCommsTest.skipTest

CartLocaleCommsTest.**skipTest**(*reason*)
Skip this test.

mpi_array.comms_test.CartLocaleCommsTest.subTest

CartLocaleCommsTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.comms_test.CartLocaleCommsTest.tearDown

CartLocaleCommsTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.comms_test.CartLocaleCommsTest.tearDownClass

CartLocaleCommsTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.comms_test.CartLocaleCommsTest.test_alloc_locale_buffer

CartLocaleCommsTest.**test_alloc_locale_buffer**()

mpi_array.comms_test.CartLocaleCommsTest.test_construct_invalid_cart_comm

CartLocaleCommsTest.**test_construct_invalid_cart_comm**()

mpi_array.comms_test.CartLocaleCommsTest.test_construct_invalid_dims

CartLocaleCommsTest.**test_construct_invalid_dims**()

mpi_array.comms_test.CartLocaleCommsTest.test_construct_no_shared

CartLocaleCommsTest.**test_construct_no_shared**()

mpi_array.comms_test.CartLocaleCommsTest.test_construct_shared

CartLocaleCommsTest.**test_construct_shared**()

Attributes

longMessage
maxDiff

mpi_array.comms_test.CartLocaleCommsTest.longMessage

CartLocaleCommsTest.**longMessage = True**

mpi_array.comms_test.CartLocaleCommsTest.maxDiff

CartLocaleCommsTest.**maxDiff = 640**

mpi_array.comms_test.CreateDistributionTest

class mpi_array.comms_test.**CreateDistributionTest**(*methodName='runTest'*)

Bases: *mpi_array.unittest.TestCase*

Tests for *mpi_array.comms.create_distribution()*.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.
check_is_single_locale_distribution(distrib)	Asserts for checking that the *distrib* `Distribution` is single-locale.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	

Continued on next page

Table 1.100 – continued from previous page

run([result])	
setUp()	Hook method for setting up the test fixture before exercising it.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_check_distrib_type()	
test_check_locale_type()	
test_create_distribution_single_locale()	Tests for mpi_array.comms.create_distribution().
test_create_distribution_slab()	Tests for mpi_array.comms.create_distribution().
test_create_locale_comms_invalid_args()	Test that mpi_array.comms.create_locale_comms() raises exception for invalid arguments.
test_create_single_locale_distribution()	Tests for mpi_array.comms.create_single_locale_distribution().

mpi_array.comms_test.CreateDistributionTest.__init__

CreateDistributionTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.comms_test.CreateDistributionTest.addCleanup

CreateDistributionTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.comms_test.CreateDistributionTest.addTypeEqualityFunc

CreateDistributionTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.comms_test.CreateDistributionTest.assertArraySplitEqual

`CreateDistributionTest.`**`assertArraySplitEqual`**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.comms_test.CreateDistributionTest.check_is_single_locale_distribution

`CreateDistributionTest.`**`check_is_single_locale_distribution`**(*distrib*)

Asserts for checking that the *distrib* `Distribution` is single-locale.

mpi_array.comms_test.CreateDistributionTest.countTestCases

`CreateDistributionTest.`**`countTestCases`**()

mpi_array.comms_test.CreateDistributionTest.debug

`CreateDistributionTest.`**`debug`**()

Run the test without collecting errors in a TestResult

mpi_array.comms_test.CreateDistributionTest.defaultTestResult

`CreateDistributionTest.`**`defaultTestResult`**()

mpi_array.comms_test.CreateDistributionTest.doCleanups

`CreateDistributionTest.`**`doCleanups`**()

Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.comms_test.CreateDistributionTest.id

`CreateDistributionTest.`**`id`**()

mpi_array.comms_test.CreateDistributionTest.run

CreateDistributionTest.**run**(*result=None*)

mpi_array.comms_test.CreateDistributionTest.setUp

CreateDistributionTest.**setUp**()
Hook method for setting up the test fixture before exercising it.

mpi_array.comms_test.CreateDistributionTest.setUpClass

CreateDistributionTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.comms_test.CreateDistributionTest.shortDescription

CreateDistributionTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.comms_test.CreateDistributionTest.skipTest

CreateDistributionTest.**skipTest**(*reason*)
Skip this test.

mpi_array.comms_test.CreateDistributionTest.subTest

CreateDistributionTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.comms_test.CreateDistributionTest.tearDown

CreateDistributionTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.comms_test.CreateDistributionTest.tearDownClass

CreateDistributionTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.comms_test.CreateDistributionTest.test_check_distrib_type

CreateDistributionTest.**test_check_distrib_type**()

mpi_array.comms_test.CreateDistributionTest.test_check_locale_type

CreateDistributionTest.**test_check_locale_type**()

mpi_array.comms_test.CreateDistributionTest.test_create_distribution_single_locale

CreateDistributionTest.**test_create_distribution_single_locale**()
Tests for *mpi_array.comms.create_distribution()*.

mpi_array.comms_test.CreateDistributionTest.test_create_distribution_slab

CreateDistributionTest.**test_create_distribution_slab**()
Tests for *mpi_array.comms.create_distribution()*.

mpi_array.comms_test.CreateDistributionTest.test_create_locale_comms_invalid_args

CreateDistributionTest.**test_create_locale_comms_invalid_args**()
Test that *mpi_array.comms.create_locale_comms()* raises exception for invalid arguments.

mpi_array.comms_test.CreateDistributionTest.test_create_single_locale_distribution

CreateDistributionTest.**test_create_single_locale_distribution**()
Tests for *mpi_array.comms.create_single_locale_distribution()*.

Attributes

longMessage
maxDiff

mpi_array.comms_test.CreateDistributionTest.longMessage

CreateDistributionTest.**longMessage** = True

mpi_array.comms_test.CreateDistributionTest.maxDiff

CreateDistributionTest.**maxDiff** = 640

1.16 The mpi_array.distribution Module

Apportionment of arrays over locales.

1.16.1 Classes

GlobaleExtent([slice, start, stop, halo, struct])	Indexing extent for an entire globale array.
HaloSubExtent([globale_extent, slice, halo, ...])	Indexing extent for single region of a larger globale extent.
LocaleExtent([peer_rank, inter_locale_rank, ...])	Indexing extent for single region of array residing on a locale.
CartLocaleExtent([peer_rank, ...])	Indexing extents for single tile of cartesian domain distribution.
Distribution(globale_extent, locale_extents)	Describes the apportionment of array extents amongst locales.
ClonedDistribution(globale_extent, num_locales)	Distribution where entire globale extent elements occur on every locale.
SingleLocaleDistribution(globale_extent, ...)	Distribution where entire globale extent elements occur on just a single locale.
BlockPartition(globale_extent, dims, ...[, ...])	Block partition of an array (shape) over locales.

mpi_array.distribution.GlobaleExtent

class mpi_array.distribution.**GlobaleExtent**(*slice=None*, *start=None*, *stop=None*, *halo=None*, *struct=None*)

Bases: mpi_array.indexing.HaloIndexingExtent

Indexing extent for an entire globale array.

Methods

__init__([slice, start, stop, halo, struct])	Construct.
calc_intersection(other)	Returns the indexing extent which is the intersection of this extent with the *other* extent.
calc_intersection_split(other)	Returns `(leftovers, intersection)` pair, where `intersection` is the `IndexingExtent` object (possibly `None`) indicating the intersection of this (*self*) extent with the `other` extent and `leftovers` is a list of `IndexingExtent` objects indicating regions of `self` which do not intersect with the `other` extent.
create_struct_dtype_from_ndim(ndim)	Creates a `numpy.dtype` structure for holding start and stop indices.
create_struct_instance(ndim)	Creates a struct instance with `numpy.dtype` `self.struct_dtype_dict[ndim]`.
get_struct_dtype(ndim)	
get_struct_dtype_from_ndim(ndim)	
globale_to_locale_extent_h(gext)	Return `gext` converted to locale index.
globale_to_locale_h(gidx)	Convert globale array index to locale array index.
globale_to_locale_n(gidx)	Convert globale array index to locale array index.
globale_to_locale_slice_h(gslice)	Return `gslice` converted to locale slice.
globale_to_locale_slice_n(gslice)	Return `gslice` converted to locale slice.
locale_to_globale_extent_h(lext)	Return `lext` converted to globale index.
locale_to_globale_h(lidx)	Convert locale array index to globale array index.
locale_to_globale_n(lidx)	Convert locale array index to globale array index.
locale_to_globale_slice_h(lslice)	Return `lslice` converted to globale slice.
locale_to_globale_slice_n(lslice)	Return `lslice` converted to globale slice.

Continued on next page

Table 1.103 – continued from previous page

split(a, index)	Split this extent into two extents by cutting along axis *a* at index *index*.
to_slice()	Same as *to_slice_n()*.
to_slice_h()	Returns "tuple of slice" equivalent of this indexing extent including halo.
to_slice_n()	Returns "tuple of slice" equivalent of this indexing extent without halo ("no halo").
to_tuple()	Convert this instance to a tuple which can be passed to constructor (or used as a dict key).

mpi_array.distribution.GlobaleExtent.__init__

GlobaleExtent.**__init__**(*slice=None*, *start=None*, *stop=None*, *halo=None*, *struct=None*)

Construct.

Parameters

- **slice** (sequence of slice) – Per axis start and stop indices defining the extent (**not including ghost elements**).
- **start** (sequence of int) – Per axis *start* indices defining the start of extent (**not including ghost elements**).
- **stop** (sequence of int) – Per axis *stop* indices defining the extent (**not including ghost elements**).
- **halo** ((len(*slice*), 2) shaped array of int) – A (len(self.start), 2) shaped array of int indicating the per-axis number of outer ghost elements. halo[:, 0] is the number of elements on the low-index *side* and halo[:,1] is the number of elements on the high-index *side*.

mpi_array.distribution.GlobaleExtent.calc_intersection

GlobaleExtent.**calc_intersection**(*other*)

Returns the indexing extent which is the intersection of this extent with the *other* extent.

Parameters **other** (IndexingExtent) – Perform intersection calculation using this extent.

Return type IndexingExtent

Returns None if the extents do not intersect, otherwise returns the extent of overlapping indices.

mpi_array.distribution.GlobaleExtent.calc_intersection_split

GlobaleExtent.**calc_intersection_split**(*other*)

Returns (leftovers, intersection) pair, where intersection is the IndexingExtent object (possibly None) indicating the intersection of this (*self*) extent with the other extent and leftovers is a list of IndexingExtent objects indicating regions of self which do not intersect with the other extent.

Parameters **other** (IndexingExtent) – Perform intersection calculation using this extent.

Return type tuple

Returns (leftovers, intersection) pair.

mpi_array.distribution.GlobaleExtent.create_struct_dtype_from_ndim

GlobaleExtent.**create_struct_dtype_from_ndim**(*ndim*)
Creates a numpy.dtype structure for holding start and stop indices.

Return type numpy.dtype

Returns numpy.dtype with "start" and "stop" multi-index fields of dimension *ndim*.

mpi_array.distribution.GlobaleExtent.create_struct_instance

GlobaleExtent.**create_struct_instance**(*ndim*)
Creates a struct instance with numpy.dtype self.struct_dtype_dict[ndim].

Return type struct

Returns A struct.

mpi_array.distribution.GlobaleExtent.get_struct_dtype

GlobaleExtent.**get_struct_dtype**(*ndim*)

mpi_array.distribution.GlobaleExtent.get_struct_dtype_from_ndim

GlobaleExtent.**get_struct_dtype_from_ndim**(*ndim*)

mpi_array.distribution.GlobaleExtent.globale_to_locale_extent_h

GlobaleExtent.**globale_to_locale_extent_h**(*gext*)
Return gext converted to locale index.

mpi_array.distribution.GlobaleExtent.globale_to_locale_h

GlobaleExtent.**globale_to_locale_h**(*gidx*)
Convert globale array index to locale array index.

Parameters **gidx** (sequence of int) – Globale index.

Return type numpy.ndarray

Returns Locale index.

mpi_array.distribution.GlobaleExtent.globale_to_locale_n

GlobaleExtent.**globale_to_locale_n**(*gidx*)
Convert globale array index to locale array index.

Parameters **gidx** (sequence of int) – Globale index.

Return type numpy.ndarray

Returns Locale index.

mpi_array.distribution.GlobaleExtent.globale_to_locale_slice_h

GlobaleExtent.**globale_to_locale_slice_h**(*gslice*)
Return gslice converted to locale slice.

mpi_array.distribution.GlobaleExtent.globale_to_locale_slice_n

GlobaleExtent.**globale_to_locale_slice_n**(*gslice*)
Return gslice converted to locale slice.

mpi_array.distribution.GlobaleExtent.locale_to_globale_extent_h

GlobaleExtent.**locale_to_globale_extent_h**(*lext*)
Return lext converted to globale index.

mpi_array.distribution.GlobaleExtent.locale_to_globale_h

GlobaleExtent.**locale_to_globale_h**(*lidx*)
Convert locale array index to globale array index.

Parameters **lidx** (sequence of int) – Locale index.

Return type numpy.ndarray

Returns Globale index.

mpi_array.distribution.GlobaleExtent.locale_to_globale_n

GlobaleExtent.**locale_to_globale_n**(*lidx*)
Convert locale array index to globale array index.

Parameters **lidx** (sequence of int) – Locale index.

Return type numpy.ndarray

Returns Globale index.

mpi_array.distribution.GlobaleExtent.locale_to_globale_slice_h

GlobaleExtent.**locale_to_globale_slice_h**(*lslice*)
Return lslice converted to globale slice.

mpi_array.distribution.GlobaleExtent.locale_to_globale_slice_n

GlobaleExtent.**locale_to_globale_slice_n**(*lslice*)
Return lslice converted to globale slice.

mpi_array.distribution.GlobaleExtent.split

GlobaleExtent.**split**(*a*, *index*)
Split this extent into two extents by cutting along axis `a` at index `index`.

Parameters

- **a** (int) – Cut along this axis.
- **index** (int) – Location of cut.

Return type tuple

Returns A `(lo, hi)` pair.

mpi_array.distribution.GlobaleExtent.to_slice

GlobaleExtent.**to_slice**()
Same as *to_slice_n()*.

mpi_array.distribution.GlobaleExtent.to_slice_h

GlobaleExtent.**to_slice_h**()
Returns "tuple of slice" equivalent of this indexing extent including halo.

Return type tuple of slice elements

Returns Tuple of slice equivalent to this indexing extent including halo.

mpi_array.distribution.GlobaleExtent.to_slice_n

GlobaleExtent.**to_slice_n**()
Returns "tuple of slice" equivalent of this indexing extent without halo ("no halo").

Return type tuple of slice elements

Returns Tuple of slice equivalent to this no-halo indexing extent.

mpi_array.distribution.GlobaleExtent.to_tuple

GlobaleExtent.**to_tuple**()
Convert this instance to a tuple which can be passed to constructor (or used as a dict key).

Return type tuple

Returns The tuple representation of this object.

Attributes

HALO
HALO_STR
HI

Continued on next page

Table 1.104 – continued from previous page

LO	
START	
START_N	
START_N_STR	
START_STR	
STOP	
STOP_N	
STOP_N_STR	
STOP_STR	
halo	A `(len(self.start), 2)` shaped array of `int` indicating the per-axis number of outer ghost elements.
ndim	Dimension of indexing.
shape	Same as *shape_n*.
shape_h	The shape of the tile with "halo" elements.
shape_n	The shape of the tile without "halo" elements ("no halo").
size_h	Integer indicating the number of elements in this extent including halo.
size_n	Integer indicating the number of elements in this extent without halo ("no halo")
start	Same as *start_n*.
start_h	The start index of the tile with "halo" elements.
start_n	The start index of the tile without "halo" elements ("no halo").
stop	Same as *stop_n*.
stop_h	The stop index of the tile with "halo" elements.
stop_n	The stop index of the tile without "halo" elements ("no halo").
struct_dtype_dict	

mpi_array.distribution.GlobaleExtent.HALO

GlobaleExtent.**HALO** = **2**

mpi_array.distribution.GlobaleExtent.HALO_STR

GlobaleExtent.**HALO_STR** = **'halo'**

mpi_array.distribution.GlobaleExtent.HI

GlobaleExtent.**HI** = **1**

mpi_array.distribution.GlobaleExtent.LO

GlobaleExtent.**LO** = **0**

mpi_array.distribution.GlobaleExtent.START

GlobaleExtent.**START = 0**

mpi_array.distribution.GlobaleExtent.START_N

GlobaleExtent.**START_N = 0**

mpi_array.distribution.GlobaleExtent.START_N_STR

GlobaleExtent.**START_N_STR = 'start'**

mpi_array.distribution.GlobaleExtent.START_STR

GlobaleExtent.**START_STR = 'start'**

mpi_array.distribution.GlobaleExtent.STOP

GlobaleExtent.**STOP = 1**

mpi_array.distribution.GlobaleExtent.STOP_N

GlobaleExtent.**STOP_N = 1**

mpi_array.distribution.GlobaleExtent.STOP_N_STR

GlobaleExtent.**STOP_N_STR = 'stop'**

mpi_array.distribution.GlobaleExtent.STOP_STR

GlobaleExtent.**STOP_STR = 'stop'**

mpi_array.distribution.GlobaleExtent.halo

GlobaleExtent.**halo**
: A `(len(self.start), 2)` shaped array of `int` indicating the per-axis number of outer ghost elements. `halo[:,0]` is the number of elements on the low-index *side* and `halo[:,1]` is the number of elements on the high-index *side*.

mpi_array.distribution.GlobaleExtent.ndim

GlobaleExtent.**ndim**
: Dimension of indexing.

mpi_array.distribution.GlobaleExtent.shape

GlobaleExtent.**shape**
Same as *shape_n*.

mpi_array.distribution.GlobaleExtent.shape_h

GlobaleExtent.**shape_h**
The shape of the tile with "halo" elements.

mpi_array.distribution.GlobaleExtent.shape_n

GlobaleExtent.**shape_n**
The shape of the tile without "halo" elements ("no halo").

mpi_array.distribution.GlobaleExtent.size_h

GlobaleExtent.**size_h**
Integer indicating the number of elements in this extent including halo.

mpi_array.distribution.GlobaleExtent.size_n

GlobaleExtent.**size_n**
Integer indicating the number of elements in this extent without halo ("no halo")

mpi_array.distribution.GlobaleExtent.start

GlobaleExtent.**start**
Same as *start_n*.

mpi_array.distribution.GlobaleExtent.start_h

GlobaleExtent.**start_h**
The start index of the tile with "halo" elements.

mpi_array.distribution.GlobaleExtent.start_n

GlobaleExtent.**start_n**
The start index of the tile without "halo" elements ("no halo").

mpi_array.distribution.GlobaleExtent.stop

GlobaleExtent.**stop**
Same as *stop_n*.

mpi_array.distribution.GlobaleExtent.stop_h

GlobaleExtent.**stop_h**
The stop index of the tile with "halo" elements.

mpi_array.distribution.GlobaleExtent.stop_n

GlobaleExtent.**stop_n**
The stop index of the tile without "halo" elements ("no halo").

mpi_array.distribution.GlobaleExtent.struct_dtype_dict

GlobaleExtent.**struct_dtype_dict = defaultdict(<function HaloIndexingExtent.<lambda>>, {})**

mpi_array.distribution.HaloSubExtent

class mpi_array.distribution.**HaloSubExtent**(*globale_extent=None*, *slice=None*, *halo=0*, *start=None*, *stop=None*, *struct=None*)
Bases: *mpi_array.indexing.HaloIndexingExtent*

Indexing extent for single region of a larger globale extent. Simply over-rides construction to trim the halo to the the *globale_extent* (with halo) bounds.

Methods

__init__([globale_extent, slice, halo, ...])	Construct.
calc_intersection(other)	Returns the indexing extent which is the intersection of this extent with the *other* extent.
calc_intersection_split(other)	Returns `(leftovers, intersection)` pair, where `intersection` is the `IndexingExtent` object (possibly `None`) indicating the intersection of this (*self*) extent with the `other` extent and `leftovers` is a list of `IndexingExtent` objects indicating regions of `self` which do not intersect with the `other` extent.
create_struct_dtype_from_ndim(ndim)	Creates a `numpy.dtype` structure for holding start and stop indices.
create_struct_instance(ndim)	Creates a struct instance with `numpy.dtype` `self.struct_dtype_dict[ndim]`.
get_struct_dtype(ndim)	
get_struct_dtype_from_ndim(ndim)	
globale_to_locale_extent_h(gext)	Return `gext` converted to locale index.
globale_to_locale_h(gidx)	Convert globale array index to locale array index.
globale_to_locale_n(gidx)	Convert globale array index to locale array index.
globale_to_locale_slice_h(gslice)	Return `gslice` converted to locale slice.
globale_to_locale_slice_n(gslice)	Return `gslice` converted to locale slice.
locale_to_globale_extent_h(lext)	Return `lext` converted to globale index.
locale_to_globale_h(lidx)	Convert locale array index to globale array index.

Continued on next page

Table 1.105 – continued from previous page

locale_to_globale_n(lidx)	Convert locale array index to globale array index.
locale_to_globale_slice_h(lslice)	Return `lslice` converted to globale slice.
locale_to_globale_slice_n(lslice)	Return `lslice` converted to globale slice.
split(a, index)	Split this extent into two extents by cutting along axis *a* at index *index*.
to_slice()	Same as *to_slice_n()*.
to_slice_h()	Returns "`tuple` of `slice`" equivalent of this indexing extent including halo.
to_slice_n()	Returns "`tuple` of `slice`" equivalent of this indexing extent without halo ("no halo").
to_tuple()	Convert this instance to a `tuple` which can be passed to constructor (or used as a `dict` key).

mpi_array.distribution.HaloSubExtent.__init__

`HaloSubExtent.`**`__init__`**(*globale_extent=None*, *slice=None*, *halo=0*, *start=None*, *stop=None*, *struct=None*)

Construct. Takes care of trimming the halo of this extent so that this extent does not stray outside the halo region of the *globale_extent*

Parameters

- **globale_extent** (*GlobaleExtent*) – The indexing extent of the entire array.
- **slice** (sequence of `slice`) – Per-axis start and stop indices (**not including ghost elements**).
- **halo** (`(len(`*split*`), 2)` shaped array of `int`) – Desired halo, a `(len(self.start), 2)` shaped array of `int` indicating the per-axis number of outer ghost elements. `halo[:,0]` is the number of ghost elements on the low-index *side* and `halo[:,1]` is the number of ghost elements on the high-index *side*. **Note**: that the halo will be truncated so that this halo extent does not extend beyond the halo *globale_extent*.
- **start** (sequence of `slice`) – Per-axis start indices (**not including ghost elements**).
- **stop** (sequence of `slice`) – Per-axis stop indices (**not including ghost elements**).

mpi_array.distribution.HaloSubExtent.calc_intersection

`HaloSubExtent.`**`calc_intersection`**(*other*)

Returns the indexing extent which is the intersection of this extent with the *other* extent.

Parameters **other** (`IndexingExtent`) – Perform intersection calculation using this extent.

Return type `IndexingExtent`

Returns `None` if the extents do not intersect, otherwise returns the extent of overlapping indices.

mpi_array.distribution.HaloSubExtent.calc_intersection_split

`HaloSubExtent.`**`calc_intersection_split`**(*other*)

Returns `(leftovers, intersection)` pair, where `intersection` is the `IndexingExtent` object (possibly `None`) indicating the intersection of this (*self*) extent with the `other` extent and `leftovers` is a list of `IndexingExtent` objects indicating regions of `self` which do not intersect with the `other` extent.

Parameters **other** (IndexingExtent) – Perform intersection calculation using this extent.

Return type tuple

Returns (leftovers, intersection) pair.

mpi_array.distribution.HaloSubExtent.create_struct_dtype_from_ndim

HaloSubExtent.**create_struct_dtype_from_ndim**(*ndim*)

Creates a numpy.dtype structure for holding start and stop indices.

Return type numpy.dtype

Returns numpy.dtype with "start" and "stop" multi-index fields of dimension *ndim*.

mpi_array.distribution.HaloSubExtent.create_struct_instance

HaloSubExtent.**create_struct_instance**(*ndim*)

Creates a struct instance with numpy.dtype self.struct_dtype_dict[ndim].

Return type struct

Returns A struct.

mpi_array.distribution.HaloSubExtent.get_struct_dtype

HaloSubExtent.**get_struct_dtype**(*ndim*)

mpi_array.distribution.HaloSubExtent.get_struct_dtype_from_ndim

HaloSubExtent.**get_struct_dtype_from_ndim**(*ndim*)

mpi_array.distribution.HaloSubExtent.globale_to_locale_extent_h

HaloSubExtent.**globale_to_locale_extent_h**(*gext*)

Return gext converted to locale index.

mpi_array.distribution.HaloSubExtent.globale_to_locale_h

HaloSubExtent.**globale_to_locale_h**(*gidx*)

Convert globale array index to locale array index.

Parameters **gidx** (sequence of int) – Globale index.

Return type numpy.ndarray

Returns Locale index.

mpi_array.distribution.HaloSubExtent.globale_to_locale_n

HaloSubExtent.**globale_to_locale_n**(*gidx*)
Convert globale array index to locale array index.

Parameters **gidx** (sequence of int) – Globale index.

Return type numpy.ndarray

Returns Locale index.

mpi_array.distribution.HaloSubExtent.globale_to_locale_slice_h

HaloSubExtent.**globale_to_locale_slice_h**(*gslice*)
Return gslice converted to locale slice.

mpi_array.distribution.HaloSubExtent.globale_to_locale_slice_n

HaloSubExtent.**globale_to_locale_slice_n**(*gslice*)
Return gslice converted to locale slice.

mpi_array.distribution.HaloSubExtent.locale_to_globale_extent_h

HaloSubExtent.**locale_to_globale_extent_h**(*lext*)
Return lext converted to globale index.

mpi_array.distribution.HaloSubExtent.locale_to_globale_h

HaloSubExtent.**locale_to_globale_h**(*lidx*)
Convert locale array index to globale array index.

Parameters **lidx** (sequence of int) – Locale index.

Return type numpy.ndarray

Returns Globale index.

mpi_array.distribution.HaloSubExtent.locale_to_globale_n

HaloSubExtent.**locale_to_globale_n**(*lidx*)
Convert locale array index to globale array index.

Parameters **lidx** (sequence of int) – Locale index.

Return type numpy.ndarray

Returns Globale index.

mpi_array.distribution.HaloSubExtent.locale_to_globale_slice_h

HaloSubExtent.**locale_to_globale_slice_h**(*lslice*)
Return lslice converted to globale slice.

mpi_array.distribution.HaloSubExtent.locale_to_globale_slice_n

HaloSubExtent.**locale_to_globale_slice_n**(*lslice*)
Return `lslice` converted to globale slice.

mpi_array.distribution.HaloSubExtent.split

HaloSubExtent.**split**(*a*, *index*)
Split this extent into two extents by cutting along axis `a` at index *index*.

Parameters

- **a** (`int`) – Cut along this axis.
- **index** (`int`) – Location of cut.

Return type `tuple`

Returns A `(lo, hi)` pair.

mpi_array.distribution.HaloSubExtent.to_slice

HaloSubExtent.**to_slice**()
Same as *to_slice_n()*.

mpi_array.distribution.HaloSubExtent.to_slice_h

HaloSubExtent.**to_slice_h**()
Returns "`tuple` of `slice`" equivalent of this indexing extent including halo.

Return type `tuple` of `slice` elements

Returns Tuple of slice equivalent to this indexing extent including halo.

mpi_array.distribution.HaloSubExtent.to_slice_n

HaloSubExtent.**to_slice_n**()
Returns "`tuple` of `slice`" equivalent of this indexing extent without halo ("no halo").

Return type `tuple` of `slice` elements

Returns Tuple of slice equivalent to this no-halo indexing extent.

mpi_array.distribution.HaloSubExtent.to_tuple

HaloSubExtent.**to_tuple**()
Convert this instance to a `tuple` which can be passed to constructor (or used as a `dict` key).

Return type `tuple`

Returns The `tuple` representation of this object.

Attributes

HALO	
HALO_STR	
HI	
LO	
START	
START_N	
START_N_STR	
START_STR	
STOP	
STOP_N	
STOP_N_STR	
STOP_STR	
halo	A `(len(self.start), 2)` shaped array of `int` indicating the per-axis number of outer ghost elements.
ndim	Dimension of indexing.
shape	Same as *shape_n*.
shape_h	The shape of the tile with "halo" elements.
shape_n	The shape of the tile without "halo" elements ("no halo").
size_h	Integer indicating the number of elements in this extent including halo.
size_n	Integer indicating the number of elements in this extent without halo ("no halo")
start	Same as *start_n*.
start_h	The start index of the tile with "halo" elements.
start_n	The start index of the tile without "halo" elements ("no halo").
stop	Same as *stop_n*.
stop_h	The stop index of the tile with "halo" elements.
stop_n	The stop index of the tile without "halo" elements ("no halo").
struct_dtype_dict	

mpi_array.distribution.HaloSubExtent.HALO

HaloSubExtent.**HALO = 2**

mpi_array.distribution.HaloSubExtent.HALO_STR

HaloSubExtent.**HALO_STR = 'halo'**

mpi_array.distribution.HaloSubExtent.HI

HaloSubExtent.**HI = 1**

mpi_array.distribution.HaloSubExtent.LO

HaloSubExtent.**LO = 0**

mpi_array.distribution.HaloSubExtent.START

HaloSubExtent.**START** = 0

mpi_array.distribution.HaloSubExtent.START_N

HaloSubExtent.**START_N** = 0

mpi_array.distribution.HaloSubExtent.START_N_STR

HaloSubExtent.**START_N_STR** = 'start'

mpi_array.distribution.HaloSubExtent.START_STR

HaloSubExtent.**START_STR** = 'start'

mpi_array.distribution.HaloSubExtent.STOP

HaloSubExtent.**STOP** = 1

mpi_array.distribution.HaloSubExtent.STOP_N

HaloSubExtent.**STOP_N** = 1

mpi_array.distribution.HaloSubExtent.STOP_N_STR

HaloSubExtent.**STOP_N_STR** = 'stop'

mpi_array.distribution.HaloSubExtent.STOP_STR

HaloSubExtent.**STOP_STR** = 'stop'

mpi_array.distribution.HaloSubExtent.halo

HaloSubExtent.**halo**
A `(len(self.start), 2)` shaped array of `int` indicating the per-axis number of outer ghost elements. `halo[:,0]` is the number of elements on the low-index *side* and `halo[:,1]` is the number of elements on the high-index *side*.

mpi_array.distribution.HaloSubExtent.ndim

HaloSubExtent.**ndim**
Dimension of indexing.

mpi_array.distribution.HaloSubExtent.shape

HaloSubExtent.**shape**
Same as *shape_n*.

mpi_array.distribution.HaloSubExtent.shape_h

HaloSubExtent.**shape_h**
The shape of the tile with "halo" elements.

mpi_array.distribution.HaloSubExtent.shape_n

HaloSubExtent.**shape_n**
The shape of the tile without "halo" elements ("no halo").

mpi_array.distribution.HaloSubExtent.size_h

HaloSubExtent.**size_h**
Integer indicating the number of elements in this extent including halo.

mpi_array.distribution.HaloSubExtent.size_n

HaloSubExtent.**size_n**
Integer indicating the number of elements in this extent without halo ("no halo")

mpi_array.distribution.HaloSubExtent.start

HaloSubExtent.**start**
Same as *start_n*.

mpi_array.distribution.HaloSubExtent.start_h

HaloSubExtent.**start_h**
The start index of the tile with "halo" elements.

mpi_array.distribution.HaloSubExtent.start_n

HaloSubExtent.**start_n**
The start index of the tile without "halo" elements ("no halo").

mpi_array.distribution.HaloSubExtent.stop

HaloSubExtent.**stop**
Same as *stop_n*.

mpi_array.distribution.HaloSubExtent.stop_h

HaloSubExtent.**stop_h**
The stop index of the tile with "halo" elements.

mpi_array.distribution.HaloSubExtent.stop_n

HaloSubExtent.**stop_n**
The stop index of the tile without "halo" elements ("no halo").

mpi_array.distribution.HaloSubExtent.struct_dtype_dict

HaloSubExtent.**struct_dtype_dict = defaultdict(<function HaloIndexingExtent.<lambda>>, {})**

mpi_array.distribution.LocaleExtent

class mpi_array.distribution.**LocaleExtent**(*peer_rank=None*, *inter_locale_rank=None*, *globale_extent=None*, *slice=None*, *halo=0*, *start=None*, *stop=None*, *struct=None*)
Bases: *mpi_array.distribution.HaloSubExtent*

Indexing extent for single region of array residing on a locale. Extends *HaloSubExtent* by storing additional {peer_rank} and *inter_locale_rank* rank integers indicating the process responsible for exchanging the data to/from this extent.

Methods

__init__([peer_rank, inter_locale_rank, ...])	Construct.
calc_intersection(other)	Returns the indexing extent which is the intersection of this extent with the *other* extent.
calc_intersection_split(other)	Returns (leftovers, intersection) pair, where intersection is the IndexingExtent object (possibly None) indicating the intersection of this (*self*) extent with the other extent and leftovers is a list of IndexingExtent objects indicating regions of self which do not intersect with the other extent.
create_struct_dtype_from_ndim(ndim)	Creates a numpy.dtype structure for holding start and stop indices.
create_struct_instance(ndim)	Creates a struct instance with numpy.dtype self.struct_dtype_dict[ndim].
get_struct_dtype(ndim)	
get_struct_dtype_from_ndim(ndim)	
globale_to_locale_extent_h(gext)	Return gext converted to locale index.
globale_to_locale_h(gidx)	Convert globale array index to locale array index.
globale_to_locale_n(gidx)	Convert globale array index to locale array index.
globale_to_locale_slice_h(gslice)	Return gslice converted to locale slice.
globale_to_locale_slice_n(gslice)	Return gslice converted to locale slice.

Continued on next page

Table 1.107 – continued from previous page

halo_slab_extent(axis, dir)	Returns indexing extent of the halo *slab* for specified axis.
locale_to_globale_extent_h(lext)	Return `lext` converted to globale index.
locale_to_globale_h(lidx)	Convert locale array index to globale array index.
locale_to_globale_n(lidx)	Convert locale array index to globale array index.
locale_to_globale_slice_h(lslice)	Return `lslice` converted to globale slice.
locale_to_globale_slice_n(lslice)	Return `lslice` converted to globale slice.
no_halo_extent(axis)	Returns the indexing extent identical to this extent, except has the halo trimmed from the axis specified by *axis*.
split(a, index)	Split this extent into two extents by cutting along axis *a* at index *index*.
to_slice()	Same as *to_slice_n()*.
to_slice_h()	Returns "`tuple` of `slice`" equivalent of this indexing extent including halo.
to_slice_n()	Returns "`tuple` of `slice`" equivalent of this indexing extent without halo ("no halo").
to_tuple()	Convert this instance to a `tuple` which can be passed to constructor (or used as a `dict` key).

mpi_array.distribution.LocaleExtent.__init__

LocaleExtent.**__init__**(*peer_rank=None*, *inter_locale_rank=None*, *globale_extent=None*, *slice=None*, *halo=0*, *start=None*, *stop=None*, *struct=None*)

Construct.

Parameters

- **peer_rank** (`int`) – Rank of MPI process in `peer_comm` communicator which corresponds to *inter_locale_rank* rank of *inter_locale_comm*.
- **inter_locale_rank** (`int`) – Rank of MPI process in `inter_locale_comm` communicator.
- **globale_extent** (*GlobaleExtent*) – The indexing extent of the entire array.
- **slice** (sequence of `slice`) – Per-axis start and stop indices (**not including ghost elements**).
- **halo** (`(len(`*split*`), 2)` shaped array of `int`) – Desired halo, a `(len(self.start), 2)` shaped array of `int` indicating the per-axis number of outer ghost elements. `halo[:,0]` is the number of ghost elements on the low-index *side* and `halo[:,1]` is the number of ghost elements on the high-index *side*. **Note**: that the halo will be truncated so that this halo extent does not extend beyond the halo *globale_extent*.
- **start** (sequence of `slice`) – Per-axis start indices (**not including ghost elements**).
- **stop** (sequence of `slice`) – Per-axis stop indices (**not including ghost elements**).

mpi_array.distribution.LocaleExtent.calc_intersection

LocaleExtent.**calc_intersection**(*other*)

Returns the indexing extent which is the intersection of this extent with the *other* extent.

Parameters **other** (`IndexingExtent`) – Perform intersection calculation using this extent.

Return type IndexingExtent

Returns None if the extents do not intersect, otherwise returns the extent of overlapping indices.

mpi_array.distribution.LocaleExtent.calc_intersection_split

LocaleExtent.**calc_intersection_split**(*other*)

Returns (leftovers, intersection) pair, where intersection is the IndexingExtent object (possibly None) indicating the intersection of this (*self*) extent with the other extent and leftovers is a list of IndexingExtent objects indicating regions of self which do not intersect with the other extent.

Parameters **other** (IndexingExtent) – Perform intersection calculation using this extent.

Return type tuple

Returns (leftovers, intersection) pair.

mpi_array.distribution.LocaleExtent.create_struct_dtype_from_ndim

static LocaleExtent.**create_struct_dtype_from_ndim**(*ndim*)

Creates a numpy.dtype structure for holding start and stop indices.

Return type numpy.dtype

Returns numpy.dtype with "start" and "stop" multi-index fields of dimension *ndim*.

mpi_array.distribution.LocaleExtent.create_struct_instance

LocaleExtent.**create_struct_instance**(*ndim*)

Creates a struct instance with numpy.dtype self.struct_dtype_dict[ndim].

Return type struct

Returns A struct.

mpi_array.distribution.LocaleExtent.get_struct_dtype

LocaleExtent.**get_struct_dtype**(*ndim*)

mpi_array.distribution.LocaleExtent.get_struct_dtype_from_ndim

LocaleExtent.**get_struct_dtype_from_ndim**(*ndim*)

mpi_array.distribution.LocaleExtent.globale_to_locale_extent_h

LocaleExtent.**globale_to_locale_extent_h**(*gext*)

Return gext converted to locale index.

mpi_array.distribution.LocaleExtent.globale_to_locale_h

LocaleExtent.**globale_to_locale_h**(*gidx*)
Convert globale array index to locale array index.

Parameters **gidx** (sequence of int) – Globale index.

Return type numpy.ndarray

Returns Locale index.

mpi_array.distribution.LocaleExtent.globale_to_locale_n

LocaleExtent.**globale_to_locale_n**(*gidx*)
Convert globale array index to locale array index.

Parameters **gidx** (sequence of int) – Globale index.

Return type numpy.ndarray

Returns Locale index.

mpi_array.distribution.LocaleExtent.globale_to_locale_slice_h

LocaleExtent.**globale_to_locale_slice_h**(*gslice*)
Return gslice converted to locale slice.

mpi_array.distribution.LocaleExtent.globale_to_locale_slice_n

LocaleExtent.**globale_to_locale_slice_n**(*gslice*)
Return gslice converted to locale slice.

mpi_array.distribution.LocaleExtent.halo_slab_extent

LocaleExtent.**halo_slab_extent**(*axis*, *dir*)
Returns indexing extent of the halo *slab* for specified axis.

Parameters

- **axis** (int) – Indexing extent of halo slab for this axis.
- **dir** (*LO* or *HI*) – Indicates low-index halo slab or high-index halo slab.

Return type IndexingExtent

Returns Indexing extent for halo slab.

Todo

Provide an example code here.

---

mpi_array.distribution.LocaleExtent.locale_to_globale_extent_h

LocaleExtent.**locale_to_globale_extent_h**(*lext*)
Return `lext` converted to globale index.

mpi_array.distribution.LocaleExtent.locale_to_globale_h

LocaleExtent.**locale_to_globale_h**(*lidx*)
Convert locale array index to globale array index.

Parameters **lidx** (sequence of `int`) – Locale index.

Return type `numpy.ndarray`

Returns Globale index.

mpi_array.distribution.LocaleExtent.locale_to_globale_n

LocaleExtent.**locale_to_globale_n**(*lidx*)
Convert locale array index to globale array index.

Parameters **lidx** (sequence of `int`) – Locale index.

Return type `numpy.ndarray`

Returns Globale index.

mpi_array.distribution.LocaleExtent.locale_to_globale_slice_h

LocaleExtent.**locale_to_globale_slice_h**(*lslice*)
Return `lslice` converted to globale slice.

mpi_array.distribution.LocaleExtent.locale_to_globale_slice_n

LocaleExtent.**locale_to_globale_slice_n**(*lslice*)
Return `lslice` converted to globale slice.

mpi_array.distribution.LocaleExtent.no_halo_extent

LocaleExtent.**no_halo_extent**(*axis*)
Returns the indexing extent identical to this extent, except has the halo trimmed from the axis specified by *`axis`*.

Parameters **axis** (`int` or sequence of `int`) – Axis (or axes) for which halo is trimmed.

Return type `IndexingExtent`

Returns Indexing extent with halo trimmed from specified axis (or axes) *`axis`*.

Todo

Provide an example code here.

---

mpi_array.distribution.LocaleExtent.split

LocaleExtent.**split**(*a*, *index*)
Split this extent into two extents by cutting along axis `a` at index `index`.

Parameters

- **a** (int) – Cut along this axis.
- **index** (int) – Location of cut.

Return type tuple

Returns A `(lo, hi)` pair.

mpi_array.distribution.LocaleExtent.to_slice

LocaleExtent.**to_slice**()
Same as *to_slice_n()*.

mpi_array.distribution.LocaleExtent.to_slice_h

LocaleExtent.**to_slice_h**()
Returns "tuple of slice" equivalent of this indexing extent including halo.

Return type tuple of slice elements

Returns Tuple of slice equivalent to this indexing extent including halo.

mpi_array.distribution.LocaleExtent.to_slice_n

LocaleExtent.**to_slice_n**()
Returns "tuple of slice" equivalent of this indexing extent without halo ("no halo").

Return type tuple of slice elements

Returns Tuple of slice equivalent to this no-halo indexing extent.

mpi_array.distribution.LocaleExtent.to_tuple

LocaleExtent.**to_tuple**()
Convert this instance to a tuple which can be passed to constructor (or used as a dict key).

Return type tuple

Returns The tuple representation of this object.

Attributes

HALO
HALO_STR
HI

Continued on next page

Table 1.108 – continued from previous page

INTER_LOCALE_RANK	
INTER_LOCALE_RANK_STR	
LO	
PEER_RANK	
PEER_RANK_STR	
START	
START_N	
START_N_STR	
START_STR	
STOP	
STOP_N	
STOP_N_STR	
STOP_STR	
halo	A `(len(self.start), 2)` shaped array of `int` indicating the per-axis number of outer ghost elements.
inter_locale_rank	An `int` indicating the rank of the process in the `inter_locale_comm` responsible for exchanging data to/from this extent.
ndim	Dimension of indexing.
peer_rank	An `int` indicating the rank of the process in the `peer_comm` communicator which corresponds to the *inter_locale_rank* in the `inter_locale_comm` communicator.
shape	Same as *shape_n*.
shape_h	The shape of the tile with "halo" elements.
shape_n	The shape of the tile without "halo" elements ("no halo").
size_h	Integer indicating the number of elements in this extent including halo.
size_n	Integer indicating the number of elements in this extent without halo ("no halo")
start	Same as *start_n*.
start_h	The start index of the tile with "halo" elements.
start_n	The start index of the tile without "halo" elements ("no halo").
stop	Same as *stop_n*.
stop_h	The stop index of the tile with "halo" elements.
stop_n	The stop index of the tile without "halo" elements ("no halo").
struct_dtype_dict	

mpi_array.distribution.LocaleExtent.HALO

`LocaleExtent.`**`HALO = 2`**

mpi_array.distribution.LocaleExtent.HALO_STR

`LocaleExtent.`**`HALO_STR = 'halo'`**

mpi_array.distribution.LocaleExtent.HI

LocaleExtent.**HI = 1**

mpi_array.distribution.LocaleExtent.INTER_LOCALE_RANK

LocaleExtent.**INTER_LOCALE_RANK = 4**

mpi_array.distribution.LocaleExtent.INTER_LOCALE_RANK_STR

LocaleExtent.**INTER_LOCALE_RANK_STR = 'inter_locale_rank'**

mpi_array.distribution.LocaleExtent.LO

LocaleExtent.**LO = 0**

mpi_array.distribution.LocaleExtent.PEER_RANK

LocaleExtent.**PEER_RANK = 3**

mpi_array.distribution.LocaleExtent.PEER_RANK_STR

LocaleExtent.**PEER_RANK_STR = 'peer_rank'**

mpi_array.distribution.LocaleExtent.START

LocaleExtent.**START = 0**

mpi_array.distribution.LocaleExtent.START_N

LocaleExtent.**START_N = 0**

mpi_array.distribution.LocaleExtent.START_N_STR

LocaleExtent.**START_N_STR = 'start'**

mpi_array.distribution.LocaleExtent.START_STR

LocaleExtent.**START_STR = 'start'**

mpi_array.distribution.LocaleExtent.STOP

LocaleExtent.**STOP = 1**

mpi_array.distribution.LocaleExtent.STOP_N

LocaleExtent.**STOP_N = 1**

mpi_array.distribution.LocaleExtent.STOP_N_STR

LocaleExtent.**STOP_N_STR = 'stop'**

mpi_array.distribution.LocaleExtent.STOP_STR

LocaleExtent.**STOP_STR = 'stop'**

mpi_array.distribution.LocaleExtent.halo

LocaleExtent.**halo**
A `(len(self.start), 2)` shaped array of `int` indicating the per-axis number of outer ghost elements. `halo[:,0]` is the number of elements on the low-index *side* and `halo[:,1]` is the number of elements on the high-index *side*.

mpi_array.distribution.LocaleExtent.inter_locale_rank

LocaleExtent.**inter_locale_rank**
An `int` indicating the rank of the process in the `inter_locale_comm` responsible for exchanging data to/from this extent.

mpi_array.distribution.LocaleExtent.ndim

LocaleExtent.**ndim**
Dimension of indexing.

mpi_array.distribution.LocaleExtent.peer_rank

LocaleExtent.**peer_rank**
An `int` indicating the rank of the process in the `peer_comm` communicator which corresponds to the *`inter_locale_rank`* in the `inter_locale_comm` communicator.

mpi_array.distribution.LocaleExtent.shape

LocaleExtent.**shape**
Same as *`shape_n`*.

mpi_array.distribution.LocaleExtent.shape_h

LocaleExtent.**shape_h**
The shape of the tile with "halo" elements.

mpi_array.distribution.LocaleExtent.shape_n

LocaleExtent.**shape_n**
The shape of the tile without "halo" elements ("no halo").

mpi_array.distribution.LocaleExtent.size_h

LocaleExtent.**size_h**
Integer indicating the number of elements in this extent including halo.

mpi_array.distribution.LocaleExtent.size_n

LocaleExtent.**size_n**
Integer indicating the number of elements in this extent without halo ("no halo")

mpi_array.distribution.LocaleExtent.start

LocaleExtent.**start**
Same as *start_n*.

mpi_array.distribution.LocaleExtent.start_h

LocaleExtent.**start_h**
The start index of the tile with "halo" elements.

mpi_array.distribution.LocaleExtent.start_n

LocaleExtent.**start_n**
The start index of the tile without "halo" elements ("no halo").

mpi_array.distribution.LocaleExtent.stop

LocaleExtent.**stop**
Same as *stop_n*.

mpi_array.distribution.LocaleExtent.stop_h

LocaleExtent.**stop_h**
The stop index of the tile with "halo" elements.

mpi_array.distribution.LocaleExtent.stop_n

LocaleExtent.**stop_n**
The stop index of the tile without "halo" elements ("no halo").

mpi_array.distribution.LocaleExtent.struct_dtype_dict

LocaleExtent.**struct_dtype_dict** = defaultdict(<function LocaleExtent.<lambda>>, {})

mpi_array.distribution.CartLocaleExtent

class mpi_array.distribution.**CartLocaleExtent**(*peer_rank=None*, *inter_locale_rank=None*, *cart_coord=None*, *cart_shape=None*, *globale_extent=None*, *slice=None*, *halo=None*, *start=None*, *stop=None*, *struct=None*)

Bases: *mpi_array.distribution.LocaleExtent*

Indexing extents for single tile of cartesian domain distribution.

Methods

__init__([peer_rank, inter_locale_rank, ...])	Construct.
calc_intersection(other)	Returns the indexing extent which is the intersection of this extent with the *other* extent.
calc_intersection_split(other)	Returns `(leftovers, intersection)` pair, where `intersection` is the `IndexingExtent` object (possibly `None`) indicating the intersection of this (*self*) extent with the `other` extent and `leftovers` is a list of `IndexingExtent` objects indicating regions of `self` which do not intersect with the `other` extent.
create_struct_dtype_from_ndim(ndim)	Creates a `numpy.dtype` structure for holding start and stop indices.
create_struct_instance(ndim)	Creates a struct instance with `numpy.dtype` `self.struct_dtype_dict[ndim]`.
get_struct_dtype(ndim)	
get_struct_dtype_from_ndim(ndim)	
globale_to_locale_extent_h(gext)	Return `gext` converted to locale index.
globale_to_locale_h(gidx)	Convert globale array index to locale array index.
globale_to_locale_n(gidx)	Convert globale array index to locale array index.
globale_to_locale_slice_h(gslice)	Return `gslice` converted to locale slice.
globale_to_locale_slice_n(gslice)	Return `gslice` converted to locale slice.
halo_slab_extent(axis, dir)	Returns indexing extent of the halo *slab* for specified axis.
locale_to_globale_extent_h(lext)	Return `lext` converted to globale index.
locale_to_globale_h(lidx)	Convert locale array index to globale array index.
locale_to_globale_n(lidx)	Convert locale array index to globale array index.
locale_to_globale_slice_h(lslice)	Return `lslice` converted to globale slice.
locale_to_globale_slice_n(lslice)	Return `lslice` converted to globale slice.
no_halo_extent(axis)	Returns the indexing extent identical to this extent, except has the halo trimmed from the axis specified by *axis*.
split(a, index)	Split this extent into two extents by cutting along axis *a* at index *index*.
to_slice()	Same as *to_slice_n()*.

Continued on next page

Table 1.109 – continued from previous page

to_slice_h()	Returns "tuple of slice" equivalent of this indexing extent including halo.
to_slice_n()	Returns "tuple of slice" equivalent of this indexing extent without halo ("no halo").
to_tuple()	Convert this instance to a tuple which can be passed to constructor (or used as a dict key).

mpi_array.distribution.CartLocaleExtent.__init__

CartLocaleExtent.**__init__**(*peer_rank=None*, *inter_locale_rank=None*, *cart_coord=None*, *cart_shape=None*, *globale_extent=None*, *slice=None*, *halo=None*, *start=None*, *stop=None*, *struct=None*)

Construct.

Parameters

- **peer_rank** (int) – Rank of MPI process in peer_comm communicator which corresponds to the *inter_locale_rank* peer_rank in the cart_comm cartesian communicator.
- **inter_locale_rank** (int) – Rank of MPI process in cart_comm cartesian communicator which corresponds to the *peer_comm* peer_rank in the peer_comm communicator.
- **cart_coord** (sequence of int) – Coordinate index (mpi4py.MPI.CartComm.Get_coordinate()) of this *LocaleExtent* in the cartesian domain distribution.
- **cart_shape** (sequence of int) – Number of *LocaleExtent* regions in each axis direction of the cartesian distribution.
- **globale_extent** (*GlobaleExtent*) – The indexing extent of the entire array.
- **slice** (sequence of slice) – Per-axis start and stop indices (**not including ghost elements**).
- **halo** ((len(*split*), 2) shaped array of int) – Desired halo, a (len(self.start), 2) shaped array of int indicating the per-axis number of outer ghost elements. halo[:,0] is the number of ghost elements on the low-index *side* and halo[:,1] is the number of ghost elements on the high-index *side*. **Note**: that the halo will be truncated so that this halo extent does not extend beyond the halo *globale_extent*.
- **start** (sequence of slice) – Per-axis start indices (**not including ghost elements**).
- **stop** (sequence of slice) – Per-axis stop indices (**not including ghost elements**).

mpi_array.distribution.CartLocaleExtent.calc_intersection

CartLocaleExtent.**calc_intersection**(*other*)

Returns the indexing extent which is the intersection of this extent with the *other* extent.

Parameters **other** (IndexingExtent) – Perform intersection calculation using this extent.

Return type IndexingExtent

Returns None if the extents do not intersect, otherwise returns the extent of overlapping indices.

mpi_array.distribution.CartLocaleExtent.calc_intersection_split

CartLocaleExtent.**calc_intersection_split**(*other*)
Returns `(leftovers, intersection)` pair, where `intersection` is the `IndexingExtent` object (possibly `None`) indicating the intersection of this (*`self`*) extent with the `other` extent and `leftovers` is a list of `IndexingExtent` objects indicating regions of `self` which do not intersect with the `other` extent.

Parameters **other** (`IndexingExtent`) – Perform intersection calculation using this extent.

Return type `tuple`

Returns `(leftovers, intersection)` pair.

mpi_array.distribution.CartLocaleExtent.create_struct_dtype_from_ndim

static CartLocaleExtent.**create_struct_dtype_from_ndim**(*ndim*)
Creates a `numpy.dtype` structure for holding start and stop indices.

Return type `numpy.dtype`

Returns `numpy.dtype` with `"start"` and `"stop"` multi-index fields of dimension *`ndim`*.

mpi_array.distribution.CartLocaleExtent.create_struct_instance

CartLocaleExtent.**create_struct_instance**(*ndim*)
Creates a struct instance with `numpy.dtype` `self.struct_dtype_dict[ndim]`.

Return type struct

Returns A struct.

mpi_array.distribution.CartLocaleExtent.get_struct_dtype

CartLocaleExtent.**get_struct_dtype**(*ndim*)

mpi_array.distribution.CartLocaleExtent.get_struct_dtype_from_ndim

CartLocaleExtent.**get_struct_dtype_from_ndim**(*ndim*)

mpi_array.distribution.CartLocaleExtent.globale_to_locale_extent_h

CartLocaleExtent.**globale_to_locale_extent_h**(*gext*)
Return `gext` converted to locale index.

mpi_array.distribution.CartLocaleExtent.globale_to_locale_h

CartLocaleExtent.**globale_to_locale_h**(*gidx*)
Convert globale array index to locale array index.

Parameters **gidx** (sequence of `int`) – Globale index.

Return type numpy.ndarray

Returns Locale index.

mpi_array.distribution.CartLocaleExtent.globale_to_locale_n

CartLocaleExtent.**globale_to_locale_n**(*gidx*)
Convert globale array index to locale array index.

Parameters **gidx** (sequence of int) – Globale index.

Return type numpy.ndarray

Returns Locale index.

mpi_array.distribution.CartLocaleExtent.globale_to_locale_slice_h

CartLocaleExtent.**globale_to_locale_slice_h**(*gslice*)
Return gslice converted to locale slice.

mpi_array.distribution.CartLocaleExtent.globale_to_locale_slice_n

CartLocaleExtent.**globale_to_locale_slice_n**(*gslice*)
Return gslice converted to locale slice.

mpi_array.distribution.CartLocaleExtent.halo_slab_extent

CartLocaleExtent.**halo_slab_extent**(*axis*, *dir*)
Returns indexing extent of the halo *slab* for specified axis.

Parameters

- **axis** (int) – Indexing extent of halo slab for this axis.
- **dir** (*LO* or *HI*) – Indicates low-index halo slab or high-index halo slab.

Return type IndexingExtent

Returns Indexing extent for halo slab.

Todo

Provide an example code here.

---

mpi_array.distribution.CartLocaleExtent.locale_to_globale_extent_h

CartLocaleExtent.**locale_to_globale_extent_h**(*lext*)
Return lext converted to globale index.

mpi_array.distribution.CartLocaleExtent.locale_to_globale_h

CartLocaleExtent.**locale_to_globale_h**(*lidx*)
Convert locale array index to globale array index.

Parameters **lidx** (sequence of int) – Locale index.

Return type numpy.ndarray

Returns Globale index.

mpi_array.distribution.CartLocaleExtent.locale_to_globale_n

CartLocaleExtent.**locale_to_globale_n**(*lidx*)
Convert locale array index to globale array index.

Parameters **lidx** (sequence of int) – Locale index.

Return type numpy.ndarray

Returns Globale index.

mpi_array.distribution.CartLocaleExtent.locale_to_globale_slice_h

CartLocaleExtent.**locale_to_globale_slice_h**(*lslice*)
Return `lslice` converted to globale slice.

mpi_array.distribution.CartLocaleExtent.locale_to_globale_slice_n

CartLocaleExtent.**locale_to_globale_slice_n**(*lslice*)
Return `lslice` converted to globale slice.

mpi_array.distribution.CartLocaleExtent.no_halo_extent

CartLocaleExtent.**no_halo_extent**(*axis*)
Returns the indexing extent identical to this extent, except has the halo trimmed from the axis specified by *axis*.

Parameters **axis** (int or sequence of int) – Axis (or axes) for which halo is trimmed.

Return type IndexingExtent

Returns Indexing extent with halo trimmed from specified axis (or axes) *axis*.

Todo

Provide an example code here.

---

mpi_array.distribution.CartLocaleExtent.split

CartLocaleExtent.**split**(*a*, *index*)
Split this extent into two extents by cutting along axis *a* at index *index*.

Parameters

- **a** (int) – Cut along this axis.
- **index** (int) – Location of cut.

Return type tuple

Returns A (lo, hi) pair.

mpi_array.distribution.CartLocaleExtent.to_slice

CartLocaleExtent.**to_slice**()
Same as *to_slice_n()*.

mpi_array.distribution.CartLocaleExtent.to_slice_h

CartLocaleExtent.**to_slice_h**()
Returns "tuple of slice" equivalent of this indexing extent including halo.

Return type tuple of slice elements

Returns Tuple of slice equivalent to this indexing extent including halo.

mpi_array.distribution.CartLocaleExtent.to_slice_n

CartLocaleExtent.**to_slice_n**()
Returns "tuple of slice" equivalent of this indexing extent without halo ("no halo").

Return type tuple of slice elements

Returns Tuple of slice equivalent to this no-halo indexing extent.

mpi_array.distribution.CartLocaleExtent.to_tuple

CartLocaleExtent.**to_tuple**()
Convert this instance to a tuple which can be passed to constructor (or used as a dict key).

Return type tuple

Returns The tuple representation of this object.

Attributes

CART_COORD
CART_COORD_STR
CART_SHAPE
CART_SHAPE_STR
HALO
HALO_STR
HI
INTER_LOCALE_RANK

Continued on next page

Table 1.110 – continued from previous page

INTER_LOCALE_RANK_STR	
LO	
PEER_RANK	
PEER_RANK_STR	
START	
START_N	
START_N_STR	
START_STR	
STOP	
STOP_N	
STOP_N_STR	
STOP_STR	
cart_coord	A `tuple` of `int` indicating the coordinate index (`mpi4py.MPI.CartComm.Get_coordinate()`) of this *LocaleExtent* in the cartesian domain distribution.
cart_rank	An `int` indicating the rank of the process in the `cart_comm` cartesian communicator which corresponds to the `{peer_rank}` rank in the `peer_comm` communicator.
cart_shape	A `tuple` of `int` indicating the number of *LocaleExtent* regions in each axis direction of the cartesian distribution.
halo	A `(len(self.start), 2)` shaped array of `int` indicating the per-axis number of outer ghost elements.
inter_locale_rank	An `int` indicating the rank of the process in the `inter_locale_comm` responsible for exchanging data to/from this extent.
ndim	Dimension of indexing.
peer_rank	An `int` indicating the rank of the process in the `peer_comm` communicator which corresponds to the *inter_locale_rank* in the `inter_locale_comm` communicator.
shape	Same as *shape_n*.
shape_h	The shape of the tile with "halo" elements.
shape_n	The shape of the tile without "halo" elements ("no halo").
size_h	Integer indicating the number of elements in this extent including halo.
size_n	Integer indicating the number of elements in this extent without halo ("no halo")
start	Same as *start_n*.
start_h	The start index of the tile with "halo" elements.
start_n	The start index of the tile without "halo" elements ("no halo").
stop	Same as *stop_n*.
stop_h	The stop index of the tile with "halo" elements.
stop_n	The stop index of the tile without "halo" elements ("no halo").
struct_dtype_dict	

mpi_array.distribution.CartLocaleExtent.CART_COORD

CartLocaleExtent.**CART_COORD = 5**

mpi_array.distribution.CartLocaleExtent.CART_COORD_STR

CartLocaleExtent.**CART_COORD_STR = 'cart_coord'**

mpi_array.distribution.CartLocaleExtent.CART_SHAPE

CartLocaleExtent.**CART_SHAPE = 6**

mpi_array.distribution.CartLocaleExtent.CART_SHAPE_STR

CartLocaleExtent.**CART_SHAPE_STR = 'cart_shape'**

mpi_array.distribution.CartLocaleExtent.HALO

CartLocaleExtent.**HALO = 2**

mpi_array.distribution.CartLocaleExtent.HALO_STR

CartLocaleExtent.**HALO_STR = 'halo'**

mpi_array.distribution.CartLocaleExtent.HI

CartLocaleExtent.**HI = 1**

mpi_array.distribution.CartLocaleExtent.INTER_LOCALE_RANK

CartLocaleExtent.**INTER_LOCALE_RANK = 4**

mpi_array.distribution.CartLocaleExtent.INTER_LOCALE_RANK_STR

CartLocaleExtent.**INTER_LOCALE_RANK_STR = 'inter_locale_rank'**

mpi_array.distribution.CartLocaleExtent.LO

CartLocaleExtent.**LO = 0**

mpi_array.distribution.CartLocaleExtent.PEER_RANK

CartLocaleExtent.**PEER_RANK = 3**

mpi_array.distribution.CartLocaleExtent.PEER_RANK_STR

CartLocaleExtent.**PEER_RANK_STR = 'peer_rank'**

mpi_array.distribution.CartLocaleExtent.START

CartLocaleExtent.**START = 0**

mpi_array.distribution.CartLocaleExtent.START_N

CartLocaleExtent.**START_N = 0**

mpi_array.distribution.CartLocaleExtent.START_N_STR

CartLocaleExtent.**START_N_STR = 'start'**

mpi_array.distribution.CartLocaleExtent.START_STR

CartLocaleExtent.**START_STR = 'start'**

mpi_array.distribution.CartLocaleExtent.STOP

CartLocaleExtent.**STOP = 1**

mpi_array.distribution.CartLocaleExtent.STOP_N

CartLocaleExtent.**STOP_N = 1**

mpi_array.distribution.CartLocaleExtent.STOP_N_STR

CartLocaleExtent.**STOP_N_STR = 'stop'**

mpi_array.distribution.CartLocaleExtent.STOP_STR

CartLocaleExtent.**STOP_STR = 'stop'**

mpi_array.distribution.CartLocaleExtent.cart_coord

CartLocaleExtent.**cart_coord**
A `tuple` of `int` indicating the coordinate index (`mpi4py.MPI.CartComm.Get_coordinate()`) of this *`LocaleExtent`* in the cartesian domain distribution.

mpi_array.distribution.CartLocaleExtent.cart_rank

`CartLocaleExtent.`**`cart_rank`**
: An `int` indicating the rank of the process in the `cart_comm` cartesian communicator which corresponds to the `{peer_rank}` rank in the `peer_comm` communicator.

mpi_array.distribution.CartLocaleExtent.cart_shape

`CartLocaleExtent.`**`cart_shape`**
: A `tuple` of `int` indicating the number of *`LocaleExtent`* regions in each axis direction of the cartesian distribution.

mpi_array.distribution.CartLocaleExtent.halo

`CartLocaleExtent.`**`halo`**
: A `(len(self.start), 2)` shaped array of `int` indicating the per-axis number of outer ghost elements. `halo[:,0]` is the number of elements on the low-index *side* and `halo[:,1]` is the number of elements on the high-index *side*.

mpi_array.distribution.CartLocaleExtent.inter_locale_rank

`CartLocaleExtent.`**`inter_locale_rank`**
: An `int` indicating the rank of the process in the `inter_locale_comm` responsible for exchanging data to/from this extent.

mpi_array.distribution.CartLocaleExtent.ndim

`CartLocaleExtent.`**`ndim`**
: Dimension of indexing.

mpi_array.distribution.CartLocaleExtent.peer_rank

`CartLocaleExtent.`**`peer_rank`**
: An `int` indicating the rank of the process in the `peer_comm` communicator which corresponds to the *`inter_locale_rank`* in the `inter_locale_comm` communicator.

mpi_array.distribution.CartLocaleExtent.shape

`CartLocaleExtent.`**`shape`**
: Same as *`shape_n`*.

mpi_array.distribution.CartLocaleExtent.shape_h

`CartLocaleExtent.`**`shape_h`**
: The shape of the tile with "halo" elements.

mpi_array.distribution.CartLocaleExtent.shape_n

CartLocaleExtent.**shape_n**
: The shape of the tile without "halo" elements ("no halo").

mpi_array.distribution.CartLocaleExtent.size_h

CartLocaleExtent.**size_h**
: Integer indicating the number of elements in this extent including halo.

mpi_array.distribution.CartLocaleExtent.size_n

CartLocaleExtent.**size_n**
: Integer indicating the number of elements in this extent without halo ("no halo")

mpi_array.distribution.CartLocaleExtent.start

CartLocaleExtent.**start**
: Same as *start_n*.

mpi_array.distribution.CartLocaleExtent.start_h

CartLocaleExtent.**start_h**
: The start index of the tile with "halo" elements.

mpi_array.distribution.CartLocaleExtent.start_n

CartLocaleExtent.**start_n**
: The start index of the tile without "halo" elements ("no halo").

mpi_array.distribution.CartLocaleExtent.stop

CartLocaleExtent.**stop**
: Same as *stop_n*.

mpi_array.distribution.CartLocaleExtent.stop_h

CartLocaleExtent.**stop_h**
: The stop index of the tile with "halo" elements.

mpi_array.distribution.CartLocaleExtent.stop_n

CartLocaleExtent.**stop_n**
: The stop index of the tile without "halo" elements ("no halo").

mpi_array.distribution.CartLocaleExtent.struct_dtype_dict

CartLocaleExtent.**struct_dtype_dict = defaultdict(<function CartLocaleExtent.<lambda>>, {})**

mpi_array.distribution.Distribution

class mpi_array.distribution.**Distribution**(*globale_extent*, *locale_extents*, *halo=0*, *globale_extent_type=<class 'mpi_array.distribution.GlobaleExtent'>*, *locale_extent_type=<class 'mpi_array.distribution.LocaleExtent'>*, *inter_locale_rank_to_peer_rank=None*)

Bases: object

Describes the apportionment of array extents amongst locales.

Methods

__init__(globale_extent, locale_extents[, ...])	Construct.
create_globale_extent(globale_extent[, halo])	Factory method for creating *GlobaleExtent* object.
create_locale_extents(struct_locale_extents)	
create_struct_locale_extents(num_locales, ndim)	
get_extent_for_rank(inter_locale_rank)	Returns extent associated with the specified rank of the inter_locale_comm communicator.
get_peer_rank(inter_locale_rank)	Returns the peer_rank rank (of peer_comm) which is is the equivalent process of the *inter_locale_rank* rank of the inter_locale_comm communicator.
initialise_struct_locale_extents(...)	
initialise_struct_locale_extents_halos(...)	Trim locale extent halos so they don't extend beyond the *globale_extent* halo.

mpi_array.distribution.Distribution.__init__

Distribution.**__init__**(*globale_extent*, *locale_extents*, *halo=0*, *globale_extent_type=<class 'mpi_array.distribution.GlobaleExtent'>*, *locale_extent_type=<class 'mpi_array.distribution.LocaleExtent'>*, *inter_locale_rank_to_peer_rank=None*)

Construct.

Parameters

- **globale_extent** (object) – Can be specified as a *sequence-of-int* shape, *sequence-of-slice* slice or a *mpi_array.indexing.IndexingExtent*. Defines the globale extent of the array.
- **locale_extents** (sequence of object) – Can be specified as a sequence of *sequence-of-slice* slices or a sequence of *mpi_array.indexing.IndexingExtent*. Defines the distribution of the globale array over locales. The element locale_extents[r] defines the extent which is to be allocated by inter_locale_comm rank r.

- **halo** (int, sequence of int or (ndim, 2) shaped array) – Locale array halo (ghost elements).
- **globale_extent_type** (object) – The class/type for *globale_extent*.
- **locale_extent_type** (object) – The class/type for elements of *locale_extents*.
- **inter_locale_rank_to_peer_rank** (sequence of int or dict) – A dict of (inter_locale_rank, peer_rank) pairs. If a sequence, then *inter_locale_rank_to_peer_rank*[inter_locale_rank] = peer_rank.

Todo

Provide example here.

mpi_array.distribution.Distribution.create_globale_extent

Distribution.**create_globale_extent**(*globale_extent*, *halo=0*)

Factory method for creating *GlobaleExtent* object.

Parameters

- **globale_extent** (object) – Can be specified as a *sequence-of-int* shape, *sequence-of-slice* slice or a *mpi_array.indexing.IndexingExtent*. Defines the globale extent of the array.
- **halo** (int) – Globale array halo (border), currently ignored.

Return type *GlobaleExtent*

Returns A self._globale_extent_type instance.

Todo

Handle *globale_extent* for non-zero *halo*.

mpi_array.distribution.Distribution.create_locale_extents

Distribution.**create_locale_extents**(*struct_locale_extents*)

mpi_array.distribution.Distribution.create_struct_locale_extents

Distribution.**create_struct_locale_extents**(*num_locales*, *ndim*)

mpi_array.distribution.Distribution.get_extent_for_rank

Distribution.**get_extent_for_rank**(*inter_locale_rank*)

Returns extent associated with the specified rank of the inter_locale_comm communicator.

Parameters **inter_locale_rank** (int) – Return the locale extent associated with this rank.

Return type *LocaleExtent*

Returns The locale extent for the specified *inter_locale_rank* rank.

mpi_array.distribution.Distribution.get_peer_rank

Distribution.**get_peer_rank**(*inter_locale_rank*)

Returns the peer_rank rank (of peer_comm) which is is the equivalent process of the *inter_locale_rank* rank of the inter_locale_comm communicator.

Parameters **inter_locale_rank** (int) – A rank of inter_locale_comm.

Return type int

Returns The equivalent rank from peer_comm.

mpi_array.distribution.Distribution.initialise_struct_locale_extents

Distribution.**initialise_struct_locale_extents**(*struct_locale_extents*, *locale_extents_descr*)

mpi_array.distribution.Distribution.initialise_struct_locale_extents_halos

Distribution.**initialise_struct_locale_extents_halos**(*struct_locale_extents*, *globale_extent*, *halo*)

Trim locale extent halos so they don't extend beyond the *globale_extent* halo.

Attributes

HI	The "high index" indices.
LO	The "low index" indices.
globale_extent	A *GlobaleExtent* specifying the globale array indexing extent.
halo	A (ndim, 2) shaped array of int indicating the halo (ghost elements) for extents.
locale_extents	Sequence of *LocaleExtent* objects where locale_extents[r] is the extent assigned to locale with inter_locale_comm rank r.
num_locales	An int specifying the number of locales in this distribution.
peer_ranks_per_locale	A numpy.ndarray of length *num_locales*.
struct_locale_extents	A structure numpy.ndarray where struct_locale_extents[r] is the extent assigned to locale with inter_locale_comm rank r.

mpi_array.distribution.Distribution.HI

Distribution.**HI = 1**

The "high index" indices.

mpi_array.distribution.Distribution.LO

Distribution.**LO = 0**
The "low index" indices.

mpi_array.distribution.Distribution.globale_extent

Distribution.**globale_extent**
A *GlobaleExtent* specifying the globale array indexing extent.

mpi_array.distribution.Distribution.halo

Distribution.**halo**
A `(ndim, 2)` shaped array of `int` indicating the halo (ghost elements) for extents. This may differ from the *LocaleExtent.halo* value due to the locale extent halos getting trimmed to lie within the globale extent.

mpi_array.distribution.Distribution.locale_extents

Distribution.**locale_extents**
Sequence of *LocaleExtent* objects where `locale_extents[r]` is the extent assigned to locale with `inter_locale_comm` rank `r`.

mpi_array.distribution.Distribution.num_locales

Distribution.**num_locales**
An `int` specifying the number of locales in this distribution.

mpi_array.distribution.Distribution.peer_ranks_per_locale

Distribution.**peer_ranks_per_locale**
A `numpy.ndarray` of length *num_locales*. Each element of the array is a sequence of `int` such that `self.peer_ranks_per_locale[r]` are the ranks of `peer_comm` which belong to the locale associated with `self.locale_extents[r]`.

mpi_array.distribution.Distribution.struct_locale_extents

Distribution.**struct_locale_extents**
A structure `numpy.ndarray` where `struct_locale_extents[r]` is the extent assigned to locale with `inter_locale_comm` rank `r`.

mpi_array.distribution.ClonedDistribution

class mpi_array.distribution.**ClonedDistribution**(*globale_extent*, *num_locales*, *halo=0*, *inter_locale_rank_to_peer_rank=None*)

Bases: *mpi_array.distribution.Distribution*

Distribution where entire globale extent elements occur on every locale.

Methods

__init__(globale_extent, num_locales[, ...])	Initialise.
create_globale_extent(globale_extent[, halo])	Factory method for creating *GlobaleExtent* object.
create_locale_extents(struct_locale_extents)	
create_struct_locale_extents(num_locales, ndim)	
get_extent_for_rank(inter_locale_rank)	Returns extent associated with the specified rank of the `inter_locale_comm` communicator.
get_peer_rank(inter_locale_rank)	Returns the `peer_rank` rank (of `peer_comm`) which is is the equivalent process of the *inter_locale_rank* rank of the `inter_locale_comm` communicator.
initialise_struct_locale_extents(...)	
initialise_struct_locale_extents_halos(...)	Trim locale extent halos so they don't extend beyond the *globale_extent* halo.

mpi_array.distribution.ClonedDistribution.__init__

`ClonedDistribution.`**`__init__`**(*globale_extent*, *num_locales*, *halo=0*, *inter_locale_rank_to_peer_rank=None*)

Initialise.

Parameters

- **globale_extent** (`object`) – Can be specified as a *sequence-of-int* shape, *sequence-of-slice* slice or a *mpi_array.indexing.IndexingExtent*. Defines the globale extent of the array.
- **num_locales** (`int`) – Number of locales over which the globale array is cloned.
- **halo** (`int`, sequence of `int` or `(ndim, 2)` shaped array) – Locale array halo (ghost elements).
- **inter_locale_rank_to_peer_rank** (sequence of `int` or `dict`) – A `dict` of `(inter_locale_rank, peer_rank)` pairs. If a sequence, then *inter_locale_rank_to_peer_rank*`[inter_locale_rank] = peer_rank`.

Todo

Provide example here.

mpi_array.distribution.ClonedDistribution.create_globale_extent

`ClonedDistribution.`**`create_globale_extent`**(*globale_extent*, *halo=0*)

Factory method for creating *GlobaleExtent* object.

Parameters

- **globale_extent** (`object`) – Can be specified as a *sequence-of-int* shape, *sequence-of-slice* slice or a *mpi_array.indexing.IndexingExtent*. Defines the globale extent of the array.

- **halo** (int) – Globale array halo (border), currently ignored.

Return type *GlobaleExtent*

Returns A `self._globale_extent_type` instance.

Todo

Handle *globale_extent* for non-zero *halo*.

mpi_array.distribution.ClonedDistribution.create_locale_extents

ClonedDistribution.**create_locale_extents**(*struct_locale_extents*)

mpi_array.distribution.ClonedDistribution.create_struct_locale_extents

ClonedDistribution.**create_struct_locale_extents**(*num_locales*, *ndim*)

mpi_array.distribution.ClonedDistribution.get_extent_for_rank

ClonedDistribution.**get_extent_for_rank**(*inter_locale_rank*)

Returns extent associated with the specified rank of the `inter_locale_comm` communicator.

Parameters **inter_locale_rank** (int) – Return the locale extent associated with this rank.

Return type *LocaleExtent*

Returns The locale extent for the specified *inter_locale_rank* rank.

mpi_array.distribution.ClonedDistribution.get_peer_rank

ClonedDistribution.**get_peer_rank**(*inter_locale_rank*)

Returns the `peer_rank` rank (of `peer_comm`) which is is the equivalent process of the *inter_locale_rank* rank of the `inter_locale_comm` communicator.

Parameters **inter_locale_rank** (int) – A rank of `inter_locale_comm`.

Return type int

Returns The equivalent rank from `peer_comm`.

mpi_array.distribution.ClonedDistribution.initialise_struct_locale_extents

ClonedDistribution.**initialise_struct_locale_extents**(*struct_locale_extents*, *locale_extents_descr*)

mpi_array.distribution.ClonedDistribution.initialise_struct_locale_extents_halos

ClonedDistribution.**initialise_struct_locale_extents_halos**(*struct_locale_extents*, *globale_extent*, *halo*)

Trim locale extent halos so they don't extend beyond the *globale_extent* halo.

Attributes

HI	
LO	
globale_extent	A *GlobaleExtent* specifying the globale array indexing extent.
halo	A `(ndim, 2)` shaped array of `int` indicating the halo (ghost elements) for extents.
locale_extents	Sequence of *LocaleExtent* objects where `locale_extents[r]` is the extent assigned to locale with `inter_locale_comm` rank `r`.
num_locales	An `int` specifying the number of locales in this distribution.
peer_ranks_per_locale	A `numpy.ndarray` of length *num_locales*.
struct_locale_extents	A structure `numpy.ndarray` where `struct_locale_extents[r]` is the extent assigned to locale with `inter_locale_comm` rank `r`.

mpi_array.distribution.ClonedDistribution.HI

ClonedDistribution.**HI = 1**

mpi_array.distribution.ClonedDistribution.LO

ClonedDistribution.**LO = 0**

mpi_array.distribution.ClonedDistribution.globale_extent

ClonedDistribution.**globale_extent**

A *GlobaleExtent* specifying the globale array indexing extent.

mpi_array.distribution.ClonedDistribution.halo

ClonedDistribution.**halo**

A `(ndim, 2)` shaped array of `int` indicating the halo (ghost elements) for extents. This may differ from the *LocaleExtent.halo* value due to the locale extent halos getting trimmed to lie within the globale extent.

mpi_array.distribution.ClonedDistribution.locale_extents

ClonedDistribution.**locale_extents**
Sequence of *LocaleExtent* objects where `locale_extents[r]` is the extent assigned to locale with `inter_locale_comm` rank `r`.

mpi_array.distribution.ClonedDistribution.num_locales

ClonedDistribution.**num_locales**
An `int` specifying the number of locales in this distribution.

mpi_array.distribution.ClonedDistribution.peer_ranks_per_locale

ClonedDistribution.**peer_ranks_per_locale**
A `numpy.ndarray` of length *num_locales*. Each element of the array is a sequence of `int` such that `self.peer_ranks_per_locale[r]` are the ranks of `peer_comm` which belong to the locale associated with `self.locale_extents[r]`.

mpi_array.distribution.ClonedDistribution.struct_locale_extents

ClonedDistribution.**struct_locale_extents**
A structure `numpy.ndarray` where `struct_locale_extents[r]` is the extent assigned to locale with `inter_locale_comm` rank `r`.

mpi_array.distribution.SingleLocaleDistribution

class mpi_array.distribution.**SingleLocaleDistribution**(*globale_extent*, *num_locales*, *inter_locale_rank=0*, *halo=0*, *inter_locale_rank_to_peer_rank=None*)

Bases: *mpi_array.distribution.Distribution*

Distribution where entire globale extent elements occur on just a single locale.

Methods

__init__(globale_extent, num_locales[, ...])	Initialise.
create_globale_extent(globale_extent[, halo])	Factory method for creating *GlobaleExtent* object.
create_locale_extents(struct_locale_extents)	
create_struct_locale_extents(num_locales, ndim)	
get_extent_for_rank(inter_locale_rank)	Returns extent associated with the specified rank of the `inter_locale_comm` communicator.
get_peer_rank(inter_locale_rank)	Returns the `peer_rank` rank (of `peer_comm`) which is is the equivalent process of the *inter_locale_rank* rank of the `inter_locale_comm` communicator.
initialise_struct_locale_extents(...)	

Continued on next page

Table 1.115 – continued from previous page

initialise_struct_locale_extents_halos(..)	Trim locale extent halos so they don't extend beyond the *globale_extent* halo.

mpi_array.distribution.SingleLocaleDistribution.__init__

SingleLocaleDistribution.**__init__**(*globale_extent*, *num_locales*, *inter_locale_rank=0*, *halo=0*, *inter_locale_rank_to_peer_rank=None*)

Initialise.

Parameters

- **globale_extent** (object) – Can be specified as a *sequence-of-int* shape, *sequence-of-slice* slice or a *mpi_array.indexing.IndexingExtent*. Defines the globale extent of the array.
- **num_locales** (int) – Number of locales. One non-empty extent on the first (i.e. `inter_locale_rank == 0` rank) locale, empty extents on remaining locales.
- **halo** (int, sequence of int or `(ndim, 2)` shaped array) – Locale array halo (ghost elements).
- **inter_locale_rank_to_peer_rank** (sequence of int or dict) – A dict of `(inter_locale_rank, peer_rank)` pairs. If a sequence, then *inter_locale_rank_to_peer_rank*`[inter_locale_rank] = peer_rank`.

mpi_array.distribution.SingleLocaleDistribution.create_globale_extent

SingleLocaleDistribution.**create_globale_extent**(*globale_extent*, *halo=0*)

Factory method for creating *GlobaleExtent* object.

Parameters

- **globale_extent** (object) – Can be specified as a *sequence-of-int* shape, *sequence-of-slice* slice or a *mpi_array.indexing.IndexingExtent*. Defines the globale extent of the array.
- **halo** (int) – Globale array halo (border), currently ignored.

Return type *GlobaleExtent*

Returns A `self._globale_extent_type` instance.

Todo

Handle *globale_extent* for non-zero *halo*.

mpi_array.distribution.SingleLocaleDistribution.create_locale_extents

SingleLocaleDistribution.**create_locale_extents**(*struct_locale_extents*)

mpi_array.distribution.SingleLocaleDistribution.create_struct_locale_extents

SingleLocaleDistribution.**create_struct_locale_extents**(*num_locales*, *ndim*)

mpi_array.distribution.SingleLocaleDistribution.get_extent_for_rank

SingleLocaleDistribution.**get_extent_for_rank**(*inter_locale_rank*)

Returns extent associated with the specified rank of the `inter_locale_comm` communicator.

Parameters **inter_locale_rank** (`int`) – Return the locale extent associated with this rank.

Return type *LocaleExtent*

Returns The locale extent for the specified *inter_locale_rank* rank.

mpi_array.distribution.SingleLocaleDistribution.get_peer_rank

SingleLocaleDistribution.**get_peer_rank**(*inter_locale_rank*)

Returns the `peer_rank` rank (of `peer_comm`) which is is the equivalent process of the *inter_locale_rank* rank of the `inter_locale_comm` communicator.

Parameters **inter_locale_rank** (`int`) – A rank of `inter_locale_comm`.

Return type `int`

Returns The equivalent rank from `peer_comm`.

mpi_array.distribution.SingleLocaleDistribution.initialise_struct_locale_extents

SingleLocaleDistribution.**initialise_struct_locale_extents**(*struct_locale_extents*, *locale_extents_descr*)

mpi_array.distribution.SingleLocaleDistribution.initialise_struct_locale_extents_halos

SingleLocaleDistribution.**initialise_struct_locale_extents_halos**(*struct_locale_extents*, *globale_extent*, *halo*)

Trim locale extent halos so they don't extend beyond the *globale_extent* halo.

Attributes

HI	
LO	
globale_extent	A *GlobaleExtent* specifying the globale array indexing extent.
halo	A `(ndim, 2)` shaped array of `int` indicating the halo (ghost elements) for extents.

Continued on next page

Table 1.116 – continued from previous page

locale_extents	Sequence of *LocaleExtent* objects where `locale_extents[r]` is the extent assigned to locale with `inter_locale_comm` rank `r`.
num_locales	An `int` specifying the number of locales in this distribution.
peer_ranks_per_locale	A `numpy.ndarray` of length *num_locales*.
struct_locale_extents	A structure `numpy.ndarray` where `struct_locale_extents[r]` is the extent assigned to locale with `inter_locale_comm` rank `r`.

mpi_array.distribution.SingleLocaleDistribution.HI

`SingleLocaleDistribution.`**`HI = 1`**

mpi_array.distribution.SingleLocaleDistribution.LO

`SingleLocaleDistribution.`**`LO = 0`**

mpi_array.distribution.SingleLocaleDistribution.globale_extent

`SingleLocaleDistribution.`**`globale_extent`**
A *GlobaleExtent* specifying the globale array indexing extent.

mpi_array.distribution.SingleLocaleDistribution.halo

`SingleLocaleDistribution.`**`halo`**
A `(ndim, 2)` shaped array of `int` indicating the halo (ghost elements) for extents. This may differ from the *LocaleExtent.halo* value due to the locale extent halos getting trimmed to lie within the globale extent.

mpi_array.distribution.SingleLocaleDistribution.locale_extents

`SingleLocaleDistribution.`**`locale_extents`**
Sequence of *LocaleExtent* objects where `locale_extents[r]` is the extent assigned to locale with `inter_locale_comm` rank `r`.

mpi_array.distribution.SingleLocaleDistribution.num_locales

`SingleLocaleDistribution.`**`num_locales`**
An `int` specifying the number of locales in this distribution.

mpi_array.distribution.SingleLocaleDistribution.peer_ranks_per_locale

`SingleLocaleDistribution.`**`peer_ranks_per_locale`**
A `numpy.ndarray` of length *num_locales*. Each element of the array is a sequence of `int` such

that `self.peer_ranks_per_locale[r]` are the ranks of `peer_comm` which belong to the locale associated with `self.locale_extents[r]`.

mpi_array.distribution.SingleLocaleDistribution.struct_locale_extents

`SingleLocaleDistribution.`**`struct_locale_extents`**

A structure `numpy.ndarray` where `struct_locale_extents[r]` is the extent assigned to locale with `inter_locale_comm` rank `r`.

mpi_array.distribution.BlockPartition

class `mpi_array.distribution.`**`BlockPartition`**(*globale_extent*, *dims*, *cart_coord_to_cart_rank*, *halo=0*, *order='C'*, *inter_locale_rank_to_peer_rank=None*)

Bases: *mpi_array.distribution.Distribution*

Block partition of an array (shape) over locales.

Methods

__init__(globale_extent, dims, ...[, halo, ...])	Creates a block-partitioning of *shape* over locales.
create_globale_extent(globale_extent[, halo])	Factory method for creating *GlobaleExtent* object.
create_locale_extents(struct_locale_extents)	
create_struct_locale_extents(num_locales, ndim)	
get_extent_for_rank(inter_locale_rank)	Returns extent associated with the specified rank of the `inter_locale_comm` communicator.
get_peer_rank(inter_locale_rank)	Returns the `peer_rank` rank (of `peer_comm`) which is is the equivalent process of the *inter_locale_rank* rank of the `inter_locale_comm` communicator.
initialise_struct_locale_extents(...)	
initialise_struct_locale_extents_halos(...)	Trim locale extent halos so they don't extend beyond the *globale_extent* halo.

mpi_array.distribution.BlockPartition.__init__

`BlockPartition.`**`__init__`**(*globale_extent*, *dims*, *cart_coord_to_cart_rank*, *halo=0*, *order='C'*, *inter_locale_rank_to_peer_rank=None*)

Creates a block-partitioning of *shape* over locales.

Parameters

- **globale_extent** (*GlobaleExtent*) – The globale extent to be partitioned.
- **dims** (sequence of `int`) – The number of partitions along each dimension, `len(`*dims*`) == len(`*globale_extent*`.shape_n)` and `num_locales = numpy.product(`*dims*`)`.
- **halo** (`int`, sequence of `int` or `(len(`*shape*`), 2)` shaped array.) – Number of *ghost* elements added per axis (low-index number of ghost elements may differ to the number of high-index ghost elements).

- **cart_coord_to_cart_rank** (dict) – Mapping between cartesian communicator coordinate (mpi4py.MPI.CartComm.Get_coords()) and cartesian communicator rank.

mpi_array.distribution.BlockPartition.create_globale_extent

BlockPartition.**create_globale_extent**(*globale_extent*, *halo=0*)

Factory method for creating *GlobaleExtent* object.

Parameters

- **globale_extent** (object) – Can be specified as a *sequence-of-int* shape, *sequence-of-slice* slice or a *mpi_array.indexing.IndexingExtent*. Defines the globale extent of the array.
- **halo** (int) – Globale array halo (border), currently ignored.

Return type *GlobaleExtent*

Returns A self._globale_extent_type instance.

Todo

Handle *globale_extent* for non-zero *halo*.

mpi_array.distribution.BlockPartition.create_locale_extents

BlockPartition.**create_locale_extents**(*struct_locale_extents*)

mpi_array.distribution.BlockPartition.create_struct_locale_extents

BlockPartition.**create_struct_locale_extents**(*num_locales*, *ndim*)

mpi_array.distribution.BlockPartition.get_extent_for_rank

BlockPartition.**get_extent_for_rank**(*inter_locale_rank*)

Returns extent associated with the specified rank of the inter_locale_comm communicator.

Parameters **inter_locale_rank** (int) – Return the locale extent associated with this rank.

Return type *LocaleExtent*

Returns The locale extent for the specified *inter_locale_rank* rank.

mpi_array.distribution.BlockPartition.get_peer_rank

BlockPartition.**get_peer_rank**(*inter_locale_rank*)

Returns the peer_rank rank (of peer_comm) which is is the equivalent process of the *inter_locale_rank* rank of the inter_locale_comm communicator.

Parameters **inter_locale_rank** (int) – A rank of inter_locale_comm.

Return type int

Returns The equivalent rank from `peer_comm`.

mpi_array.distribution.BlockPartition.initialise_struct_locale_extents

BlockPartition.**initialise_struct_locale_extents**(*struct_locale_extents*, *locale_extents_descr*)

mpi_array.distribution.BlockPartition.initialise_struct_locale_extents_halos

BlockPartition.**initialise_struct_locale_extents_halos**(*struct_locale_extents*, *globale_extent*, *halo*)

Trim locale extent halos so they don't extend beyond the *globale_extent* halo.

Attributes

HI	
LO	
globale_extent	A *GlobaleExtent* specifying the globale array indexing extent.
halo	A `(ndim, 2)` shaped array of `int` indicating the halo (ghost elements) for extents.
locale_extents	Sequence of *LocaleExtent* objects where `locale_extents[r]` is the extent assigned to locale with `inter_locale_comm` rank `r`.
num_locales	An `int` specifying the number of locales in this distribution.
peer_ranks_per_locale	A `numpy.ndarray` of length *num_locales*.
struct_locale_extents	A structure `numpy.ndarray` where `struct_locale_extents[r]` is the extent assigned to locale with `inter_locale_comm` rank `r`.

mpi_array.distribution.BlockPartition.HI

BlockPartition.**HI = 1**

mpi_array.distribution.BlockPartition.LO

BlockPartition.**LO = 0**

mpi_array.distribution.BlockPartition.globale_extent

BlockPartition.**globale_extent**

A *GlobaleExtent* specifying the globale array indexing extent.

mpi_array.distribution.BlockPartition.halo

BlockPartition.**halo**
A `(ndim, 2)` shaped array of `int` indicating the halo (ghost elements) for extents. This may differ from the *`LocaleExtent.halo`* value due to the locale extent halos getting trimmed to lie within the globale extent.

mpi_array.distribution.BlockPartition.locale_extents

BlockPartition.**locale_extents**
Sequence of *`LocaleExtent`* objects where `locale_extents[r]` is the extent assigned to locale with `inter_locale_comm` rank `r`.

mpi_array.distribution.BlockPartition.num_locales

BlockPartition.**num_locales**
An `int` specifying the number of locales in this distribution.

mpi_array.distribution.BlockPartition.peer_ranks_per_locale

BlockPartition.**peer_ranks_per_locale**
A `numpy.ndarray` of length *`num_locales`*. Each element of the array is a sequence of `int` such that `self.peer_ranks_per_locale[r]` are the ranks of `peer_comm` which belong to the locale associated with `self.locale_extents[r]`.

mpi_array.distribution.BlockPartition.struct_locale_extents

BlockPartition.**struct_locale_extents**
A structure `numpy.ndarray` where `struct_locale_extents[r]` is the extent assigned to locale with `inter_locale_comm` rank `r`.

1.17 The `mpi_array.distribution_test` Module

Module defining *`mpi_array.distribution`* unit-tests. Execute as:

```
python -m mpi_array.distribution_test
```

1.17.1 Classes

`CartLocaleExtentTest`([methodName])	`unittest.TestCase` for *`mpi_array.distribution.CartLocaleExtent`*.
`BlockPartitionTest`([methodName])	`unittest.TestCase` for *`mpi_array.distribution.BlockPartition`*.

mpi_array.distribution_test.CartLocaleExtentTest

class mpi_array.distribution_test.**CartLocaleExtentTest**(*methodName='runTest'*)

Bases: *mpi_array.unittest.TestCase*

unittest.TestCase for *mpi_array.distribution.CartLocaleExtent*.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	Hook method for setting up the test fixture before exercising it.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_construct_attribs()	Assertions for properties.
test_extent_calcs_1d_thick_tiles()	Tests *mpi_array.distribution.CartLocaleExtent.halo_slab_extent()* and *mpi_array.distribution.CartLocaleExtent.no_halo_extent()* methods when halo size is smaller than the tile size.
test_extent_calcs_1d_thin_tiles()	Tests *mpi_array.distribution.CartLocaleExtent.halo_slab_extent()* and *mpi_array.distribution.CartLocaleExtent.no_halo_extent()* methods when halo size is larger than the tile size, 1D fixture.

Continued on next page

Table 1.120 – continued from previous page

test_extent_calcs_2d_thick_tiles()	Tests *mpi_array.distribution.CartLocaleExtent.halo_slab_extent()* and *mpi_array.distribution.CartLocaleExtent.no_halo_extent()* methods when halo size is smaller than the tile size, 2D fixture.
test_repr()	Tests mpi_array.distribution.CartLocaleExtent.__repr__().

mpi_array.distribution_test.CartLocaleExtentTest.__init__

CartLocaleExtentTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.distribution_test.CartLocaleExtentTest.addCleanup

CartLocaleExtentTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.distribution_test.CartLocaleExtentTest.addTypeEqualityFunc

CartLocaleExtentTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.distribution_test.CartLocaleExtentTest.assertArraySplitEqual

CartLocaleExtentTest.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.distribution_test.CartLocaleExtentTest.countTestCases

CartLocaleExtentTest.**countTestCases**()

mpi_array.distribution_test.CartLocaleExtentTest.debug

CartLocaleExtentTest.**debug**()
Run the test without collecting errors in a TestResult

mpi_array.distribution_test.CartLocaleExtentTest.defaultTestResult

CartLocaleExtentTest.**defaultTestResult**()

mpi_array.distribution_test.CartLocaleExtentTest.doCleanups

CartLocaleExtentTest.**doCleanups**()
Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.distribution_test.CartLocaleExtentTest.id

CartLocaleExtentTest.**id**()

mpi_array.distribution_test.CartLocaleExtentTest.run

CartLocaleExtentTest.**run**(*result=None*)

mpi_array.distribution_test.CartLocaleExtentTest.setUp

CartLocaleExtentTest.**setUp**()
Hook method for setting up the test fixture before exercising it.

mpi_array.distribution_test.CartLocaleExtentTest.setUpClass

CartLocaleExtentTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.distribution_test.CartLocaleExtentTest.shortDescription

CartLocaleExtentTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.distribution_test.CartLocaleExtentTest.skipTest

CartLocaleExtentTest.**skipTest**(*reason*)
Skip this test.

mpi_array.distribution_test.CartLocaleExtentTest.subTest

CartLocaleExtentTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.distribution_test.CartLocaleExtentTest.tearDown

CartLocaleExtentTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.distribution_test.CartLocaleExtentTest.tearDownClass

CartLocaleExtentTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.distribution_test.CartLocaleExtentTest.test_construct_attribs

CartLocaleExtentTest.**test_construct_attribs**()
Assertions for properties.

mpi_array.distribution_test.CartLocaleExtentTest.test_extent_calcs_1d_thick_tiles

CartLocaleExtentTest.**test_extent_calcs_1d_thick_tiles**()
Tests *mpi_array.distribution.CartLocaleExtent.halo_slab_extent()* and *mpi_array.distribution.CartLocaleExtent.no_halo_extent()* methods when halo size is smaller than the tile size.

mpi_array.distribution_test.CartLocaleExtentTest.test_extent_calcs_1d_thin_tiles

CartLocaleExtentTest.**test_extent_calcs_1d_thin_tiles**()
Tests *mpi_array.distribution.CartLocaleExtent.halo_slab_extent()* and *mpi_array.distribution.CartLocaleExtent.no_halo_extent()* methods when halo size is larger than the tile size, 1D fixture.

mpi_array.distribution_test.CartLocaleExtentTest.test_extent_calcs_2d_thick_tiles

CartLocaleExtentTest.**test_extent_calcs_2d_thick_tiles**()
Tests *mpi_array.distribution.CartLocaleExtent.halo_slab_extent()* and *mpi_array.distribution.CartLocaleExtent.no_halo_extent()* methods when halo size is smaller than the tile size, 2D fixture.

mpi_array.distribution_test.CartLocaleExtentTest.test_repr

CartLocaleExtentTest.**test_repr**()
Tests mpi_array.distribution.CartLocaleExtent.__repr__().

Attributes

longMessage
maxDiff

mpi_array.distribution_test.CartLocaleExtentTest.longMessage

CartLocaleExtentTest.**longMessage** = True

mpi_array.distribution_test.CartLocaleExtentTest.maxDiff

CartLocaleExtentTest.**maxDiff** = 640

mpi_array.distribution_test.BlockPartitionTest

class mpi_array.distribution_test.**BlockPartitionTest**(*methodName='runTest'*)
Bases: *mpi_array.unittest.TestCase*

unittest.TestCase for *mpi_array.distribution.BlockPartition*.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
do_test_construct_1d_with_halo([halo])	Test *mpi_array.distribution.BlockPartition* construction.
do_test_construct_2d_with_halo([halo])	Test *mpi_array.distribution.BlockPartition* construction.
id()	
run([result])	
setUp()	Initialise self.root_logger.

Continued on next page

Table 1.122 – continued from previous page

setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_construct_1d_empty_tiles()	Test *mpi_array.distribution.BlockPartition* construction when the partition leads to empty extents.
test_construct_1d_no_halo()	Test *mpi_array.distribution.BlockPartition* construction.
test_construct_1d_with_halo()	Test *mpi_array.distribution.BlockPartition* construction.
test_construct_2d_no_halo()	Test *mpi_array.distribution.BlockPartition* construction.
test_construct_2d_with_halo()	Test *mpi_array.distribution.BlockPartition* construction.
test_construct_single_locale_1d()	Test *mpi_array.distribution.BlockPartition* construction.

mpi_array.distribution_test.BlockPartitionTest.__init__

BlockPartitionTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.distribution_test.BlockPartitionTest.addCleanup

BlockPartitionTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.distribution_test.BlockPartitionTest.addTypeEqualityFunc

BlockPartitionTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.distribution_test.BlockPartitionTest.assertArraySplitEqual

BlockPartitionTest.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.distribution_test.BlockPartitionTest.countTestCases

BlockPartitionTest.**countTestCases**()

mpi_array.distribution_test.BlockPartitionTest.debug

BlockPartitionTest.**debug**()

Run the test without collecting errors in a TestResult

mpi_array.distribution_test.BlockPartitionTest.defaultTestResult

BlockPartitionTest.**defaultTestResult**()

mpi_array.distribution_test.BlockPartitionTest.doCleanups

BlockPartitionTest.**doCleanups**()

Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.distribution_test.BlockPartitionTest.do_test_construct_1d_with_halo

BlockPartitionTest.**do_test_construct_1d_with_halo**(*halo=0*)

Test *mpi_array.distribution.BlockPartition* construction.

mpi_array.distribution_test.BlockPartitionTest.do_test_construct_2d_with_halo

BlockPartitionTest.**do_test_construct_2d_with_halo**(*halo=0*)

Test *mpi_array.distribution.BlockPartition* construction.

mpi_array.distribution_test.BlockPartitionTest.id

BlockPartitionTest.**id**()

mpi_array.distribution_test.BlockPartitionTest.run

BlockPartitionTest.**run**(*result=None*)

mpi_array.distribution_test.BlockPartitionTest.setUp

BlockPartitionTest.**setUp**()
Initialise self.root_logger.

mpi_array.distribution_test.BlockPartitionTest.setUpClass

BlockPartitionTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.distribution_test.BlockPartitionTest.shortDescription

BlockPartitionTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.distribution_test.BlockPartitionTest.skipTest

BlockPartitionTest.**skipTest**(*reason*)
Skip this test.

mpi_array.distribution_test.BlockPartitionTest.subTest

BlockPartitionTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.distribution_test.BlockPartitionTest.tearDown

BlockPartitionTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.distribution_test.BlockPartitionTest.tearDownClass

BlockPartitionTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.distribution_test.BlockPartitionTest.test_construct_1d_empty_tiles

BlockPartitionTest.**test_construct_1d_empty_tiles**()
Test *mpi_array.distribution.BlockPartition* construction when the partition leads to empty extents.

mpi_array.distribution_test.BlockPartitionTest.test_construct_1d_no_halo

BlockPartitionTest.**test_construct_1d_no_halo**()
Test *mpi_array.distribution.BlockPartition* construction.

mpi_array.distribution_test.BlockPartitionTest.test_construct_1d_with_halo

BlockPartitionTest.**test_construct_1d_with_halo**()
Test *mpi_array.distribution.BlockPartition* construction.

mpi_array.distribution_test.BlockPartitionTest.test_construct_2d_no_halo

BlockPartitionTest.**test_construct_2d_no_halo**()
Test *mpi_array.distribution.BlockPartition* construction.

mpi_array.distribution_test.BlockPartitionTest.test_construct_2d_with_halo

BlockPartitionTest.**test_construct_2d_with_halo**()
Test *mpi_array.distribution.BlockPartition* construction.

mpi_array.distribution_test.BlockPartitionTest.test_construct_single_locale_1d

BlockPartitionTest.**test_construct_single_locale_1d**()
Test *mpi_array.distribution.BlockPartition* construction.

Attributes

longMessage
maxDiff

mpi_array.distribution_test.BlockPartitionTest.longMessage

BlockPartitionTest.**longMessage** = True

mpi_array.distribution_test.BlockPartitionTest.maxDiff

BlockPartitionTest.**maxDiff** = 640

1.18 The `mpi_array.globale` Module

Defines *gndarray* class and factory functions for creating multi-dimensional distributed arrays (Partitioned Global Address Space).

1.18.1 Classes

gndarray	A Partitioned Global Address Space array with `numpy.ndarray` API.
PerAxisRmaHaloUpdater(locale_extents, dtype, ...)	Helper class for performing halo data transfer using RMA via MPI windows (`mpi4py.MPI.Win` objects).
RmaRedistributeUpdater(dst, src[, casting])	Helper class for redistributing array to new distribution.

mpi_array.globale.gndarray

class `mpi_array.globale.`**gndarray**

Bases: `numpy.lib.mixins.NDArrayOperatorsMixin`

A Partitioned Global Address Space array with `numpy.ndarray` API.

Methods

all(**unused_kwargs)	
calculate_copyfrom_updates(src[, casting])	
copy([order])	
copyfrom(src[, casting])	Copy the elements of the *src* array to corresponding elements of the *self* array.
fill(value)	Fill the array (excluding ghost elements) with a scalar value.
fill_h(value)	Fill all array elements (including ghost elements) with a scalar value.
free()	Collective (all samp:*peer_comm* processes) free of MPI windows (and locale array memory).
get_view([slice, start, stop, halo])	Returns `(ary, extent)` pair, where `ary` is a view from the locale extent array corresponding to the specified extent arguments.
initialise_windows()	Creates the RMA windows required for inter-locale (and peer) one-sided RMA comms.
inter_locale_barrier()	
intra_locale_barrier()	
locale_get([slice, start, stop, halo])	Collective over *self*.`comms.intra_locale_comm` to get a portion of the globale array.
peer_rank_get([slice, start, stop, halo])	Non-collective, one-sided fetch of data to this peer rank process.
reshape(shape)	Returns an array containing the same data with a new shape equal to *shape*.
update()	

mpi_array.globale.gndarray.all

gndarray.**all**(**unused_kwargs)

mpi_array.globale.gndarray.calculate_copyfrom_updates

gndarray.**calculate_copyfrom_updates**(*src*, *casting='same_kind'*)

mpi_array.globale.gndarray.copy

gndarray.**copy**(*order='C'*)

mpi_array.globale.gndarray.copyfrom

gndarray.**copyfrom**(*src*, *casting='same_kind'*)

Copy the elements of the *src* array to corresponding elements of the *self* array.

Parameters

- **src** (*gndarray*) – Global array from which elements are copied.
- **casting** (str) – See *casting* parameter in numpy.copyto().

mpi_array.globale.gndarray.fill

gndarray.**fill**(*value*)

Fill the array (excluding ghost elements) with a scalar value.

Parameters **value** (*scalar*) – All non-ghost elements will be assigned this value.

mpi_array.globale.gndarray.fill_h

gndarray.**fill_h**(*value*)

Fill all array elements (including ghost elements) with a scalar value.

Parameters **value** (*scalar*) – All elements will be assigned this value.

mpi_array.globale.gndarray.free

gndarray.**free**()

Collective (all samp:*peer_comm* processes) free of MPI windows (and locale array memory).

mpi_array.globale.gndarray.get_view

gndarray.**get_view**(*slice=None*, *start=None*, *stop=None*, *halo=0*)

Returns `(ary, extent)` pair, where `ary` is a view from the locale extent array corresponding to the specified extent arguments. If any of the globale slice lies outside the locale extent, then `ary` is `None`. The `extent` element is a *mpi_array.distribution.LocaleExtent* instance which corresponds to the specified extent arguments.

mpi_array.globale.gndarray.initialise_windows

gndarray.**initialise_windows**()
Creates the RMA windows required for inter-locale (and peer) one-sided RMA comms.

mpi_array.globale.gndarray.inter_locale_barrier

gndarray.**inter_locale_barrier**()

mpi_array.globale.gndarray.intra_locale_barrier

gndarray.**intra_locale_barrier**()

mpi_array.globale.gndarray.locale_get

gndarray.**locale_get**(*slice=None*, *start=None*, *stop=None*, *halo=0*)
Collective over `self.comms.intra_locale_comm` to get a portion of the globale array. Returns a view from the locale extent of the array if possible, otherwise allocates shared memory and performs one-sided RMA to fetch data from remote locales.

mpi_array.globale.gndarray.peer_rank_get

gndarray.**peer_rank_get**(*slice=None*, *start=None*, *stop=None*, *halo=0*)
Non-collective, one-sided fetch of data to this peer rank process. Returns a view from the locale extent of the array if possible, otherwise allocates non-shared memory and performs one-sided RMA to fetch data from remote locales.

mpi_array.globale.gndarray.reshape

gndarray.**reshape**(*shape*)
Returns an array containing the same data with a new shape equal to *shape*.

mpi_array.globale.gndarray.update

gndarray.**update**()

Attributes

comms_and_distrib
distribution
dtype
halo_updater
lndarray_proxy
locale_comms
ndim

Continued on next page

Table 1.126 – continued from previous page

num_locales
order
rank_logger
rank_view_h
rank_view_n
rma_window_buffer
root_logger
shape
this_locale
view_h
view_n

mpi_array.globale.gndarray.comms_and_distrib

gndarray.**comms_and_distrib**

mpi_array.globale.gndarray.distribution

gndarray.**distribution**

mpi_array.globale.gndarray.dtype

gndarray.**dtype**

mpi_array.globale.gndarray.halo_updater

gndarray.**halo_updater**

mpi_array.globale.gndarray.lndarray_proxy

gndarray.**lndarray_proxy**

mpi_array.globale.gndarray.locale_comms

gndarray.**locale_comms**

mpi_array.globale.gndarray.ndim

gndarray.**ndim**

mpi_array.globale.gndarray.num_locales

gndarray.**num_locales**

mpi_array.globale.gndarray.order

gndarray.**order**

mpi_array.globale.gndarray.rank_logger

gndarray.**rank_logger**

mpi_array.globale.gndarray.rank_view_h

gndarray.**rank_view_h**

mpi_array.globale.gndarray.rank_view_n

gndarray.**rank_view_n**

mpi_array.globale.gndarray.rma_window_buffer

gndarray.**rma_window_buffer**

mpi_array.globale.gndarray.root_logger

gndarray.**root_logger**

mpi_array.globale.gndarray.shape

gndarray.**shape**

mpi_array.globale.gndarray.this_locale

gndarray.**this_locale**

mpi_array.globale.gndarray.view_h

gndarray.**view_h**

mpi_array.globale.gndarray.view_n

gndarray.**view_n**

mpi_array.globale.PerAxisRmaHaloUpdater

class mpi_array.globale.**PerAxisRmaHaloUpdater**(*locale_extents*, *dtype*, *order*, *inter_locale_win*, *dst_buffer*)

Bases: mpi_array.globale.CommLogger

Helper class for performing halo data transfer using RMA via MPI windows (mpi4py.MPI.Win objects).

Methods

__init__(locale_extents, dtype, order, ...)	Initialise.
calc_halo_updates()	Calculates the per-axis halo-region updates for all inter-locale ranks (of the inter_locale_comm).
do_update_halos(halo_updates)	Performs the data exchange required to update the halo (ghost) elements of the array buffer *dst_buffer*.buffer.
update_halos()	Performs the data exchange required to update the halo (ghost) elements of the array buffer *dst_buffer*.buffer.

mpi_array.globale.PerAxisRmaHaloUpdater.__init__

PerAxisRmaHaloUpdater.**__init__**(*locale_extents*, *dtype*, *order*, *inter_locale_win*, *dst_buffer*)

Initialise.

Parameters

- **locale_extents** (sequence of mpi_array.distributon.LocaleExtent) – locale_extents[r] is the extent of the array elements which reside on rank r of the inter_locale_comm communicator.
- **dtype** (numpy.dtype) – Data type of elements in array.
- **order** (str) – The array order, 'C' for C memory layout.
- **inter_locale_win** (mpi4py.MPI.Win) – The window used to exchange halo element data.
- **dst_buffer** (memoryview) – The buffer into which the halo elements are written.

mpi_array.globale.PerAxisRmaHaloUpdater.calc_halo_updates

PerAxisRmaHaloUpdater.**calc_halo_updates**()

Calculates the per-axis halo-region updates for all inter-locale ranks (of the inter_locale_comm).

Return type tuple pair

Returns A (rank_2_updates_dict, bool_sequence) pair where *rank_2_updates* is a dict of *inter_locale_rank, halos_update*, where inter_locale_rank is an int indicating the rank of the process in inter_locale_comm and halos_update is a mpi_array.update.MpiHalosUpdate containing the description of regions which are required to be fetched from remote processes. The bool_sequence is of length ndim and bool_sequence[a] is True indicates that halo updates are required on axis a.

mpi_array.globale.PerAxisRmaHaloUpdater.do_update_halos

PerAxisRmaHaloUpdater.**do_update_halos**(*halo_updates*)

Performs the data exchange required to update the halo (ghost) elements of the array buffer *dst_buffer*.buffer. Can be called peer_comm collectively.

Parameters **halo_updates** (mpi_array.update.MpiHalosUpdate) – A dict of per inter_locale_rank halo region updates. See *calc_halo_updates()*.

mpi_array.globale.PerAxisRmaHaloUpdater.update_halos

PerAxisRmaHaloUpdater.**update_halos**()

Performs the data exchange required to update the halo (ghost) elements of the array buffer *dst_buffer*.buffer. Can be called peer_comm collectively.

Attributes

HI	Halo "high index" indices.
LO	Halo "low index" indices.
dst_buffer	A memoryview which provides the buffer into which the halo data is written.
dtype	The numpy.dtype of the data to be exchanged in the halo update.
halo_updates	The (rank_2_updates_dict, bool_sequence) pair calculated by *calc_halo_updates()*.
locale_extents	Sequence of *mpi_array.distribution.LocaleExtent* objects which define the partitioning of the array.
order	Array order str, 'C' for C memory layout.
rank_logger	
root_logger	

mpi_array.globale.PerAxisRmaHaloUpdater.HI

PerAxisRmaHaloUpdater.**HI = 1**

Halo "high index" indices.

mpi_array.globale.PerAxisRmaHaloUpdater.LO

PerAxisRmaHaloUpdater.**LO = 0**

Halo "low index" indices.

mpi_array.globale.PerAxisRmaHaloUpdater.dst_buffer

PerAxisRmaHaloUpdater.**dst_buffer**

A memoryview which provides the buffer into which the halo data is written.

mpi_array.globale.PerAxisRmaHaloUpdater.dtype

PerAxisRmaHaloUpdater.**dtype**
The numpy.dtype of the data to be exchanged in the halo update.

mpi_array.globale.PerAxisRmaHaloUpdater.halo_updates

PerAxisRmaHaloUpdater.**halo_updates**
The (rank_2_updates_dict, bool_sequence) pair calculated by *calc_halo_updates()*.

mpi_array.globale.PerAxisRmaHaloUpdater.locale_extents

PerAxisRmaHaloUpdater.**locale_extents**
Sequence of *mpi_array.distribution.LocaleExtent* objects which define the partitioning of the array.

mpi_array.globale.PerAxisRmaHaloUpdater.order

PerAxisRmaHaloUpdater.**order**
Array order str, 'C' for C memory layout.

mpi_array.globale.PerAxisRmaHaloUpdater.rank_logger

PerAxisRmaHaloUpdater.**rank_logger**

mpi_array.globale.PerAxisRmaHaloUpdater.root_logger

PerAxisRmaHaloUpdater.**root_logger**

mpi_array.globale.RmaRedistributeUpdater

class mpi_array.globale.**RmaRedistributeUpdater**(*dst*, *src*, *casting='same_kind'*)
Bases: *mpi_array.update.UpdatesForRedistribute*

Helper class for redistributing array to new distribution. Calculates sequence of mpi_array.distribution.ExtentUpdate objects which are used to copy elements from remote *src* locales to local *dst* locales.

Methods

__init__(dst, src[, casting])	
barrier()	MPI barrier.
calc_can_use_existing_src_peer_comm()	Returns True if self._src.locale_comms.peer_comm can be used to redistribute to the distribution of the self._dst array.

Continued on next page

Table 1.129 – continued from previous page

calc_intersection_split(dst_extent, src_extent)	Calculates intersection between *dst_extent* and *{src_extent}*.
check_updates()	Runs consistency checks on the calculated updates, assumes that the dst_distrib and src_distrib distributed as a partitioning (no locale extent overlaps except for halo).
create_pair_extent_update(dst_extent, ...)	Factory method which creates sequence of of mpi_array.distribution.MpiPairExtentUpdate objects.
do_locale_cpy2_update()	Performs direct copy updates.
do_locale_rma_update()	Performs RMA to get elements from remote locales to update the locale extent array.
do_locale_update()	
do_update()	
get_cpy2_src_extents(dst_inter_locale_rank)	
initialise()	
initialise_cpy2_updates()	
initialise_rget_updates()	
initialise_updates()	
wait_all(req_list)	

mpi_array.globale.RmaRedistributeUpdater.__init__

RmaRedistributeUpdater.**__init__**(*dst*, *src*, *casting='same_kind'*)

mpi_array.globale.RmaRedistributeUpdater.barrier

RmaRedistributeUpdater.**barrier**()
MPI barrier.

mpi_array.globale.RmaRedistributeUpdater.calc_can_use_existing_src_peer_comm

RmaRedistributeUpdater.**calc_can_use_existing_src_peer_comm**()
Returns True if self._src.locale_comms.peer_comm can be used to redistribute to the distribution of the self._dst array.

Return type bool

Returns True if self._src.locale_comms.peer_comm is a super-set of the processes of self._dst.locale_comms.peer_comm

mpi_array.globale.RmaRedistributeUpdater.calc_intersection_split

RmaRedistributeUpdater.**calc_intersection_split**(*dst_extent*, *src_extent*)
Calculates intersection between *dst_extent* and *{src_extent}*. Any regions of *dst_extent* which **do not** intersect with *src_extent* are returned as a list of *left-over* type(*dst_extent*) elements. The regions of *dst_extent* which **do** intersect with *src_extent* are returned as a list of *update* PairExtentUpdate elements. Returns tuple pair (leftovers, updates)

Parameters

- **dst_extent** (HaloIndexingExtent) – Extent which is to receive update from intersection with *src_extent*.
- **src_extent** (HaloIndexingExtent) – Extent which is to provide update for the intersecting region of *dst_extent*.

Return type tuple

Returns Returns tuple pair of (leftovers, updates).

mpi_array.globale.RmaRedistributeUpdater.check_updates

RmaRedistributeUpdater.**check_updates**()

Runs consistency checks on the calculated updates, assumes that the dst_distrib and src_distrib distributed as a partitioning (no locale extent overlaps except for halo).

Raises **RuntimeError** – If update inconsistency discovered.

mpi_array.globale.RmaRedistributeUpdater.create_pair_extent_update

RmaRedistributeUpdater.**create_pair_extent_update**(*dst_extent*, *src_extent*, *intersection_extent*)

Factory method which creates sequence of of mpi_array.distribution.MpiPairExtentUpdate objects.

mpi_array.globale.RmaRedistributeUpdater.do_locale_cpy2_update

RmaRedistributeUpdater.**do_locale_cpy2_update**()

Performs direct copy updates.

mpi_array.globale.RmaRedistributeUpdater.do_locale_rma_update

RmaRedistributeUpdater.**do_locale_rma_update**()

Performs RMA to get elements from remote locales to update the locale extent array.

mpi_array.globale.RmaRedistributeUpdater.do_locale_update

RmaRedistributeUpdater.**do_locale_update**()

mpi_array.globale.RmaRedistributeUpdater.do_update

RmaRedistributeUpdater.**do_update**()

mpi_array.globale.RmaRedistributeUpdater.get_cpy2_src_extents

RmaRedistributeUpdater.**get_cpy2_src_extents**(*dst_inter_locale_rank*)

mpi_array.globale.RmaRedistributeUpdater.initialise

RmaRedistributeUpdater.**initialise**()

mpi_array.globale.RmaRedistributeUpdater.initialise_cpy2_updates

RmaRedistributeUpdater.**initialise_cpy2_updates**()

mpi_array.globale.RmaRedistributeUpdater.initialise_rget_updates

RmaRedistributeUpdater.**initialise_rget_updates**()

mpi_array.globale.RmaRedistributeUpdater.initialise_updates

RmaRedistributeUpdater.**initialise_updates**()

mpi_array.globale.RmaRedistributeUpdater.wait_all

RmaRedistributeUpdater.**wait_all**(*req_list*)

1.18.2 Functions

copyto(dst, src[, casting])	Copy the elements of the *src* array to corresponding elements of the dst array.

mpi_array.globale.copyto

mpi_array.globale.**copyto**(*dst*, *src*, *casting='same_kind'*, ***kwargs*)

Copy the elements of the *src* array to corresponding elements of the dst array.

Parameters

- **dst** (*gndarray*) – Global array which receives elements.
- **src** (*gndarray*) – Global array from which elements are copied.
- **casting** (str) – See *casting* parameter in numpy.copyto().

1.19 The mpi_array.globale_test Module

Module defining *mpi_array.globale* unit-tests. Execute as:

```
python -m mpi_array.globale_test
```

and with parallelism:

```
mpirun -n  2 python -m mpi_array.globale_test
mpirun -n  4 python -m mpi_array.globale_test
mpirun -n 27 python -m mpi_array.globale_test
```

1.19.1 Classes

GndarrayTest([methodName])	unittest.TestCase for *mpi_array.globale.gndarray*.

mpi_array.globale_test.GndarrayTest

class mpi_array.globale_test.**GndarrayTest**(*methodName='runTest'*)

Bases: *mpi_array.unittest.TestCase*

unittest.TestCase for *mpi_array.globale.gndarray*.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
do_test_construct_empty_locale_extent([...])	Test constructing a *mpi_array.globale.gndarray* when locale extent is empty.
do_test_copyto_diff_locale_types([halo, ...])	Tests for *mpi_array.globale.copyto()*.
do_test_copyto_same_locale_types([halo, ...])	Tests for *mpi_array.globale.copyto()*.
do_test_peer_rank_get([locale_type])	Test for *mpi_array.globale.gndarray.peer_rank_get()*.
id()	
run([result])	
setUp()	
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.

Continued on next page

Table 1.132 – continued from previous page

subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_all()	Tests for mpi_array.globale.gndarray.all().
test_attr()	Test various attributes of mpi_array.globale.gndarray.
test_construct_empty_locale_extent_locale_type_nothing()	Test constructing a mpi_array.globale.gndarray when locale extent is empty.
test_construct_empty_locale_extent_locale_type_process()	Test constructing a mpi_array.globale.gndarray when locale extent is empty.
test_construct_with_structured_array_dtype()	Construct a a mpi_array.globale.gndarray structured array.
test_copyto_arg_check()	Tests for mpi_array.globale.copyto().
test_copyto_diff_locale_types_no_halo_different_mpi()	Tests for mpi_array.globale.copyto().
test_copyto_diff_locale_types_no_halo_same_mpi()	Tests for mpi_array.globale.copyto().
test_copyto_diff_locale_types_wt_halo_different_mpi()	Tests for mpi_array.globale.copyto().
test_copyto_diff_locale_types_wt_halo_same_mpi()	Tests for mpi_array.globale.copyto().
test_copyto_same_node_locale_types_no_halo_same_mpi_type()	Tests for mpi_array.globale.copyto().
test_copyto_same_node_locale_types_wt_halo_same_mpi_type()	Tests for mpi_array.globale.copyto().
test_copyto_same_process_locale_types_no_halo_same_mpi_type()	Tests for mpi_array.globale.copyto().
test_copyto_same_process_locale_types_wt_halo_same_mpi_type()	Tests for mpi_array.globale.copyto().
test_get_item_and_set_item()	Test the mpi_array.globale.gndarray.__getitem__() and mpi_array.globale.gndarray.__setitem__() methods.
test_peer_rank_get_locale_type_node()	Test for mpi_array.globale.gndarray.peer_rank_get().
test_peer_rank_get_locale_type_process()	Test for mpi_array.globale.gndarray.peer_rank_get().
test_update()	Test for mpi_array.globale.gndarray.update(), 1D and 2D shaped data with 1D distribution.
test_update_block()	Test for mpi_array.globale.gndarray.update(), block 2D distribution.

mpi_array.globale_test.GndarrayTest.__init__

GndarrayTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.globale_test.GndarrayTest.addCleanup

GndarrayTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.globale_test.GndarrayTest.addTypeEqualityFunc

GndarrayTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.globale_test.GndarrayTest.assertArraySplitEqual

GndarrayTest.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.globale_test.GndarrayTest.countTestCases

GndarrayTest.**countTestCases**()

mpi_array.globale_test.GndarrayTest.debug

GndarrayTest.**debug**()

Run the test without collecting errors in a TestResult

mpi_array.globale_test.GndarrayTest.defaultTestResult

GndarrayTest.**defaultTestResult**()

mpi_array.globale_test.GndarrayTest.doCleanups

GndarrayTest.**doCleanups**()

Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.globale_test.GndarrayTest.do_test_construct_empty_locale_extent

GndarrayTest.**do_test_construct_empty_locale_extent**(*locale_type='process'*, *halo=0*)
Test constructing a *mpi_array.globale.gndarray* when locale extent is empty.

mpi_array.globale_test.GndarrayTest.do_test_copyto_diff_locale_types

GndarrayTest.**do_test_copyto_diff_locale_types**(*halo=0*, *node_slab_dtype='int32'*, *proc_blok_dtype='int32'*)
Tests for *mpi_array.globale.copyto()*.

mpi_array.globale_test.GndarrayTest.do_test_copyto_same_locale_types

GndarrayTest.**do_test_copyto_same_locale_types**(*halo=0*, *dst_dtype='int32'*, *src_dtype='int32'*, *locale_type='process'*)
Tests for *mpi_array.globale.copyto()*.

mpi_array.globale_test.GndarrayTest.do_test_peer_rank_get

GndarrayTest.**do_test_peer_rank_get**(*locale_type='process'*)
Test for *mpi_array.globale.gndarray.peer_rank_get()*.

mpi_array.globale_test.GndarrayTest.id

GndarrayTest.**id**()

mpi_array.globale_test.GndarrayTest.run

GndarrayTest.**run**(*result=None*)

mpi_array.globale_test.GndarrayTest.setUp

GndarrayTest.**setUp**()

mpi_array.globale_test.GndarrayTest.setUpClass

GndarrayTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.globale_test.GndarrayTest.shortDescription

GndarrayTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.globale_test.GndarrayTest.skipTest

GndarrayTest.**skipTest**(*reason*)
Skip this test.

mpi_array.globale_test.GndarrayTest.subTest

GndarrayTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.globale_test.GndarrayTest.tearDown

GndarrayTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.globale_test.GndarrayTest.tearDownClass

GndarrayTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.globale_test.GndarrayTest.test_all

GndarrayTest.**test_all**()
Tests for *mpi_array.globale.gndarray.all()*.

mpi_array.globale_test.GndarrayTest.test_attr

GndarrayTest.**test_attr**()
Test various attributes of *mpi_array.globale.gndarray*.

mpi_array.globale_test.GndarrayTest.test_construct_empty_locale_extent_locale_type_node

GndarrayTest.**test_construct_empty_locale_extent_locale_type_node**()
Test constructing a *mpi_array.globale.gndarray* when locale extent is empty.

mpi_array.globale_test.GndarrayTest.test_construct_empty_locale_extent_locale_type_process

GndarrayTest.**test_construct_empty_locale_extent_locale_type_process**()
Test constructing a *mpi_array.globale.gndarray* when locale extent is empty.

mpi_array.globale_test.GndarrayTest.test_construct_with_structured_array_dtype

GndarrayTest.**test_construct_with_structured_array_dtype**()
Construct a a *mpi_array.globale.gndarray* structured array.

mpi_array.globale_test.GndarrayTest.test_copyto_arg_check

GndarrayTest.**test_copyto_arg_check**()
Tests for *mpi_array.globale.copyto()*.

mpi_array.globale_test.GndarrayTest.test_copyto_diff_locale_types_no_halo_diff_dtype

GndarrayTest.**test_copyto_diff_locale_types_no_halo_diff_dtype**()
Tests for *mpi_array.globale.copyto()*.

mpi_array.globale_test.GndarrayTest.test_copyto_diff_locale_types_no_halo_same_dtype

GndarrayTest.**test_copyto_diff_locale_types_no_halo_same_dtype**()
Tests for *mpi_array.globale.copyto()*.

mpi_array.globale_test.GndarrayTest.test_copyto_diff_locale_types_wt_halo_diff_dtype

GndarrayTest.**test_copyto_diff_locale_types_wt_halo_diff_dtype**()
Tests for *mpi_array.globale.copyto()*.

mpi_array.globale_test.GndarrayTest.test_copyto_diff_locale_types_wt_halo_same_dtype

GndarrayTest.**test_copyto_diff_locale_types_wt_halo_same_dtype**()
Tests for *mpi_array.globale.copyto()*.

mpi_array.globale_test.GndarrayTest.test_copyto_same_node_locale_types_no_halo_same_dtype

GndarrayTest.**test_copyto_same_node_locale_types_no_halo_same_dtype**()
Tests for *mpi_array.globale.copyto()*.

mpi_array.globale_test.GndarrayTest.test_copyto_same_node_locale_types_wt_halo_same_dtype

GndarrayTest.**test_copyto_same_node_locale_types_wt_halo_same_dtype**()
Tests for *mpi_array.globale.copyto()*.

mpi_array.globale_test.GndarrayTest.test_copyto_same_process_locale_types_no_halo_same_dtype

GndarrayTest.**test_copyto_same_process_locale_types_no_halo_same_dtype**()
Tests for *mpi_array.globale.copyto()*.

mpi_array.globale_test.GndarrayTest.test_copyto_same_process_locale_types_wt_halo_same_dtype

GndarrayTest.**test_copyto_same_process_locale_types_wt_halo_same_dtype**()
Tests for *mpi_array.globale.copyto()*.

mpi_array.globale_test.GndarrayTest.test_get_item_and_set_item

GndarrayTest.**test_get_item_and_set_item**()
Test the mpi_array.globale.gndarray.__getitem__() and mpi_array.globale.gndarray.__setitem__() methods.

mpi_array.globale_test.GndarrayTest.test_peer_rank_get_locale_type_node

GndarrayTest.**test_peer_rank_get_locale_type_node**()
Test for *mpi_array.globale.gndarray.peer_rank_get()*.

mpi_array.globale_test.GndarrayTest.test_peer_rank_get_locale_type_process

GndarrayTest.**test_peer_rank_get_locale_type_process**()
Test for *mpi_array.globale.gndarray.peer_rank_get()*.

mpi_array.globale_test.GndarrayTest.test_update

GndarrayTest.**test_update**()
Test for *mpi_array.globale.gndarray.update()*, 1D and 2D shaped data with 1D distribution.

mpi_array.globale_test.GndarrayTest.test_update_block

GndarrayTest.**test_update_block**()
Test for *mpi_array.globale.gndarray.update()*, block 2D distribution.

Attributes

longMessage
maxDiff

mpi_array.globale_test.GndarrayTest.longMessage

GndarrayTest.**longMessage** = True

mpi_array.globale_test.GndarrayTest.maxDiff

GndarrayTest.**maxDiff** = 640

1.20 The mpi_array.globale_creation Module

Defines *mpi_array.globale.gndarray* creation functions.

1.20.1 Ones and zeros

empty([shape, dtype, order, ...])	Creates array of uninitialised elements.
empty_like(ary[, dtype, order, subok])	Return a new array with the same shape and type as a given array.
eye(N[, M, k, dtype])	Not implemented.
identity(n[, dtype])	Not implemented.
ones([shape, dtype, comms_and_distrib, order])	Creates array of one-initialised elements.
ones_like(ary, *args, **kwargs)	Return a new one-initialised array with the same shape and type as a given array.
zeros([shape, dtype, order, comms_and_distrib])	Creates array of zero-initialised elements.
zeros_like(ary, *args, **kwargs)	Return a new zero-initialised array with the same shape and type as a given array.
full([shape, fill_value])	Return a new array of given shape and type, filled with `fill_value`.
full_like(ary, fill_value, *args, **kwargs)	Return a new array with the same shape and type as a given array.

mpi_array.globale_creation.empty

mpi_array.globale_creation.**empty**(*shape=None*, *dtype='float64'*, *order='C'*, *comms_and_distrib=None*, *intra_partition_dims=None*, ***kwargs*)

Creates array of uninitialised elements.

Parameters

- **shape** (`None` or sequence of `int`) – **Global** shape to be distributed amongst memory nodes.
- **dtype** (`numpy.dtype`) – Data type of array elements.
- **order** (*'C', 'F'*) – Only `'C'` implemented. Whether to store multi-dimensional data in row-major (C-style) or column-major (Fortran-style) order in memory.
- **comms_and_distrib** (`numpy.dtype`) – Data type of array elements.

Return type *mpi_array.globale.gndarray*

Returns Newly created array with uninitialised elements.

mpi_array.globale_creation.empty_like

mpi_array.globale_creation.**empty_like**(*ary*, *dtype=None*, *order='K'*, *subok=True*, ***kwargs*)

Return a new array with the same shape and type as a given array.

Parameters

- **ary** (`numpy.ndarray`) – Copy attributes from this array.
- **dtype** (`numpy.dtype`) – Specifies different dtype for the returned array.
- **order** (*'C', 'F', 'A', or 'K'*) – Only `'K'` implemented. Overrides the memory layout of the result. `'C'` means C-order, `'F'` means F-order, `'A'` means `'F'` if a is Fortran contiguous, `'C'` otherwise. `'K'` means match the layout of *ary* as closely as possible.
- **subok** (`bool`) – Ignored. If True, then the newly created array will use the sub-class type of *ary*, otherwise it will be a base-class array. Defaults to True.

Return type `type(ary)`

Returns Array of uninitialized (arbitrary) data with the same shape and type as *ary*.

mpi_array.globale_creation.eye

`mpi_array.globale_creation.`**`eye`**(*N*, *M=None*, *k=0*, *dtype=<class 'float'>*)

Not implemented. Return a 2-D array with ones on the diagonal and zeros elsewhere.

mpi_array.globale_creation.identity

`mpi_array.globale_creation.`**`identity`**(*n*, *dtype=None*)

Not implemented. Return the identity array.

mpi_array.globale_creation.ones

`mpi_array.globale_creation.`**`ones`**(*shape=None*, *dtype='float64'*, *comms_and_distrib=None*, *order='C'*, ***kwargs*)

Creates array of one-initialised elements.

Parameters

- **`shape`** (`None` or sequence of `int`) – **Global** shape to be distributed amongst memory nodes.
- **`dtype`** (`numpy.dtype`) – Data type of array elements.
- **`order`** (*'C', 'F'*) – Only `'C'` implemented. Whether to store multi-dimensional data in row-major (C-style) or column-major (Fortran-style) order in memory.
- **`comms_and_distrib`** (`numpy.dtype`) – Data type of array elements.

Return type *mpi_array.globale.gndarray*

Returns Newly created array with one-initialised elements.

mpi_array.globale_creation.ones_like

`mpi_array.globale_creation.`**`ones_like`**(*ary*, **args*, ***kwargs*)

Return a new one-initialised array with the same shape and type as a given array.

Parameters

- **`ary`** (*mpi_array.globale.gndarray*) – Copy attributes from this array.
- **`dtype`** (`numpy.dtype`) – Specifies different dtype for the returned array.
- **`order`** (*'C', 'F', 'A', or 'K'*) – Only `'K'` implemented. Overrides the memory layout of the result. `'C'` means C-order, `'F'` means F-order, `'A'` means `'F'` if a is Fortran contiguous, `'C'` otherwise. `'K'` means match the layout of *ary* as closely as possible.
- **`subok`** (`bool`) – Ignored. If True, then the newly created array will use the sub-class type of *ary*, otherwise it will be a base-class array. Defaults to True.

Return type *mpi_array.globale.gndarray*

Returns Array of one-initialized data with the same shape and type as *ary*.

mpi_array.globale_creation.zeros

mpi_array.globale_creation.**zeros**(*shape=None*, *dtype='float64'*, *order='C'*, *comms_and_distrib=None*, ***kwargs*)

Creates array of zero-initialised elements.

Parameters

- **shape** (None or sequence of int) – **Global** shape to be distributed amongst memory nodes.
- **dtype** (numpy.dtype) – Data type of array elements.
- **order** (*'C', 'F'*) – Only 'C' implemented. Whether to store multi-dimensional data in row-major (C-style) or column-major (Fortran-style) order in memory.
- **comms_and_distrib** (numpy.dtype) – Data type of array elements.

Return type *mpi_array.globale.gndarray*

Returns Newly created array with zero-initialised elements.

mpi_array.globale_creation.zeros_like

mpi_array.globale_creation.**zeros_like**(*ary*, **args*, ***kwargs*)

Return a new zero-initialised array with the same shape and type as a given array.

Parameters

- **ary** (*mpi_array.globale.gndarray*) – Copy attributes from this array.
- **dtype** (numpy.dtype) – Specifies different dtype for the returned array.
- **order** (*'C', 'F', 'A', or 'K'*) – Only 'K' implemented. Overrides the memory layout of the result. 'C' means C-order, 'F' means F-order, 'A' means 'F' if a is Fortran contiguous, 'C' otherwise. 'K' means match the layout of *ary* as closely as possible.
- **subok** (bool) – Ignored. If True, then the newly created array will use the sub-class type of *ary*, otherwise it will be a base-class array. Defaults to True.

Return type *mpi_array.globale.gndarray*

Returns Array of zero-initialized data with the same shape and type as *ary*.

mpi_array.globale_creation.full

mpi_array.globale_creation.**full**(*shape=None*, *fill_value=0*, **args*, ***kwargs*)

Return a new array of given shape and type, filled with fill_value.

Parameters

- **shape** (None or sequence of int) – **Global** shape to be distributed amongst memory nodes.
- **fill_value** (*scalar*) – Fill value.
- **dtype** (numpy.dtype) – Data type of array elements.
- **order** (*'C', 'F'*) – Only 'C' implemented. Whether to store multi-dimensional data in row-major (C-style) or column-major (Fortran-style) order in memory.
- **comms_and_distrib** (numpy.dtype) – Data type of array elements.

Return type *mpi_array.globale.gndarray*

Returns Newly created array with uninitialised elements.

mpi_array.globale_creation.full_like

mpi_array.globale_creation.**full_like**(*ary*, *fill_value*, **args*, ***kwargs*)

Return a new array with the same shape and type as a given array.

Parameters

- **ary** (numpy.ndarray) – Copy attributes from this array.
- **fill_value** (*scalar*) – Fill value.
- **dtype** (numpy.dtype) – Specifies different dtype for the returned array.
- **order** (*'C', 'F', 'A', or 'K'*) – Only 'K' implemented. Overrides the memory layout of the result. 'C' means C-order, 'F' means F-order, 'A' means 'F' if a is Fortran contiguous, 'C' otherwise. 'K' means match the layout of *ary* as closely as possible.
- **subok** (bool) – Ignored. If True, then the newly created array will use the sub-class type of *ary*, otherwise it will be a base-class array. Defaults to True.

Return type type(ary)

Returns Array of uninitialized (arbitrary) data with the same shape and type as *ary*.

1.20.2 From existing data

array(a[, dtype, copy, order, subok, ndmin])	Create a *mpi_array.globale.gndarray* from an existing *array-like* object.
asarray(a[, dtype, order])	Converts *a* (potentially via a copy) to a *mpi_array.globale.gndarray*.
asanyarray(a[, dtype, order])	Convert the input to an ndarray, but pass *mpi_array.globale.gndarray* subclasses through.
copy(ary, **kwargs)	Return an array copy of the given object.

mpi_array.globale_creation.array

mpi_array.globale_creation.**array**(*a*, *dtype=None*, *copy=True*, *order='K'*, *subok=False*, *ndmin=0*, ***kwargs*)

Create a *mpi_array.globale.gndarray* from an existing *array-like* object.

Parameters

- **object** (*array_like*) – An array, any object exposing the array interface, an object whose __array__ method returns an array, or any (nested) sequence.
- **dtype** (numpy.dtype) – The desired data-type for the array. If not given, then the type will be determined as the minimum type required to hold the objects in the sequence. This argument can only be used to *upcast* the array.
- **copy** (bool) – If True, then the object is copied. Otherwise, a copy will only be made if __array__ returns a copy, if a is a nested sequence, or if a copy is needed to satisfy any of the other requirements (dtype, order, etc.).

- **order** (*'K', 'A', 'C', 'F'*) – Only C implemented. Specify the memory layout of the array. If object is not an array, the newly created array will be in C order (row major)
- **subok** (bool) – If True, then sub-classes will be passed-through, otherwise the returned array will be forced to be a base-class array.
- **ndmin** (*int*) – Specifies the minimum number of dimensions that the resulting array should have. Ones will be pre-pended to the shape as needed to meet this requirement.

Return type *mpi_array.globale.gndarray*

Returns An array object satisfying the specified requirements.

See also:

asarray(), *asanyarray()*

mpi_array.globale_creation.asarray

mpi_array.globale_creation.**asarray**(*a*, *dtype=None*, *order=None*, ***kwargs*)

Converts *a* (potentially via a copy) to a *mpi_array.globale.gndarray*. The *kwargs* are as for the mpi_array.comms.create_distributon() function and determine the distribution for the returned *mpi_array.globale.gndarray*.

Parameters

- **a** (scalar, tuple, list, numpy.ndarray, etc) – Object converted to a *mpi_array.globale.gndarray*.
- **dtype** (numpy.dtype) – The numpy.dtype for the returned *mpi_array.globale.gndarray*.
- **order** (*'C', 'F'*) – Only 'C' implemented. Whether to store multi-dimensional data in row-major (C-style) or column-major (Fortran-style) order in memory. Defaults to 'C'.

Return type *mpi_array.globale.gndarray*

Returns The object a converted to an instance of *mpi_array.globale.gndarray*.

See also:

array(), *asanyarray()*

mpi_array.globale_creation.asanyarray

mpi_array.globale_creation.**asanyarray**(*a*, *dtype=None*, *order=None*, ***kwargs*)

Convert the input to an ndarray, but pass *mpi_array.globale.gndarray* subclasses through.

Parameters

- **a** (scalar, tuple, list, numpy.ndarray, etc) – Object converted to a *mpi_array.globale.gndarray*.
- **dtype** (numpy.dtype) – The numpy.dtype for the returned *mpi_array.globale.gndarray*.
- **order** (*'C', 'F'*) – Only 'C' implemented. Whether to store multi-dimensional data in row-major (C-style) or column-major (Fortran-style) order in memory. Defaults to 'C'.

Return type *mpi_array.globale.gndarray*

Returns The object a converted to an instance of *mpi_array.globale.gndarray*.

See also:

array(), *asarray()*

mpi_array.globale_creation.copy

mpi_array.globale_creation.**copy**(*ary*, ***kwargs*)
Return an array copy of the given object.

Parameters

- **ary** (*mpi_array.globale.gndarray*) – Array to copy.
- **order** (*'C', 'F'*) – Only 'C' implemented. Whether to store multi-dimensional data in row-major (C-style) or column-major (Fortran-style) order in memory. Defaults to 'C'.

Return type *mpi_array.globale.gndarray*

Returns A copy of *ary*.

1.21 The mpi_array.globale_creation_test Module

Module for testing creation/factory functions which generate instances of *mpi_array.globale.gndarray*. Execute as:

```
python -m mpi_array.globale_creation_test
```

and with parallelism:

```
mpirun -n  2 python -m mpi_array.globale_creation_test
mpirun -n  4 python -m mpi_array.globale_creation_test
mpirun -n 27 python -m mpi_array.globale_creation_test
```

1.21.1 Classes

GndarrayCreationTest([methodName])	unittest.TestCase for *mpi_array.globale.gndarray()* instance generation.

mpi_array.globale_creation_test.GndarrayCreationTest

class mpi_array.globale_creation_test.**GndarrayCreationTest**(*methodName='runTest'*)
Bases: *mpi_array.unittest.TestCase*

unittest.TestCase for *mpi_array.globale.gndarray()* instance generation.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.

Continued on next page

Table 1.137 – continued from previous page

addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	Hook method for setting up the test fixture before exercising it.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_asanyarray_with_subclass()	`unittest.TestCase` for *mpi_array.globale_creation.asanyarray()*.
test_asanyarray_with_tuple()	`unittest.TestCase` for *mpi_array.globale_creation.asanyarray()*.
test_asarray_with_scalar()	`unittest.TestCase` for *mpi_array.globale_creation.asarray()*.
test_asarray_with_subclass()	`unittest.TestCase` for *mpi_array.globale_creation.asarray()*.
test_asarray_with_tuple()	`unittest.TestCase` for *mpi_array.globale_creation.asarray()*.
test_copy_non_shared_1d()	Test for *mpi_array.globale_creation.copy()*.
test_copy_shared_1d()	Test for *mpi_array.globale_creation.copy()*.
test_empty_non_shared_1d()	Test for *mpi_array.globale_creation.empty()* and *mpi_array.globale_creation.empty_like()*.
test_empty_scalar()	Test for `mpi_array.globale.empty()` and `mpi_array.globale.empty_like()`.
test_empty_shared_1d()	Test for `mpi_array.globale.empty()` and `mpi_array.globale.empty_like()`.
test_ones_non_shared_1d()	Test for *mpi_array.globale_creation.ones()* and *mpi_array.globale_creation.ones_like()*.

Continued on next page

Table 1.137 – continued from previous page

test_ones_shared_1d()	Test for *mpi_array.globale_creation.ones()* and *mpi_array.globale_creation.ones_like()*.
test_zeros_non_shared_1d()	Test for *mpi_array.globale_creation.zeros()* and *mpi_array.globale_creation.zeros_like()*.
test_zeros_shared_1d()	Test for *mpi_array.globale_creation.zeros()* and *mpi_array.globale_creation.zeros_like()*.

mpi_array.globale_creation_test.GndarrayCreationTest.__init__

GndarrayCreationTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.globale_creation_test.GndarrayCreationTest.addCleanup

GndarrayCreationTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.globale_creation_test.GndarrayCreationTest.addTypeEqualityFunc

GndarrayCreationTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.globale_creation_test.GndarrayCreationTest.assertArraySplitEqual

GndarrayCreationTest.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.globale_creation_test.GndarrayCreationTest.countTestCases

GndarrayCreationTest.**countTestCases**()

mpi_array.globale_creation_test.GndarrayCreationTest.debug

GndarrayCreationTest.**debug**()
Run the test without collecting errors in a TestResult

mpi_array.globale_creation_test.GndarrayCreationTest.defaultTestResult

GndarrayCreationTest.**defaultTestResult**()

mpi_array.globale_creation_test.GndarrayCreationTest.doCleanups

GndarrayCreationTest.**doCleanups**()
Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.globale_creation_test.GndarrayCreationTest.id

GndarrayCreationTest.**id**()

mpi_array.globale_creation_test.GndarrayCreationTest.run

GndarrayCreationTest.**run**(*result=None*)

mpi_array.globale_creation_test.GndarrayCreationTest.setUp

GndarrayCreationTest.**setUp**()
Hook method for setting up the test fixture before exercising it.

mpi_array.globale_creation_test.GndarrayCreationTest.setUpClass

GndarrayCreationTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.globale_creation_test.GndarrayCreationTest.shortDescription

GndarrayCreationTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.globale_creation_test.GndarrayCreationTest.skipTest

GndarrayCreationTest.**skipTest**(*reason*)
Skip this test.

mpi_array.globale_creation_test.GndarrayCreationTest.subTest

GndarrayCreationTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.globale_creation_test.GndarrayCreationTest.tearDown

GndarrayCreationTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.globale_creation_test.GndarrayCreationTest.tearDownClass

GndarrayCreationTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.globale_creation_test.GndarrayCreationTest.test_asanyarray_with_subclass

GndarrayCreationTest.**test_asanyarray_with_subclass**()
unittest.TestCase for *mpi_array.globale_creation.asanyarray()*.

mpi_array.globale_creation_test.GndarrayCreationTest.test_asanyarray_with_tuple

GndarrayCreationTest.**test_asanyarray_with_tuple**()
unittest.TestCase for *mpi_array.globale_creation.asanyarray()*.

mpi_array.globale_creation_test.GndarrayCreationTest.test_asarray_with_scalar

GndarrayCreationTest.**test_asarray_with_scalar**()
unittest.TestCase for *mpi_array.globale_creation.asarray()*.

mpi_array.globale_creation_test.GndarrayCreationTest.test_asarray_with_subclass

GndarrayCreationTest.**test_asarray_with_subclass**()
unittest.TestCase for *mpi_array.globale_creation.asarray()*.

mpi_array.globale_creation_test.GndarrayCreationTest.test_asarray_with_tuple

GndarrayCreationTest.**test_asarray_with_tuple**()
unittest.TestCase for *mpi_array.globale_creation.asarray()*.

mpi_array.globale_creation_test.GndarrayCreationTest.test_copy_non_shared_1d

GndarrayCreationTest.**test_copy_non_shared_1d**()
Test for *mpi_array.globale_creation.copy()*.

mpi_array.globale_creation_test.GndarrayCreationTest.test_copy_shared_1d

GndarrayCreationTest.**test_copy_shared_1d**()
Test for *mpi_array.globale_creation.copy()*.

mpi_array.globale_creation_test.GndarrayCreationTest.test_empty_non_shared_1d

GndarrayCreationTest.**test_empty_non_shared_1d**()
Test for *mpi_array.globale_creation.empty()* and *mpi_array.globale_creation.empty_like()*.

mpi_array.globale_creation_test.GndarrayCreationTest.test_empty_scalar

GndarrayCreationTest.**test_empty_scalar**()
Test for mpi_array.globale.empty() and mpi_array.globale.empty_like().

mpi_array.globale_creation_test.GndarrayCreationTest.test_empty_shared_1d

GndarrayCreationTest.**test_empty_shared_1d**()
Test for mpi_array.globale.empty() and mpi_array.globale.empty_like().

mpi_array.globale_creation_test.GndarrayCreationTest.test_ones_non_shared_1d

GndarrayCreationTest.**test_ones_non_shared_1d**()
Test for *mpi_array.globale_creation.ones()* and *mpi_array.globale_creation.ones_like()*.

mpi_array.globale_creation_test.GndarrayCreationTest.test_ones_shared_1d

GndarrayCreationTest.**test_ones_shared_1d**()
Test for *mpi_array.globale_creation.ones()* and *mpi_array.globale_creation.ones_like()*.

mpi_array.globale_creation_test.GndarrayCreationTest.test_zeros_non_shared_1d

GndarrayCreationTest.**test_zeros_non_shared_1d**()
Test for *mpi_array.globale_creation.zeros()* and *mpi_array.globale_creation.zeros_like()*.

mpi_array.globale_creation_test.GndarrayCreationTest.test_zeros_shared_1d

GndarrayCreationTest.**test_zeros_shared_1d**()
Test for *mpi_array.globale_creation.zeros()* and *mpi_array.globale_creation.zeros_like()*.

Attributes

longMessage
maxDiff

mpi_array.globale_creation_test.GndarrayCreationTest.longMessage

GndarrayCreationTest.**longMessage = True**

mpi_array.globale_creation_test.GndarrayCreationTest.maxDiff

GndarrayCreationTest.**maxDiff = 640**

1.22 The mpi_array.globale_ufunc Module

Defines numpy.ufunc functions for *mpi_array.globale.gndarray*.

1.22.1 Classes

GndarrayArrayUfuncExecutor(array_like_obj, ...)	Instances execute a ufunc for a *mpi_array.globale.gndarray*.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor

class mpi_array.globale_ufunc.**GndarrayArrayUfuncExecutor**(*array_like_obj*, *ufunc*, *method*, **inputs*, ***kwargs*)

Bases: object

Instances execute a ufunc for a *mpi_array.globale.gndarray*. Takes care of creating outputs, remote fetching of required parts of inputs and forwarding call to numpy.ufunc instance to perform the computation on the locale numpy.ndarray instances.

Methods

__init__(array_like_obj, ufunc, method, ...)	Initialise.
GndarrayArrayUfuncExecutor.check_equivalent_inter_locale_comms	

Continued on next page

Table 1.140 – continued from previous page

create_outputs(outputs, result_shape, ...)	Returns list of output *mpi_array.globale.gndarray* instances.
execute()	Perform the ufunc operation.
execute___call__()	
execute_accumulate()	Not implemented.
execute_at()	Not implemented.
execute_outer()	Not implemented.
execute_reduce()	Not implemented.
execute_reduceat()	Not implemented.
get_best_match_input(result_shape)	Returns the element of *inputs* whose globale shape best matches *result_shape*.
get_input_extents(locale_info)	Returns tuple of `(locale_extent, globale_extent)` pairs, one for each of the *inputs*.
get_inputs_shapes()	Returns a *shape* `tuple` for each element of *inputs*.
get_numpy_ufunc_peer_rank_inputs_outputs()	Returns two element tuple of `(input_arrays, output_arrays)` which are to be passed to the `numpy.ufunc` object *ufunc*.
need_remote_data(gndarray_outputs)	Returns `True` if any locale needs to fetch remote input data in order to compute the all elements of the outputs *gndarray_outputs*.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.__init__

GndarrayArrayUfuncExecutor.**__init__**(*array_like_obj*, *ufunc*, *method*, **inputs*, ***kwargs*)
Initialise.

Parameters

- **array_like_obj** (*mpi_array.globale.gndarray*) – The *mpi_array.globale.gndarray* which triggered the __array_ufunc__ call.
- **ufunc** (`numpy.ufunc`) – The ufunc to be executed.
- **method** (`str`) – The name of the method of *ufunc* which is to be executed.
- **inputs** (*array like*) – The ufunc inputs.
- **kwargs** (*keyword args*) – The ufunc keyword arguments.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.create_outputs

GndarrayArrayUfuncExecutor.**create_outputs**(*outputs*, *result_shape*, *result_types*)
Returns list of output *mpi_array.globale.gndarray* instances.

Parameters

- **outputs** (`None` or `tuple` of *mpi_array.globale.gndarray*) – Output arrays passed in as the `out` argument of the `numpy.ufunc`.
- **result_shape** (sequence of `int`) – The shape of all output arrays.
- **result_types** (sequence of `numpy.dtype`) – The `dtype` of each output array. Note that this is the list for all outputs including any in the `outputs` argument. This determines the number of output arrays.

Return type list of *mpi_array.globale.gndarray*

Returns A list of length `len(result_types)` elements, each element is a *mpi_array.globale.gndarray*.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.execute

`GndarrayArrayUfuncExecutor.`**`execute`**`()`

Perform the ufunc operation. Call is forwarded to one of: *execute___call__()*, *execute_accumulate()*, *execute_at()* , *execute_outer()*, *execute_reduce()* or *execute_reduceat()*.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.execute___call__

`GndarrayArrayUfuncExecutor.`**`execute___call__`**`()`

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.execute_accumulate

`GndarrayArrayUfuncExecutor.`**`execute_accumulate`**`()`

Not implemented.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.execute_at

`GndarrayArrayUfuncExecutor.`**`execute_at`**`()`

Not implemented.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.execute_outer

`GndarrayArrayUfuncExecutor.`**`execute_outer`**`()`

Not implemented.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.execute_reduce

`GndarrayArrayUfuncExecutor.`**`execute_reduce`**`()`

Not implemented.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.execute_reduceat

`GndarrayArrayUfuncExecutor.`**`execute_reduceat`**`()`

Not implemented.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.get_best_match_input

`GndarrayArrayUfuncExecutor.`**`get_best_match_input`**`(`*result_shape*`)`

Returns the element of *inputs* whose globale shape best matches *result_shape*.

Return type None or *mpi_array.globale.gndarray*.

Returns The input array whose shape matches *result_shape*, or None if none of the inputs are a good match.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.get_input_extents

GndarrayArrayUfuncExecutor.**get_input_extents**(*locale_info*)

Returns tuple of (locale_extent, globale_extent) pairs, one for each of the *inputs*.

Parameters **locale_info** (*mpi_array.comms.ThisLocaleInfo*) – The rank info required for constructing a *mpi_array.distribution.LocaleExtent* instance for input types which are not *mpi_array.globale.gndarray*.

Return type tuple

Returns Pairs which indicate the locale extent of the ufunc *inputs*.

See also:

get_extents()

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.get_inputs_shapes

GndarrayArrayUfuncExecutor.**get_inputs_shapes**()

Returns a *shape* tuple for each element of *inputs*.

Return type tuple

Returns Shape of each ufunc input.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.get_numpy_ufunc_peer_rank_inputs_outputs

GndarrayArrayUfuncExecutor.**get_numpy_ufunc_peer_rank_inputs_outputs**(*gndarray_outputs*)

Returns two element tuple of (input_arrays, output_arrays) which are to be passed to the numpy.ufunc object *ufunc*.

Parameters **gndarray_outputs** (sequence of *mpi_array.globale.gndarray*) – The output arrays. All arrays should be the same shape and same distribution.

Return type None or tuple

Returns A tuple (input_arrays, output_arrays) of inputs and outputs which are to be passed to numpy.ufunc call. Returns None if the output locale extents are empty (i.e. no array elements to compute on this locale).

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.need_remote_data

GndarrayArrayUfuncExecutor.**need_remote_data**(*gndarray_outputs*)

Returns True if any locale needs to fetch remote input data in order to compute the all elements of the outputs *gndarray_outputs*.

Parameters **gndarray_outputs** (sequence of *mpi_array.globale.gndarray*) – Check whether any of the locales require remote data in order to compute these outputs.

Return type bool

Returns True if remote fetch of input data is required in order to compute ufunc for the given outputs.

Attributes

array_like_obj	The *mpi_array.globale.gndarray* object which triggered the construction of this *GndarrayArrayUfuncExecutor* object.
casting	A `str` indicating the casting mode.
inputs	The sequence of ufunc inputs.
inter_locale_comm	The inter-locale `mpi4py.MPI.Comm` communicator.
intra_locale_comm	The intra-locale `mpi4py.MPI.Comm` communicator.
method	A `str` indicating the method of the *ufunc* to be executed.
outputs	The ufunc *mpi_array.globale.gndarray* output arrays.
peer_comm	The peer `mpi4py.MPI.Comm` communicator.
ufunc	The `numpy.ufunc` to be executed.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.array_like_obj

GndarrayArrayUfuncExecutor.**array_like_obj**
The *mpi_array.globale.gndarray* object which triggered the construction of this *GndarrayArrayUfuncExecutor* object.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.casting

GndarrayArrayUfuncExecutor.**casting**
A `str` indicating the casting mode.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.inputs

GndarrayArrayUfuncExecutor.**inputs**
The sequence of ufunc inputs.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.inter_locale_comm

GndarrayArrayUfuncExecutor.**inter_locale_comm**
The inter-locale `mpi4py.MPI.Comm` communicator.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.intra_locale_comm

GndarrayArrayUfuncExecutor.**intra_locale_comm**
The intra-locale `mpi4py.MPI.Comm` communicator.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.method

GndarrayArrayUfuncExecutor.**method**
A `str` indicating the method of the *ufunc* to be executed.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.outputs

GndarrayArrayUfuncExecutor.**outputs**
The ufunc *mpi_array.globale.gndarray* output arrays.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.peer_comm

GndarrayArrayUfuncExecutor.**peer_comm**
The peer `mpi4py.MPI.Comm` communicator.

mpi_array.globale_ufunc.GndarrayArrayUfuncExecutor.ufunc

GndarrayArrayUfuncExecutor.**ufunc**
The `numpy.ufunc` to be executed.

1.22.2 Functions

get_dtype_and_ndim(array_like)	Returns `(dtype, ndim)` pair for the given *array_like* argument.
ufunc_result_type(ufunc_types, inputs[, ...])	Attempts to calculate the result type from given ufunc *inputs* and ufunc types (`numpy.ufunc.types`).
broadcast_shape(*shape_args)	Returns the `numpy` broadcast shape for the give shape arguments.
shape_extend_dims(ndim, shape)	Returns `shape` pre-prepended with ones so returned 1D array has length *ndim*.
gndarray_array_ufunc(array_like_obj, ufunc, ...)	The implementation for `mpi_array.globale.gndarray.__array_ufunc__()`.

mpi_array.globale_ufunc.get_dtype_and_ndim

mpi_array.globale_ufunc.**get_dtype_and_ndim**(*array_like*)
Returns `(dtype, ndim)` pair for the given *array_like* argument. If the *array_like* has *both* `"dtype"` and `"ndim"` attributes, then the return tuple is `(`*array_like*`.dtype, `*array_like*`.ndim)`. Otherwise, returns `(numpy.asanyarray(`*array_like*`).dtype, numpy.asanyarray(`*array_like*`).ndim)`.

Parameters **array_like** (castable to `numpy.ndarray`) – Returns dtype and ndim for this object.

Return type two element `tuple`

Returns The `numpy.dtype` and integer `ndim` properties for *array_like*.

Example:

```
>>> get_dtype_and_ndim(1.0)
(dtype('float64'), 0)
>>> get_dtype_and_ndim((1.0, 2.0, 3.0, 4.0))
(dtype('float64'), 1)
>>> get_dtype_and_ndim([(1.0, 2.0, 3.0, 4.0), (5.0, 6.0, 7.0, 8.0)])
(dtype('float64'), 2)
```

mpi_array.globale_ufunc.ufunc_result_type

mpi_array.globale_ufunc.**ufunc_result_type**(*ufunc_types*, *inputs*, *outputs=None*, *casting='safe'*, *input_match_casting='safe'*)

Attempts to calculate the result type from given ufunc `inputs` and ufunc types (`numpy.ufunc.types`). Like `numpy.result_type`, but handles *mpi_array.globale.gndarray* in the `inputs` and handles multiple `outputs` cases.

Parameters

- **ufunc_types** (sequence of *str*) – The `numpy.ufunc.types` attribute, e.g. `['??->?', 'bb->b', 'BB->B', 'hh->h', 'HH->H', ..., 'mm->m', 'mM->M', 'OO->O']`.
- **inputs** (sequence of `object`) – The inputs (e.g. `numpy.ndarray`, scalars or *mpi_array.globale.gndarray*) to a `numpy.ufunc` call.
- **outputs** (`None` or sequence of `object`) – The output arrays these are explicitly checked casting correctness.
- **casting** (`str` *'no', 'equiv', 'safe', 'same_kind', 'unsafe'*) – Casting mode applied to outputs. See `numpy.can_cast()`.
- **input_match_casting** (`str` *'no', 'equiv', 'safe', 'same_kind', 'unsafe'*) – Casting mode applied to match `ufunc_types` inputs with the `inputs`. See `numpy.can_cast()`.

Return type `tuple` of `numpy.dtype`

Returns A tuple of `numpy.dtype` indicating the output types produced for the given inputs.

Raises **ValueError** – If the the inputs (and outputs) cannot be cast to an appropriate element of `ufunc_types`.

Example:

```
>>> import numpy as np
>>> import mpi_array as mpia
>>> inp = (
... np.zeros((10,10,10), dtype='float16'),
... 16.0,
... mpia.zeros((10,10,10), dtype='float32'),
... )
>>> ufunc_result_type(['eee->e?', 'fff->f?', 'ddd->d?'], inputs=inp)
(dtype('float32'), dtype('bool'))
>>> out = (mpia.zeros((10,10,10), dtype="float64"),)
>>> ufunc_result_type(['eee->e?', 'fff->f?', 'ddd->d?'], inputs=inp, outputs=out)
(dtype('float64'), dtype('bool'))
>>> out += (mpia.zeros((10, 10, 10), dtype="uint16"),)
>>> ufunc_result_type(['eee->e?', 'fff->f?', 'ddd->d?'], inputs=inp, outputs=out)
(dtype('float64'), dtype('uint16'))
>>> mpia.free_all(inp + out)
```

mpi_array.globale_ufunc.broadcast_shape

mpi_array.globale_ufunc.**broadcast_shape**(**shape_args*)

Returns the `numpy` broadcast shape for the give shape arguments.

Parameters **shape2, ...** (*shape1,*) – Array shapes to be broadcast.

Return type sequence of *int*

Returns The broadcast shape.

Examples:

```
>>> broadcast_shape((4,), (4,))
(4,)
>>> broadcast_shape((4, 1), (1, 5))
(4, 5)
>>> broadcast_shape((4, 1, 3, 7), (1, 8, 1, 7))
(4, 8, 3, 7)
>>> broadcast_shape((3, 7), ())
(3, 7)
```

mpi_array.globale_ufunc.shape_extend_dims

mpi_array.globale_ufunc.**shape_extend_dims**(*ndim*, *shape*)

Returns `shape` pre-prepended with ones so returned 1D array has length *`ndim`*.

Parameters

- **ndim** (`int`) – Length of returned 1D sequence.
- **shape** (sequence of `object`) – Length of returned 1D sequence.

Return type `tuple`

Returns Sequence pre-pended with one elements so that sequence length equals *`ndim`*.

Example:

```
>>> shape_extend_dims(5, (3, 1, 5))
(1, 1, 3, 1, 5)
>>> shape_extend_dims(3, (3, 1, 5))
(3, 1, 5)
>>> shape_extend_dims(1, (3, 1, 5))
(3, 1, 5)
```

mpi_array.globale_ufunc.gndarray_array_ufunc

mpi_array.globale_ufunc.**gndarray_array_ufunc**(*array_like_obj*, *ufunc*, *method*, **inputs*, ***kwargs*)

The implementation for `mpi_array.globale.gndarray.__array_ufunc__()`.

1.23 The `mpi_array.globale_ufunc_test` Module

Module defining *`mpi_array.globale`* unit-tests. Execute as:

```
python -m mpi_array.globale_ufunc_test
```

and with parallelism:

```
mpirun -n  2 python -m mpi_array.globale_ufunc_test
mpirun -n  4 python -m mpi_array.globale_ufunc_test
mpirun -n 27 python -m mpi_array.globale_ufunc_test
```

1.23.1 Classes

UfuncResultTypeTest([methodName])	unittest.TestCase for *mpi_array.globale_ufunc.ufunc_result_type()*.
BroadcastShapeTest([methodName])	unittest.TestCase for *mpi_array.globale_ufunc.broadcast_shape()*.
GndarrayUfuncTest([methodName])	unittest.TestCase for *mpi_array.globale_ufunc*.
ToGndarrayConverter(**kwargs)	Base class for converting numpy.ndarray objects to *mpi_array.globale.gndarray* objects.

mpi_array.globale_ufunc_test.UfuncResultTypeTest

class mpi_array.globale_ufunc_test.**UfuncResultTypeTest**(*methodName='runTest'*)

Bases: *mpi_array.unittest.TestCase*

unittest.TestCase for *mpi_array.globale_ufunc.ufunc_result_type()*.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	Hook method for setting up the test fixture before exercising it.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_example()	

Continued on next page

Table 1.144 – continued from previous page

test_multiple_output()	unittest.TestCase for *mpi_array.globale_ufunc.ufunc_result_type()*,
test_single_output()	unittest.TestCase for *mpi_array.globale_ufunc.ufunc_result_type()*,
test_tuple_input()	unittest.TestCase for *mpi_array.globale_ufunc.ufunc_result_type()*,

mpi_array.globale_ufunc_test.UfuncResultTypeTest.__init__

UfuncResultTypeTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.globale_ufunc_test.UfuncResultTypeTest.addCleanup

UfuncResultTypeTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.globale_ufunc_test.UfuncResultTypeTest.addTypeEqualityFunc

UfuncResultTypeTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.globale_ufunc_test.UfuncResultTypeTest.assertArraySplitEqual

UfuncResultTypeTest.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.globale_ufunc_test.UfuncResultTypeTest.countTestCases

UfuncResultTypeTest.**countTestCases**()

mpi_array.globale_ufunc_test.UfuncResultTypeTest.debug

UfuncResultTypeTest.**debug**()
Run the test without collecting errors in a TestResult

mpi_array.globale_ufunc_test.UfuncResultTypeTest.defaultTestResult

UfuncResultTypeTest.**defaultTestResult**()

mpi_array.globale_ufunc_test.UfuncResultTypeTest.doCleanups

UfuncResultTypeTest.**doCleanups**()
Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.globale_ufunc_test.UfuncResultTypeTest.id

UfuncResultTypeTest.**id**()

mpi_array.globale_ufunc_test.UfuncResultTypeTest.run

UfuncResultTypeTest.**run**(*result=None*)

mpi_array.globale_ufunc_test.UfuncResultTypeTest.setUp

UfuncResultTypeTest.**setUp**()
Hook method for setting up the test fixture before exercising it.

mpi_array.globale_ufunc_test.UfuncResultTypeTest.setUpClass

UfuncResultTypeTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.globale_ufunc_test.UfuncResultTypeTest.shortDescription

UfuncResultTypeTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.globale_ufunc_test.UfuncResultTypeTest.skipTest

UfuncResultTypeTest.**skipTest**(*reason*)
Skip this test.

mpi_array.globale_ufunc_test.UfuncResultTypeTest.subTest

UfuncResultTypeTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.globale_ufunc_test.UfuncResultTypeTest.tearDown

UfuncResultTypeTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.globale_ufunc_test.UfuncResultTypeTest.tearDownClass

UfuncResultTypeTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.globale_ufunc_test.UfuncResultTypeTest.test_example

UfuncResultTypeTest.**test_example**()

mpi_array.globale_ufunc_test.UfuncResultTypeTest.test_multiple_output

UfuncResultTypeTest.**test_multiple_output**()
unittest.TestCase for *mpi_array.globale_ufunc.ufunc_result_type()*, with multiple output arrays.

mpi_array.globale_ufunc_test.UfuncResultTypeTest.test_single_output

UfuncResultTypeTest.**test_single_output**()
unittest.TestCase for *mpi_array.globale_ufunc.ufunc_result_type()*, with single output array.

mpi_array.globale_ufunc_test.UfuncResultTypeTest.test_tuple_input

UfuncResultTypeTest.**test_tuple_input**()
unittest.TestCase for *mpi_array.globale_ufunc.ufunc_result_type()*, with single output array.

Attributes

longMessage	
maxDiff	

mpi_array.globale_ufunc_test.UfuncResultTypeTest.longMessage

UfuncResultTypeTest.**longMessage** = True

mpi_array.globale_ufunc_test.UfuncResultTypeTest.maxDiff

UfuncResultTypeTest.**maxDiff** = 640

mpi_array.globale_ufunc_test.BroadcastShapeTest

class mpi_array.globale_ufunc_test.**BroadcastShapeTest**(*methodName='runTest'*)

Bases: mpi_array.unittest.TestCase

unittest.TestCase for mpi_array.globale_ufunc.broadcast_shape().

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	Hook method for setting up the test fixture before exercising it.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.

Continued on next page

Table 1.146 – continued from previous page

tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_broadcastable()	Asserts for variety of broadcastable shapes.
test_non_broadcastable()	Test that *mpi_array.globale_ufunc.broadcast_shape()* raises a ValueError if shapes are not broadcastable.

mpi_array.globale_ufunc_test.BroadcastShapeTest.__init__

BroadcastShapeTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.globale_ufunc_test.BroadcastShapeTest.addCleanup

BroadcastShapeTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.globale_ufunc_test.BroadcastShapeTest.addTypeEqualityFunc

BroadcastShapeTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.globale_ufunc_test.BroadcastShapeTest.assertArraySplitEqual

BroadcastShapeTest.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.globale_ufunc_test.BroadcastShapeTest.countTestCases

BroadcastShapeTest.**countTestCases**()

mpi_array.globale_ufunc_test.BroadcastShapeTest.debug

BroadcastShapeTest.**debug**()
Run the test without collecting errors in a TestResult

mpi_array.globale_ufunc_test.BroadcastShapeTest.defaultTestResult

BroadcastShapeTest.**defaultTestResult**()

mpi_array.globale_ufunc_test.BroadcastShapeTest.doCleanups

BroadcastShapeTest.**doCleanups**()
Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.globale_ufunc_test.BroadcastShapeTest.id

BroadcastShapeTest.**id**()

mpi_array.globale_ufunc_test.BroadcastShapeTest.run

BroadcastShapeTest.**run**(*result=None*)

mpi_array.globale_ufunc_test.BroadcastShapeTest.setUp

BroadcastShapeTest.**setUp**()
Hook method for setting up the test fixture before exercising it.

mpi_array.globale_ufunc_test.BroadcastShapeTest.setUpClass

BroadcastShapeTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.globale_ufunc_test.BroadcastShapeTest.shortDescription

BroadcastShapeTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.globale_ufunc_test.BroadcastShapeTest.skipTest

BroadcastShapeTest.**skipTest**(*reason*)
Skip this test.

mpi_array.globale_ufunc_test.BroadcastShapeTest.subTest

BroadcastShapeTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.globale_ufunc_test.BroadcastShapeTest.tearDown

BroadcastShapeTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.globale_ufunc_test.BroadcastShapeTest.tearDownClass

BroadcastShapeTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.globale_ufunc_test.BroadcastShapeTest.test_broadcastable

BroadcastShapeTest.**test_broadcastable**()
Asserts for variety of broadcastable shapes.

mpi_array.globale_ufunc_test.BroadcastShapeTest.test_non_broadcastable

BroadcastShapeTest.**test_non_broadcastable**()
Test that *mpi_array.globale_ufunc.broadcast_shape()* raises a ValueError if shapes are not broadcastable.

Attributes

longMessage
maxDiff

mpi_array.globale_ufunc_test.BroadcastShapeTest.longMessage

BroadcastShapeTest.**longMessage** = True

mpi_array.globale_ufunc_test.BroadcastShapeTest.maxDiff

BroadcastShapeTest.**maxDiff** = 640

mpi_array.globale_ufunc_test.GndarrayUfuncTest

class mpi_array.globale_ufunc_test.**GndarrayUfuncTest**(*methodName='runTest'*)

Bases: *mpi_array.unittest.TestCase*

unittest.TestCase for *mpi_array.globale_ufunc*.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
compare_results(mpi_cln_npy_result_ary, ...)	Asserts that all elements of the *mpi_array.globale.gndarray* *mpi_cln_npy_result_ary* equal all elements of the *mpi_array.globale.gndarray* *mpi_result_ary*.
convert_func_args_to_gndarrays(converter, ...)	**type converter** *ToGndarrayConverter*
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
do_block_distribution_test(...)	Converts numpy.ndarray elements of func_args to *mpi_array.globale.gndarray* instances distributed as the *mpi_array.comms.DT_BLOCK* distribution type.
do_cloned_distribution_test(...)	Converts numpy.ndarray elements of func_args to *mpi_array.globale.gndarray* instances distributed as the *mpi_array.comms.DT_CLONED* distribution type.
do_convert_execute_and_compare(...)	Compares the result of *func* called with self.convert_func_args_to_gndarrays(*converter*, *func_args*) converted arguments with the *mpi_cln_npy_result_ary* array (which should have been produced by calling mpi_array.globale_creation.asarray(*func*(*func_args))).
do_multi_distribution_tests(func, *func_args)	Compares result of *func* called with numpy.ndarray arguments and result of *func* called with *mpi_array.globale.gndarray* arguments.

Continued on next page

Table 1.148 – continued from previous page

do_single_locale_distribution_test(...)	Converts `numpy.ndarray` elements of `func_args` to *mpi_array.globale.gndarray* instances distributed as the *mpi_array.comms.DT_SINGLE_LOCALE* distribution type.
do_test_umath([halo, gshape])	Test binary op for a *mpi_array.globale.gndarray* object and a scalar.
do_test_umath_broadcast([halo, dims])	Test binary op for a *mpi_array.globale.gndarray* objects and an *array-like* object which requires requiring broadcast to result shape.
do_test_umath_broadcast_upsized_result([halo])	Test binary op for two *mpi_array.globale.gndarray* objects with the resulting *mpi_array.globale.gndarray* object having different (larger) shape than that of both inputs.
do_test_umath_distributed_broadcast([...])	Test binary op for two *mpi_array.globale.gndarray* objects which requires remote fetch of data when broadcasting to result shape.
id()	
run([result])	
setUp()	Initialise `numpy.random.seed()`.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_umath_broadcast_halo()	Test binary op for a *mpi_array.globale.gndarray* objects and an *array-like* object which requires requiring broadcast to result shape.
test_umath_broadcast_no_halo()	Test binary op for a *mpi_array.globale.gndarray* objects and an *array-like* object which requires requiring broadcast to result shape.
test_umath_broadcast_upsized_result()	Test binary op for two *mpi_array.globale.gndarray* objects with the resulting *mpi_array.globale.gndarray* object having different (larger) shape than that of both inputs.
test_umath_distributed_broadcast_halo()	Test binary op for two *mpi_array.globale.gndarray* objects which requires remote fetch of data when broadcasting to result shape.
test_umath_distributed_broadcast_no_halo()	Test binary op for two *mpi_array.globale.gndarray* objects which requires remote fetch of data when broadcasting to result shape.
test_umath_halo()	Test binary op for a *mpi_array.globale.gndarray* object and a scalar, test halo is preserved.

Continued on next page

Table 1.148 – continued from previous page

test_umath_multiply()	Asserts that binary ufunc multiplication (numpy.multiply) computation for *mpi_array.globale.gndarray* arguments produces same results as for numpy.ndarray arguments.
test_umath_no_halo()	Test binary op for a *mpi_array.globale.gndarray* object and a scalar.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.__init__

GndarrayUfuncTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.addCleanup

GndarrayUfuncTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.globale_ufunc_test.GndarrayUfuncTest.addTypeEqualityFunc

GndarrayUfuncTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.assertArraySplitEqual

GndarrayUfuncTest.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.

Parameters

- **splt1** (list of numpy.ndarray) – First object in equality comparison.
- **splt2** (list of numpy.ndarray) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of splt2.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.compare_results

GndarrayUfuncTest.**compare_results**(*mpi_cln_npy_result_ary*, *mpi_result_ary*)

Asserts that all elements of the *mpi_array.globale.gndarray* *mpi_cln_npy_result_ary* equal all elements of the *mpi_array.globale.gndarray* *mpi_result_ary*.

Parameters

- **mpi_cln_npy_result_ary** (*mpi_array.globale.gndarray*) – The result returned by *func*(**func_args*) converted to a cloned-distribution *mpi_array.globale.gndarray*.
- **mpi_result_ary** (*mpi_array.globale.gndarray*) – The result array from mpi_array.globale.gndarray.__array_ufunc__() execution.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.convert_func_args_to_gndarrays

GndarrayUfuncTest.**convert_func_args_to_gndarrays**(*converter*, *func_args*)

Parameters

- **converter** (*ToGndarrayConverter*) – Used to convert the numpy.ndarray instances of *func_args* to *mpi_array.globale.gndarray* instances.
- **func_args** (sequence of numpy.ndarray or array-like objects) – Sequence of array-like objects. Can be comprised of misture of numpy.ndarray instances, scalars or (broadcastable) sequences (e.g. tuple of scalars) elements.

Return type list

Returns The *func_args* list with numpy.ndarray instances converted to *mpi_array.globale.gndarray* instances.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.countTestCases

GndarrayUfuncTest.**countTestCases**()

mpi_array.globale_ufunc_test.GndarrayUfuncTest.debug

GndarrayUfuncTest.**debug**()

Run the test without collecting errors in a TestResult

mpi_array.globale_ufunc_test.GndarrayUfuncTest.defaultTestResult

GndarrayUfuncTest.**defaultTestResult**()

mpi_array.globale_ufunc_test.GndarrayUfuncTest.doCleanups

GndarrayUfuncTest.**doCleanups**()

Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.do_block_distribution_test

GndarrayUfuncTest.**do_block_distribution_test**(*mpi_cln_npy_result_ary*, *func*, **func_args*)

Converts numpy.ndarray elements of func_args to *mpi_array.globale.gndarray* instances distributed as the *mpi_array.comms.DT_BLOCK* distribution type.

Parameters

- **mpi_cln_npy_result_ary** (*mpi_array.globale.gndarray*) – The result returned by *func*(**func_args*) converted to a cloned-distribution *mpi_array.globale.gndarray*.
- **func** (*callable*) – Function which computes a new array from the **func_args* arguments.
- **func_args** (sequence of numpy.ndarray or array-like objects) – The arguments for the *func* function. Can be comprised of numpy.ndarray, scalars or broadcastable sequence (e.g. tuple of scalars) elements.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.do_cloned_distribution_test

GndarrayUfuncTest.**do_cloned_distribution_test**(*mpi_cln_npy_result_ary*, *func*, **func_args*)

Converts numpy.ndarray elements of func_args to *mpi_array.globale.gndarray* instances distributed as the *mpi_array.comms.DT_CLONED* distribution type.

Parameters

- **mpi_cln_npy_result_ary** (*mpi_array.globale.gndarray*) – The result returned by *func*(**func_args*) converted to a cloned-distribution *mpi_array.globale.gndarray*.
- **func** (*callable*) – Function which computes a new array from the **func_args* arguments.
- **func_args** (sequence of numpy.ndarray or array-like objects) – The arguments for the *func* function. Can be comprised of numpy.ndarray, scalars or broadcastable sequence (e.g. tuple of scalars) elements.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.do_convert_execute_and_compare

GndarrayUfuncTest.**do_convert_execute_and_compare**(*mpi_cln_npy_result_ary*, *converter*, *func*, **func_args*)

Compares the result of *func* called with self.convert_func_args_to_gndarrays(*converter*, *func_args*) converted arguments with the *mpi_cln_npy_result_ary* array (which should have been produced by calling mpi_array.globale_creation.asarray(*func*(*func_args))).

Parameters

- **mpi_cln_npy_result_ary** (*mpi_array.globale.gndarray*) – The result returned by *func*(**func_args*) converted to a cloned-distribution *mpi_array.globale.gndarray*.
- **converter** (*ToGndarrayConverter*) – Used to convert the numpy.ndarray instances of *func_args* to mpi_array.globale.gndarray instances.

- **func** (*callable*) – Function which computes a new array from the **func_args* arguments and for arguments converted with *converter*.
- **func_args** (sequence of numpy.ndarray or array-like objects) – The arguments for the *func* function. Can be comprised of numpy.ndarray, scalars or broadcastable sequence (e.g. tuple of scalars) elements.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.do_multi_distribution_tests

GndarrayUfuncTest.**do_multi_distribution_tests**(*func*, **func_args*)

Compares result of *func* called with numpy.ndarray arguments and result of *func* called with *mpi_array.globale.gndarray* arguments. Executes *func*(**func_args*) and compares the result with *func*(**self.convert_func_args_to_gndarrays(converter, func_args)*), where multiple versions instances of converter are used to generate different distributions for the *mpi_array.globale.gndarray* *func* arguments.

Parameters

- **func** (*callable*) – Function which computes a new array from the **func_args* arguments.
- **func_args** (sequence of numpy.ndarray or array-like objects) – The arguments for the *func* function. Can be comprised of numpy.ndarray, scalars or broadcastable sequence (e.g. tuple of scalars) elements.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.do_single_locale_distribution_test

GndarrayUfuncTest.**do_single_locale_distribution_test**(*mpi_cln_npy_result_ary*, *func*, **func_args*)

Converts numpy.ndarray elements of func_args to *mpi_array.globale.gndarray* instances distributed as the *mpi_array.comms.DT_SINGLE_LOCALE* distribution type.

Parameters

- **mpi_cln_npy_result_ary** (*mpi_array.globale.gndarray*) – The result returned by *func*(**func_args*) converted to a cloned-distribution *mpi_array.globale.gndarray*.
- **func** (*callable*) – Function which computes a new array from the **func_args* arguments.
- **func_args** (sequence of numpy.ndarray or array-like objects) – The arguments for the *func* function. Can be comprised of numpy.ndarray, scalars or broadcastable sequence (e.g. tuple of scalars) elements.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.do_test_umath

GndarrayUfuncTest.**do_test_umath**(*halo=0*, *gshape=(32*, *48)*)

Test binary op for a *mpi_array.globale.gndarray* object and a scalar.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.do_test_umath_broadcast

GndarrayUfuncTest.**do_test_umath_broadcast**(*halo=0*, *dims=(0*, *0*, *0)*)

Test binary op for a *mpi_array.globale.gndarray* objects and an *array-like* object which requires

requiring broadcast to result shape.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.do_test_umath_broadcast_upsized_result

GndarrayUfuncTest.**do_test_umath_broadcast_upsized_result**(*halo_a=0*, *halo_b=0*, *dims_a=(0, 0)*, *dims_b=(0, 0, 0)*)

Test binary op for two *mpi_array.globale.gndarray* objects with the resulting *mpi_array.globale.gndarray* object having different (larger) shape than that of both inputs.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.do_test_umath_distributed_broadcast

GndarrayUfuncTest.**do_test_umath_distributed_broadcast**(*halo_a=0*, *halo_b=0*)

Test binary op for two *mpi_array.globale.gndarray* objects which requires remote fetch of data when broadcasting to result shape.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.id

GndarrayUfuncTest.**id**()

mpi_array.globale_ufunc_test.GndarrayUfuncTest.run

GndarrayUfuncTest.**run**(*result=None*)

mpi_array.globale_ufunc_test.GndarrayUfuncTest.setUp

GndarrayUfuncTest.**setUp**()

Initialise `numpy.random.seed()`.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.setUpClass

GndarrayUfuncTest.**setUpClass**()

Hook method for setting up class fixture before running tests in the class.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.shortDescription

GndarrayUfuncTest.**shortDescription**()

Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.skipTest

GndarrayUfuncTest.**skipTest**(*reason*)

Skip this test.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.subTest

GndarrayUfuncTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.tearDown

GndarrayUfuncTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.tearDownClass

GndarrayUfuncTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.test_umath_broadcast_halo

GndarrayUfuncTest.**test_umath_broadcast_halo**()
Test binary op for a *mpi_array.globale.gndarray* objects and an *array-like* object which requires requiring broadcast to result shape.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.test_umath_broadcast_no_halo

GndarrayUfuncTest.**test_umath_broadcast_no_halo**()
Test binary op for a *mpi_array.globale.gndarray* objects and an *array-like* object which requires requiring broadcast to result shape.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.test_umath_broadcast_upsized_result

GndarrayUfuncTest.**test_umath_broadcast_upsized_result**()
Test binary op for two *mpi_array.globale.gndarray* objects with the resulting *mpi_array.globale.gndarray* object having different (larger) shape than that of both inputs.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.test_umath_distributed_broadcast_halo

GndarrayUfuncTest.**test_umath_distributed_broadcast_halo**()
Test binary op for two *mpi_array.globale.gndarray* objects which requires remote fetch of data when broadcasting to result shape. Ghost elements added to arrays.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.test_umath_distributed_broadcast_no_halo

GndarrayUfuncTest.**test_umath_distributed_broadcast_no_halo**()
Test binary op for two *mpi_array.globale.gndarray* objects which requires remote fetch of data when broadcasting to result shape.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.test_umath_halo

GndarrayUfuncTest.**test_umath_halo**()
Test binary op for a *mpi_array.globale.gndarray* object and a scalar, test halo is preserved.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.test_umath_multiply

GndarrayUfuncTest.**test_umath_multiply**()
Asserts that binary ufunc multiplication (numpy.multiply) computation for *mpi_array.globale.gndarray* arguments produces same results as for numpy.ndarray arguments. Tries various argument combinations and different distribution types for the *mpi_array.globale.gndarray* arguments.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.test_umath_no_halo

GndarrayUfuncTest.**test_umath_no_halo**()
Test binary op for a *mpi_array.globale.gndarray* object and a scalar.

Attributes

longMessage	
maxDiff	
rank_logger	A logging.Logger object.

mpi_array.globale_ufunc_test.GndarrayUfuncTest.longMessage

GndarrayUfuncTest.**longMessage** = True

mpi_array.globale_ufunc_test.GndarrayUfuncTest.maxDiff

GndarrayUfuncTest.**maxDiff** = 640

mpi_array.globale_ufunc_test.GndarrayUfuncTest.rank_logger

GndarrayUfuncTest.**rank_logger**
A logging.Logger object.

mpi_array.globale_ufunc_test.ToGndarrayConverter

class mpi_array.globale_ufunc_test.**ToGndarrayConverter**(***kwargs*)
Bases: object

Base class for converting numpy.ndarray objects to *mpi_array.globale.gndarray* objects.

Methods

__init__(**kwargs)	The `kwargs` are passed directly to the *mpi_array.globale_creation.asarray()* function in `__call__()`.

mpi_array.globale_ufunc_test.ToGndarrayConverter.__init__

`ToGndarrayConverter.`**`__init__`**`(`***kwargs*`)`

The `kwargs` are passed directly to the *mpi_array.globale_creation.asarray()* function in `__call__()`.

1.24 The `mpi_array.indexing` Module

Various calculations for array indexing and array indexing extents.

1.24.1 Classes and Functions

IndexingExtent([slice, start, stop, struct])	
HaloIndexingExtent([slice, start, stop, ...])	Indexing bounds with ghost (halo) elements, for a single tile of domain decomposition.
calc_intersection_split(dst_extent, ...)	Calculates intersection between *dst_extent* and *{src_extent}*.

mpi_array.indexing.IndexingExtent

class `mpi_array.indexing.`**`IndexingExtent`**`(`*slice=None*, *start=None*, *stop=None*, *struct=None*`)`

Bases: `object`

Methods

__init__([slice, start, stop, struct])	Construct, must specify either *slice* or both of *start* and *stop*.
calc_intersection(other)	Returns the indexing extent which is the intersection of this extent with the *other* extent.
calc_intersection_split(other)	Returns `(leftovers, intersection)` pair, where `intersection` is the *IndexingExtent* object (possibly `None`) indicating the intersection of this (*self*) extent with the `other` extent and `leftovers` is a list of *IndexingExtent* objects indicating regions of `self` which do not intersect with the `other` extent.
create_struct_dtype_from_ndim(ndim)	Creates a `numpy.dtype` structure for holding start and stop indices.
create_struct_instance(ndim)	Creates a struct instance with `numpy.dtype` `self.struct_dtype_dict[ndim]`.
get_struct_dtype(ndim)	

Continued on next page

Table 1.152 – continued from previous page

get_struct_dtype_from_ndim(ndim)	
split(a, index)	Split this extent into two extents by cutting along axis *a* at index *index*.
to_slice()	Returns "tuple of slice" equivalent of this indexing extent.
to_tuple()	Convert this instance to a tuple which can be passed to constructor (or used as a dict key).

mpi_array.indexing.IndexingExtent.__init__

IndexingExtent.**__init__**(*slice=None*, *start=None*, *stop=None*, *struct=None*)

Construct, must specify either *slice* or both of *start* and *stop*.

Parameters

- **slice** (sequence of slice) – Per axis start and stop indices defining the extent.
- **start** (sequence of int) – Per axis *start* indices defining the start of extent.
- **stop** (sequence of int) – Per axis *stop* indices defining the extent.

mpi_array.indexing.IndexingExtent.calc_intersection

IndexingExtent.**calc_intersection**(*other*)

Returns the indexing extent which is the intersection of this extent with the *other* extent.

Parameters **other** (*IndexingExtent*) – Perform intersection calculation using this extent.

Return type *IndexingExtent*

Returns None if the extents do not intersect, otherwise returns the extent of overlapping indices.

mpi_array.indexing.IndexingExtent.calc_intersection_split

IndexingExtent.**calc_intersection_split**(*other*)

Returns (leftovers, intersection) pair, where intersection is the *IndexingExtent* object (possibly None) indicating the intersection of this (*self*) extent with the other extent and leftovers is a list of *IndexingExtent* objects indicating regions of self which do not intersect with the other extent.

Parameters **other** (*IndexingExtent*) – Perform intersection calculation using this extent.

Return type tuple

Returns (leftovers, intersection) pair.

mpi_array.indexing.IndexingExtent.create_struct_dtype_from_ndim

static IndexingExtent.**create_struct_dtype_from_ndim**(*ndim*)

Creates a numpy.dtype structure for holding start and stop indices.

Return type numpy.dtype

Returns numpy.dtype with "start" and "stop" multi-index fields of dimension *ndim*.

mpi_array.indexing.IndexingExtent.create_struct_instance

IndexingExtent.**create_struct_instance**(*ndim*)
Creates a struct instance with numpy.dtype self.struct_dtype_dict[ndim].

Return type struct

Returns A struct.

mpi_array.indexing.IndexingExtent.get_struct_dtype

IndexingExtent.**get_struct_dtype**(*ndim*)

mpi_array.indexing.IndexingExtent.get_struct_dtype_from_ndim

static IndexingExtent.**get_struct_dtype_from_ndim**(*ndim*)

mpi_array.indexing.IndexingExtent.split

IndexingExtent.**split**(*a*, *index*)
Split this extent into two extents by cutting along axis *a* at index *index*.

Parameters

- **a** (int) – Cut along this axis.
- **index** (int) – Location of cut.

Return type tuple

Returns A (lo, hi) pair.

mpi_array.indexing.IndexingExtent.to_slice

IndexingExtent.**to_slice**()
Returns "tuple of slice" equivalent of this indexing extent.

Return type tuple of slice elements

Returns Tuple of slice equivalent to this indexing extent.

mpi_array.indexing.IndexingExtent.to_tuple

IndexingExtent.**to_tuple**()
Convert this instance to a tuple which can be passed to constructor (or used as a dict key).

Return type tuple

Returns The tuple representation of this object.

Attributes

START	
START_STR	
STOP	
STOP_STR	Indexing bounds for a single tile of domain decomposition.
ndim	Dimension of indexing.
shape	Sequence of `int` indicating the shape of this extent (including halo).
start	Sequence of `int` indicating the per-axis start indices of this extent (including halo).
stop	Sequence of `int` indicating the per-axis stop indices of this extent (including halo).
struct_dtype_dict	

mpi_array.indexing.IndexingExtent.START

IndexingExtent.**START** = 0

mpi_array.indexing.IndexingExtent.START_STR

IndexingExtent.**START_STR** = 'start'

mpi_array.indexing.IndexingExtent.STOP

IndexingExtent.**STOP** = 1

mpi_array.indexing.IndexingExtent.STOP_STR

IndexingExtent.**STOP_STR** = 'stop'
Indexing bounds for a single tile of domain decomposition.

mpi_array.indexing.IndexingExtent.ndim

IndexingExtent.**ndim**
Dimension of indexing.

mpi_array.indexing.IndexingExtent.shape

IndexingExtent.**shape**
Sequence of `int` indicating the shape of this extent (including halo).

mpi_array.indexing.IndexingExtent.start

IndexingExtent.**start**
Sequence of `int` indicating the per-axis start indices of this extent (including halo).

mpi_array.indexing.IndexingExtent.stop

IndexingExtent.**stop**
Sequence of int indicating the per-axis stop indices of this extent (including halo).

mpi_array.indexing.IndexingExtent.struct_dtype_dict

IndexingExtent.**struct_dtype_dict = defaultdict(<function IndexingExtent.<lambda>>, {})**

mpi_array.indexing.HaloIndexingExtent

class mpi_array.indexing.**HaloIndexingExtent**(*slice=None*, *start=None*, *stop=None*, *halo=None*, *struct=None*)
Bases: *mpi_array.indexing.IndexingExtent*

Indexing bounds with ghost (halo) elements, for a single tile of domain decomposition.

Example:

```
>>> from mpi_array.indexing import HaloIndexingExtent
>>>
>>> hie = HaloIndexingExtent(start=(10,), stop=(20,), halo=((2,4),))
>>> print("hie.start_n = %s" % (hie.start_n,)) # start without halo
hie.start_n = [10]
>>> print("hie.start_h = %s" % (hie.start_h,)) # start with halo
hie.start_h = [8]
>>> print("hie.stop_n  = %s" % (hie.stop_n,))  # stop without halo
hie.stop_n  = [20]
>>> print("hie.stop_h  = %s" % (hie.stop_h,))  # stop with halo
hie.stop_h  = [24]
```

Methods

__init__([slice, start, stop, halo, struct])	Construct.
calc_intersection(other)	Returns the indexing extent which is the intersection of this extent with the *other* extent.
calc_intersection_split(other)	Returns (leftovers, intersection) pair, where intersection is the *IndexingExtent* object (possibly None) indicating the intersection of this (*self*) extent with the other extent and leftovers is a list of *IndexingExtent* objects indicating regions of self which do not intersect with the other extent.
create_struct_dtype_from_ndim(ndim)	Creates a numpy.dtype structure for holding start and stop indices.
create_struct_instance(ndim)	Creates a struct instance with numpy.dtype self.struct_dtype_dict[ndim].
get_struct_dtype(ndim)	
get_struct_dtype_from_ndim(ndim)	
globale_to_locale_extent_h(gext)	Return gext converted to locale index.
globale_to_locale_h(gidx)	Convert globale array index to locale array index.

Continued on next page

Table 1.154 – continued from previous page

globale_to_locale_n(gidx)	Convert globale array index to locale array index.
globale_to_locale_slice_h(gslice)	Return `gslice` converted to locale slice.
globale_to_locale_slice_n(gslice)	Return `gslice` converted to locale slice.
locale_to_globale_extent_h(lext)	Return `lext` converted to globale index.
locale_to_globale_h(lidx)	Convert locale array index to globale array index.
locale_to_globale_n(lidx)	Convert locale array index to globale array index.
locale_to_globale_slice_h(lslice)	Return `lslice` converted to globale slice.
locale_to_globale_slice_n(lslice)	Return `lslice` converted to globale slice.
split(a, index)	Split this extent into two extents by cutting along axis *a* at index *index*.
to_slice()	Same as *to_slice_n()*.
to_slice_h()	Returns "`tuple` of `slice`" equivalent of this indexing extent including halo.
to_slice_n()	Returns "`tuple` of `slice`" equivalent of this indexing extent without halo ("no halo").
to_tuple()	Convert this instance to a `tuple` which can be passed to constructor (or used as a `dict` key).

mpi_array.indexing.HaloIndexingExtent.__init__

`HaloIndexingExtent.`**`__init__`**(*slice=None*, *start=None*, *stop=None*, *halo=None*, *struct=None*)

Construct.

Parameters

- **`slice`** (sequence of `slice`) – Per axis start and stop indices defining the extent (**not including ghost elements**).
- **`start`** (sequence of `int`) – Per axis *start* indices defining the start of extent (**not including ghost elements**).
- **`stop`** (sequence of `int`) – Per axis *stop* indices defining the extent (**not including ghost elements**).
- **`halo`** (`(len(`*slice*`), 2)` shaped array of `int`) – A `(len(self.start), 2)` shaped array of `int` indicating the per-axis number of outer ghost elements. `halo[:, 0]` is the number of elements on the low-index *side* and `halo[:,1]` is the number of elements on the high-index *side*.

mpi_array.indexing.HaloIndexingExtent.calc_intersection

`HaloIndexingExtent.`**`calc_intersection`**(*other*)

Returns the indexing extent which is the intersection of this extent with the *other* extent.

Parameters **`other`** (*IndexingExtent*) – Perform intersection calculation using this extent.

Return type *IndexingExtent*

Returns `None` if the extents do not intersect, otherwise returns the extent of overlapping indices.

mpi_array.indexing.HaloIndexingExtent.calc_intersection_split

HaloIndexingExtent.**calc_intersection_split**(*other*)

Returns `(leftovers, intersection)` pair, where `intersection` is the *IndexingExtent* object (possibly `None`) indicating the intersection of this (*self*) extent with the `other` extent and `leftovers` is a list of *IndexingExtent* objects indicating regions of `self` which do not intersect with the `other` extent.

Parameters **other** (*IndexingExtent*) – Perform intersection calculation using this extent.

Return type `tuple`

Returns `(leftovers, intersection)` pair.

mpi_array.indexing.HaloIndexingExtent.create_struct_dtype_from_ndim

static HaloIndexingExtent.**create_struct_dtype_from_ndim**(*ndim*)

Creates a `numpy.dtype` structure for holding start and stop indices.

Return type `numpy.dtype`

Returns `numpy.dtype` with `"start"` and `"stop"` multi-index fields of dimension *ndim*.

mpi_array.indexing.HaloIndexingExtent.create_struct_instance

HaloIndexingExtent.**create_struct_instance**(*ndim*)

Creates a struct instance with `numpy.dtype` `self.struct_dtype_dict[ndim]`.

Return type struct

Returns A struct.

mpi_array.indexing.HaloIndexingExtent.get_struct_dtype

HaloIndexingExtent.**get_struct_dtype**(*ndim*)

mpi_array.indexing.HaloIndexingExtent.get_struct_dtype_from_ndim

HaloIndexingExtent.**get_struct_dtype_from_ndim**(*ndim*)

mpi_array.indexing.HaloIndexingExtent.globale_to_locale_extent_h

HaloIndexingExtent.**globale_to_locale_extent_h**(*gext*)

Return `gext` converted to locale index.

mpi_array.indexing.HaloIndexingExtent.globale_to_locale_h

HaloIndexingExtent.**globale_to_locale_h**(*gidx*)

Convert globale array index to locale array index.

Parameters **gidx** (sequence of `int`) – Globale index.

Return type numpy.ndarray

Returns Locale index.

mpi_array.indexing.HaloIndexingExtent.globale_to_locale_n

HaloIndexingExtent.**globale_to_locale_n**(*gidx*)
Convert globale array index to locale array index.

Parameters **gidx** (sequence of int) – Globale index.

Return type numpy.ndarray

Returns Locale index.

mpi_array.indexing.HaloIndexingExtent.globale_to_locale_slice_h

HaloIndexingExtent.**globale_to_locale_slice_h**(*gslice*)
Return `gslice` converted to locale slice.

mpi_array.indexing.HaloIndexingExtent.globale_to_locale_slice_n

HaloIndexingExtent.**globale_to_locale_slice_n**(*gslice*)
Return `gslice` converted to locale slice.

mpi_array.indexing.HaloIndexingExtent.locale_to_globale_extent_h

HaloIndexingExtent.**locale_to_globale_extent_h**(*lext*)
Return `lext` converted to globale index.

mpi_array.indexing.HaloIndexingExtent.locale_to_globale_h

HaloIndexingExtent.**locale_to_globale_h**(*lidx*)
Convert locale array index to globale array index.

Parameters **lidx** (sequence of int) – Locale index.

Return type numpy.ndarray

Returns Globale index.

mpi_array.indexing.HaloIndexingExtent.locale_to_globale_n

HaloIndexingExtent.**locale_to_globale_n**(*lidx*)
Convert locale array index to globale array index.

Parameters **lidx** (sequence of int) – Locale index.

Return type numpy.ndarray

Returns Globale index.

mpi_array.indexing.HaloIndexingExtent.locale_to_globale_slice_h

HaloIndexingExtent.**locale_to_globale_slice_h**(*lslice*)
Return `lslice` converted to globale slice.

mpi_array.indexing.HaloIndexingExtent.locale_to_globale_slice_n

HaloIndexingExtent.**locale_to_globale_slice_n**(*lslice*)
Return `lslice` converted to globale slice.

mpi_array.indexing.HaloIndexingExtent.split

HaloIndexingExtent.**split**(*a*, *index*)
Split this extent into two extents by cutting along axis *a* at index *index*.

Parameters

- **a** (int) – Cut along this axis.
- **index** (int) – Location of cut.

Return type tuple

Returns A `(lo, hi)` pair.

mpi_array.indexing.HaloIndexingExtent.to_slice

HaloIndexingExtent.**to_slice**()
Same as *to_slice_n()*.

mpi_array.indexing.HaloIndexingExtent.to_slice_h

HaloIndexingExtent.**to_slice_h**()
Returns "tuple of slice" equivalent of this indexing extent including halo.

Return type tuple of slice elements

Returns Tuple of slice equivalent to this indexing extent including halo.

mpi_array.indexing.HaloIndexingExtent.to_slice_n

HaloIndexingExtent.**to_slice_n**()
Returns "tuple of slice" equivalent of this indexing extent without halo ("no halo").

Return type tuple of slice elements

Returns Tuple of slice equivalent to this no-halo indexing extent.

mpi_array.indexing.HaloIndexingExtent.to_tuple

HaloIndexingExtent.**to_tuple**()

Convert this instance to a `tuple` which can be passed to constructor (or used as a `dict` key).

Return type `tuple`

Returns The `tuple` representation of this object.

Attributes

HALO	
HALO_STR	
HI	The "high index" indices.
LO	The "low index" indices.
START	
START_N	
START_N_STR	
START_STR	
STOP	
STOP_N	
STOP_N_STR	
STOP_STR	
halo	A `(len(self.start), 2)` shaped array of `int` indicating the per-axis number of outer ghost elements.
ndim	Dimension of indexing.
shape	Same as *shape_n*.
shape_h	The shape of the tile with "halo" elements.
shape_n	The shape of the tile without "halo" elements ("no halo").
size_h	Integer indicating the number of elements in this extent including halo.
size_n	Integer indicating the number of elements in this extent without halo ("no halo")
start	Same as *start_n*.
start_h	The start index of the tile with "halo" elements.
start_n	The start index of the tile without "halo" elements ("no halo").
stop	Same as *stop_n*.
stop_h	The stop index of the tile with "halo" elements.
stop_n	The stop index of the tile without "halo" elements ("no halo").
struct_dtype_dict	

mpi_array.indexing.HaloIndexingExtent.HALO

HaloIndexingExtent.**HALO** = 2

mpi_array.indexing.HaloIndexingExtent.HALO_STR

HaloIndexingExtent.**HALO_STR = 'halo'**

mpi_array.indexing.HaloIndexingExtent.HI

HaloIndexingExtent.**HI = 1**
The "high index" indices.

mpi_array.indexing.HaloIndexingExtent.LO

HaloIndexingExtent.**LO = 0**
The "low index" indices.

mpi_array.indexing.HaloIndexingExtent.START

HaloIndexingExtent.**START = 0**

mpi_array.indexing.HaloIndexingExtent.START_N

HaloIndexingExtent.**START_N = 0**

mpi_array.indexing.HaloIndexingExtent.START_N_STR

HaloIndexingExtent.**START_N_STR = 'start'**

mpi_array.indexing.HaloIndexingExtent.START_STR

HaloIndexingExtent.**START_STR = 'start'**

mpi_array.indexing.HaloIndexingExtent.STOP

HaloIndexingExtent.**STOP = 1**

mpi_array.indexing.HaloIndexingExtent.STOP_N

HaloIndexingExtent.**STOP_N = 1**

mpi_array.indexing.HaloIndexingExtent.STOP_N_STR

HaloIndexingExtent.**STOP_N_STR = 'stop'**

mpi_array.indexing.HaloIndexingExtent.STOP_STR

HaloIndexingExtent.**STOP_STR = 'stop'**

mpi_array.indexing.HaloIndexingExtent.halo

HaloIndexingExtent.**halo**
A `(len(self.start), 2)` shaped array of `int` indicating the per-axis number of outer ghost elements. `halo[:,0]` is the number of elements on the low-index *side* and `halo[:,1]` is the number of elements on the high-index *side*.

mpi_array.indexing.HaloIndexingExtent.ndim

HaloIndexingExtent.**ndim**
Dimension of indexing.

mpi_array.indexing.HaloIndexingExtent.shape

HaloIndexingExtent.**shape**
Same as *shape_n*.

mpi_array.indexing.HaloIndexingExtent.shape_h

HaloIndexingExtent.**shape_h**
The shape of the tile with "halo" elements.

mpi_array.indexing.HaloIndexingExtent.shape_n

HaloIndexingExtent.**shape_n**
The shape of the tile without "halo" elements ("no halo").

mpi_array.indexing.HaloIndexingExtent.size_h

HaloIndexingExtent.**size_h**
Integer indicating the number of elements in this extent including halo.

mpi_array.indexing.HaloIndexingExtent.size_n

HaloIndexingExtent.**size_n**
Integer indicating the number of elements in this extent without halo ("no halo")

mpi_array.indexing.HaloIndexingExtent.start

HaloIndexingExtent.**start**
Same as *start_n*.

mpi_array.indexing.HaloIndexingExtent.start_h

HaloIndexingExtent.**start_h**
The start index of the tile with "halo" elements.

mpi_array.indexing.HaloIndexingExtent.start_n

HaloIndexingExtent.**start_n**
The start index of the tile without "halo" elements ("no halo").

mpi_array.indexing.HaloIndexingExtent.stop

HaloIndexingExtent.**stop**
Same as *stop_n*.

mpi_array.indexing.HaloIndexingExtent.stop_h

HaloIndexingExtent.**stop_h**
The stop index of the tile with "halo" elements.

mpi_array.indexing.HaloIndexingExtent.stop_n

HaloIndexingExtent.**stop_n**
The stop index of the tile without "halo" elements ("no halo").

mpi_array.indexing.HaloIndexingExtent.struct_dtype_dict

HaloIndexingExtent.**struct_dtype_dict = defaultdict(<function HaloIndexingExtent.<lambda>>, {})**

mpi_array.indexing.calc_intersection_split

mpi_array.indexing.**calc_intersection_split**(*dst_extent*, *src_extent*, *update_factory*, *update_dst_halo*)
Calculates intersection between *dst_extent* and *{src_extent}*. Any regions of *dst_extent* which **do not** intersect with *src_extent* are returned as a list of *left-over* type(*dst_extent*) elements. The regions of *dst_extent* which **do** intersect with *src_extent* are returned as a list of *update* elements. The *update* elements are created with a call to the factory object update_factory:

```
update_factory(dst_extent, src_extent, intersection)
```

Returns tuple pair (leftovers, updates).

Parameters

- **dst_extent** (*HaloIndexingExtent*) – Extent which is to receive update from intersection with *src_extent*.
- **src_extent** (*HaloIndexingExtent*) – Extent which is to provide update for the intersecting region of *dst_extent*.

- **update_factory** (callable object) – Object called to create instances of mpi_array.decomposition.PairUpdateExtent.
- **update_dst_halo** (bool) – If true, then the halo of *dst_extent* is include when calculating the intersection with *src_extent*.

Return type tuple

Returns Returns tuple pair of (leftovers, updates).

1.25 The mpi_array.indexing_test Module

Module defining *mpi_array.indexing* unit-tests. Execute as:

```
python -m mpi_array.indexing_test
```

1.25.1 Classes

IndexingExtentTest([methodName])	unittest.TestCase for mpi_array.indexing.IndexingExtentTest.
HaloIndexingExtentTest([methodName])	unittest.TestCase for mpi_array.indexing.HaloIndexingExtentTest.

mpi_array.indexing_test.IndexingExtentTest

class mpi_array.indexing_test.**IndexingExtentTest**(*methodName='runTest'*)

Bases: *mpi_array.unittest.TestCase*

unittest.TestCase for mpi_array.indexing.IndexingExtentTest.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	Hook method for setting up the test fixture before exercising it.

Continued on next page

Table 1.157 – continued from previous page

setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_assign_different_dimension_index()	Test for `IndexingExtent.start = ...()`.
test_attributes()	Tests *mpi_array.indexing.IndexingExtent.start* and *mpi_array.indexing.IndexingExtent.stop* and *mpi_array.indexing.IndexingExtent.shape* attributes.
test_calc_intersection_split()	Test for *mpi_array.indexing.IndexingExtent.calc_intersection_split()*.
test_intersection_1d()	Tests *mpi_array.indexing.IndexingExtent.calc_intersection()* method, 1D indexing.
test_intersection_2d()	Tests *mpi_array.indexing.IndexingExtent.calc_intersection()* method, 2D indexing.
test_repr()	Test for `repr(IndexingExtent(start=(1,2,3), stop=(8,9,10)))`.
test_split()	Test for *mpi_array.indexing.IndexingExtent.split()*.
test_to_tuple()	Test for `IndexingExtent.to_tuple()`.

mpi_array.indexing_test.IndexingExtentTest.__init__

`IndexingExtentTest.`**`__init__`**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.indexing_test.IndexingExtentTest.addCleanup

`IndexingExtentTest.`**`addCleanup`**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.indexing_test.IndexingExtentTest.addTypeEqualityFunc

IndexingExtentTest.**addTypeEqualityFunc**(*typeobj*, *function*)
Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.indexing_test.IndexingExtentTest.assertArraySplitEqual

IndexingExtentTest.**assertArraySplitEqual**(*splt1*, *splt2*)
Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.indexing_test.IndexingExtentTest.countTestCases

IndexingExtentTest.**countTestCases**()

mpi_array.indexing_test.IndexingExtentTest.debug

IndexingExtentTest.**debug**()
Run the test without collecting errors in a TestResult

mpi_array.indexing_test.IndexingExtentTest.defaultTestResult

IndexingExtentTest.**defaultTestResult**()

mpi_array.indexing_test.IndexingExtentTest.doCleanups

IndexingExtentTest.**doCleanups**()
Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.indexing_test.IndexingExtentTest.id

IndexingExtentTest.**id**()

mpi_array.indexing_test.IndexingExtentTest.run

IndexingExtentTest.**run**(*result=None*)

mpi_array.indexing_test.IndexingExtentTest.setUp

IndexingExtentTest.**setUp**()
Hook method for setting up the test fixture before exercising it.

mpi_array.indexing_test.IndexingExtentTest.setUpClass

IndexingExtentTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.indexing_test.IndexingExtentTest.shortDescription

IndexingExtentTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.indexing_test.IndexingExtentTest.skipTest

IndexingExtentTest.**skipTest**(*reason*)
Skip this test.

mpi_array.indexing_test.IndexingExtentTest.subTest

IndexingExtentTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.indexing_test.IndexingExtentTest.tearDown

IndexingExtentTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.indexing_test.IndexingExtentTest.tearDownClass

IndexingExtentTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.indexing_test.IndexingExtentTest.test_assign_different_dimension_index

IndexingExtentTest.**test_assign_different_dimension_index**()
Test for `IndexingExtent.start = ...()`.

mpi_array.indexing_test.IndexingExtentTest.test_attributes

IndexingExtentTest.**test_attributes**()
Tests *mpi_array.indexing.IndexingExtent.start* and *mpi_array.indexing.IndexingExtent.stop* and *mpi_array.indexing.IndexingExtent.shape* attributes.

mpi_array.indexing_test.IndexingExtentTest.test_calc_intersection_split

IndexingExtentTest.**test_calc_intersection_split**()
Test for *mpi_array.indexing.IndexingExtent.calc_intersection_split()*.

mpi_array.indexing_test.IndexingExtentTest.test_intersection_1d

IndexingExtentTest.**test_intersection_1d**()
Tests *mpi_array.indexing.IndexingExtent.calc_intersection()* method, 1D indexing.

mpi_array.indexing_test.IndexingExtentTest.test_intersection_2d

IndexingExtentTest.**test_intersection_2d**()
Tests *mpi_array.indexing.IndexingExtent.calc_intersection()* method, 2D indexing.

mpi_array.indexing_test.IndexingExtentTest.test_repr

IndexingExtentTest.**test_repr**()
Test for `repr(IndexingExtent(start=(1,2,3), stop=(8,9,10)))`.

mpi_array.indexing_test.IndexingExtentTest.test_split

IndexingExtentTest.**test_split**()
Test for *mpi_array.indexing.IndexingExtent.split()*.

mpi_array.indexing_test.IndexingExtentTest.test_to_tuple

IndexingExtentTest.**test_to_tuple**()
Test for `IndexingExtent.to_tuple()`.

Attributes

longMessage	
maxDiff	

mpi_array.indexing_test.IndexingExtentTest.longMessage

IndexingExtentTest.**longMessage = True**

mpi_array.indexing_test.IndexingExtentTest.maxDiff

IndexingExtentTest.**maxDiff = 640**

mpi_array.indexing_test.HaloIndexingExtentTest

class mpi_array.indexing_test.**HaloIndexingExtentTest**(*methodName='runTest'*)

Bases: *mpi_array.unittest.TestCase*

unittest.TestCase for mpi_array.indexing.HaloIndexingExtentTest.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	Hook method for setting up the test fixture before exercising it.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.

Continued on next page

Table 1.159 – continued from previous page

tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_attributes()	unittest.TestCase for mpi_array.indexing.HaloIndexingExtentTest
test_calc_intersection_split()	Tests for *mpi_array.indexing.calc_intersection_split*.
test_globale_and_locale_extent_conversion()	Test for *mpi_array.indexing.HaloIndexingExtent.globale_to_locale_h()*, and *mpi_array.indexing.HaloIndexingExtent.locale_to_globale_h()*.
test_globale_and_locale_index_conversion()	Test for *mpi_array.indexing.HaloIndexingExtent.globale_to_locale_h()*, and *mpi_array.indexing.HaloIndexingExtent.locale_to_globale_h()*.
test_globale_and_locale_slice_conversion()	Test for *mpi_array.indexing.HaloIndexingExtent.globale_to_locale_slice_h()*, and *mpi_array.indexing.HaloIndexingExtent.locale_to_globale_slice_h()*.
test_repr()	Test for repr(HaloIndexingExtent(start=(1,2,3), stop=(8,9,10))).
test_start_stop_shape()	unittest.TestCase for *mpi_array.indexing.HaloIndexingExtent*
test_to_slice()	unittest.TestCase for *mpi_array.indexing.HaloIndexingExtent*
test_to_tuple()	Test for HaloIndexingExtent.to_tuple().

mpi_array.indexing_test.HaloIndexingExtentTest.__init__

HaloIndexingExtentTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.indexing_test.HaloIndexingExtentTest.addCleanup

HaloIndexingExtentTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.indexing_test.HaloIndexingExtentTest.addTypeEqualityFunc

HaloIndexingExtentTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.indexing_test.HaloIndexingExtentTest.assertArraySplitEqual

HaloIndexingExtentTest.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.indexing_test.HaloIndexingExtentTest.countTestCases

HaloIndexingExtentTest.**countTestCases**()

mpi_array.indexing_test.HaloIndexingExtentTest.debug

HaloIndexingExtentTest.**debug**()

Run the test without collecting errors in a TestResult

mpi_array.indexing_test.HaloIndexingExtentTest.defaultTestResult

HaloIndexingExtentTest.**defaultTestResult**()

mpi_array.indexing_test.HaloIndexingExtentTest.doCleanups

HaloIndexingExtentTest.**doCleanups**()

Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.indexing_test.HaloIndexingExtentTest.id

HaloIndexingExtentTest.**id**()

mpi_array.indexing_test.HaloIndexingExtentTest.run

HaloIndexingExtentTest.**run**(*result=None*)

mpi_array.indexing_test.HaloIndexingExtentTest.setUp

HaloIndexingExtentTest.**setUp**()
Hook method for setting up the test fixture before exercising it.

mpi_array.indexing_test.HaloIndexingExtentTest.setUpClass

HaloIndexingExtentTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.indexing_test.HaloIndexingExtentTest.shortDescription

HaloIndexingExtentTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.indexing_test.HaloIndexingExtentTest.skipTest

HaloIndexingExtentTest.**skipTest**(*reason*)
Skip this test.

mpi_array.indexing_test.HaloIndexingExtentTest.subTest

HaloIndexingExtentTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.indexing_test.HaloIndexingExtentTest.tearDown

HaloIndexingExtentTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.indexing_test.HaloIndexingExtentTest.tearDownClass

HaloIndexingExtentTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.indexing_test.HaloIndexingExtentTest.test_attributes

HaloIndexingExtentTest.**test_attributes**()
unittest.TestCase for mpi_array.indexing.HaloIndexingExtentTest attributes.

mpi_array.indexing_test.HaloIndexingExtentTest.test_calc_intersection_split

HaloIndexingExtentTest.**test_calc_intersection_split**()
Tests for *mpi_array.indexing.calc_intersection_split*.

mpi_array.indexing_test.HaloIndexingExtentTest.test_globale_and_locale_extent_conversion

HaloIndexingExtentTest.**test_globale_and_locale_extent_conversion**()
Test for *mpi_array.indexing.HaloIndexingExtent.globale_to_locale_h()*, and *mpi_array.indexing.HaloIndexingExtent.locale_to_globale_h()*.

mpi_array.indexing_test.HaloIndexingExtentTest.test_globale_and_locale_index_conversion

HaloIndexingExtentTest.**test_globale_and_locale_index_conversion**()
Test for *mpi_array.indexing.HaloIndexingExtent.globale_to_locale_h()*, and *mpi_array.indexing.HaloIndexingExtent.locale_to_globale_h()*.

mpi_array.indexing_test.HaloIndexingExtentTest.test_globale_and_locale_slice_conversion

HaloIndexingExtentTest.**test_globale_and_locale_slice_conversion**()
Test for *mpi_array.indexing.HaloIndexingExtent.globale_to_locale_slice_h()*, and *mpi_array.indexing.HaloIndexingExtent.locale_to_globale_slice_h()*.

mpi_array.indexing_test.HaloIndexingExtentTest.test_repr

HaloIndexingExtentTest.**test_repr**()
Test for `repr(HaloIndexingExtent(start=(1,2,3), stop=(8,9,10)))`.

mpi_array.indexing_test.HaloIndexingExtentTest.test_start_stop_shape

HaloIndexingExtentTest.**test_start_stop_shape**()
`unittest.TestCase` for *mpi_array.indexing.HaloIndexingExtent* attributes: `start`, `stop`, and `shape`.

mpi_array.indexing_test.HaloIndexingExtentTest.test_to_slice

HaloIndexingExtentTest.**test_to_slice**()
`unittest.TestCase` for *mpi_array.indexing.HaloIndexingExtent* methods: `to_slice`, `to_slice_n`, and `to_slice_h`.

mpi_array.indexing_test.HaloIndexingExtentTest.test_to_tuple

HaloIndexingExtentTest.**test_to_tuple**()
Test for `HaloIndexingExtent.to_tuple()`.

Attributes

longMessage
maxDiff

mpi_array.indexing_test.HaloIndexingExtentTest.longMessage

HaloIndexingExtentTest.**longMessage = True**

mpi_array.indexing_test.HaloIndexingExtentTest.maxDiff

HaloIndexingExtentTest.**maxDiff = 640**

1.26 The `mpi_array.init` Module

Initialisation which needs to occur prior to `MPI_Init`.

Parts of this source borrows from the airspeed velocity (asv) file benchmark.py.

See the LICENSE.

1.26.1 Functions

create_linux_process_time()	Uses `ctypes` to create a `time.process_time()` on the `'Linux'` platform.
create_darwin_process_time()	Uses `ctypes` to create a `time.process_time()` on the `'darwin'` (OSX) platform.
initialise_process_time_timer()	Loads (or creates) `time.process_time()` function and caches the function in `mpi_array.init._process_time`.
get_process_time_timer()	The best timer we can use is `time.process_time()`, but it is not available in the Python stdlib until Python 3.3.

mpi_array.init.create_linux_process_time

mpi_array.init.**create_linux_process_time**()
Uses `ctypes` to create a `time.process_time()` on the `'Linux'` platform.

Return type function

Returns A `time.process_time()` equivalent.

mpi_array.init.create_darwin_process_time

mpi_array.init.**create_darwin_process_time**()
Uses `ctypes` to create a `time.process_time()` on the `'darwin'` (OSX) platform.

Return type function

Returns A `time.process_time()` equivalent.

mpi_array.init.initialise_process_time_timer

`mpi_array.init.`**`initialise_process_time_timer`**`()`

Loads (or creates) `time.process_time()` function and caches the function in `mpi_array.init._process_time`.

Return type `function`

Returns The `time.process_time()` function, if available, otherwise, if possible, an equivalent created using `ctypes`, otherwise `timeit.default_timer()`.

mpi_array.init.get_process_time_timer

`mpi_array.init.`**`get_process_time_timer`**`()`

The best timer we can use is `time.process_time()`, but it is not available in the Python stdlib until Python 3.3. This is a `ctypes` backport for Pythons that don't have it.

Return type `function`

Returns The `time.process_time()` function, if available, otherwise, if possible, an equivalent created using `ctypes`, otherwise `timeit.default_timer()`.

1.27 The `mpi_array.license` Module

License and copyright info.

1.27.1 License

1.27.2 Copyright

1.27.3 Functions

version()	Returns *mpi_array* version string.
license()	Returns the *mpi_array* license string.
copyright()	Returns the *mpi_array* copyright string.

mpi_array.license.version

mpi_array.license.**version**()
Returns *mpi_array* version string.

Return type str

Returns Version string.

mpi_array.license.license

mpi_array.license.**license**()
Returns the *mpi_array* license string.

Return type str

Returns License string.

mpi_array.license.copyright

mpi_array.license.**copyright**()
Returns the *mpi_array* copyright string.

Return type str

Returns Copyright string.

1.28 The mpi_array.locale Module

Defines *LndarrayProxy* class and factory functions for creating multi-dimensional arrays where memory is allocated using mpi4py.MPI.Win.Allocate_shared() or mpi4py.MPI.Win.Allocate().

1.28.1 Classes

lndarray	Sub-class of numpy.ndarray which requires *buffer* to be specified for instantiation.

mpi_array.locale.lndarray

class mpi_array.locale.**lndarray**
Bases: numpy.ndarray

Sub-class of numpy.ndarray which requires *buffer* to be specified for instantiation.

Methods

__new__([shape, dtype, buffer, offset, ...])	Construct, at least one of :samp:{shape} or `decomp` should be specified (i.e.
__array_finalize__(obj)	Sets *md* attribute for *self* from *obj* if required.

mpi_array.locale.lndarray.__new__

static `lndarray.`**`__new__`**(*shape=None*, *dtype=dtype('float64')*, *buffer=None*, *offset=0*, *strides=None*, *order=None*)

Construct, at least one of :samp:{shape} or `decomp` should be specified (i.e. at least one should not be `None`).

Parameters

- **shape** (`None` or sequence of `int`) – **Local** shape of the array, this parameter is ignored.
- **dtype** (`numpy.dtype`) – Data type for elements of the array.
- **buffer** (`buffer`) – The sequence of bytes providing array element storage. Raises `ValueError` if *buffer* `is None`.
- **offset** (`None` or `int`) – Offset of array data in buffer, i.e where array begins in buffer (in buffer bytes).
- **strides** (`None` or sequence of `int`) – Strides of data in memory.
- **order** ({`C`, `F`} or `None`) – Row-major (C-style) or column-major (Fortran-style) order.

mpi_array.locale.lndarray.__array_finalize__

`lndarray.`**`__array_finalize__`**(*obj*)

Sets *md* attribute for *self* from *obj* if required.

Parameters **obj** (`object` or `None`) – Object from which attributes are set.

Attributes

md	Meta-data object of type *NdarrayMetaData*.

mpi_array.locale.lndarray.md

`lndarray.`**`md`**

Meta-data object of type *NdarrayMetaData*.

LndarrayProxy	Proxy for *lndarray* instances.
PartitionViewSlices	Stores multiple `tuple`-of-`slice` objects indicating the slice (tile) of the *lndarray* on which a `intra_locale_comm` rank MPI process operates.

mpi_array.locale.LndarrayProxy

class mpi_array.locale.**LndarrayProxy**
Bases: object

Proxy for *lndarray* instances. Provides peer_rank views of the array for parallelism.

Methods

calculate_intra_partition(intra_locale_size, ...)	Splits *extent* into self.intra_locale_size number of tiles.
fill(value)	Fill the array with a scalar value (excludes ghost elements).
fill_h(value)	Fill the array with a scalar value (including ghost elements).
free()	Release locale array memory and assign None to self attributes.

mpi_array.locale.LndarrayProxy.calculate_intra_partition

LndarrayProxy.**calculate_intra_partition**(*intra_locale_size*, *intra_locale_dims*, *intra_locale_rank*, *extent*, *halo*)
Splits *extent* into self.intra_locale_size number of tiles.

mpi_array.locale.LndarrayProxy.fill

LndarrayProxy.**fill**(*value*)
Fill the array with a scalar value (excludes ghost elements).

Parameters **value** (*scalar*) – All non-ghost elements are assigned this value.

mpi_array.locale.LndarrayProxy.fill_h

LndarrayProxy.**fill_h**(*value*)
Fill the array with a scalar value (including ghost elements).

Parameters **value** (*scalar*) – All elements (including ghost elements) are assigned this value.

mpi_array.locale.LndarrayProxy.free

LndarrayProxy.**free**()
Release locale array memory and assign None to self attributes.

Attributes

HI	The "high index" indices.

Continued on next page

Table 1.168 – continued from previous page

LO	The "low index" indices.
dtype	A `numpy.dtype` object describing the element type of this array.
halo	The number of ghost cells for intra locale partitioning of the extent.
intra_partition	A *PartitionViewSlices* containing slices for this rank (of `peer_comm`).
intra_partition_dims	A sequence of integers indicating the number of partitions along each axis which determines the per-rank views of the locale extent array.
lndarray	An *lndarray* instance containing array data in (potentially) shared memory.
locale_extent	A `LocaleExtent` describing the portion of the array assigned to this locale.
md	Meta-data object of type *NdarrayMetaData*.
rank_view_h	A tile view (including halo elements) of the array for this rank of `peer_comm`.
rank_view_n	A tile view of the array for this rank of `peer_comm`.
rank_view_partition_h	Rank tile view from the paritioning of entire `self._lndarray` (i.e.
rank_view_slice_h	Sequence of `slice` objects used to generate *rank_view_h*.
rank_view_slice_n	Sequence of `slice` objects used to generate *rank_view_n*.
shape	The shape of the locale array (including halo), i.e.
view_h	The entire *LndarrayProxy* view including halo (i.e.
view_n	View of entire array without halo.

mpi_array.locale.LndarrayProxy.HI

`LndarrayProxy.`**`HI = 1`**
The "high index" indices.

mpi_array.locale.LndarrayProxy.LO

`LndarrayProxy.`**`LO = 0`**
The "low index" indices.

mpi_array.locale.LndarrayProxy.dtype

`LndarrayProxy.`**`dtype`**
A `numpy.dtype` object describing the element type of this array.

mpi_array.locale.LndarrayProxy.halo

`LndarrayProxy.`**`halo`**
The number of ghost cells for intra locale partitioning of the extent. This is an upper bound on the per-rank partitions, with the halo possibly trimmed by the halo extent (due to being on globale boundary).

mpi_array.locale.LndarrayProxy.intra_partition

LndarrayProxy.**intra_partition**
A *PartitionViewSlices* containing slices for this rank (of peer_comm).

mpi_array.locale.LndarrayProxy.intra_partition_dims

LndarrayProxy.**intra_partition_dims**
A sequence of integers indicating the number of partitions along each axis which determines the per-rank views of the locale extent array.

mpi_array.locale.LndarrayProxy.lndarray

LndarrayProxy.**lndarray**
An *lndarray* instance containing array data in (potentially) shared memory.

mpi_array.locale.LndarrayProxy.locale_extent

LndarrayProxy.**locale_extent**
A LocaleExtent describing the portion of the array assigned to this locale.

mpi_array.locale.LndarrayProxy.md

LndarrayProxy.**md**
Meta-data object of type *NdarrayMetaData*.

mpi_array.locale.LndarrayProxy.rank_view_h

LndarrayProxy.**rank_view_h**
A tile view (including halo elements) of the array for this rank of peer_comm.

mpi_array.locale.LndarrayProxy.rank_view_n

LndarrayProxy.**rank_view_n**
A tile view of the array for this rank of peer_comm.

mpi_array.locale.LndarrayProxy.rank_view_partition_h

LndarrayProxy.**rank_view_partition_h**
Rank tile view from the paritioning of entire self._lndarray (i.e. partition of halo array). Same as self.rank_view_n when halo is zero.

mpi_array.locale.LndarrayProxy.rank_view_slice_h

LndarrayProxy.**rank_view_slice_h**
Sequence of slice objects used to generate *rank_view_h*.

mpi_array.locale.LndarrayProxy.rank_view_slice_n

LndarrayProxy.**rank_view_slice_n**
Sequence of `slice` objects used to generate *rank_view_n*.

mpi_array.locale.LndarrayProxy.shape

LndarrayProxy.**shape**
The shape of the locale array (including halo), i.e. `self.lndarray.shape`.

mpi_array.locale.LndarrayProxy.view_h

LndarrayProxy.**view_h**
The entire *LndarrayProxy* view including halo (i.e. :samp:{self}).

mpi_array.locale.LndarrayProxy.view_n

LndarrayProxy.**view_n**
View of entire array without halo.

mpi_array.locale.PartitionViewSlices

class mpi_array.locale.**PartitionViewSlices**
Bases: `tuple`

Stores multiple `tuple`-of-`slice` objects indicating the slice (tile) of the *lndarray* on which a `intra_locale_comm` rank MPI process operates.

Methods

count(...)	
index((value, [start, ...)	Raises ValueError if the value is not present.

mpi_array.locale.PartitionViewSlices.count

PartitionViewSlices.**count**(*value*) → integer – return number of occurrences of value

mpi_array.locale.PartitionViewSlices.index

PartitionViewSlices.**index**(*value*[, *start*[, *stop*]]) → integer – return first index of value.
Raises ValueError if the value is not present.

Attributes

lndarray_view_slice_n	Slice for generating a view of a *lndarray* with the halo removed.
rank_view_partition_slice_h	Slice indicating tile of the halo array.
rank_view_relative_slice_n	*Relative* slice which can be used to remove the
rank_view_slice_h	The slice *rank_view_slice_n* with halo added.
rank_view_slice_n	Slice indicating tile of the non-halo array.

mpi_array.locale.PartitionViewSlices.lndarray_view_slice_n

PartitionViewSlices.**lndarray_view_slice_n**
Slice for generating a view of a *lndarray* with the halo removed.

mpi_array.locale.PartitionViewSlices.rank_view_partition_slice_h

PartitionViewSlices.**rank_view_partition_slice_h**
Slice indicating tile of the halo array.

mpi_array.locale.PartitionViewSlices.rank_view_relative_slice_n

PartitionViewSlices.**rank_view_relative_slice_n**
Relative slice which can be used to remove the halo elements from a view generated using *rank_view_slice_h*.

mpi_array.locale.PartitionViewSlices.rank_view_slice_h

PartitionViewSlices.**rank_view_slice_h**
The slice *rank_view_slice_n* with halo added.

mpi_array.locale.PartitionViewSlices.rank_view_slice_n

PartitionViewSlices.**rank_view_slice_n**
Slice indicating tile of the non-halo array.

1.28.2 Factory Functions

empty([shape, dtype, comms_and_distrib, ...])	Creates array of uninitialised elements.
empty_like(ary[, dtype])	Return a new array with the same shape and type as a given array.
zeros([shape, dtype, comms_and_distrib, order])	Creates array of zero-initialised elements.
zeros_like(ary, *args, **kwargs)	Return a new zero-initialised array with the same shape and type as a given array.
ones([shape, dtype, comms_and_distrib, order])	Creates array of one-initialised elements.
ones_like(ary, *args, **kwargs)	Return a new one-initialised array with the same shape and type as a given array.
copy(ary)	Return an array copy of the given object.

mpi_array.locale.empty

mpi_array.locale.**empty**(*shape=None*, *dtype='float64'*, *comms_and_distrib=None*, *order='C'*, *return_rma_window_buffer=False*, *intra_partition_dims=None*, ***kwargs*)

Creates array of uninitialised elements.

Parameters

- **shape** (None or sequence of int) – **Global** shape to be distributed amongst memory nodes.
- **dtype** (numpy.dtype) – Data type of array elements.
- **comms_and_distrib** (numpy.dtype) – Data type of array elements.

Return type *LndarrayProxy*

Returns Newly created array with uninitialised elements.

mpi_array.locale.empty_like

mpi_array.locale.**empty_like**(*ary*, *dtype=None*)

Return a new array with the same shape and type as a given array.

Parameters

- **ary** (numpy.ndarray) – Copy attributes from this array.
- **dtype** (numpy.dtype) – Specifies different dtype for the returned array.

Return type type(ary)

Returns Array of uninitialized (arbitrary) data with the same shape and type as *a*.

mpi_array.locale.zeros

mpi_array.locale.**zeros**(*shape=None*, *dtype='float64'*, *comms_and_distrib=None*, *order='C'*, ***kwargs*)

Creates array of zero-initialised elements.

Parameters

- **shape** (None or sequence of int) – **Global** shape to be distributed amongst memory nodes.
- **dtype** (numpy.dtype) – Data type of array elements.
- **comms_and_distrib** (numpy.dtype) – Data type of array elements.

Return type *LndarrayProxy*

Returns Newly created array with zero-initialised elements.

mpi_array.locale.zeros_like

mpi_array.locale.**zeros_like**(*ary*, **args*, ***kwargs*)

Return a new zero-initialised array with the same shape and type as a given array.

Parameters

- **ary** (*LndarrayProxy*) – Copy attributes from this array.

- **dtype** (numpy.dtype) – Specifies different dtype for the returned array.

Return type *LndarrayProxy*

Returns Array of zero-initialized data with the same shape and type as *ary*.

mpi_array.locale.ones

mpi_array.locale.**ones**(*shape=None*, *dtype='float64'*, *comms_and_distrib=None*, *order='C'*, ***kwargs*)

Creates array of one-initialised elements.

Parameters

- **shape** (None or sequence of int) – **Global** shape to be distributed amongst memory nodes.
- **dtype** (numpy.dtype) – Data type of array elements.
- **comms_and_distrib** (numpy.dtype) – Data type of array elements.

Return type *LndarrayProxy*

Returns Newly created array with one-initialised elements.

mpi_array.locale.ones_like

mpi_array.locale.**ones_like**(*ary*, **args*, ***kwargs*)

Return a new one-initialised array with the same shape and type as a given array.

Parameters

- **ary** (*LndarrayProxy*) – Copy attributes from this array.
- **dtype** (numpy.dtype) – Specifies different dtype for the returned array.

Return type *LndarrayProxy*

Returns Array of one-initialized data with the same shape and type as *ary*.

mpi_array.locale.copy

mpi_array.locale.**copy**(*ary*)

Return an array copy of the given object.

Parameters **ary** (*LndarrayProxy*) – Array to copy.

Return type *LndarrayProxy*

Returns A copy of ary.

1.28.3 Utilities

NdarrayMetaData(offset, strides, order)	Encapsulates, strides, offset and order argument of LndarrayProxy.__new__().

mpi_array.locale.NdarrayMetaData

class mpi_array.locale.**NdarrayMetaData**(*offset*, *strides*, *order*)
Bases: object

Encapsulates, strides, offset and order argument of LndarrayProxy.__new__().

Methods

__init__(offset, strides, order)	Construct.

mpi_array.locale.NdarrayMetaData.__init__

NdarrayMetaData.**__init__**(*offset*, *strides*, *order*)
Construct.

Parameters

- **offset** (None or int) – Offset of array data in buffer.
- **strides** (None or sequence of int) – Strides of data in memory.
- **order** ({C, F} or None) – Row-major (C-style) or column-major (Fortran-style) order.

Attributes

order

mpi_array.locale.NdarrayMetaData.order

NdarrayMetaData.**order**

1.29 The mpi_array.locale_test Module

Module defining *mpi_array.locale* unit-tests. Execute as:

```
python -m mpi_array.locale_test
```

1.29.1 Classes

WinLndarrayTest([methodName])	Tests for mpi_array.locale.win_lndarray.
LndarrayTest([methodName])	unittest.TestCase for *mpi_array.locale.lndarray*.
LndarrayProxyTest([methodName])	unittest.TestCase for *mpi_array.locale.LndarrayProxy*.

mpi_array.locale_test.WinLndarrayTest

class mpi_array.locale_test.**WinLndarrayTest**(*methodName='runTest'*)

Bases: *mpi_array.unittest.TestCase*

Tests for mpi_array.locale.win_lndarray.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	Hook method for setting up the test fixture before exercising it.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_construct()	Tests for mpi_array.locale.win_lndarray construction.
test_construct_with_invalid_comm()	Tests that ValueError is raised for invalid communicator argument passed to mpi_array.locale.win_lndarray constructor.

mpi_array.locale_test.WinLndarrayTest.__init__

WinLndarrayTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.locale_test.WinLndarrayTest.addCleanup

WinLndarrayTest.**addCleanup**(*function*, **args*, ***kwargs*)
Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.locale_test.WinLndarrayTest.addTypeEqualityFunc

WinLndarrayTest.**addTypeEqualityFunc**(*typeobj*, *function*)
Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.locale_test.WinLndarrayTest.assertArraySplitEqual

WinLndarrayTest.**assertArraySplitEqual**(*splt1*, *splt2*)
Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.locale_test.WinLndarrayTest.countTestCases

WinLndarrayTest.**countTestCases**()

mpi_array.locale_test.WinLndarrayTest.debug

WinLndarrayTest.**debug**()
Run the test without collecting errors in a TestResult

mpi_array.locale_test.WinLndarrayTest.defaultTestResult

WinLndarrayTest.**defaultTestResult**()

mpi_array.locale_test.WinLndarrayTest.doCleanups

WinLndarrayTest.**doCleanups**()
Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.locale_test.WinLndarrayTest.id

WinLndarrayTest.**id**()

mpi_array.locale_test.WinLndarrayTest.run

WinLndarrayTest.**run**(*result=None*)

mpi_array.locale_test.WinLndarrayTest.setUp

WinLndarrayTest.**setUp**()
Hook method for setting up the test fixture before exercising it.

mpi_array.locale_test.WinLndarrayTest.setUpClass

WinLndarrayTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.locale_test.WinLndarrayTest.shortDescription

WinLndarrayTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.locale_test.WinLndarrayTest.skipTest

WinLndarrayTest.**skipTest**(*reason*)
Skip this test.

mpi_array.locale_test.WinLndarrayTest.subTest

WinLndarrayTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.locale_test.WinLndarrayTest.tearDown

WinLndarrayTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.locale_test.WinLndarrayTest.tearDownClass

WinLndarrayTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.locale_test.WinLndarrayTest.test_construct

WinLndarrayTest.**test_construct**()
Tests for `mpi_array.locale.win_lndarray` construction.

mpi_array.locale_test.WinLndarrayTest.test_construct_with_invalid_comm

WinLndarrayTest.**test_construct_with_invalid_comm**()
Tests that ValueError is raised for invalid communicator argument passed to `mpi_array.locale.win_lndarray` constructor.

Attributes

longMessage
maxDiff

mpi_array.locale_test.WinLndarrayTest.longMessage

WinLndarrayTest.**longMessage = True**

mpi_array.locale_test.WinLndarrayTest.maxDiff

WinLndarrayTest.**maxDiff = 640**

mpi_array.locale_test.LndarrayTest

class mpi_array.locale_test.**LndarrayTest**(*methodName='runTest'*)
Bases: *mpi_array.unittest.TestCase*

unittest.TestCase for *mpi_array.locale.lndarray*.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.

Continued on next page

Table 1.178 – continued from previous page

assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	Hook method for setting up the test fixture before exercising it.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_construct()	Tests mpi_array.locale.lndarray.__new__().
test_numpy_sum()	Test numpy.sum() reduction using a mpi_array.locale.lndarray as argument.
test_view()	Tests mpi_array.locale.lndarray.__getitem__().

mpi_array.locale_test.LndarrayTest.__init__

LndarrayTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.locale_test.LndarrayTest.addCleanup

LndarrayTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.locale_test.LndarrayTest.addTypeEqualityFunc

LndarrayTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.locale_test.LndarrayTest.assertArraySplitEqual

LndarrayTest.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.locale_test.LndarrayTest.countTestCases

LndarrayTest.**countTestCases**()

mpi_array.locale_test.LndarrayTest.debug

LndarrayTest.**debug**()

Run the test without collecting errors in a TestResult

mpi_array.locale_test.LndarrayTest.defaultTestResult

LndarrayTest.**defaultTestResult**()

mpi_array.locale_test.LndarrayTest.doCleanups

LndarrayTest.**doCleanups**()

Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.locale_test.LndarrayTest.id

LndarrayTest.**id**()

mpi_array.locale_test.LndarrayTest.run

LndarrayTest.**run**(*result=None*)

mpi_array.locale_test.LndarrayTest.setUp

LndarrayTest.**setUp**()
Hook method for setting up the test fixture before exercising it.

mpi_array.locale_test.LndarrayTest.setUpClass

LndarrayTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.locale_test.LndarrayTest.shortDescription

LndarrayTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.locale_test.LndarrayTest.skipTest

LndarrayTest.**skipTest**(*reason*)
Skip this test.

mpi_array.locale_test.LndarrayTest.subTest

LndarrayTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.locale_test.LndarrayTest.tearDown

LndarrayTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.locale_test.LndarrayTest.tearDownClass

LndarrayTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.locale_test.LndarrayTest.test_construct

LndarrayTest.**test_construct**()
Tests *mpi_array.locale.lndarray.__new__()*.

mpi_array.locale_test.LndarrayTest.test_numpy_sum

LndarrayTest.**test_numpy_sum**()
Test numpy.sum() reduction using a *mpi_array.locale.lndarray* as argument.

mpi_array.locale_test.LndarrayTest.test_view

LndarrayTest.**test_view**()
Tests mpi_array.locale.lndarray.__getitem__().

Attributes

longMessage
maxDiff

mpi_array.locale_test.LndarrayTest.longMessage

LndarrayTest.**longMessage** = True

mpi_array.locale_test.LndarrayTest.maxDiff

LndarrayTest.**maxDiff** = 640

mpi_array.locale_test.LndarrayProxyTest

class mpi_array.locale_test.**LndarrayProxyTest**(*methodName='runTest'*)
Bases: *mpi_array.unittest.TestCase*

unittest.TestCase for *mpi_array.locale.LndarrayProxy*.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.

Continued on next page

Table 1.180 – continued from previous page

do_test_views_2d([halo])	Test for mpi_array.locale.LndarrayProxy.rank_view_n() and mpi_array.locale.LndarrayProxy.rank_view_h().
id()	
run([result])	
setUp()	Hook method for setting up the test fixture before exercising it.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_construct_arg_checking()	Test for mpi_array.locale.LndarrayProxy.__new__().
test_copy_non_shared_1d()	Test for _locale.copy().
test_copy_shared_1d()	Test for _locale.copy().
test_empty_non_shared_1d()	Test for _locale.empty() and _locale.empty_like().
test_empty_shared_1d()	Test for _locale.empty() and _locale.empty_like().
test_fill()	Test for mpi_array.locale.LndarrayProxy.fill().
test_get_and_set_item()	
test_ones_non_shared_1d()	Test for _locale.ones() and _locale.ones_like().
test_ones_shared_1d()	Test for _locale.ones() and _locale.ones_like().
test_views_2d_halo()	
test_views_2d_no_halo()	
test_zeros_non_shared_1d()	Test for _locale.zeros() and _locale.zeros_like().
test_zeros_shared_1d()	Test for _locale.zeros() and _locale.zeros_like().

mpi_array.locale_test.LndarrayProxyTest.__init__

LndarrayProxyTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.locale_test.LndarrayProxyTest.addCleanup

LndarrayProxyTest.**addCleanup**(*function*, **args*, ***kwargs*)
Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.locale_test.LndarrayProxyTest.addTypeEqualityFunc

LndarrayProxyTest.**addTypeEqualityFunc**(*typeobj*, *function*)
Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.locale_test.LndarrayProxyTest.assertArraySplitEqual

LndarrayProxyTest.**assertArraySplitEqual**(*splt1*, *splt2*)
Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.locale_test.LndarrayProxyTest.countTestCases

LndarrayProxyTest.**countTestCases**()

mpi_array.locale_test.LndarrayProxyTest.debug

LndarrayProxyTest.**debug**()
Run the test without collecting errors in a TestResult

mpi_array.locale_test.LndarrayProxyTest.defaultTestResult

LndarrayProxyTest.**defaultTestResult**()

mpi_array.locale_test.LndarrayProxyTest.doCleanups

LndarrayProxyTest.**doCleanups**()
Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.locale_test.LndarrayProxyTest.do_test_views_2d

LndarrayProxyTest.**do_test_views_2d**(*halo=0*)
Test for *mpi_array.locale.LndarrayProxy.rank_view_n()* and *mpi_array.locale.LndarrayProxy.rank_view_h()*.

mpi_array.locale_test.LndarrayProxyTest.id

LndarrayProxyTest.**id**()

mpi_array.locale_test.LndarrayProxyTest.run

LndarrayProxyTest.**run**(*result=None*)

mpi_array.locale_test.LndarrayProxyTest.setUp

LndarrayProxyTest.**setUp**()
Hook method for setting up the test fixture before exercising it.

mpi_array.locale_test.LndarrayProxyTest.setUpClass

LndarrayProxyTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.locale_test.LndarrayProxyTest.shortDescription

LndarrayProxyTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.locale_test.LndarrayProxyTest.skipTest

LndarrayProxyTest.**skipTest**(*reason*)
Skip this test.

mpi_array.locale_test.LndarrayProxyTest.subTest

LndarrayProxyTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.locale_test.LndarrayProxyTest.tearDown

LndarrayProxyTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.locale_test.LndarrayProxyTest.tearDownClass

LndarrayProxyTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.locale_test.LndarrayProxyTest.test_construct_arg_checking

LndarrayProxyTest.**test_construct_arg_checking**()
Test for `mpi_array.locale.LndarrayProxy.__new__()`.

mpi_array.locale_test.LndarrayProxyTest.test_copy_non_shared_1d

LndarrayProxyTest.**test_copy_non_shared_1d**()
Test for `_locale.copy()`.

mpi_array.locale_test.LndarrayProxyTest.test_copy_shared_1d

LndarrayProxyTest.**test_copy_shared_1d**()
Test for `_locale.copy()`.

mpi_array.locale_test.LndarrayProxyTest.test_empty_non_shared_1d

LndarrayProxyTest.**test_empty_non_shared_1d**()
Test for `_locale.empty()` and `_locale.empty_like()`.

mpi_array.locale_test.LndarrayProxyTest.test_empty_shared_1d

LndarrayProxyTest.**test_empty_shared_1d**()
Test for `_locale.empty()` and `_locale.empty_like()`.

mpi_array.locale_test.LndarrayProxyTest.test_fill

LndarrayProxyTest.**test_fill**()
Test for *mpi_array.locale.LndarrayProxy.fill()*.

mpi_array.locale_test.LndarrayProxyTest.test_get_and_set_item

LndarrayProxyTest.**test_get_and_set_item**()

mpi_array.locale_test.LndarrayProxyTest.test_ones_non_shared_1d

LndarrayProxyTest.**test_ones_non_shared_1d**()
Test for _locale.ones() and _locale.ones_like().

mpi_array.locale_test.LndarrayProxyTest.test_ones_shared_1d

LndarrayProxyTest.**test_ones_shared_1d**()
Test for _locale.ones() and _locale.ones_like().

mpi_array.locale_test.LndarrayProxyTest.test_views_2d_halo

LndarrayProxyTest.**test_views_2d_halo**()

mpi_array.locale_test.LndarrayProxyTest.test_views_2d_no_halo

LndarrayProxyTest.**test_views_2d_no_halo**()

mpi_array.locale_test.LndarrayProxyTest.test_zeros_non_shared_1d

LndarrayProxyTest.**test_zeros_non_shared_1d**()
Test for _locale.zeros() and _locale.zeros_like().

mpi_array.locale_test.LndarrayProxyTest.test_zeros_shared_1d

LndarrayProxyTest.**test_zeros_shared_1d**()
Test for _locale.zeros() and _locale.zeros_like().

Attributes

longMessage
maxDiff

mpi_array.locale_test.LndarrayProxyTest.longMessage

LndarrayProxyTest.**longMessage** = True

mpi_array.locale_test.LndarrayProxyTest.maxDiff

LndarrayProxyTest.**maxDiff** = 640

1.30 The mpi_array.logging Module

Default initialisation of python logging.

Some simple wrappers of python built-in `logging` module for *mpi_array* logging.

1.30.1 Classes and Functions

SplitStreamHandler([outstr, errstr, splitlevel])	A python `logging.handlers` `Handler` class for splitting logging messages to different streams depending on the logging-level.
initialise_loggers(names[, log_level, ...])	Initialises specified loggers to generate output at the specified logging level.
get_rank_logger(name[, comm, ranks])	Returns `logging.Logger` object for message logging.
get_root_logger(name[, comm, root_rank])	Returns a `logging.Logger` object with time-stamp, *comm*`.Get_name()` and *comm*`.Get_rank()` in the message.
LoggerFactory()	Factory for generating `logging.Logger` instances.

mpi_array.logging.SplitStreamHandler

class mpi_array.logging.**SplitStreamHandler**(*outstr=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*, *errstr=<_io.TextIOWrapper name='<stderr>' mode='w' encoding='UTF-8'>*, *splitlevel=30*)

Bases: `mpi_array.logging._Python3SplitStreamHandler`

A python `logging.handlers` `Handler` class for splitting logging messages to different streams depending on the logging-level.

Methods

__init__([outstr, errstr, splitlevel])	Initialise with a pair of streams and a threshold level which determines the stream where the messages are writting.
acquire()	Acquire the I/O thread lock.
addFilter(filter)	Add the specified filter to this handler.
close()	Tidy up any resources used by the handler.
createLock()	Acquire a thread lock for serializing access to the underlying I/O.
emit(record)	Emit a record.
filter(record)	Determine if a record is loggable by consulting all the filters.
flush()	Flushes the stream.
format(record)	Format the specified record.
get_name()	
handle(record)	Conditionally emit the specified logging record.
handleError(record)	Handle errors which occur during an emit() call.
release()	Release the I/O thread lock.

Continued on next page

Table 1.183 – continued from previous page

removeFilter(filter)	Remove the specified filter from this handler.
setFormatter(fmt)	Set the formatter for this handler.
setLevel(level)	Set the logging level of this handler.
set_name(name)	

mpi_array.logging.SplitStreamHandler.__init__

SplitStreamHandler.**__init__**(*outstr=<_io.TextIOWrapper name='<stdout>' mode='w' encoding='UTF-8'>*, *errstr=<_io.TextIOWrapper name='<stderr>' mode='w' encoding='UTF-8'>*, *splitlevel=30*)

Initialise with a pair of streams and a threshold level which determines the stream where the messages are writting.

Parameters

- **outstr** (*file-like*) – Logging messages are written to this stream if the message level is less than `self.splitLevel`.
- **errstr** (*stream*) – Logging messages are written to this stream if the message level is greater-than-or-equal-to `self.splitLevel`.
- **splitlevel** (*int*) – Logging level threshold determining split streams for log messages.

mpi_array.logging.SplitStreamHandler.acquire

SplitStreamHandler.**acquire**()

Acquire the I/O thread lock.

mpi_array.logging.SplitStreamHandler.addFilter

SplitStreamHandler.**addFilter**(*filter*)

Add the specified filter to this handler.

mpi_array.logging.SplitStreamHandler.close

SplitStreamHandler.**close**()

Tidy up any resources used by the handler.

This version removes the handler from an internal map of handlers, _handlers, which is used for handler lookup by name. Subclasses should ensure that this gets called from overridden close() methods.

mpi_array.logging.SplitStreamHandler.createLock

SplitStreamHandler.**createLock**()

Acquire a thread lock for serializing access to the underlying I/O.

mpi_array.logging.SplitStreamHandler.emit

SplitStreamHandler.**emit**(*record*)
Emit a record.

If a formatter is specified, it is used to format the record. The record is then written to the stream with a trailing newline. If exception information is present, it is formatted using traceback.print_exception and appended to the stream. If the stream has an 'encoding' attribute, it is used to determine how to do the output to the stream.

mpi_array.logging.SplitStreamHandler.filter

SplitStreamHandler.**filter**(*record*)
Determine if a record is loggable by consulting all the filters.

The default is to allow the record to be logged; any filter can veto this and the record is then dropped. Returns a zero value if a record is to be dropped, else non-zero.

Changed in version 3.2: Allow filters to be just callables.

mpi_array.logging.SplitStreamHandler.flush

SplitStreamHandler.**flush**()
Flushes the stream.

mpi_array.logging.SplitStreamHandler.format

SplitStreamHandler.**format**(*record*)
Format the specified record.

If a formatter is set, use it. Otherwise, use the default formatter for the module.

mpi_array.logging.SplitStreamHandler.get_name

SplitStreamHandler.**get_name**()

mpi_array.logging.SplitStreamHandler.handle

SplitStreamHandler.**handle**(*record*)
Conditionally emit the specified logging record.

Emission depends on filters which may have been added to the handler. Wrap the actual emission of the record with acquisition/release of the I/O thread lock. Returns whether the filter passed the record for emission.

mpi_array.logging.SplitStreamHandler.handleError

SplitStreamHandler.**handleError**(*record*)
Handle errors which occur during an emit() call.

This method should be called from handlers when an exception is encountered during an emit() call. If raiseExceptions is false, exceptions get silently ignored. This is what is mostly wanted for a logging system - most users will not care about errors in the logging system, they are more interested in application errors. You could, however, replace this with a custom handler if you wish. The record which was being processed is passed in to this method.

mpi_array.logging.SplitStreamHandler.release

SplitStreamHandler.**release**()
Release the I/O thread lock.

mpi_array.logging.SplitStreamHandler.removeFilter

SplitStreamHandler.**removeFilter**(*filter*)
Remove the specified filter from this handler.

mpi_array.logging.SplitStreamHandler.setFormatter

SplitStreamHandler.**setFormatter**(*fmt*)
Set the formatter for this handler.

mpi_array.logging.SplitStreamHandler.setLevel

SplitStreamHandler.**setLevel**(*level*)
Set the logging level of this handler. level must be an int or a str.

mpi_array.logging.SplitStreamHandler.set_name

SplitStreamHandler.**set_name**(*name*)

Attributes

name
terminator

mpi_array.logging.SplitStreamHandler.name

SplitStreamHandler.**name**

mpi_array.logging.SplitStreamHandler.terminator

SplitStreamHandler.**terminator** = '\n'

mpi_array.logging.initialise_loggers

mpi_array.logging.**initialise_loggers**(*names*, *log_level=30*, *handler_class=<class 'mpi_array.logging.SplitStreamHandler'>*)

Initialises specified loggers to generate output at the specified logging level. If the specified named loggers do not exist, they are created.

Parameters

- **names** (list of str) – List of logger names.
- **log_level** (int) – Log level for messages, typically one of logging.DEBUG, logging.INFO, logging.WARN, logging.ERROR or logging.CRITICAL. See Logging Levels.
- **handler_class** (One of the logging.handlers classes.) – The handler class for output of log messages, for example *SplitStreamHandler* or logging.StreamHandler.

mpi_array.logging.get_rank_logger

mpi_array.logging.**get_rank_logger**(*name*, *comm=None*, *ranks=None*)

Returns logging.Logger object for message logging.

Parameters

- **name** (str) – Name of logger (note that the name of logger actually created will be *name* + ".rank." + ("%04d" % *comm*.Get_rank())).
- **comm** (mpi4py.MPI.Comm) – MPI communicator. Used for determining the rank of this process. If None uses mpi4py.MPI.COMM_WORLD.
- **ranks** (None or list-of-int) – Limits logging output to ranks specified in this list. If None, all ranks produce logging output.

Return type logging.Logger

Returns Logger object.

mpi_array.logging.get_root_logger

mpi_array.logging.**get_root_logger**(*name*, *comm=None*, *root_rank=0*)

Returns a logging.Logger object with time-stamp, *comm*.Get_name() and *comm*.Get_rank() in the message. Logging output limited to the MPI rank specified by *root_rank*.

Parameters

- **name** (str) – Name of logger (note that the name of logger actually created will be *name* + ".rank." + ("%04d" % *comm*.Get_rank())).
- **comm** (mpi4py.MPI.Comm) – MPI communicator. Used for determining the rank of this process. If None uses mpi4py.MPI.COMM_WORLD.
- **root_rank** (int) – Logging output is limited to this rank, the returned logging.Logger objects on other ranks have a logging.NullHandler.

Return type logging.Logger

Returns Logger object.

mpi_array.logging.LoggerFactory

class mpi_array.logging.**LoggerFactory**
Bases: object

Factory for generating logging.Logger instances.

Methods

__init__()	
get_formatter([prefix_string])	Returns logging.Formatter object which produces messages with *time* and prefix_string prefix.
get_rank_logger(name[, comm, ranks, rank_string])	Returns a logging.Logger object with time-stamp, *comm*.Get_name() and *comm*.Get_rank() in the message.
get_root_logger(name[, comm, root_rank])	Returns a logging.Logger object with time-stamp, *comm*.Get_name() and *comm*.Get_rank() in the message.

mpi_array.logging.LoggerFactory.__init__

LoggerFactory.**__init__**()

mpi_array.logging.LoggerFactory.get_formatter

LoggerFactory.**get_formatter**(*prefix_string='MPIARY\ '*)
Returns logging.Formatter object which produces messages with *time* and prefix_string prefix.

Parameters **prefix_string** (str or None) – Prefix for all logging messages.

Return type logging.Formatter

Returns Regular formatter for logging.

mpi_array.logging.LoggerFactory.get_rank_logger

LoggerFactory.**get_rank_logger**(*name*, *comm=None*, *ranks=None*, *rank_string='rank'*)
Returns a logging.Logger object with time-stamp, *comm*.Get_name() and *comm*.Get_rank() in the message.

Parameters

- **name** (str) – Name of logger (note that the name of logger actuallycreated will be *name* + "." + *comm*.Get_name() + ".rank." + ("%04d" % *comm*.Get_rank())).
- **comm** (mpi4py.MPI.Comm) – MPI communicator. Used for determining the rank of this process. If None uses mpi4py.MPI.COMM_WORLD.
- **ranks** (None or list-of-int) – Limits logging output to ranks specified in this list. If None, all ranks produce logging output.

Return type logging.Logger

Returns Logger object.

mpi_array.logging.LoggerFactory.get_root_logger

LoggerFactory.**get_root_logger**(*name*, *comm=None*, *root_rank=0*)

Returns a logging.Logger object with time-stamp, *comm*.Get_name() and *comm*.Get_rank() in the message. Logging output limited to the MPI rank specified by *root_rank*.

Parameters

- **name** (str) – Name of logger (note that the name of logger actually created will be *name* + ".rank." + ("%04d" % *comm*.Get_rank())).
- **comm** (mpi4py.MPI.Comm) – MPI communicator. Used for determining the rank of this process. If None uses mpi4py.MPI.COMM_WORLD.
- **root_rank** (int) – Logging output is limited to this rank, the returned logging.Logger objects on other ranks have a logging.NullHandler.

Return type logging.Logger

Returns Logger object.

1.30.2 Attributes

mpi_array.logging.**logger_factory = <mpi_array.logging.LoggerFactory object>**

Factory for creating logging.Logger objects. Can set value to different instance in order to customise logging output.

1.31 The mpi_array.rtd Module

Sets up mock modules for readthedocs.org sphinx builds. See this FAQ.

1.31.1 Functions

initialise_mock_modules(module_name_list)	Updates system modules (sys.modules.update()) with unittest.mock.MagicMock objects.

mpi_array.rtd.initialise_mock_modules

mpi_array.rtd.**initialise_mock_modules**(*module_name_list*)

Updates system modules (sys.modules.update()) with unittest.mock.MagicMock objects.

Parameters **module_name_list** (sequence of str) – List of module names to be replaced/initialised with MagicMock instances in sys.modules.

1.31.2 Attributes

mpi_array.rtd.**MOCK_MODULES = ['mpi4py', 'mpi4py.MPI']**

List of module names

1.32 The mpi_array.tests Module

Module for running all *mpi_array* unit-tests, including unittest test-cases and doctest tests for module doc-strings and sphinx (RST) documentation. Execute as:

```
python -m mpi_array.tests
```

1.33 The mpi_array.types Module

Convert numpy.dtype to mpi4pi.MPI.Datatype.

1.33.1 Functions

to_datatype(dtyp)	Converts a numpy.dtype to a mpi4py.MPI.Datatype.

mpi_array.types.to_datatype

mpi_array.types.**to_datatype**(*dtyp*)

Converts a numpy.dtype to a mpi4py.MPI.Datatype.

Parameters **dtyp** (numpy.dtype) – This gets converted to a mpi4py.MPI.Datatype.

Return type mpi4py.MPI.Datatype

Returns A mpi4py.MPI.Datatype for the given *dtyp*.

1.33.2 Utilities

create_lookup()	Creates a collections.defaultdict of (numpy.dtype, mpi4py.MPI.Datatype) key-value pairs.
create_datatype(dtyp)	Creates a mpi4py.MPI.Datatype from a given numpy.dtype.
find_or_create_datatype(dtyp)	Converts a numpy.dtype to a mpi4py.MPI.Datatype.

mpi_array.types.create_lookup

mpi_array.types.**create_lookup**()

Creates a collections.defaultdict of (numpy.dtype, mpi4py.MPI.Datatype) key-value pairs. Populated with basic types from mpi4py.MPI._typedict.

Return type collections.defaultdict

Returns A collections.defaultdict with default element being None.

mpi_array.types.create_datatype

mpi_array.types.**create_datatype**(*dtyp*)
Creates a mpi4py.MPI.Datatype from a given numpy.dtype.

Parameters **dtyp** (numpy.dtype) – This gets converted to a mpi4py.MPI.Datatype.

Return type mpi4py.MPI.Datatype

Returns A mpi4py.MPI.Datatype for the given *dtyp*.

mpi_array.types.find_or_create_datatype

mpi_array.types.**find_or_create_datatype**(*dtyp*)
Converts a numpy.dtype to a mpi4py.MPI.Datatype. Uses a lookup dictionary to find the MPI datatype which matches the *dtype* numpy.dtype. If no match is found, a new instance of a mpi4py.MPI.Datatype is created (and added to the lookup dictionary).

Parameters **dtyp** (numpy.dtype) – This gets converted to a mpi4py.MPI.Datatype.

Return type mpi4py.MPI.Datatype

Returns A mpi4py.MPI.Datatype for the given *dtyp*.

1.34 The mpi_array.types_test Module

Module defining *mpi_array.types* unit-tests. Execute as:

```
python -m mpi_array.types_test
```

1.34.1 Classes

TypesTest([methodName])	unittest.TestCase for *mpi_array.types.to_datatype*.

mpi_array.types_test.TypesTest

class mpi_array.types_test.**TypesTest**(*methodName='runTest'*)
Bases: *mpi_array.unittest.TestCase*

unittest.TestCase for *mpi_array.types.to_datatype*.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.

Continued on next page

Table 1.190 – continued from previous page

assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	Set up, assign logging.Logger object.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_basic()	Test for mpi_array.types.to_datatype converting basic numpy.dtype to MPI data types.
test_contiguous()	Test for mpi_array.types.to_datatype converting contiguous numpy.dtype to MPI data types.
test_struct()	Test for mpi_array.types.to_datatype converting structure numpy.dtype to MPI data types.
test_struct_bcast()	Tests MPI communications with structured arrays.

mpi_array.types_test.TypesTest.__init__

TypesTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.types_test.TypesTest.addCleanup

TypesTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.types_test.TypesTest.addTypeEqualityFunc

TypesTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.types_test.TypesTest.assertArraySplitEqual

TypesTest.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.types_test.TypesTest.countTestCases

TypesTest.**countTestCases**()

mpi_array.types_test.TypesTest.debug

TypesTest.**debug**()

Run the test without collecting errors in a TestResult

mpi_array.types_test.TypesTest.defaultTestResult

TypesTest.**defaultTestResult**()

mpi_array.types_test.TypesTest.doCleanups

TypesTest.**doCleanups**()

Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.types_test.TypesTest.id

TypesTest.**id**()

mpi_array.types_test.TypesTest.run

TypesTest.**run**(*result=None*)

mpi_array.types_test.TypesTest.setUp

TypesTest.**setUp**()
Set up, assign logging.Logger object.

mpi_array.types_test.TypesTest.setUpClass

TypesTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.types_test.TypesTest.shortDescription

TypesTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.types_test.TypesTest.skipTest

TypesTest.**skipTest**(*reason*)
Skip this test.

mpi_array.types_test.TypesTest.subTest

TypesTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.types_test.TypesTest.tearDown

TypesTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.types_test.TypesTest.tearDownClass

TypesTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.types_test.TypesTest.test_basic

TypesTest.**test_basic**()
Test for *mpi_array.types.to_datatype* converting basic numpy.dtype to MPI data types.

mpi_array.types_test.TypesTest.test_contiguous

TypesTest.**test_contiguous**()
Test for *mpi_array.types.to_datatype* converting contiguous numpy.dtype to MPI data types.

mpi_array.types_test.TypesTest.test_struct

TypesTest.**test_struct**()
Test for *mpi_array.types.to_datatype* converting structure numpy.dtype to MPI data types.

mpi_array.types_test.TypesTest.test_struct_bcast

TypesTest.**test_struct_bcast**()
Tests MPI communications with structured arrays.

Attributes

longMessage
maxDiff

mpi_array.types_test.TypesTest.longMessage

TypesTest.**longMessage** = True

mpi_array.types_test.TypesTest.maxDiff

TypesTest.**maxDiff** = 640

1.35 The mpi_array.utils Module

Various utilities.

1.35.1 Functions

get_shared_mem_usage_percent_string([...])	Returns a string indicating the current percentage of available shared memory which is allocated.
log_shared_memory_alloc(logger, pfx, ...[, ...])	Generates logging message which indicates amount of shared-memory allocated using call to mpi4py.MPI.Win.Allocate_shared().
log_memory_alloc(logger, pfx, ...[, buffer])	Generates logging message which indicates amount of memory allocated using call to mpi4py.MPI.Win.Allocate().

mpi_array.utils.get_shared_mem_usage_percent_string

mpi_array.utils.**get_shared_mem_usage_percent_string**(*shm_file_name='/dev/shm'*)
Returns a string indicating the current percentage of available shared memory which is allocated.

Parameters **shm_file_name** (str) – Absolute path of shared-memory file.

mpi_array.utils.log_shared_memory_alloc

mpi_array.utils.**log_shared_memory_alloc**(*logger*, *pfx*, *num_rank_bytes*, *rank_shape*, *dtype*, *buffer=None*)
Generates logging message which indicates amount of shared-memory allocated using call to mpi4py.MPI.Win.Allocate_shared().

mpi_array.utils.log_memory_alloc

mpi_array.utils.**log_memory_alloc**(*logger*, *pfx*, *num_rank_bytes*, *rank_shape*, *dtype*, *buffer=None*)
Generates logging message which indicates amount of memory allocated using call to mpi4py.MPI.Win.Allocate().

1.36 The mpi_array.unittest Module

Some simple wrappers of python built-in unittest module for *mpi_array* unit-tests.

1.36.1 Classes and Functions

TestCase([methodName])	Extends unittest.TestCase with the assertArraySplitEqual().

mpi_array.unittest.TestCase

class mpi_array.unittest.**TestCase**(*methodName='runTest'*)
Bases: unittest.case.TestCase

Extends unittest.TestCase with the *assertArraySplitEqual()*.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.

Continued on next page

Table 1.194 – continued from previous page

assertArraySplitEqual(splt1, splt2)	Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	Hook method for setting up the test fixture before exercising it.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.unittest.TestCase.__init__

`TestCase.`**`__init__`**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.unittest.TestCase.addCleanup

`TestCase.`**`addCleanup`**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.unittest.TestCase.addTypeEqualityFunc

`TestCase.`**`addTypeEqualityFunc`**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.unittest.TestCase.assertArraySplitEqual

TestCase.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.

Parameters

- **splt1** (list of numpy.ndarray) – First object in equality comparison.
- **splt2** (list of numpy.ndarray) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of splt2.

mpi_array.unittest.TestCase.countTestCases

TestCase.**countTestCases**()

mpi_array.unittest.TestCase.debug

TestCase.**debug**()

Run the test without collecting errors in a TestResult

mpi_array.unittest.TestCase.defaultTestResult

TestCase.**defaultTestResult**()

mpi_array.unittest.TestCase.doCleanups

TestCase.**doCleanups**()

Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.unittest.TestCase.id

TestCase.**id**()

mpi_array.unittest.TestCase.run

TestCase.**run**(*result=None*)

mpi_array.unittest.TestCase.setUp

TestCase.**setUp**()

Hook method for setting up the test fixture before exercising it.

mpi_array.unittest.TestCase.setUpClass

TestCase.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.unittest.TestCase.shortDescription

TestCase.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.unittest.TestCase.skipTest

TestCase.**skipTest**(*reason*)
Skip this test.

mpi_array.unittest.TestCase.subTest

TestCase.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.unittest.TestCase.tearDown

TestCase.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.unittest.TestCase.tearDownClass

TestCase.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

Attributes

longMessage
maxDiff

mpi_array.unittest.TestCase.longMessage

TestCase.**longMessage** = **True**

mpi_array.unittest.TestCase.maxDiff

TestCase.**maxDiff** = **640**

TestProgram(*args, **kwargs)	A command-line program that runs a set of tests, extends `unittest.TestProgram` by using logging rather than standard stream.
TextTestRunner(*args, **kwargs)	A test runner class that displays results in textual form.
TextTestResult(stream, descriptions, verbosity)	
main(module_name[, log_level, ...])	Like `unittest.main()`, initialises `logging.Logger` objects and instantiates a *TestProgram* to discover and run *TestCase* objects.

mpi_array.unittest.TestProgram

class `mpi_array.unittest.`**`TestProgram`**(**args*, ***kwargs*)

Bases: `unittest.main.TestProgram`

A command-line program that runs a set of tests, extends `unittest.TestProgram` by using logging rather than standard stream.

Methods

__init__(*args, **kwargs)	
createTests()	
parseArgs(argv)	
runTests()	
usageExit([msg])	

mpi_array.unittest.TestProgram.__init__

`TestProgram.`**`__init__`**(**args*, ***kwargs*)

mpi_array.unittest.TestProgram.createTests

`TestProgram.`**`createTests`**()

mpi_array.unittest.TestProgram.parseArgs

`TestProgram.`**`parseArgs`**(*argv*)

mpi_array.unittest.TestProgram.runTests

`TestProgram.`**`runTests`**()

mpi_array.unittest.TestProgram.usageExit

`TestProgram.`**`usageExit`**(*msg=None*)

Attributes

buffer
catchbreak
failfast
module
progName
verbosity
warnings

mpi_array.unittest.TestProgram.buffer

TestProgram.**buffer** = None

mpi_array.unittest.TestProgram.catchbreak

TestProgram.**catchbreak** = None

mpi_array.unittest.TestProgram.failfast

TestProgram.**failfast** = None

mpi_array.unittest.TestProgram.module

TestProgram.**module** = None

mpi_array.unittest.TestProgram.progName

TestProgram.**progName** = None

mpi_array.unittest.TestProgram.verbosity

TestProgram.**verbosity** = 1

mpi_array.unittest.TestProgram.warnings

TestProgram.**warnings** = None

mpi_array.unittest.TextTestRunner

class mpi_array.unittest.**TextTestRunner**(**args*, ***kwargs*)

Bases: unittest.runner.TextTestRunner

A test runner class that displays results in textual form.

Extends unittest.TextTestRunner with logging output instead of sys.stderr output.

Methods

__init__(*args, **kwargs)	
run(test)	Run the given test case or test suite.

mpi_array.unittest.TextTestRunner.__init__

TextTestRunner.**__init__**(**args*, ***kwargs*)

mpi_array.unittest.TextTestRunner.run

TextTestRunner.**run**(*test*)

Run the given test case or test suite.

mpi_array.unittest.TextTestResult

class mpi_array.unittest.**TextTestResult**(*stream*, *descriptions*, *verbosity*)

Bases: unittest.runner.TextTestResult

Methods

__init__(stream, descriptions, verbosity)	
addError(test, err)	
addExpectedFailure(test, err)	
addFailure(test, err)	
addSkip(test, reason)	
addSubTest(test, subtest, err)	Called at the end of a subtest.
addSuccess(test)	
addUnexpectedSuccess(test)	
getDescription(test)	
printErrorList(flavour, errors)	
printErrors()	
startTest(test)	
startTestRun()	Called once before any tests are executed.
stop()	Indicates that the tests should be aborted.
stopTest(test)	Called when the given test has been run
stopTestRun()	Called once after all tests are executed.
wasSuccessful()	Tells whether or not this result was a success.

mpi_array.unittest.TextTestResult.__init__

TextTestResult.**__init__**(*stream*, *descriptions*, *verbosity*)

mpi_array.unittest.TextTestResult.addError

TextTestResult.**addError**(*test*, *err*)

mpi_array.unittest.TextTestResult.addExpectedFailure

TextTestResult.**addExpectedFailure**(*test*, *err*)

mpi_array.unittest.TextTestResult.addFailure

TextTestResult.**addFailure**(*test*, *err*)

mpi_array.unittest.TextTestResult.addSkip

TextTestResult.**addSkip**(*test*, *reason*)

mpi_array.unittest.TextTestResult.addSubTest

TextTestResult.**addSubTest**(*test*, *subtest*, *err*)
Called at the end of a subtest. 'err' is None if the subtest ended successfully, otherwise it's a tuple of values as returned by sys.exc_info().

mpi_array.unittest.TextTestResult.addSuccess

TextTestResult.**addSuccess**(*test*)

mpi_array.unittest.TextTestResult.addUnexpectedSuccess

TextTestResult.**addUnexpectedSuccess**(*test*)

mpi_array.unittest.TextTestResult.getDescription

TextTestResult.**getDescription**(*test*)

mpi_array.unittest.TextTestResult.printErrorList

TextTestResult.**printErrorList**(*flavour*, *errors*)

mpi_array.unittest.TextTestResult.printErrors

TextTestResult.**printErrors**()

mpi_array.unittest.TextTestResult.startTest

TextTestResult.**startTest**(*test*)

mpi_array.unittest.TextTestResult.startTestRun

TextTestResult.**startTestRun**()
Called once before any tests are executed.

See startTest for a method called before each test.

mpi_array.unittest.TextTestResult.stop

TextTestResult.**stop**()
Indicates that the tests should be aborted.

mpi_array.unittest.TextTestResult.stopTest

TextTestResult.**stopTest**(*test*)
Called when the given test has been run

mpi_array.unittest.TextTestResult.stopTestRun

TextTestResult.**stopTestRun**()
Called once after all tests are executed.

See stopTest for a method called after each test.

mpi_array.unittest.TextTestResult.wasSuccessful

TextTestResult.**wasSuccessful**()
Tells whether or not this result was a success.

Attributes

separator1
separator2

mpi_array.unittest.TextTestResult.separator1

TextTestResult.**separator1** = '=='

mpi_array.unittest.TextTestResult.separator2

TextTestResult.**separator2** = '--'

mpi_array.unittest.main

mpi_array.unittest.**main**(*module_name*, *log_level=10*, *init_logger_names=None*, *verbosity=None*, *failfast=None*)
Like unittest.main(), initialises logging.Logger objects and instantiates a *TestProgram* to dis-

cover and run *TestCase* objects. Loads a set of tests from module and runs them; this is primarily for making test modules conveniently executable. The simplest use for this function is to include the following line at the end of a test module:

```
mpi_array.unittest.main(__name__)
```

If `__name__ == "__main__"`, then *discoverable* `unittest.TestCase` test cases are executed. Logging level can be explicitly set for a group of modules using:

```
import logging

mpi_array.unittest.main(
    __name__,
    logging.DEBUG,
    [__name__, "module_name_0", "module_name_1", "package.module_name_2"]
)
```

Parameters

- **module_name** (`str`) – If *module_name* `== "__main__"` then unit-tests are *discovered* and run.
- **log_level** (`int`) – The default logging level for all `mpi_array.logging.Logger` objects.
- **init_logger_names** (sequence of `str`) – List of logger names to initialise (using *mpi_array.logging.initialise_loggers()*). If `None`, then the list defaults to [*module_name*, `"mpi_array"`]. If list is empty no loggers are initialised.

1.37 The `mpi_array.update` Module

Helper classes for calculating sub-extent intersections in order to perform remote array element copying/updates.

1.37.1 Classes and Functions

ExtentAndRegion(locale_extent[, region_extent])	Container for *mpi_array.distribution.LocaleExtent* and an update region (*mpi_array.indexing.IndexingExtent*).
MpiExtentAndRegion(locale_extent, region_extent)	
ExtentUpdate(dst_extent_info, src_extent_info)	Source and destination indexing info for updating a sub-extent region.
PairExtentUpdate(dst_extent, src_extent, ...)	Source and destination indexing info for updating a sub-extent region.
MpiPairExtentUpdate(dst_extent, src_extent, ...)	Source and destination indexing info for updating the whole of a halo portion.
MpiPairExtentUpdateDifferentDtypes(...)	Over-rides *MpiPairExtentUpdate.do_get()* to buffer-copy and subsequent casting when source and destination arrays have different `numpy.dtype`.
HaloSingleExtentUpdate(dst_extent, ...)	Source and destination indexing info for updating a halo portion.

Continued on next page

Table 1.202 – continued from previous page

MpiHaloSingleExtentUpdate(dst_extent, ...)	Source and destination indexing info for updating the whole of a halo portion.
UpdatesForRedistribute(dst_distrib, src_distrib)	Collection of update extents for re-distribution of array elements from one distribution to another.
RmaUpdateExecutor(inter_win, dst_lndarray, ...)	Performs one-sided fetch of data from remote (source) locale arrays to update destination locale array.

mpi_array.update.ExtentAndRegion

class mpi_array.update.**ExtentAndRegion**(*locale_extent*, *region_extent=None*)

Bases: object

Container for *mpi_array.distribution.LocaleExtent* and an update region (*mpi_array.indexing.IndexingExtent*).

Methods

__init__(locale_extent[, region_extent])	

mpi_array.update.ExtentAndRegion.__init__

ExtentAndRegion.**__init__**(*locale_extent*, *region_extent=None*)

Attributes

locale_extent	
region_extent	

mpi_array.update.ExtentAndRegion.locale_extent

ExtentAndRegion.**locale_extent**

mpi_array.update.ExtentAndRegion.region_extent

ExtentAndRegion.**region_extent**

mpi_array.update.MpiExtentAndRegion

class mpi_array.update.**MpiExtentAndRegion**(*locale_extent*, *region_extent*, *dtype=None*, *order=None*, *mpi_data_type=None*, *mpi_order=None*)

Bases: *mpi_array.update.ExtentAndRegion*

Methods

__init__(locale_extent, region_extent[, ...])	
create_data_type(dtype[, order])	
initialise_mpi_data_type(dtype, order)	

mpi_array.update.MpiExtentAndRegion.__init__

MpiExtentAndRegion.**__init__**(*locale_extent*, *region_extent*, *dtype=None*, *order=None*, *mpi_data_type=None*, *mpi_order=None*)

mpi_array.update.MpiExtentAndRegion.create_data_type

MpiExtentAndRegion.**create_data_type**(*dtype*, *order='C'*)

mpi_array.update.MpiExtentAndRegion.initialise_mpi_data_type

MpiExtentAndRegion.**initialise_mpi_data_type**(*dtype*, *order*)

Attributes

locale_extent	
mpi_data_type	
region_extent	

mpi_array.update.MpiExtentAndRegion.locale_extent

MpiExtentAndRegion.**locale_extent**

mpi_array.update.MpiExtentAndRegion.mpi_data_type

MpiExtentAndRegion.**mpi_data_type**

mpi_array.update.MpiExtentAndRegion.region_extent

MpiExtentAndRegion.**region_extent**

mpi_array.update.ExtentUpdate

class mpi_array.update.**ExtentUpdate**(*dst_extent_info*, *src_extent_info*)

Bases: object

Source and destination indexing info for updating a sub-extent region.

Methods

__init__(dst_extent_info, src_extent_info)	Initialise.

mpi_array.update.ExtentUpdate.__init__

ExtentUpdate.**__init__**(*dst_extent_info*, *src_extent_info*)

Initialise.

Parameters

- **dst_extent_info** (*ExtentAndRegion*) – Info containing locale extent which is to receive region update.
- **dst_extent_info** – Info containing locale extent from which the region update is read.

Attributes

dst_extent	The locale `LocaleExtent` which is to receive sub-array update.
src_extent	The locale `CartLocaleExtent` from which the sub-array update is read.

mpi_array.update.ExtentUpdate.dst_extent

ExtentUpdate.**dst_extent**

The locale `LocaleExtent` which is to receive sub-array update.

mpi_array.update.ExtentUpdate.src_extent

ExtentUpdate.**src_extent**

The locale `CartLocaleExtent` from which the sub-array update is read.

mpi_array.update.PairExtentUpdate

class mpi_array.update.**PairExtentUpdate**(*dst_extent*, *src_extent*, *dst_update_extent*, *src_update_extent*)

Bases: *mpi_array.update.ExtentUpdate*

Source and destination indexing info for updating a sub-extent region.

Methods

__init__(dst_extent, src_extent, ...)	
copyto(dst_array, src_array, casting)	Copies the *src_update_extent* region from *src_array*

mpi_array.update.PairExtentUpdate.__init__

PairExtentUpdate.**__init__**(*dst_extent*, *src_extent*, *dst_update_extent*, *src_update_extent*)

mpi_array.update.PairExtentUpdate.copyto

PairExtentUpdate.**copyto**(*dst_array*, *src_array*, *casting*)

Copies the *src_update_extent* region from *src_array* to the *dst_update_extent* region of *dst_array*

Attributes

dst_extent	The locale LocaleExtent which is to receive sub-array update.
dst_update_extent	The locale sub-extent (IndexingExtent) to be updated.
src_extent	The locale CartLocaleExtent from which the sub-array update is read.
src_update_extent	The locale sub-extent (IndexingExtent) from which the update is read.

mpi_array.update.PairExtentUpdate.dst_extent

PairExtentUpdate.**dst_extent**

The locale LocaleExtent which is to receive sub-array update.

mpi_array.update.PairExtentUpdate.dst_update_extent

PairExtentUpdate.**dst_update_extent**

The locale sub-extent (IndexingExtent) to be updated.

mpi_array.update.PairExtentUpdate.src_extent

PairExtentUpdate.**src_extent**

The locale CartLocaleExtent from which the sub-array update is read.

mpi_array.update.PairExtentUpdate.src_update_extent

PairExtentUpdate.**src_update_extent**

The locale sub-extent (IndexingExtent) from which the update is read.

mpi_array.update.MpiPairExtentUpdate

class mpi_array.update.**MpiPairExtentUpdate**(*dst_extent*, *src_extent*, *dst_update_extent*, *src_update_extent*)

Bases: *mpi_array.update.ExtentUpdate*

Source and destination indexing info for updating the whole of a halo portion. Extends *ExtentUpdate* with API to create `mpi4py.MPI.Datatype` instances (using `mpi4py.MPI.Datatype.Create_subarray()`) for convenient transfer of sub-array data.

Methods

__init__(dst_extent, src_extent, ...)	
conclude()	
copyto(dst_array, src_array, casting)	Copies the *src_update_extent* region from *src_array*
do_get(mpi_win, target_src_rank, ...)	Performs calls `mpi4py.MPI.Win.Get()` method of `mpi_win` to perform the RMA data-transfer.
do_rget(mpi_win, target_src_rank, ...)	Performs calls `mpi4py.MPI.Win.Rget()` method of `mpi_win` to perform the RMA data-transfer.
initialise_data_types(dst_dtype, src_dtype, ...)	Assigns new instances of *mpi4py.MPI.Datatype* for the *dst_data_type* and *src_data_type* attributes.

mpi_array.update.MpiPairExtentUpdate.__init__

`MpiPairExtentUpdate.`**`__init__`**(*dst_extent*, *src_extent*, *dst_update_extent*, *src_update_extent*)

mpi_array.update.MpiPairExtentUpdate.conclude

`MpiPairExtentUpdate.`**`conclude`**`()`

mpi_array.update.MpiPairExtentUpdate.copyto

`MpiPairExtentUpdate.`**`copyto`**(*dst_array*, *src_array*, *casting*)

Copies the *src_update_extent* region from *src_array* to the *dst_update_extent* region of *dst_array*

mpi_array.update.MpiPairExtentUpdate.do_get

`MpiPairExtentUpdate.`**`do_get`**(*mpi_win*, *target_src_rank*, *origin_dst_buffer*)

Performs calls `mpi4py.MPI.Win.Get()` method of `mpi_win` to perform the RMA data-transfer.

Parameters

- **mpi_win** (`mpi4py.MPI.Win`) – Window used to retrieve update region for array.
- **target_src_rank** (`int`) – The rank of the target process in `mpi_win.group.rank`.
- **origin_dst_buffer** (`memoryview`) – The destination memory for the update, size of buffer should correspond to the size of the *dst_extent*.

mpi_array.update.MpiPairExtentUpdate.do_rget

MpiPairExtentUpdate.**do_rget**(*mpi_win*, *target_src_rank*, *origin_dst_buffer*)

Performs calls `mpi4py.MPI.Win.Rget()` method of `mpi_win` to perform the RMA data-transfer.

Parameters

- **mpi_win** (`mpi4py.MPI.Win`) – Window used to retrieve update region for array.
- **target_src_rank** (`int`) – The rank of the target process in `mpi_win.group.rank`.
- **origin_dst_buffer** (`memoryview`) – The destination memory for the update, size of buffer should correspond to the size of the *dst_extent*.

mpi_array.update.MpiPairExtentUpdate.initialise_data_types

MpiPairExtentUpdate.**initialise_data_types**(*dst_dtype*, *src_dtype*, *dst_order*, *src_order*)

Assigns new instances of *mpi4py.MPI.Datatype* for the *dst_data_type* and *src_data_type* attributes. Only creates new instances when the *dst_dtype*, *src_dtype* or *order* do not match existing instances.

Parameters

- **dst_dtype** (`numpy.dtype`) – The array element type of the array which is to receive data.
- **src_dtype** (`numpy.dtype`) – The array element type of the array from which data is copied.
- **order** (`str`) – Array memory layout, `"C"` for C array, or `"F"` for fortran array.

Attributes

casting	A `str` indicating the casting allowed between different `numpy.dtype` elements.
dst_data_type	A `mpi4py.MPI.Datatype` object created using `mpi4py.MPI.Datatype.Create_subarray()` which defines the sub-array of halo elements which are to receive update values.
dst_dtype	A `numpy.dtype` object indicating the element type of the destination array.
dst_extent	The locale `LocaleExtent` which is to receive sub-array update.
dst_update_extent	The locale sub-extent (`IndexingExtent`) to be updated.
src_data_type	A `mpi4py.MPI.Datatype` object created using `mpi4py.MPI.Datatype.Create_subarray()` which defines the sub-array of halo elements from which receive update values.
src_dtype	A `numpy.dtype` object indicating the element type of the source array.

Continued on next page

Table 1.212 – continued from previous page

src_extent	The locale `CartLocaleExtent` from which the sub-array update is read.
src_update_extent	The locale sub-extent (`IndexingExtent`) from which the update is read.

mpi_array.update.MpiPairExtentUpdate.casting

`MpiPairExtentUpdate.`**`casting`**
A `str` indicating the casting allowed between different `numpy.dtype` elements. See the `casting` parameter for the `numpy.copyto()` function.

mpi_array.update.MpiPairExtentUpdate.dst_data_type

`MpiPairExtentUpdate.`**`dst_data_type`**
A `mpi4py.MPI.Datatype` object created using `mpi4py.MPI.Datatype.Create_subarray()` which defines the sub-array of halo elements which are to receive update values.

mpi_array.update.MpiPairExtentUpdate.dst_dtype

`MpiPairExtentUpdate.`**`dst_dtype`**
A `numpy.dtype` object indicating the element type of the destination array.

mpi_array.update.MpiPairExtentUpdate.dst_extent

`MpiPairExtentUpdate.`**`dst_extent`**
The locale `LocaleExtent` which is to receive sub-array update.

mpi_array.update.MpiPairExtentUpdate.dst_update_extent

`MpiPairExtentUpdate.`**`dst_update_extent`**
The locale sub-extent (`IndexingExtent`) to be updated.

mpi_array.update.MpiPairExtentUpdate.src_data_type

`MpiPairExtentUpdate.`**`src_data_type`**
A `mpi4py.MPI.Datatype` object created using `mpi4py.MPI.Datatype.Create_subarray()` which defines the sub-array of halo elements from which receive update values.

mpi_array.update.MpiPairExtentUpdate.src_dtype

`MpiPairExtentUpdate.`**`src_dtype`**
A `numpy.dtype` object indicating the element type of the source array.

mpi_array.update.MpiPairExtentUpdate.src_extent

MpiPairExtentUpdate.**src_extent**
The locale CartLocaleExtent from which the sub-array update is read.

mpi_array.update.MpiPairExtentUpdate.src_update_extent

MpiPairExtentUpdate.**src_update_extent**
The locale sub-extent (IndexingExtent) from which the update is read.

mpi_array.update.MpiPairExtentUpdateDifferentDtypes

class mpi_array.update.**MpiPairExtentUpdateDifferentDtypes**(*dst_extent*, *src_extent*, *dst_update_extent*, *src_update_extent*)

Bases: *mpi_array.update.MpiPairExtentUpdate*

Over-rides *MpiPairExtentUpdate.do_get()* to buffer-copy and subsequent casting when source and destination arrays have different numpy.dtype.

Methods

__init__(dst_extent, src_extent, ...)	
conclude()	
copyto(dst_array, src_array, casting)	Copies the *src_update_extent* region from *src_array*
do_get(mpi_win, target_src_rank, ...)	Performs calls mpi4py.MPI.Win.Get() method of mpi_win to perform the RMA data-transfer.
do_rget(mpi_win, target_src_rank, ...)	
initialise_data_types(dst_dtype, src_dtype, ...)	Assigns new instances of *mpi4py.MPI.Datatype* for the *dst_data_type* and *src_data_type* attributes.

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.__init__

MpiPairExtentUpdateDifferentDtypes.**__init__**(*dst_extent*, *src_extent*, *dst_update_extent*, *src_update_extent*)

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.conclude

MpiPairExtentUpdateDifferentDtypes.**conclude**()

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.copyto

MpiPairExtentUpdateDifferentDtypes.**copyto**(*dst_array*, *src_array*, *casting*)
Copies the *src_update_extent* region from *src_array* to the *dst_update_extent* region of *dst_array*

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.do_get

MpiPairExtentUpdateDifferentDtypes.**do_get**(*mpi_win*, *target_src_rank*, *origin_dst_buffer*)

Performs calls `mpi4py.MPI.Win.Get()` method of `mpi_win` to perform the RMA data-transfer. Uses a locally allocated buffer to receive the data and then uses `numpy.copyto()` to convert the *src_dtype* to the *dst_dtype*.

Parameters

- **mpi_win** (`mpi4py.MPI.Win.Get`) – Window used to retrieve update region for array.
- **target_src_rank** (`int`) – The rank of the target process in `mpi_win.group.rank`.
- **origin_dst_buffer** (`memoryview`) – The destination memory for the update, size of buffer should correspond to the size of the *dst_extent*.

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.do_rget

MpiPairExtentUpdateDifferentDtypes.**do_rget**(*mpi_win*, *target_src_rank*, *origin_dst_buffer*)

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.initialise_data_types

MpiPairExtentUpdateDifferentDtypes.**initialise_data_types**(*dst_dtype*, *src_dtype*, *dst_order*, *src_order*)

Assigns new instances of *mpi4py.MPI.Datatype* for the *dst_data_type* and *src_data_type* attributes. Only creates new instances when the *dst_dtype*, *src_dtype* or *order* do not match existing instances.

Parameters

- **dst_dtype** (`numpy.dtype`) – The array element type of the array which is to receive data.
- **src_dtype** (`numpy.dtype`) – The array element type of the array from which data is copied.
- **order** (`str`) – Array memory layout, `"C"` for C array, or `"F"` for fortran array.

Attributes

casting	A `str` indicating the casting allowed between different `numpy.dtype` elements.
dst_data_type	A `mpi4py.MPI.Datatype` object created using `mpi4py.MPI.Datatype.Create_subarray()` which defines the sub-array of halo elements which are to receive update values.
dst_dtype	A `numpy.dtype` object indicating the element type of the destination array.

Continued on next page

Table 1.214 – continued from previous page

dst_extent	The locale `LocaleExtent` which is to receive sub-array update.
dst_update_extent	The locale sub-extent (`IndexingExtent`) to be updated.
src_data_type	A `mpi4py.MPI.Datatype` object created using `mpi4py.MPI.Datatype.Create_subarray()` which defines the sub-array of halo elements from which receive update values.
src_dtype	A `numpy.dtype` object indicating the element type of the source array.
src_extent	The locale `CartLocaleExtent` from which the sub-array update is read.
src_update_extent	The locale sub-extent (`IndexingExtent`) from which the update is read.

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.casting

`MpiPairExtentUpdateDifferentDtypes.`**`casting`**
A `str` indicating the casting allowed between different `numpy.dtype` elements. See the `casting` parameter for the `numpy.copyto()` function.

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.dst_data_type

`MpiPairExtentUpdateDifferentDtypes.`**`dst_data_type`**
A `mpi4py.MPI.Datatype` object created using `mpi4py.MPI.Datatype.Create_subarray()` which defines the sub-array of halo elements which are to receive update values.

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.dst_dtype

`MpiPairExtentUpdateDifferentDtypes.`**`dst_dtype`**
A `numpy.dtype` object indicating the element type of the destination array.

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.dst_extent

`MpiPairExtentUpdateDifferentDtypes.`**`dst_extent`**
The locale `LocaleExtent` which is to receive sub-array update.

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.dst_update_extent

`MpiPairExtentUpdateDifferentDtypes.`**`dst_update_extent`**
The locale sub-extent (`IndexingExtent`) to be updated.

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.src_data_type

`MpiPairExtentUpdateDifferentDtypes.`**`src_data_type`**
A `mpi4py.MPI.Datatype` object created using `mpi4py.MPI.Datatype.`

Create_subarray() which defines the sub-array of halo elements from which receive update values.

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.src_dtype

MpiPairExtentUpdateDifferentDtypes.**src_dtype**
A numpy.dtype object indicating the element type of the source array.

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.src_extent

MpiPairExtentUpdateDifferentDtypes.**src_extent**
The locale CartLocaleExtent from which the sub-array update is read.

mpi_array.update.MpiPairExtentUpdateDifferentDtypes.src_update_extent

MpiPairExtentUpdateDifferentDtypes.**src_update_extent**
The locale sub-extent (IndexingExtent) from which the update is read.

mpi_array.update.HaloSingleExtentUpdate

class mpi_array.update.**HaloSingleExtentUpdate**(*dst_extent*, *src_extent*, *update_extent*)
Bases: *mpi_array.update.ExtentUpdate*

Source and destination indexing info for updating a halo portion.

Methods

__init__(dst_extent, src_extent, update_extent)	

mpi_array.update.HaloSingleExtentUpdate.__init__

HaloSingleExtentUpdate.**__init__**(*dst_extent*, *src_extent*, *update_extent*)

Attributes

dst_extent	The locale LocaleExtent which is to receive sub-array update.
src_extent	The locale CartLocaleExtent from which the sub-array update is read.
update_extent	The IndexingExtent indicating the halo sub-array which is to be updated.

mpi_array.update.HaloSingleExtentUpdate.dst_extent

HaloSingleExtentUpdate.**dst_extent**
The locale LocaleExtent which is to receive sub-array update.

mpi_array.update.HaloSingleExtentUpdate.src_extent

HaloSingleExtentUpdate.**src_extent**
The locale CartLocaleExtent from which the sub-array update is read.

mpi_array.update.HaloSingleExtentUpdate.update_extent

HaloSingleExtentUpdate.**update_extent**
The IndexingExtent indicating the halo sub-array which is to be updated.

mpi_array.update.MpiHaloSingleExtentUpdate

class mpi_array.update.**MpiHaloSingleExtentUpdate**(*dst_extent*, *src_extent*, *update_extent*)
Bases: *mpi_array.update.ExtentUpdate*

Source and destination indexing info for updating the whole of a halo portion. Extends *ExtentUpdate* with API to create mpi4py.MPI.Datatype instances (using mpi4py.MPI.Datatype.Create_subarray()) for convenient transfer of sub-array data.

Methods

__init__(dst_extent, src_extent, update_extent)	
initialise_data_types(dtype, order)	Assigns new instances of *mpi4py.MPI.Datatype* for the *dst_data_type* and *src_data_type* attributes.

mpi_array.update.MpiHaloSingleExtentUpdate.__init__

MpiHaloSingleExtentUpdate.**__init__**(*dst_extent*, *src_extent*, *update_extent*)

mpi_array.update.MpiHaloSingleExtentUpdate.initialise_data_types

MpiHaloSingleExtentUpdate.**initialise_data_types**(*dtype*, *order*)
Assigns new instances of *mpi4py.MPI.Datatype* for the *dst_data_type* and *src_data_type* attributes. Only creates new instances when the *dtype* and *order* do not match existing instances.

Parameters

- **dtype** (numpy.dtype) – The array element type.
- **order** (str) – Array memory layout, "C" for C array, or "F" for fortran array.

Attributes

dst_data_type	A `mpi4py.MPI.Datatype` object created using `mpi4py.MPI.Datatype.Create_subarray()` which defines the sub-array of halo elements which are to receive update values.
dst_extent	The locale `LocaleExtent` which is to receive sub-array update.
src_data_type	A `mpi4py.MPI.Datatype` object created using `mpi4py.MPI.Datatype.Create_subarray()` which defines the sub-array of halo elements from which receive update values.
src_extent	The locale `CartLocaleExtent` from which the sub-array update is read.
update_extent	The `IndexingExtent` indicating the halo sub-array which is to be updated.

mpi_array.update.MpiHaloSingleExtentUpdate.dst_data_type

`MpiHaloSingleExtentUpdate.`**`dst_data_type`**
A `mpi4py.MPI.Datatype` object created using `mpi4py.MPI.Datatype.Create_subarray()` which defines the sub-array of halo elements which are to receive update values.

mpi_array.update.MpiHaloSingleExtentUpdate.dst_extent

`MpiHaloSingleExtentUpdate.`**`dst_extent`**
The locale `LocaleExtent` which is to receive sub-array update.

mpi_array.update.MpiHaloSingleExtentUpdate.src_data_type

`MpiHaloSingleExtentUpdate.`**`src_data_type`**
A `mpi4py.MPI.Datatype` object created using `mpi4py.MPI.Datatype.Create_subarray()` which defines the sub-array of halo elements from which receive update values.

mpi_array.update.MpiHaloSingleExtentUpdate.src_extent

`MpiHaloSingleExtentUpdate.`**`src_extent`**
The locale `CartLocaleExtent` from which the sub-array update is read.

mpi_array.update.MpiHaloSingleExtentUpdate.update_extent

`MpiHaloSingleExtentUpdate.`**`update_extent`**
The `IndexingExtent` indicating the halo sub-array which is to be updated.

mpi_array.update.UpdatesForRedistribute

class mpi_array.update.**UpdatesForRedistribute**(*dst_distrib*, *src_distrib*, *peer_rank_translator=None*)

Bases: object

Collection of update extents for re-distribution of array elements from one distribution to another.

Methods

__init__(dst_distrib, src_distrib[, ...])	
calc_intersection_split(dst_extent, src_extent)	Calculates intersection between *dst_extent* and *{src_extent}*.
check_updates()	Runs consistency checks on the calculated updates, assumes that the dst_distrib and src_distrib distributed as a partitioning (no locale extent overlaps except for halo).
create_pair_extent_update(dst_extent, ...)	Factory method for creating *PairExtentUpdate* objects.
get_cpy2_src_extents(dst_inter_locale_rank)	
initialise()	
initialise_cpy2_updates()	
initialise_rget_updates()	
initialise_updates()	

mpi_array.update.UpdatesForRedistribute.__init__

UpdatesForRedistribute.**__init__**(*dst_distrib*, *src_distrib*, *peer_rank_translator=None*)

mpi_array.update.UpdatesForRedistribute.calc_intersection_split

UpdatesForRedistribute.**calc_intersection_split**(*dst_extent*, *src_extent*)

Calculates intersection between *dst_extent* and *{src_extent}*. Any regions of *dst_extent* which **do not** intersect with *src_extent* are returned as a list of *left-over* type(*dst_extent*) elements. The regions of *dst_extent* which **do** intersect with *src_extent* are returned as a list of *update* *PairExtentUpdate* elements. Returns tuple pair (leftovers, updates)

Parameters

- **dst_extent** (HaloIndexingExtent) – Extent which is to receive update from intersection with *src_extent*.
- **src_extent** (HaloIndexingExtent) – Extent which is to provide update for the intersecting region of *dst_extent*.

Return type tuple

Returns Returns tuple pair of (leftovers, updates).

mpi_array.update.UpdatesForRedistribute.check_updates

UpdatesForRedistribute.**check_updates**()

Runs consistency checks on the calculated updates, assumes that the `dst_distrib` and `src_distrib` distributed as a partitioning (no locale extent overlaps except for halo).

Raises **RuntimeError** – If update inconsistency discovered.

mpi_array.update.UpdatesForRedistribute.create_pair_extent_update

UpdatesForRedistribute.**create_pair_extent_update**(*dst_extent*, *src_extent*, *intersection_extent*)

Factory method for creating *PairExtentUpdate* objects.

Parameters

- **dst_extent** (*mpi_array.distribution.LocaleExtent*) – Destination extent.
- **src_extent** (*mpi_array.distribution.LocaleExtent*) – Source extent.
- **src_extent** – The intersection of *src_extent* and *dst_extent* which defines the region of array elements which are to be transferred from source to destination.

Return type *PairExtentUpdate*

Returns Object Defining the source sub-array and destination sub-array.

mpi_array.update.UpdatesForRedistribute.get_cpy2_src_extents

UpdatesForRedistribute.**get_cpy2_src_extents**(*dst_inter_locale_rank*)

mpi_array.update.UpdatesForRedistribute.initialise

UpdatesForRedistribute.**initialise**()

mpi_array.update.UpdatesForRedistribute.initialise_cpy2_updates

UpdatesForRedistribute.**initialise_cpy2_updates**()

mpi_array.update.UpdatesForRedistribute.initialise_rget_updates

UpdatesForRedistribute.**initialise_rget_updates**()

mpi_array.update.UpdatesForRedistribute.initialise_updates

UpdatesForRedistribute.**initialise_updates**()

mpi_array.update.RmaUpdateExecutor

class mpi_array.update.**RmaUpdateExecutor**(*inter_win*, *dst_lndarray*, *src_inter_win_rank_attr*, *rank_logger=None*, *casting='same_kind'*)

Bases: object

Performs one-sided fetch of data from remote (source) locale arrays to update destination locale array.

Methods

__init__(inter_win, dst_lndarray, ...[, ...])	
do_direct_cpy2_update(updates, src_lndarray)	Does direct copy update to *dst_lndarray* from the specified *src_lndarray* array.
do_locale_rma_update(updates)	Performs RMA to get elements from remote (source) locales to update the (destination) locale extent array.
get_src_win_rank(src_extent)	Returns target rank integer (*inter_win*) for specified *src_extent* extent.

mpi_array.update.RmaUpdateExecutor.__init__

RmaUpdateExecutor.**__init__**(*inter_win*, *dst_lndarray*, *src_inter_win_rank_attr*, *rank_logger=None*, *casting='same_kind'*)

mpi_array.update.RmaUpdateExecutor.do_direct_cpy2_update

RmaUpdateExecutor.**do_direct_cpy2_update**(*updates*, *src_lndarray*)

Does direct copy update to *dst_lndarray* from the specified *src_lndarray* array.

Parameters

- **updates** (sequence of *PairExtentUpdate*) – Sequence of destination and source extents.
- **src_lndarray** (numpy.ndarray) – Elements copied from this array.

mpi_array.update.RmaUpdateExecutor.do_locale_rma_update

RmaUpdateExecutor.**do_locale_rma_update**(*updates*)

Performs RMA to get elements from remote (source) locales to update the (destination) locale extent array.

Parameters **updates** (sequence of *PairExtentUpdate*) – Sequence of destination and source extents.

mpi_array.update.RmaUpdateExecutor.get_src_win_rank

RmaUpdateExecutor.**get_src_win_rank**(*src_extent*)

Returns target rank integer (*inter_win*) for specified *src_extent* extent.

Parameters **src_extent** (*mpi_array.distribution.LocaleExtent*) – Return target rank for this extent.

Return type int

Returns Target rank for window *inter_win*.

Attributes

dst_lndarray	The destination numpy.ndarray for remote fetches.
inter_win	The mpi4py.MPI.Win instance used for remote fetch of data.
random_state	A numpy.random.RandomState instance, used to permute target rank ordering to alleviate swamping single rank with *get* requests from multiple source ranks.
rank_logger	The logging.Logger used for log messages.

mpi_array.update.RmaUpdateExecutor.dst_lndarray

RmaUpdateExecutor.**dst_lndarray**
The destination numpy.ndarray for remote fetches.

mpi_array.update.RmaUpdateExecutor.inter_win

RmaUpdateExecutor.**inter_win**
The mpi4py.MPI.Win instance used for remote fetch of data.

mpi_array.update.RmaUpdateExecutor.random_state

RmaUpdateExecutor.**random_state**
A numpy.random.RandomState instance, used to permute target rank ordering to alleviate swamping single rank with *get* requests from multiple source ranks.

mpi_array.update.RmaUpdateExecutor.rank_logger

RmaUpdateExecutor.**rank_logger**
The logging.Logger used for log messages.

1.38 The mpi_array.update_test Module

Module defining *mpi_array.update* unit-tests. Execute as:

```
python -m mpi_array.update_test
```

1.38.1 Classes

MpiPairExtentUpdateTest([methodName])	Tests for mpi_array.distribution.MpiPairExtentUpdate.

Continued on next page

Table 1.222 – continued from previous page

MpiHaloSingleExtentUpdateTest([methodName])	Tests for mpi_array.distribution.MpiHaloSingleExtentUpdate.
HalosUpdateTest([methodName])	Tests for mpi_array.distribution.HalosUpdate.
UpdatesForRedistributeTest([methodName])	Tests for *mpi_array.update.UpdatesForRedistribute*.

mpi_array.update_test.MpiPairExtentUpdateTest

class mpi_array.update_test.**MpiPairExtentUpdateTest**(*methodName='runTest'*)

Bases: *mpi_array.unittest.TestCase*

Tests for mpi_array.distribution.MpiPairExtentUpdate.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_construct()	Tests for mpi_array.distribution.MpiPairExtentUpdate.__init__().
test_data_type()	Tests for mpi_array.distribution.MpiHaloSingleExtentUpdate.__str__().
test_str()	Tests for mpi_array.distribution.MpiPairExtentUpdate.__str__().

mpi_array.update_test.MpiPairExtentUpdateTest.__init__

MpiPairExtentUpdateTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.update_test.MpiPairExtentUpdateTest.addCleanup

MpiPairExtentUpdateTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.update_test.MpiPairExtentUpdateTest.addTypeEqualityFunc

MpiPairExtentUpdateTest.**addTypeEqualityFunc**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.update_test.MpiPairExtentUpdateTest.assertArraySplitEqual

MpiPairExtentUpdateTest.**assertArraySplitEqual**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.update_test.MpiPairExtentUpdateTest.countTestCases

MpiPairExtentUpdateTest.**countTestCases**()

mpi_array.update_test.MpiPairExtentUpdateTest.debug

MpiPairExtentUpdateTest.**debug**()

Run the test without collecting errors in a TestResult

mpi_array.update_test.MpiPairExtentUpdateTest.defaultTestResult

MpiPairExtentUpdateTest.**defaultTestResult**()

mpi_array.update_test.MpiPairExtentUpdateTest.doCleanups

MpiPairExtentUpdateTest.**doCleanups**()
Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.update_test.MpiPairExtentUpdateTest.id

MpiPairExtentUpdateTest.**id**()

mpi_array.update_test.MpiPairExtentUpdateTest.run

MpiPairExtentUpdateTest.**run**(*result=None*)

mpi_array.update_test.MpiPairExtentUpdateTest.setUp

MpiPairExtentUpdateTest.**setUp**()

mpi_array.update_test.MpiPairExtentUpdateTest.setUpClass

MpiPairExtentUpdateTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.update_test.MpiPairExtentUpdateTest.shortDescription

MpiPairExtentUpdateTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.update_test.MpiPairExtentUpdateTest.skipTest

MpiPairExtentUpdateTest.**skipTest**(*reason*)
Skip this test.

mpi_array.update_test.MpiPairExtentUpdateTest.subTest

MpiPairExtentUpdateTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.update_test.MpiPairExtentUpdateTest.tearDown

MpiPairExtentUpdateTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.update_test.MpiPairExtentUpdateTest.tearDownClass

MpiPairExtentUpdateTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.update_test.MpiPairExtentUpdateTest.test_construct

MpiPairExtentUpdateTest.**test_construct**()
Tests for mpi_array.distribution.MpiPairExtentUpdate.__init__().

mpi_array.update_test.MpiPairExtentUpdateTest.test_data_type

MpiPairExtentUpdateTest.**test_data_type**()
Tests for mpi_array.distribution.MpiHaloSingleExtentUpdate.__str__().

mpi_array.update_test.MpiPairExtentUpdateTest.test_str

MpiPairExtentUpdateTest.**test_str**()
Tests for mpi_array.distribution.MpiPairExtentUpdate.__str__().

Attributes

longMessage
maxDiff

mpi_array.update_test.MpiPairExtentUpdateTest.longMessage

MpiPairExtentUpdateTest.**longMessage** = True

mpi_array.update_test.MpiPairExtentUpdateTest.maxDiff

MpiPairExtentUpdateTest.**maxDiff** = 640

mpi_array.update_test.MpiHaloSingleExtentUpdateTest

class mpi_array.update_test.**MpiHaloSingleExtentUpdateTest**(*methodName='runTest'*)
Bases: *mpi_array.unittest.TestCase*

Tests for mpi_array.distribution.MpiHaloSingleExtentUpdate.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_construct()	Tests for `mpi_array.distribution.MpiHaloSingleExtentUpdate.__init__()`.
test_data_type()	Tests for `mpi_array.distribution.MpiHaloSingleExtentUpdate.__str__()`.
test_str()	Tests for `mpi_array.distribution.MpiHaloSingleExtentUpdate.__str__()`.

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.__init__

MpiHaloSingleExtentUpdateTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.addCleanup

MpiHaloSingleExtentUpdateTest.**addCleanup**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.addTypeEqualityFunc

MpiHaloSingleExtentUpdateTest.**addTypeEqualityFunc**(*typeobj*, *function*)
Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.assertArraySplitEqual

MpiHaloSingleExtentUpdateTest.**assertArraySplitEqual**(*splt1*, *splt2*)
Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.countTestCases

MpiHaloSingleExtentUpdateTest.**countTestCases**()

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.debug

MpiHaloSingleExtentUpdateTest.**debug**()
Run the test without collecting errors in a TestResult

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.defaultTestResult

MpiHaloSingleExtentUpdateTest.**defaultTestResult**()

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.doCleanups

MpiHaloSingleExtentUpdateTest.**doCleanups**()
Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.id

MpiHaloSingleExtentUpdateTest.**id**()

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.run

MpiHaloSingleExtentUpdateTest.**run**(*result=None*)

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.setUp

MpiHaloSingleExtentUpdateTest.**setUp**()

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.setUpClass

MpiHaloSingleExtentUpdateTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.shortDescription

MpiHaloSingleExtentUpdateTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.skipTest

MpiHaloSingleExtentUpdateTest.**skipTest**(*reason*)
Skip this test.

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.subTest

MpiHaloSingleExtentUpdateTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.tearDown

MpiHaloSingleExtentUpdateTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.tearDownClass

MpiHaloSingleExtentUpdateTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.test_construct

MpiHaloSingleExtentUpdateTest.**test_construct**()
Tests for `mpi_array.distribution.MpiHaloSingleExtentUpdate.__init__()`.

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.test_data_type

MpiHaloSingleExtentUpdateTest.**test_data_type**()
Tests for mpi_array.distribution.MpiHaloSingleExtentUpdate.__str__().

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.test_str

MpiHaloSingleExtentUpdateTest.**test_str**()
Tests for mpi_array.distribution.MpiHaloSingleExtentUpdate.__str__().

Attributes

longMessage
maxDiff

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.longMessage

MpiHaloSingleExtentUpdateTest.**longMessage** = True

mpi_array.update_test.MpiHaloSingleExtentUpdateTest.maxDiff

MpiHaloSingleExtentUpdateTest.**maxDiff** = 640

mpi_array.update_test.HalosUpdateTest

class mpi_array.update_test.**HalosUpdateTest**(*methodName='runTest'*)
Bases: *mpi_array.unittest.TestCase*

Tests for mpi_array.distribution.HalosUpdate.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares list of numpy.ndarray results returned by numpy.mpi_array() and mpi_array.split.mpi_array() functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	

Continued on next page

Table 1.227 – continued from previous page

run([result])	
setUp()	
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_construct()	Tests for `mpi_array.distribution.HalosUpdate.__init__()`.

mpi_array.update_test.HalosUpdateTest.__init__

`HalosUpdateTest.`**`__init__`**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.update_test.HalosUpdateTest.addCleanup

`HalosUpdateTest.`**`addCleanup`**(*function*, **args*, ***kwargs*)

Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.update_test.HalosUpdateTest.addTypeEqualityFunc

`HalosUpdateTest.`**`addTypeEqualityFunc`**(*typeobj*, *function*)

Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.update_test.HalosUpdateTest.assertArraySplitEqual

`HalosUpdateTest.`**`assertArraySplitEqual`**(*splt1*, *splt2*)

Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (list of numpy.ndarray) – First object in equality comparison.
- **splt2** (list of numpy.ndarray) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of splt2.

mpi_array.update_test.HalosUpdateTest.countTestCases

HalosUpdateTest.**countTestCases**()

mpi_array.update_test.HalosUpdateTest.debug

HalosUpdateTest.**debug**()
Run the test without collecting errors in a TestResult

mpi_array.update_test.HalosUpdateTest.defaultTestResult

HalosUpdateTest.**defaultTestResult**()

mpi_array.update_test.HalosUpdateTest.doCleanups

HalosUpdateTest.**doCleanups**()
Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.update_test.HalosUpdateTest.id

HalosUpdateTest.**id**()

mpi_array.update_test.HalosUpdateTest.run

HalosUpdateTest.**run**(*result=None*)

mpi_array.update_test.HalosUpdateTest.setUp

HalosUpdateTest.**setUp**()

mpi_array.update_test.HalosUpdateTest.setUpClass

HalosUpdateTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.update_test.HalosUpdateTest.shortDescription

HalosUpdateTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.update_test.HalosUpdateTest.skipTest

HalosUpdateTest.**skipTest**(*reason*)
Skip this test.

mpi_array.update_test.HalosUpdateTest.subTest

HalosUpdateTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.update_test.HalosUpdateTest.tearDown

HalosUpdateTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.update_test.HalosUpdateTest.tearDownClass

HalosUpdateTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.update_test.HalosUpdateTest.test_construct

HalosUpdateTest.**test_construct**()
Tests for `mpi_array.distribution.HalosUpdate.__init__()`.

Attributes

longMessage
maxDiff

mpi_array.update_test.HalosUpdateTest.longMessage

HalosUpdateTest.**longMessage** = True

mpi_array.update_test.HalosUpdateTest.maxDiff

HalosUpdateTest.**maxDiff** = 640

mpi_array.update_test.UpdatesForRedistributeTest

class mpi_array.update_test.**UpdatesForRedistributeTest**(*methodName='runTest'*)

Bases: *mpi_array.unittest.TestCase*

Tests for *mpi_array.update.UpdatesForRedistribute*.

Methods

__init__([methodName])	Create an instance of the class that will use the named test method when executed.
addCleanup(function, *args, **kwargs)	Add a function, with arguments, to be called when the test is completed.
addTypeEqualityFunc(typeobj, function)	Add a type specific assertEqual style function to compare a type.
assertArraySplitEqual(splt1, splt2)	Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.
countTestCases()	
debug()	Run the test without collecting errors in a TestResult
defaultTestResult()	
doCleanups()	Execute all cleanup functions.
id()	
run([result])	
setUp()	Sets `self.rank_logger` attribute.
setUpClass()	Hook method for setting up class fixture before running tests in the class.
shortDescription()	Returns a one-line description of the test, or None if no description has been provided.
skipTest(reason)	Skip this test.
subTest([msg])	Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters.
tearDown()	Hook method for deconstructing the test fixture after testing it.
tearDownClass()	Hook method for deconstructing the class fixture after running all tests in the class.
test_slab_to_block()	Tests *mpi_array.update.UpdatesForRedistribute* by calling the *mpi_array.update.UpdatesForRedistribute.check_updates* method.

mpi_array.update_test.UpdatesForRedistributeTest.__init__

UpdatesForRedistributeTest.**__init__**(*methodName='runTest'*)

Create an instance of the class that will use the named test method when executed. Raises a ValueError if the instance does not have a method with the specified name.

mpi_array.update_test.UpdatesForRedistributeTest.addCleanup

UpdatesForRedistributeTest.**addCleanup**(*function*, **args*, ***kwargs*)
Add a function, with arguments, to be called when the test is completed. Functions added are called on a LIFO basis and are called after tearDown on test failure or success.

Cleanup items are called even if setUp fails (unlike tearDown).

mpi_array.update_test.UpdatesForRedistributeTest.addTypeEqualityFunc

UpdatesForRedistributeTest.**addTypeEqualityFunc**(*typeobj*, *function*)
Add a type specific assertEqual style function to compare a type.

This method is for use by TestCase subclasses that need to register their own type equality functions to provide nicer error messages.

Args:

typeobj: The data type to call this function on when both values are of the same type in assertEqual().

function: The callable taking two arguments and an optional msg= argument that raises self.failureException with a useful error message when the two arguments are not equal.

mpi_array.update_test.UpdatesForRedistributeTest.assertArraySplitEqual

UpdatesForRedistributeTest.**assertArraySplitEqual**(*splt1*, *splt2*)
Compares `list` of `numpy.ndarray` results returned by `numpy.mpi_array()` and `mpi_array.split.mpi_array()` functions.

Parameters

- **splt1** (`list` of `numpy.ndarray`) – First object in equality comparison.
- **splt2** (`list` of `numpy.ndarray`) – Second object in equality comparison.

Raises **unittest.AssertionError** – If any element of *splt1* is not equal to the corresponding element of `splt2`.

mpi_array.update_test.UpdatesForRedistributeTest.countTestCases

UpdatesForRedistributeTest.**countTestCases**()

mpi_array.update_test.UpdatesForRedistributeTest.debug

UpdatesForRedistributeTest.**debug**()
Run the test without collecting errors in a TestResult

mpi_array.update_test.UpdatesForRedistributeTest.defaultTestResult

UpdatesForRedistributeTest.**defaultTestResult**()

mpi_array.update_test.UpdatesForRedistributeTest.doCleanups

UpdatesForRedistributeTest.**doCleanups**()
Execute all cleanup functions. Normally called for you after tearDown.

mpi_array.update_test.UpdatesForRedistributeTest.id

UpdatesForRedistributeTest.**id**()

mpi_array.update_test.UpdatesForRedistributeTest.run

UpdatesForRedistributeTest.**run**(*result=None*)

mpi_array.update_test.UpdatesForRedistributeTest.setUp

UpdatesForRedistributeTest.**setUp**()
Sets `self.rank_logger` attribute.

mpi_array.update_test.UpdatesForRedistributeTest.setUpClass

UpdatesForRedistributeTest.**setUpClass**()
Hook method for setting up class fixture before running tests in the class.

mpi_array.update_test.UpdatesForRedistributeTest.shortDescription

UpdatesForRedistributeTest.**shortDescription**()
Returns a one-line description of the test, or None if no description has been provided.

The default implementation of this method returns the first line of the specified test method's docstring.

mpi_array.update_test.UpdatesForRedistributeTest.skipTest

UpdatesForRedistributeTest.**skipTest**(*reason*)
Skip this test.

mpi_array.update_test.UpdatesForRedistributeTest.subTest

UpdatesForRedistributeTest.**subTest**(*msg=None*, ***params*)
Return a context manager that will return the enclosed block of code in a subtest identified by the optional message and keyword parameters. A failure in the subtest marks the test case as failed but resumes execution at the end of the enclosed block, allowing further test code to be executed.

mpi_array.update_test.UpdatesForRedistributeTest.tearDown

UpdatesForRedistributeTest.**tearDown**()
Hook method for deconstructing the test fixture after testing it.

mpi_array.update_test.UpdatesForRedistributeTest.tearDownClass

UpdatesForRedistributeTest.**tearDownClass**()
Hook method for deconstructing the class fixture after running all tests in the class.

mpi_array.update_test.UpdatesForRedistributeTest.test_slab_to_block

UpdatesForRedistributeTest.**test_slab_to_block**()
Tests *mpi_array.update.UpdatesForRedistribute* by calling the *mpi_array.update.UpdatesForRedistribute.check_updates* method.

Attributes

longMessage
maxDiff

mpi_array.update_test.UpdatesForRedistributeTest.longMessage

UpdatesForRedistributeTest.**longMessage** = True

mpi_array.update_test.UpdatesForRedistributeTest.maxDiff

UpdatesForRedistributeTest.**maxDiff** = 640

Python Module Index

m

Index

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

Z