

---

# **TraceR Documentation**

**Nikhil Jain, Bilge Acun, Abhinav Bhatele**

**Jul 19, 2019**



**CONTENTS:**

<b>1</b>	<b>Download and Install</b>	<b>3</b>
1.1	Dependencies . . . . .	3
1.2	Build . . . . .	3
<b>2</b>	<b>User Guide</b>	<b>5</b>
2.1	Quickstart . . . . .	5
2.2	Creating a TraceR configuration file . . . . .	5
2.3	Creating the network (CODES) configuration file . . . . .	6
2.4	Creating the job placement file . . . . .	7
2.5	Generating Traces . . . . .	7
<b>3</b>	<b>Tutorial</b>	<b>11</b>
<b>4</b>	<b>Source Code Documentation</b>	<b>13</b>
4.1	Class Hierarchy . . . . .	13
4.2	File Hierarchy . . . . .	13
4.3	Full API . . . . .	13
<b>5</b>	<b>Indices and tables</b>	<b>47</b>
	<b>Index</b>	<b>49</b>



TraceR is a trace replay tool built upon the ROSS-based CODES simulation framework. TraceR can be used for predicting network performance and understanding network behavior by simulating messaging in High Performance Computing applications on interconnection networks.



## DOWNLOAD AND INSTALL

TraceR can be downloaded from [GitHub](#).

### 1.1 Dependencies

TraceR depends on [CODES](#) and [ROSS](#).

### 1.2 Build

There are several ways to build TraceR.

1. Use [spack](#) to build TraceR and its dependencies:

```
spack install tracer
```

2. Build TraceR and its dependencies manually:

- Download and install ROSS and CODES. Set the appropriate paths: ROSS\_DIR, and CODES\_DIR in tracer/Makefile.common.
- Pick between the two trace formats supported by TraceR: OTF2 or BigSim, and accordingly build the OTF2 or Charm++ library. If using OTF2 traces (default), set SELECT\_TRACE = -DTRACER\_OTF\_TRACES=1, and ensure that otf2-config is in your PATH. If using BigSim traces, set SELECT\_TRACE = -DTRACER\_BIGSIM\_TRACES=1, and set CHARMPATH to the Charm++ installation in tracer/Makefile.common.
- Set the ARCH variable in tracer/Makefile.common or alternatively set the CXX and ARCH\_FLAGS variables. Then type:

```
cd tracer  
make
```

#### 1.2.1 Trace Formats

TraceR supports two different trace formats as input. For each format, you need to build additional software as explained below.

1. Score-P's OTF2 format (default): To use OTF2 traces, you need to download and build the [OTF2](#) library.
2. AMPI-based BigSim format: To use BigSim traces as input to TraceR, you need to download and build [Charm++](#).

The instructions to build Charm++ are in the [Charm++ manual](#). You should use the “charm++” target and pass “bigemulator” as a build option.



## USER GUIDE

Below, we provide detailed instructions for how to start doing network simulations using TraceR.

### 2.1 Quickstart

This is a basic `mpirun` command to launch a TraceR simulation in the optimistic mode:

```
mpirun -np <p> ../traceR --sync=3 -- <network_config> <tracer_config>
```

Some useful options to use with TraceR:

<b>--sync</b>	ROSS's PDES type. 1 - sequential, 2 - conservative, 3 - optimistic
<b>--nkp</b>	number of groups used for clustering LPs; recommended value for lower roll-backs: (total #LPs)/(#MPI processes)
<b>--extramem</b>	number of messages in ROSS's extra message buffer (each message is ~500 bytes, 100K should work for most cases)
<b>--max-opt-lookahead</b>	leash on optimistic execution in nanoseconds (1 microsecond is a good value)
<b>--timer-frequency</b>	frequency with which PEO should print current virtual time

### 2.2 Creating a TraceR configuration file

This is the format for the TraceR config file:

```
<global map file>
<num jobs>
<Trace path for job0> <map file for job0> <number of ranks in job0> <iterations (use_
↪1 if running in normal mode)>
<Trace path for job1> <map file for job1> <number of ranks in job1> <iterations (use_
↪1 if running in normal mode)>
...
<Trace path for jobN> <map file for jobN> <number of ranks in jobN> <iterations (use_
↪1 if running in normal mode)>
```

If you do not intend to create global or per-job map files, you can use NA instead of them.

Sample TraceR config files can be found in `examples/jacobi2d-bigsim/tracer_config` (BigSim) or `examples/stencil4d-otf/tracer_config` (OTF)

See [Creating the job placement file](#) below for how to generate global or per-job map files.

## 2.3 Creating the network (CODES) configuration file

Sample network configuration files can be found in `examples/conf`

Additional documentation on the format of the CODES config file can be found in the CODES wiki at <https://xgitlab.cels.anl.gov/codes/codes/wikis/home>

A brief summary of the format follows.

LPGROUPS, MODELNET\_GRP, PARAMS are keywords and should be used as is.

MODELNET\_GRP:

```
repetition = number of routers that have nodes connecting to them.

server = number of MPI processes/cores per router

modelnet_* = number of NICs. For torus, this value has to be 1; for dragonfly,
it should be router radix divided by 4; for the fat-tree, it should be router
radix divided by 2. For the dragonfly network, modelnet_dragonfly_router should
also be specified (as 1). For express mesh, modelnet_express_mesh_router should
also be specified as 1.

Similarly, the fat-tree config file requires specifying fattree_switch which
can be 2 or 3, depending on the number of levels in the fat-tree. Note that the
total number of cores specified in the CODES config file can be greater than
the number of MPI processes being simulated (specified in the tracer config
file).
```

Other common parameters:

```
packet_size/chunk_size (both should have the same value) = size of the packets
created by NIC for transmission on the network. Smaller the packet size, longer
the time for which simulation will run (in real time). Larger the packet size,
the less accurate the predictions are expected to be (in virtual time). Packet
sizes of 512 bytes to 4096 bytes are commonly used.

modelnet_order = torus/dragonfly/fattree/slimfly/express_mesh

modelnet_scheduler =
    fcfs: packetize messages one by one.
    round-robin: packetize message in a round robin manner.

message_size = PDES parameter (keep constant at 512)

router_delay = delay at each router for packet transmission (in nanoseconds)

soft_delay = delay caused by software stack such as that of MPI (in nanoseconds)

link_bandwidth = bandwidth of each link in the system (in GB/s)

cn_bandwidth = bandwidth of connection between NIC and router (in GB/s)

buffer_size/vc_size = size of channels used to store transient packets at routers (in
bytes). Typical value is 64*packet_size.

routing = how are packets being routed. Options depend on the network.
    torus: static/adaptive
```

(continues on next page)

(continued from previous page)

```
dragonfly: minimal/nonminimal/adaptive
fat-tree: adaptive/static
```

Network specific parameters:

```
Torus:
  n_dims = number of dimensions in the torus
  dim_length = length of each dimension

Dragonfly:
  num_routers = number of routers within a group.
  global_bandwidth = bandwidth of the links that connect groups.

Fat-tree:
  ft_type = always choose 1
  num_levels = number of levels in the fat-tree (2 or 3)
  switch_radix = radix of the switch being used
  switch_count = number of switches at leaf level.
```

## 2.4 Creating the job placement file

See the README in utils for instructions on using the tools to generate the global and job mapping files.

## 2.5 Generating Traces

### 2.5.1 Score-P

#### Installation of Score-P

1. Download from <http://www.vi-hps.org/projects/score-p/>
2. `tar -xvzf scorep-3.0.tar.gz`
3. `cd scorep-3.0`
4. `CC=mpicc CFLAGS="-O2" CXX=mpicxx CXXFLAGS="-O2" FC=mpif77 ./configure --without-gui --prefix=<SCOREP_INSTALL>`
5. `make`
6. `make install`

#### Generating OTF2 traces with an MPI program using Score-P

Detailed instructions are available at <https://silc.zih.tu-dresden.de/scorep-current/pdf/scorep.pdf>.

1. Add \$SCOREP\_INSTALL/bin to your PATH for convenience. Example:

```
export SCOREP_INSTALL=$HOME/workspace/scoreP/scorep-3.0/install
export PATH=$SCOREP_INSTALL/bin:$PATH
```

2. Add the following compile time flags to the application:

```
-I$SCOREP_INSTALL/include -I$SCOREP_INSTALL/include/scorep -DSCOREP_USER_ENABLE
```

3. Add `#include <scorep/SCOREP_User.h>` to all files where you plan to add any of the following Score-P calls (optional step):

```
SCOREP_RECORDING_OFF(); - stop recording
SCOREP_RECORDING_ON(); - start recording
```

Marking special regions: `SCOREP_USER_REGION_BY_NAME_BEGIN(regionname, SCOREP_USER_REGION_TYPE_COMMON)` and `SCOREP_USER_REGION_BY_NAME_END(regionname)`.

Region names beginning with `TRACER_WallTime_` are special: using `TRACER_WallTime_<any_name>` prints current time during simulation with tag `<any_name>`.

An example using these features is given below:

```
#include <scorep/SCOREP_User.h>
...
int main(int argc, char **argv, char **envp)
{
    MPI_Init(&argc,&argv);
    SCOREP_RECORDING_OFF(); //turn recording off for initialization/regions_
    ↪not of interest
    ...
    SCOREP_RECORDING_ON();
    //use verbatim to facilitate looping over the traces in simulation when_
    ↪simulating multiple jobs
    SCOREP_USER_REGION_BY_NAME_BEGIN("TRACER_Loop", SCOREP_USER_REGION_TYPE_
    ↪COMMON);
    // at least add this BEGIN timer call - called from only one rank
    // you can add more calls later with region names TRACER_WallTime_<any_
    ↪string of your choice>
    if(myRank == 0)
        SCOREP_USER_REGION_BY_NAME_BEGIN("TRACER_WallTime_MainLoop", SCOREP_USER_
    ↪REGION_TYPE_COMMON);
    // Application main work LOOP
    for ( int itscf = 0; itscf < nitscf; itscf++ )
    {
        ...
    }
    // time call to mark END of work - called from only one rank
    if(myRank == 0)
        SCOREP_USER_REGION_BY_NAME_END("TRACER_WallTime_MainLoop");
    // use verbatim - mark end of trace loop
    SCOREP_USER_REGION_BY_NAME_END("TRACER_Loop");
    SCOREP_RECORDING_OFF(); //turn off recording again
    ...
}
```

4. For the link step, prefix the linker line with the following:

```
LD = scorep --user --nocompiler --noopenmp --nopomp --nocuda --noopenacc --
    ↪noopencl --nomemory <your_linker>
```

5. For running, set:

```
export SCOREP_ENABLE_TRACING=1
export SCOREP_ENABLE_PROFILING=0
```

(continues on next page)

(continued from previous page)

```
export SCOREP_REDUCE_PROBE_TEST=1
export SCOREP_MPI_ENABLE_GROUPS=ENV, P2P, COLL, XNONBLOCK
```

If Score-P prints a warning about flushing traces during the run, you may avoid them using:

```
export SCOREP_TOTAL_MEMORY=256M
export SCOREP_EXPERIMENT_DIRECTORY=/p/lscratchd/<username>/...
```

6. Run the binary and traces should be generated in a folder named scorep-.\*.

## 2.5.2 BigSim

### Installation of BigSim

Compile BigSim/Charm++ for emulation (see <http://charm.cs.illinois.edu/manuals/html/bigsim/manual-1p.html> for more detail). Use any one of the following commands:

- To use UDP as BigSim/Charm++'s communication layer:

```
./build bgampi net-linux-x86_64 bigemulator --with-production --enable-tracing
./build bgampi net-darwin-x86_64 bigemulator --with-production --enable-tracing
```

Or explicitly provide the compiler optimization level:

```
./build bgampi net-linux-x86_64 bigemulator -O2
```

- To use MPI as BigSim/Charm++'s communication layer:

```
./build bgampi mpi-linux-x86_64 bigemulator --with-production --enable-tracing
```

**Note:** This build is used to compile MPI applications so that traces can be generated. Hence, the communication layer used by BigSim/Charm++ is not important. During simulation, the communication will be replayed using the network simulator from CODES. However, the computation time captured here can be important if it is not being explicitly replaced at simulation time using configuration options. So using appropriate compiler flags is important.

### Generating AMPI traces with an MPI program using BigSim

1. Compile your MPI application using BigSim/Charm++.

Example commands:

```
$CHARM_DIR/bin/ampicc -O2 simplePrg.c -o simplePrg_c
$CHARM_DIR/bin/ampiCC -O2 simplePrg.cc -o simplePrg_cxx
```

2. Emulation to generate traces. When the binary generated is run, BigSim/Charm++ runs the program on the allocated cores as if it were running as usual. Users should provide a few additional arguments to specify the number of MPI processes in the prototype systems.

If using UDP as the BigSim/Charm++'s communication layer:

```
./charmrun +p<number of real processes> ++nodelist <machine file> ./pgm
↪<program arguments> +vp<number of processes expected on the future system>
↪+x<x dim> +y<y dim> +z<z dim> +bglog
```

If using MPI as the BigSim/Charm++'s communication layer:

```
mpirun -n<number of real processes> ./pgm <program arguments> +vp<number of  
↪processes expected on the future system> +x<x dim> +y<y dim> +z<z dim> ↪  
↪+bglog
```

Number of real processes is typically equal to the number cores the emulation is being run on.

*machine file* is the list of systems the emulation should be run on (similar to machine file for MPI; refer to Charm++ website for more details).

*vp* is the number of MPI ranks that are to be emulated. For simple tests, it can be the same as the number of real processes, in which case one MPI rank is run on each real process (as it happens when a regular program is run). When the number of *vp* (virtual processes) is higher, BigSim launches user level threads to execute multiple MPI ranks within a process.

*+x +y +z* defines a 3D grid of the virtual processes. The product of these three dimensions must match the number of *vp*'s. These arguments do not have any effect on the emulation, but exist due to historical reasons.

*+bglog* instructs bigsim to write the logs to files.

3. When this run is finished, you should see many files named *bgTrace\** in the directory. The total number of such files equals the number of real processes plus one. Their names are *bgTrace*, *bgTrace0*, *bgTrace1*, and so on. Create a new folder and move all *bgTrace* files to that folder.

---

CHAPTER  
**THREE**

---

**TUTORIAL**





## SOURCE CODE DOCUMENTATION

### 4.1 Class Hierarchy

### 4.2 File Hierarchy

### 4.3 Full API

#### 4.3.1 Classes and Structs

##### Struct Coll\_lookup

- Defined in file\_tracer\_tracer-driver.h

##### Struct Documentation

**struct Coll\_lookup**

###### Public Members

*proc\_event* **remote\_event**

*proc\_event* **local\_event**

##### Struct CoreInf

- Defined in file\_tracer\_tracer-driver.h

##### Struct Documentation

**struct CoreInf**

###### Public Members

int **mapsTo**

int **jobID**

### Struct JobInf

- Defined in file\_tracer\_reader\_datatypes.h

### Struct Documentation

**struct JobInf**

#### Public Members

int **numRanks**  
char **traceDir**[256]  
char **map\_file**[256]  
int \***rankMap**  
int \***offsets**  
int **skipMsgId**  
int **numIters**

### Struct MsgEntry

- Defined in file\_tracer\_elements\_MsgEntry.h

### Struct Documentation

**struct MsgEntry**

#### Public Members

int **node**  
int **thread**  
*MsgID* **msgId**

### Struct MsgID

- Defined in file\_tracer\_elements\_MsgEntry.h

### Struct Documentation

**struct MsgID**

### Public Members

int **pe**  
int **id**  
uint64\_t **size**

### Struct `proc_msg`

- Defined in `file_tracer_tracer-driver.h`

### Struct Documentation

`struct proc_msg`

#### Public Members

*proc\_event* **proc\_event\_type**  
tw\_lpid **src**  
int **iteration**  
*TaskPair* **executed**  
int **fwd\_dep\_count**  
int **saved\_task**  
*MsgID* **msgId**  
bool **incremented\_flag**  
int **model\_net\_calls**  
unsigned int **coll\_info**  
unsigned int **coll\_info\_2**

### Struct `proc_state`

- Defined in `file_tracer_tracer-driver.h`

### Struct Documentation

`struct proc_state`

#### Public Members

tw\_stime **start\_ts**  
tw\_stime **end\_ts**  
*PE* \***my\_pe**  
clock\_t **sim\_start**

```
int my_pe_num
int my_job
```

### Struct TaskPair

- Defined in file\_tracer\_reader\_datatypes.h

### Struct Documentation

```
struct TaskPair
```

#### Public Members

```
int iter
int taskid
```

### Class CollMsgKey

- Defined in file\_tracer\_elements\_PE.h

### Class Documentation

```
class CollMsgKey
```

#### Public Functions

```
CollMsgKey (uint32_t _rank, uint32_t _comm, int64_t _seq)
bool operator< (const CollMsgKey &rhs) const
~CollMsgKey ()
```

#### Public Members

```
uint32_t rank
uint32_t comm
int64_t seq
```

### Class MsgKey

- Defined in file\_tracer\_elements\_PE.h

### Class Documentation

```
class MsgKey
```

## Public Functions

**MsgKey** (uint32\_t \_rank, uint32\_t \_tag, uint32\_t \_comm, int64\_t \_seq)

bool **operator<** (const *MsgKey* &*rhs*) const

**~MsgKey** ()

## Public Members

uint32\_t **rank**

uint32\_t **comm**

uint32\_t **tag**

int64\_t **seq**

## Class PE

- Defined in file\_tracer\_elements\_PE.h

## Class Documentation

**class PE**

## Public Functions

**PE** ()

**~PE** ()

void **goToNextIter** (int *iter*)

bool **noUnsatDep** (int *iter*, int *tInd*)

void **mark\_all\_done** (int *iter*, int *tInd*)

double **taskExecTime** (int *tInd*)

void **printStats** (int *iter*)

void **invertMsgPe** (int *iter*, int *tInd*)

double **getTaskExecTime** (int *tInd*)

void **addTaskExecTime** (int *tInd*, double *time*)

int **findTaskFromMsg** (*MsgID* \**msg*)

## Public Members

```
std::list<TaskPair> msgBuffer
Task *myTasks
bool **taskStatus
bool **taskExecuted
bool **msgStatus
bool *allMarked
double currTime
bool busy
int beforeTask
int totalTasksCount
int myNum
int myEmPE
int jobNum
int tasksCount
int currentTask
int firstTask
int currIter
int loop_start_task
std::map<int, int> *msgDestLogs
int numWth
int numEmpes
KeyType pendingMsgs
KeyType pendingRMsgs
int64_t *sendSeq
int64_t *recvSeq
std::map<int, int> pendingReqs
std::map<int, int64_t> pendingRReqs
std::vector<int64_t> collectiveSeq
std::map<int64_t, std::map<int64_t, std::map<int, int>>> pendingCollMsgs
CollKeyType pendingRCollMsgs
int64_t currentCollComm
int64_t currentCollSeq
int64_t currentCollTask
int64_t currentCollMsgSize
int currentCollRank
```

```
int currentCollPartner  
int currentCollSize  
int currentCollSendCount  
int currentCollRecvCount
```

## Class Task

- Defined in file\_tracer\_elements\_Task.h

## Class Documentation

```
class Task
```

### Public Functions

```
Task ()  
~Task ()
```

### Public Members

```
bool endEvent  
bool loopEvent  
bool loopStartEvent  
double execTime
```

## Class TraceReader

- Defined in file\_tracer\_reader\_TraceReader.h

## Class Documentation

```
class TraceReader
```

### Public Functions

```
TraceReader (char *)  
~TraceReader ()
```

### Public Members

```
int numEmPes
int totalWorkerProcs
int totalNodes
int numWth
int *allNodeOffsets
char tracePath[256]
int fileLoc
int firstLog
int totalTlineLength
```

## 4.3.2 Enums

### Enum `proc_event`

- Defined in file\_tracer\_tracer-driver.h

### Enum Documentation

`enum proc_event`

*Values:*

```
KICKOFF = 1
LOCAL
RECV_MSG
BCAST
EXEC_COMPLETE
SEND_COMP
RECV_POST
COLL_BCAST
COLL_REDUCTION
COLL_A2A
COLL_A2A_SEND_DONE
COLL_ALLGATHER
COLL_ALLGATHER_SEND_DONE
COLL_BRUCK
COLL_BRUCK_SEND_DONE
COLL_A2A_BLOCKED
COLL_A2A_BLOCKED_SEND_DONE
```



```
COLL_SCATTER_SMALL
COLL_SCATTER
COLL_SCATTER_SEND_DONE
RECV_COLL_POST
COLL_COMPLETE
```

### Enum tracer\_coll\_type

- Defined in file\_tracer\_tracer-driver.h

### Enum Documentation

**enum tracer\_coll\_type**

*Values:*

```
TRACER_COLLECTIVE_BCAST = 1
TRACER_COLLECTIVE_REDUCE
TRACER_COLLECTIVE_BARRIER
TRACER_COLLECTIVE_ALLTOALL_LARGE
TRACER_COLLECTIVE_ALLTOALL_BLOCKED
TRACER_COLLECTIVE_ALL_BRUCK
TRACER_COLLECTIVE_ALLGATHER_LARGE
TRACER_COLLECTIVE_SCATTER_SMALL
TRACER_COLLECTIVE_SCATTER
```

## 4.3.3 Functions

### Function addEventSub

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

void **addEventSub** (int *job*, char \**key*, double *val*, int *numjobs*)

### Function addMsgSizeSub

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

void **addMsgSizeSub** (int *job*, int64\_t *key*, int64\_t *val*, int *numjobs*)

### Function `bcast_msg`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

int **bcast\_msg** (*proc\_state* \*ns, int size, int iter, *MsgID* \*msgId, tw\_stime timeOffset, tw\_stime copyTime, tw\_lp \*lp, *proc\_msg* \*m)

### Function `delegate_send_msg`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **delegate\_send\_msg** (*proc\_state* \*ns, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, *Task* \*t, int taskId, tw\_stime delay)

### Function `enqueue_msg`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **enqueue\_msg** (*proc\_state* \*ns, int size, int iter, *MsgID* \*msgId, int64\_t seq, int dest\_id, tw\_stime sendOffset, *enum proc\_event* evt\_type, *proc\_msg* \*m\_local, tw\_lp \*lp)

### Function `exec_comp`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

int **exec\_comp** (*proc\_state* \*ns, int iter, int task\_id, int comm\_id, tw\_stime sendOffset, int recv, tw\_lp \*lp)

### Function `exec_task`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

tw\_stime **exec\_task** (*proc\_state* \*ns, *TaskPair* task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b)

### Function `exec_task_rev`

- Defined in `file_tracer_tracer-driver.h`

## Function Documentation

void **exec\_task\_rev** (*proc\_state* \*ns, TaskPair task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b)

## Function handle\_a2a\_blocked\_send\_comp\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_a2a\_blocked\_send\_comp\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

## Function handle\_a2a\_blocked\_send\_comp\_rev\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_a2a\_blocked\_send\_comp\_rev\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

## Function handle\_a2a\_send\_comp\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_a2a\_send\_comp\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

## Function handle\_a2a\_send\_comp\_rev\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_a2a\_send\_comp\_rev\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

## Function handle\_allgather\_send\_comp\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_allgather\_send\_comp\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `handle_allgather_send_comp_rev_event`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void `handle_allgather_send_comp_rev_event` (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `handle_bcast_event`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void `handle_bcast_event` (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `handle_bcast_rev_event`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void `handle_bcast_rev_event` (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `handle_bruck_send_comp_event`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void `handle_bruck_send_comp_event` (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `handle_bruck_send_comp_rev_event`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void `handle_bruck_send_comp_rev_event` (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `handle_coll_complete_event`

- Defined in `file_tracer_tracer-driver.h`

## Function Documentation

void **handle\_coll\_complete\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function handle\_coll\_complete\_rev\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_coll\_complete\_rev\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function handle\_coll\_recv\_post\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_coll\_recv\_post\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function handle\_coll\_recv\_post\_rev\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_coll\_recv\_post\_rev\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function handle\_exec\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_exec\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function handle\_exec\_rev\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_exec\_rev\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `handle_kickoff_event`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **handle\_kickoff\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `handle_kickoff_rev_event`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **handle\_kickoff\_rev\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `handle_local_event`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **handle\_local\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `handle_local_rev_event`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **handle\_local\_rev\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `handle_recv_event`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **handle\_recv\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `handle_recv_post_event`

- Defined in `file_tracer_tracer-driver.h`

## Function Documentation

void **handle\_recv\_post\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

## Function handle\_recv\_post\_rev\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_recv\_post\_rev\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

## Function handle\_recv\_rev\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_recv\_rev\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

## Function handle\_scatter\_send\_comp\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_scatter\_send\_comp\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

## Function handle\_scatter\_send\_comp\_rev\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_scatter\_send\_comp\_rev\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

## Function handle\_send\_comp\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **handle\_send\_comp\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `handle_send_comp_rev_event`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **handle\_send\_comp\_rev\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `isPEonThisRank`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

bool **isPEonThisRank** (int *jobID*, int *i*)

### Function `lpid_to_job`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

int **lpid\_to\_job** (int *lp\_gid*)

### Function `lpid_to_pe`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

int **lpid\_to\_pe** (int *lp\_gid*)

### Function `MsgEntry_getID`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

int **MsgEntry\_getID** (*MsgEntry* \*m)

### Function `MsgEntry_getNode`

- Defined in `file_tracer_reader_CWrapper.h`



## Function Documentation

int **MsgEntry\_getNode** (*MsgEntry* \*m)

## Function MsgEntry\_getPE

- Defined in file\_tracer\_reader\_CWrapper.h

## Function Documentation

int **MsgEntry\_getPE** (*MsgEntry* \*m)

## Function MsgEntry\_getSize

- Defined in file\_tracer\_reader\_CWrapper.h

## Function Documentation

int **MsgEntry\_getSize** (*MsgEntry* \*m)

## Function MsgEntry\_getThread

- Defined in file\_tracer\_reader\_CWrapper.h

## Function Documentation

int **MsgEntry\_getThread** (*MsgEntry* \*m)

## Function MsgID\_getID

- Defined in file\_tracer\_reader\_CWrapper.h

## Function Documentation

int **MsgID\_getID** (*MsgID* \*m)

## Function MsgID\_getPE

- Defined in file\_tracer\_reader\_CWrapper.h

## Function Documentation

int **MsgID\_getPE** (*MsgID* \*m)

### Function MsgID\_getSize

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

int **MsgID\_getSize** (*MsgID* \**m*)

### Function newMsgEntry

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

*MsgEntry* \***newMsgEntry** ()

### Function newMsgID

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

*MsgID* \***newMsgID** (int *size*, int *pe*, int *id*)

### Function ns\_to\_s

- Defined in file\_tracer\_tracer-driver.h

### Function Documentation

tw\_stime **ns\_to\_s** (tw\_stime *ns*)

### Function PE\_addTaskExecTime

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

void **PE\_addTaskExecTime** (*PE* \**p*, int *tInd*, double *time*)

### Function PE\_addToBuffer

- Defined in file\_tracer\_reader\_CWrapper.h

## Function Documentation

void **PE\_addToBuffer** (*PE* \**p*, *TaskPair* \**task\_id*)

## Function PE\_addToFrontBuffer

- Defined in file\_tracer\_reader\_CWrapper.h

## Function Documentation

void **PE\_addToFrontBuffer** (*PE* \**p*, *TaskPair* \**task\_id*)

## Function PE\_clearMsgBuffer

- Defined in file\_tracer\_reader\_CWrapper.h

## Function Documentation

void **PE\_clearMsgBuffer** (*PE* \**p*)

## Function PE\_dec\_iter

- Defined in file\_tracer\_reader\_CWrapper.h

## Function Documentation

void **PE\_dec\_iter** (*PE* \**p*)

## Function PE\_findTaskFromMsg

- Defined in file\_tracer\_reader\_CWrapper.h

## Function Documentation

int **PE\_findTaskFromMsg** (*PE* \**p*, *MsgID* \**msgId*)

## Function PE\_get\_currentTask

- Defined in file\_tracer\_reader\_CWrapper.h

## Function Documentation

int **PE\_get\_currentTask** (*PE* \**p*)

### Function `PE_get_iter`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

int **PE\_get\_iter** (*PE* \**p*)

### Function `PE_get_myEmPE`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

int **PE\_get\_myEmPE** (*PE* \**p*)

### Function `PE_get_myNum`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

int **PE\_get\_myNum** (*PE* \**p*)

### Function `PE_get_numWorkThreads`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

int **PE\_get\_numWorkThreads** (*PE* \**p*)

### Function `PE_get_taskDone`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

bool **PE\_get\_taskDone** (*PE* \**p*, int, int *tInd*)

### Function `PE_get_tasksCount`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

int **PE\_get\_tasksCount** (*PE* \**p*)

### Function PE\_get\_totalTasksCount

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

int **PE\_get\_totalTasksCount** (*PE* \**p*)

### Function PE\_getBufferSize

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

int **PE\_getBufferSize** (*PE* \**p*)

### Function PE\_getFirstTask

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

int **PE\_getFirstTask** (*PE* \**p*)

### Function PE\_getNextBuffedMsg

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

*TaskPair* **PE\_getNextBuffedMsg** (*PE* \**p*)

### Function PE\_getTaskExecTime

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

double **PE\_getTaskExecTime** (*PE* \**p*, int *tInd*)

### Function `PE_inc_iter`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

void **PE\_inc\_iter** (*PE* \**p*)

### Function `PE_invertMsgPe`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

void **PE\_invertMsgPe** (*PE* \**p*, int, int *tInd*)

### Function `PE_is_busy`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

bool **PE\_is\_busy** (*PE* \**p*)

### Function `PE_isEndEvent`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

bool **PE\_isEndEvent** (*PE* \**p*, int *task\_id*)

### Function `PE_isLoopEvent`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

bool **PE\_isLoopEvent** (*PE* \**p*, int *task\_id*)

### Function `PE_mark_all_done`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

void **PE\_mark\_all\_done** (*PE* \**p*, int *iter*, int *task\_id*)

### Function PE\_noMsgDep

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

bool **PE\_noMsgDep** (*PE* \**p*, int, int *tInd*)

### Function PE\_noUnsatDep

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

bool **PE\_noUnsatDep** (*PE* \**p*, int, int *tInd*)

### Function PE\_printStat

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

void **PE\_printStat** (*PE* \**p*, int *iter*)

### Function PE\_removeFromBuffer

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

void **PE\_removeFromBuffer** (*PE* \**p*, *TaskPair* \**task\_id*)

### Function PE\_resizeBuffer

- Defined in file\_tracer\_reader\_CWrapper.h

### Function Documentation

void **PE\_resizeBuffer** (*PE* \**p*, int *num\_elems\_to\_remove*)

### Function `PE_set_busy`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

void **PE\_set\_busy** (*PE* \**p*, bool *b*)

### Function `PE_set_currentTask`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

void **PE\_set\_currentTask** (*PE* \**p*, int *tInd*)

### Function `PE_set_taskDone`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

void **PE\_set\_taskDone** (*PE* \**p*, int, int *tInd*, bool *b*)

### Function `pe_to_job`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

int **pe\_to\_job** (int *pe*)

### Function `pe_to_lpid`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

int **pe\_to\_lpid** (int *pe*, int *job*)

### Function `perform_a2a`

- Defined in `file_tracer_tracer-driver.h`



## Function Documentation

void **perform\_a2a** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

## Function perform\_a2a\_blocked

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **perform\_a2a\_blocked** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

## Function perform\_a2a\_blocked\_rev

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **perform\_a2a\_blocked\_rev** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

## Function perform\_a2a\_rev

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **perform\_a2a\_rev** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

## Function perform\_allgather

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **perform\_allgather** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

## Function perform\_allgather\_rev

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **perform\_allgather\_rev** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

### Function `perform_allreduce`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **perform\_allreduce** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

### Function `perform_allreduce_rev`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **perform\_allreduce\_rev** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

### Function `perform_bcast`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **perform\_bcast** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

### Function `perform_bcast_rev`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **perform\_bcast\_rev** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

### Function `perform_bruck`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **perform\_bruck** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

### Function `perform_bruck_rev`

- Defined in `file_tracer_tracer-driver.h`

## Function Documentation

void **perform\_bruck\_rev** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

## Function perform\_collective

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **perform\_collective** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b)

## Function perform\_collective\_rev

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **perform\_collective\_rev** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b)

## Function perform\_reduction

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **perform\_reduction** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

## Function perform\_reduction\_rev

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **perform\_reduction\_rev** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

## Function perform\_scatter

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **perform\_scatter** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

### Function `perform_scatter_rev`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **perform\_scatter\_rev** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

### Function `perform_scatter_small`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **perform\_scatter\_small** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

### Function `perform_scatter_small_rev`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **perform\_scatter\_small\_rev** (*proc\_state* \*ns, int task\_id, tw\_lp \*lp, *proc\_msg* \*m, tw\_bf \*b, int isEvent)

### Function `proc_commit_event`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **proc\_commit\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `proc_event`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

void **proc\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function `proc_finalize`

- Defined in `file_tracer_tracer-driver.h`

## Function Documentation

void **proc\_finalize** (*proc\_state* \*ns, tw\_lp \*lp)

### Function proc\_init

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **proc\_init** (*proc\_state* \*ns, tw\_lp \*lp)

### Function proc\_rev\_event

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

void **proc\_rev\_event** (*proc\_state* \*ns, tw\_bf \*b, *proc\_msg* \*m, tw\_lp \*lp)

### Function s\_to\_ns

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

tw\_stime **s\_to\_ns** (tw\_stime ns)

### Function send\_coll\_comp

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

int **send\_coll\_comp** (*proc\_state* \*ns, tw\_stime sendOffset, int collType, tw\_lp \*lp, int isEvent, *proc\_msg* \*m)

### Function send\_coll\_comp\_rev

- Defined in file\_tracer\_tracer-driver.h

## Function Documentation

int **send\_coll\_comp\_rev** (*proc\_state* \*ns, tw\_stime sendOffset, int collType, tw\_lp \*lp, int isEvent, *proc\_msg* \*m)

### Function `send_msg`

- Defined in `file_tracer_tracer-driver.h`

### Function Documentation

int **send\_msg** (*proc\_state* \*ns, int size, int iter, *MsgID* \*msgId, int64\_t seq, int dest\_id, tw\_stime timeOffset, enum *proc\_event* evt\_type, tw\_lp \*lp, bool fillSz = false, int64\_t size2 = 0)

### Function `TraceReader_readOTF2Trace`

- Defined in `file_tracer_reader_CWrapper.h`

### Function Documentation

void **TraceReader\_readOTF2Trace** (*PE* \*pe, int my\_pe\_num, int my\_job, double \*startTime)

## 4.3.4 Variables

### Variable `copy_per_byte`

- Defined in `file_tracer_tracer-driver.h`

### Variable Documentation

double **copy\_per\_byte**

### Variable `eager_limit`

- Defined in `file_tracer_tracer-driver.h`

### Variable Documentation

double **eager\_limit**

### Variable `jobs`

- Defined in `file_tracer_tracer-driver.h`

### Variable Documentation

*JobInf* \***jobs**

### Variable `net_id`

- Defined in `file_tracer_tracer-driver.h`

### Variable Documentation

int **net\_id**

### Variable `nic_delay`

- Defined in `file_tracer_tracer-driver.h`

### Variable Documentation

tw\_stime **nic\_delay**

### Variable `print_frequency`

- Defined in `file_tracer_tracer-driver.h`

### Variable Documentation

unsigned int **print\_frequency**

### Variable `rdma_delay`

- Defined in `file_tracer_tracer-driver.h`

### Variable Documentation

tw\_stime **rdma\_delay**

### Variable `soft_delay_mpi`

- Defined in `file_tracer_tracer-driver.h`

### Variable Documentation

tw\_stime **soft\_delay\_mpi**

## 4.3.5 Defines

### Define `BCAST_DEGREE`

- Defined in `file_tracer_tracer-driver.h`

### Define Documentation

**BCAST\_DEGREE**

### Define MPI\_INTERNAL\_DELAY

- Defined in file\_tracer\_tracer-driver.h

### Define Documentation

**MPI\_INTERNAL\_DELAY**

### Define REDUCE\_DEGREE

- Defined in file\_tracer\_tracer-driver.h

### Define Documentation

**REDUCE\_DEGREE**

### Define TIME\_MULT

- Defined in file\_tracer\_elements\_Task.h

### Define Documentation

**TIME\_MULT**

### Define TRACER\_A2A\_ALG\_CUTOFF

- Defined in file\_tracer\_tracer-driver.h

### Define Documentation

**TRACER\_A2A\_ALG\_CUTOFF**

### Define TRACER\_ALLGATHER\_ALG\_CUTOFF

- Defined in file\_tracer\_tracer-driver.h

### Define Documentation

**TRACER\_ALLGATHER\_ALG\_CUTOFF**



### Define TRACER\_BLOCK\_SIZE

- Defined in file\_tracer\_tracer-driver.h

### Define Documentation

**TRACER\_BLOCK\_SIZE**

### Define TRACER\_SCATTER\_ALG\_CUTOFF

- Defined in file\_tracer\_tracer-driver.h

### Define Documentation

**TRACER\_SCATTER\_ALG\_CUTOFF**

## 4.3.6 Typedefs

### Typedef CollKeyType

- Defined in file\_tracer\_elements\_PE.h

### Typedef Documentation

**typedef** std::map<*CollMsgKey*, std::list<int>> **CollKeyType**

### Typedef CoreInf

- Defined in file\_tracer\_tracer-driver.h

### Typedef Documentation

**typedef struct** *CoreInf* **CoreInf**

### Typedef JobInf

- Defined in file\_tracer\_reader\_datatypes.h

### Typedef Documentation

**typedef struct** *JobInf* **JobInf**

### Typedef KeyType

- Defined in file\_tracer\_elements\_PE.h

### Typedef Documentation

**typedef** std::map<*MsgKey*, std::list<int>> **KeyType**

### Typedef MsgEntry

- Defined in file\_tracer\_reader\_CWrapper.h

### Typedef Documentation

**typedef struct** *MsgEntry* **MsgEntry**

### Typedef MsgID

- Defined in file\_tracer\_reader\_CWrapper.h

### Typedef Documentation

**typedef struct** *MsgID* **MsgID**

### Typedef PE

- Defined in file\_tracer\_reader\_CWrapper.h

### Typedef Documentation

**typedef struct** *PE* **PE**

### Typedef TaskPair

- Defined in file\_tracer\_reader\_datatypes.h

### Typedef Documentation

**typedef struct** *TaskPair* **TaskPair**

## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## A

addEventSub (C++ *function*), 21  
addMsgSizeSub (C++ *function*), 21

## B

BCAST (C++ *enumerator*), 20  
BCAST\_DEGREE (C *macro*), 44  
bcast\_msg (C++ *function*), 22

## C

COLL\_A2A (C++ *enumerator*), 20  
COLL\_A2A\_BLOCKED (C++ *enumerator*), 20  
COLL\_A2A\_BLOCKED\_SEND\_DONE (C++ *enumerator*), 20  
COLL\_A2A\_SEND\_DONE (C++ *enumerator*), 20  
COLL\_ALLGATHER (C++ *enumerator*), 20  
COLL\_ALLGATHER\_SEND\_DONE (C++ *enumerator*), 20  
COLL\_BCAST (C++ *enumerator*), 20  
COLL\_BRUCK (C++ *enumerator*), 20  
COLL\_BRUCK\_SEND\_DONE (C++ *enumerator*), 20  
COLL\_COMPLETE (C++ *enumerator*), 21  
Coll\_lookup (C++ *class*), 13  
Coll\_lookup::local\_event (C++ *member*), 13  
Coll\_lookup::remote\_event (C++ *member*), 13  
COLL\_REDUCTION (C++ *enumerator*), 20  
COLL\_SCATTER (C++ *enumerator*), 21  
COLL\_SCATTER\_SEND\_DONE (C++ *enumerator*), 21  
COLL\_SCATTER\_SMALL (C++ *enumerator*), 20  
CollKeyType (C++ *type*), 45  
CollMsgKey (C++ *class*), 16  
CollMsgKey::~CollMsgKey (C++ *function*), 16  
CollMsgKey::CollMsgKey (C++ *function*), 16  
CollMsgKey::comm (C++ *member*), 16  
CollMsgKey::operator< (C++ *function*), 16  
CollMsgKey::rank (C++ *member*), 16  
CollMsgKey::seq (C++ *member*), 16  
copy\_per\_byte (C++ *member*), 42  
CoreInf (C++ *class*), 13  
CoreInf (C++ *type*), 45  
CoreInf::jobID (C++ *member*), 13  
CoreInf::mapsTo (C++ *member*), 13

## D

delegate\_send\_msg (C++ *function*), 22

## E

eager\_limit (C++ *member*), 42  
enqueue\_msg (C++ *function*), 22  
exec\_comp (C++ *function*), 22  
EXEC\_COMPLETE (C++ *enumerator*), 20  
exec\_task (C++ *function*), 22  
exec\_task\_rev (C++ *function*), 23

## H

handle\_a2a\_blocked\_send\_comp\_event (C++ *function*), 23  
handle\_a2a\_blocked\_send\_comp\_rev\_event (C++ *function*), 23  
handle\_a2a\_send\_comp\_event (C++ *function*), 23  
handle\_a2a\_send\_comp\_rev\_event (C++ *function*), 23  
handle\_allgather\_send\_comp\_event (C++ *function*), 23  
handle\_allgather\_send\_comp\_rev\_event (C++ *function*), 24  
handle\_bcast\_event (C++ *function*), 24  
handle\_bcast\_rev\_event (C++ *function*), 24  
handle\_bruck\_send\_comp\_event (C++ *function*), 24  
handle\_bruck\_send\_comp\_rev\_event (C++ *function*), 24  
handle\_coll\_complete\_event (C++ *function*), 25  
handle\_coll\_complete\_rev\_event (C++ *function*), 25  
handle\_coll\_recv\_post\_event (C++ *function*), 25  
handle\_coll\_recv\_post\_rev\_event (C++ *function*), 25  
handle\_exec\_event (C++ *function*), 25  
handle\_exec\_rev\_event (C++ *function*), 25  
handle\_kickoff\_event (C++ *function*), 26  
handle\_kickoff\_rev\_event (C++ *function*), 26

`handle_local_event` (C++ *function*), 26  
`handle_local_rev_event` (C++ *function*), 26  
`handle_recv_event` (C++ *function*), 26  
`handle_recv_post_event` (C++ *function*), 27  
`handle_recv_post_rev_event` (C++ *function*), 27  
`handle_recv_rev_event` (C++ *function*), 27  
`handle_scatter_send_comp_event` (C++ *function*), 27  
`handle_scatter_send_comp_rev_event` (C++ *function*), 27  
`handle_send_comp_event` (C++ *function*), 27  
`handle_send_comp_rev_event` (C++ *function*), 28

## I

`isPEonThisRank` (C++ *function*), 28

## J

`JobInf` (C++ *class*), 14  
`JobInf` (C++ *type*), 45  
`JobInf::map_file` (C++ *member*), 14  
`JobInf::numIters` (C++ *member*), 14  
`JobInf::numRanks` (C++ *member*), 14  
`JobInf::offsets` (C++ *member*), 14  
`JobInf::rankMap` (C++ *member*), 14  
`JobInf::skipMsgId` (C++ *member*), 14  
`JobInf::traceDir` (C++ *member*), 14  
`jobs` (C++ *member*), 42

## K

`KeyType` (C++ *type*), 46  
`KICKOFF` (C++ *enumerator*), 20

## L

`LOCAL` (C++ *enumerator*), 20  
`lpid_to_job` (C++ *function*), 28  
`lpid_to_pe` (C++ *function*), 28

## M

`MPI_INTERNAL_DELAY` (C *macro*), 44  
`MsgEntry` (C++ *class*), 14  
`MsgEntry` (C++ *type*), 46  
`MsgEntry::msgId` (C++ *member*), 14  
`MsgEntry::node` (C++ *member*), 14  
`MsgEntry::thread` (C++ *member*), 14  
`MsgEntry_getID` (C++ *function*), 28  
`MsgEntry_getNode` (C++ *function*), 29  
`MsgEntry_getPE` (C++ *function*), 29  
`MsgEntry_getSize` (C++ *function*), 29  
`MsgEntry_getThread` (C++ *function*), 29  
`MsgID` (C++ *class*), 14  
`MsgID` (C++ *type*), 46

`MsgID::id` (C++ *member*), 15  
`MsgID::pe` (C++ *member*), 15  
`MsgID::size` (C++ *member*), 15  
`MsgID_getID` (C++ *function*), 29  
`MsgID_getPE` (C++ *function*), 29  
`MsgID_getSize` (C++ *function*), 30  
`MsgKey` (C++ *class*), 16  
`MsgKey::~~MsgKey` (C++ *function*), 17  
`MsgKey::comm` (C++ *member*), 17  
`MsgKey::MsgKey` (C++ *function*), 17  
`MsgKey::operator<` (C++ *function*), 17  
`MsgKey::rank` (C++ *member*), 17  
`MsgKey::seq` (C++ *member*), 17  
`MsgKey::tag` (C++ *member*), 17

## N

`net_id` (C++ *member*), 43  
`newMsgEntry` (C++ *function*), 30  
`newMsgID` (C++ *function*), 30  
`nic_delay` (C++ *member*), 43  
`ns_to_s` (C++ *function*), 30

## P

`PE` (C++ *class*), 17  
`PE` (C++ *type*), 46  
`PE::~~PE` (C++ *function*), 17  
`PE::addTaskExecTime` (C++ *function*), 17  
`PE::allMarked` (C++ *member*), 18  
`PE::beforeTask` (C++ *member*), 18  
`PE::busy` (C++ *member*), 18  
`PE::collectiveSeq` (C++ *member*), 18  
`PE::currentCollComm` (C++ *member*), 18  
`PE::currentCollMsgSize` (C++ *member*), 18  
`PE::currentCollPartner` (C++ *member*), 18  
`PE::currentCollRank` (C++ *member*), 18  
`PE::currentCollRecvCount` (C++ *member*), 19  
`PE::currentCollSendCount` (C++ *member*), 19  
`PE::currentCollSeq` (C++ *member*), 18  
`PE::currentCollSize` (C++ *member*), 19  
`PE::currentCollTask` (C++ *member*), 18  
`PE::currentTask` (C++ *member*), 18  
`PE::currIter` (C++ *member*), 18  
`PE::currTime` (C++ *member*), 18  
`PE::findTaskFromMsg` (C++ *function*), 17  
`PE::firstTask` (C++ *member*), 18  
`PE::getTaskExecTime` (C++ *function*), 17  
`PE::goToNextIter` (C++ *function*), 17  
`PE::invertMsgPe` (C++ *function*), 17  
`PE::jobNum` (C++ *member*), 18  
`PE::loop_start_task` (C++ *member*), 18  
`PE::mark_all_done` (C++ *function*), 17  
`PE::msgBuffer` (C++ *member*), 18  
`PE::msgDestLogs` (C++ *member*), 18  
`PE::msgStatus` (C++ *member*), 18

PE::myEmPE (C++ member), 18  
 PE::myNum (C++ member), 18  
 PE::myTasks (C++ member), 18  
 PE::noUnsatDep (C++ function), 17  
 PE::numEmPes (C++ member), 18  
 PE::numWth (C++ member), 18  
 PE::PE (C++ function), 17  
 PE::pendingCollMsgs (C++ member), 18  
 PE::pendingMsgs (C++ member), 18  
 PE::pendingRCollMsgs (C++ member), 18  
 PE::pendingReqs (C++ member), 18  
 PE::pendingRMsgs (C++ member), 18  
 PE::pendingRReqs (C++ member), 18  
 PE::printStats (C++ function), 17  
 PE::recvSeq (C++ member), 18  
 PE::sendSeq (C++ member), 18  
 PE::taskExecTime (C++ function), 17  
 PE::taskExecuted (C++ member), 18  
 PE::tasksCount (C++ member), 18  
 PE::taskStatus (C++ member), 18  
 PE::totalTasksCount (C++ member), 18  
 PE\_addTaskExecTime (C++ function), 30  
 PE\_addToBuffer (C++ function), 31  
 PE\_addToFrontBuffer (C++ function), 31  
 PE\_clearMsgBuffer (C++ function), 31  
 PE\_dec\_iter (C++ function), 31  
 PE\_findTaskFromMsg (C++ function), 31  
 PE\_get\_currentTask (C++ function), 31  
 PE\_get\_iter (C++ function), 32  
 PE\_get\_myEmPE (C++ function), 32  
 PE\_get\_myNum (C++ function), 32  
 PE\_get\_numWorkThreads (C++ function), 32  
 PE\_get\_taskDone (C++ function), 32  
 PE\_get\_tasksCount (C++ function), 33  
 PE\_get\_totalTasksCount (C++ function), 33  
 PE\_getBufferSize (C++ function), 33  
 PE\_getFirstTask (C++ function), 33  
 PE\_getNextBufferedMsg (C++ function), 33  
 PE\_getTaskExecTime (C++ function), 33  
 PE\_inc\_iter (C++ function), 34  
 PE\_invertMsgPe (C++ function), 34  
 PE\_is\_busy (C++ function), 34  
 PE\_isEndEvent (C++ function), 34  
 PE\_isLoopEvent (C++ function), 34  
 PE\_mark\_all\_done (C++ function), 35  
 PE\_noMsgDep (C++ function), 35  
 PE\_noUnsatDep (C++ function), 35  
 PE\_printStat (C++ function), 35  
 PE\_removeFromBuffer (C++ function), 35  
 PE\_resizeBuffer (C++ function), 35  
 PE\_set\_busy (C++ function), 36  
 PE\_set\_currentTask (C++ function), 36  
 PE\_set\_taskDone (C++ function), 36  
 pe\_to\_job (C++ function), 36

pe\_to\_lpid (C++ function), 36  
 perform\_a2a (C++ function), 37  
 perform\_a2a\_blocked (C++ function), 37  
 perform\_a2a\_blocked\_rev (C++ function), 37  
 perform\_a2a\_rev (C++ function), 37  
 perform\_allgather (C++ function), 37  
 perform\_allgather\_rev (C++ function), 37  
 perform\_allreduce (C++ function), 38  
 perform\_allreduce\_rev (C++ function), 38  
 perform\_bcast (C++ function), 38  
 perform\_bcast\_rev (C++ function), 38  
 perform\_bruck (C++ function), 38  
 perform\_bruck\_rev (C++ function), 39  
 perform\_collective (C++ function), 39  
 perform\_collective\_rev (C++ function), 39  
 perform\_reduction (C++ function), 39  
 perform\_reduction\_rev (C++ function), 39  
 perform\_scatter (C++ function), 39  
 perform\_scatter\_rev (C++ function), 40  
 perform\_scatter\_small (C++ function), 40  
 perform\_scatter\_small\_rev (C++ function), 40  
 print\_frequency (C++ member), 43  
 proc\_commit\_event (C++ function), 40  
 proc\_event (C++ enum), 20  
 proc\_event (C++ function), 40  
 proc\_finalize (C++ function), 41  
 proc\_init (C++ function), 41  
 proc\_msg (C++ class), 15  
 proc\_msg::coll\_info (C++ member), 15  
 proc\_msg::coll\_info\_2 (C++ member), 15  
 proc\_msg::executed (C++ member), 15  
 proc\_msg::fwd\_dep\_count (C++ member), 15  
 proc\_msg::incremented\_flag (C++ member), 15  
 proc\_msg::iteration (C++ member), 15  
 proc\_msg::model\_net\_calls (C++ member), 15  
 proc\_msg::msgId (C++ member), 15  
 proc\_msg::proc\_event\_type (C++ member), 15  
 proc\_msg::saved\_task (C++ member), 15  
 proc\_msg::src (C++ member), 15  
 proc\_rev\_event (C++ function), 41  
 proc\_state (C++ class), 15  
 proc\_state::end\_ts (C++ member), 15  
 proc\_state::my\_job (C++ member), 16  
 proc\_state::my\_pe (C++ member), 15  
 proc\_state::my\_pe\_num (C++ member), 15  
 proc\_state::sim\_start (C++ member), 15  
 proc\_state::start\_ts (C++ member), 15

## R

rdma\_delay (C++ member), 43  
 RECV\_COLL\_POST (C++ enumerator), 21  
 RECV\_MSG (C++ enumerator), 20  
 RECV\_POST (C++ enumerator), 20

REDUCE\_DEGREE (*C macro*), 44

## S

s\_to\_ns (*C++ function*), 41

send\_coll\_comp (*C++ function*), 41

send\_coll\_comp\_rev (*C++ function*), 41

SEND\_COMP (*C++ enumerator*), 20

send\_msg (*C++ function*), 42

soft\_delay\_mpi (*C++ member*), 43

## T

Task (*C++ class*), 19

Task::~~Task (*C++ function*), 19

Task::endEvent (*C++ member*), 19

Task::execTime (*C++ member*), 19

Task::loopEvent (*C++ member*), 19

Task::loopStartEvent (*C++ member*), 19

Task::Task (*C++ function*), 19

TaskPair (*C++ class*), 16

TaskPair (*C++ type*), 46

TaskPair::iter (*C++ member*), 16

TaskPair::taskid (*C++ member*), 16

TIME\_MULT (*C macro*), 44

TRACER\_A2A\_ALG\_CUTOFF (*C macro*), 44

TRACER\_ALLGATHER\_ALG\_CUTOFF (*C macro*), 44

TRACER\_BLOCK\_SIZE (*C macro*), 45

tracer\_coll\_type (*C++ enum*), 21

TRACER\_COLLECTIVE\_ALL\_BRUCK (*C++ enumerator*), 21

TRACER\_COLLECTIVE\_ALLGATHER\_LARGE (*C++ enumerator*), 21

TRACER\_COLLECTIVE\_ALLTOALL\_BLOCKED (*C++ enumerator*), 21

TRACER\_COLLECTIVE\_ALLTOALL\_LARGE (*C++ enumerator*), 21

TRACER\_COLLECTIVE\_BARRIER (*C++ enumerator*), 21

TRACER\_COLLECTIVE\_BCAST (*C++ enumerator*), 21

TRACER\_COLLECTIVE\_REDUCE (*C++ enumerator*), 21

TRACER\_COLLECTIVE\_SCATTER (*C++ enumerator*), 21

TRACER\_COLLECTIVE\_SCATTER\_SMALL (*C++ enumerator*), 21

TRACER\_SCATTER\_ALG\_CUTOFF (*C macro*), 45

TraceReader (*C++ class*), 19

TraceReader::~~TraceReader (*C++ function*), 19

TraceReader::allNodeOffsets (*C++ member*), 20

TraceReader::fileLoc (*C++ member*), 20

TraceReader::firstLog (*C++ member*), 20

TraceReader::numEmPes (*C++ member*), 20

TraceReader::numWth (*C++ member*), 20

TraceReader::totalNodes (*C++ member*), 20

TraceReader::totalTlineLength (*C++ member*), 20

TraceReader::totalWorkerProcs (*C++ member*), 20

TraceReader::tracePath (*C++ member*), 20

TraceReader::TraceReader (*C++ function*), 19

TraceReader\_readOTF2Trace (*C++ function*), 42