TraceR Documentation

Nikhil Jain, Bilge Acun, Abhinav Bhatele

May 22, 2019

CONTENTS:

1 Download and Install **3**
- 1.1 Dependencies . . . 3
- 1.2 Build . . . 3

2 User Guide **5**
- 2.1 Quickstart . . . 5
- 2.2 Creating a TraceR config file . . . 5
- 2.3 Creating the target system configuration . . . 6
- 2.4 Creating the job placement file . . . 6
- 2.5 Generating Traces . . . 6

3 Tutorial **9**

4 Source Code Documentation **11**
- 4.1 Class Hierarchy . . . 11
- 4.2 File Hierarchy . . . 11
- 4.3 Full API . . . 11

5 Indices and tables **77**

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.

CHAPTER

ONE

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.

CHAPTER

TWO

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 TraceR.

--sync ROSS's PDES type. 1 - sequential, 2 - conservative, 3 - optimistic

--nkp number of groups used for clustering LPs; recommended value for lower rollbacks: (total #LPs)/(#MPI processes)

--extramem number of messages in ROSS's extra message buffer (each message is ~500 bytes, 100K should work for most cases)

--max-opt-lookahead leash on optimisitc execution in nanoseconds (1 micro second is a good value)

--timer-frequency frequency with which PE0 should print current virtual time

2.2 Creating a TraceR config 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)>
```

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

See below to generate global or per-job map files.

2.3 Creating the target system configuration

2.4 Creating the job placement file

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.

Quick start

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
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

CHAPTER

THREE

TUTORIAL

CHAPTER

FOUR

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-driver.h*

Struct Documentation

struct Coll_lookup

Public Members

proc_event **remote_event**

proc_event **local_event**

Struct CoreInf

- Defined in *File tracer-driver.h*

Struct Documentation

struct CoreInf

Public Members

int **mapsTo**

int **jobID**

Struct JobInf

- Defined in *File 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 MsgEntry.h*

Struct Documentation

struct MsgEntry

Public Members

int **node**

int **thread**

MsgID **msgId**

Struct MsgID

- Defined in *File MsgEntry.h*

Struct Documentation

struct MsgID

Public Members

int **pe**

int **id**

uint64_t **size**

Struct proc_msg

- Defined in *File 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-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 datatypes.h*

Struct Documentation

struct TaskPair

Public Members

int **iter**

int **taskid**

Class CollMsgKey

- Defined in *File 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 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 PE.h*

Class Documentation

class PE

Public Functions

PE()

~PE()

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

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

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

void **printStat**()

void **check**()

void **printState**()

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 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 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-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-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 CWrapper.h*

Function Documentation

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

Function addMsgSizeSub

- Defined in *File CWrapper.h*

Function Documentation

void **addMsgSizeSub**(int *job*, int64_t *key*, int64_t *val*, int *numjobs*)

Function bcast_msg

- Defined in *File 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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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 CWrapper.h*

Function Documentation

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

Function lpid_to_job

- Defined in *File tracer-driver.h*

Function Documentation

int **lpid_to_job** (int *lp_gid*)

Function lpid_to_pe

- Defined in *File tracer-driver.h*

Function Documentation

int **lpid_to_pe** (int *lp_gid*)

Function MsgEntry_getID

- Defined in *File CWrapper.h*

Function Documentation

int **MsgEntry_getID** (*MsgEntry* *m)

Function MsgEntry_getNode

- Defined in *File CWrapper.h*

Function Documentation

int **MsgEntry_getNode** (*MsgEntry* *m*)

Function MsgEntry_getPE

- Defined in *File CWrapper.h*

Function Documentation

int **MsgEntry_getPE** (*MsgEntry* *m*)

Function MsgEntry_getSize

- Defined in *File CWrapper.h*

Function Documentation

int **MsgEntry_getSize** (*MsgEntry* *m*)

Function MsgEntry_getThread

- Defined in *File CWrapper.h*

Function Documentation

int **MsgEntry_getThread** (*MsgEntry* *m*)

Function MsgID_getID

- Defined in *File CWrapper.h*

Function Documentation

int **MsgID_getID** (*MsgID* *m*)

Function MsgID_getPE

- Defined in *File CWrapper.h*

Function Documentation

int **MsgID_getPE** (*MsgID* *m*)

Function MsgID_getSize

- Defined in *File CWrapper.h*

Function Documentation

int **MsgID_getSize** (*MsgID* *m)

Function newMsgEntry

- Defined in *File CWrapper.h*

Function Documentation

MsgEntry ***newMsgEntry** ()

Function newMsgID

- Defined in *File CWrapper.h*

Function Documentation

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

Function ns_to_s

- Defined in *File tracer-driver.h*

Function Documentation

tw_stime **ns_to_s** (tw_stime *ns*)

Function PE_addTaskExecTime

- Defined in *File CWrapper.h*

Function Documentation

void **PE_addTaskExecTime** (*PE* *p, int *tInd*, double *time*)

Function PE_addToBuffer

- Defined in *File CWrapper.h*

Function Documentation

void **PE_addToBuffer**(*PE* *p, *TaskPair* *task_id)

Function PE_addToFrontBuffer

- Defined in *File CWrapper.h*

Function Documentation

void **PE_addToFrontBuffer**(*PE* *p, *TaskPair* *task_id)

Function PE_clearMsgBuffer

- Defined in *File CWrapper.h*

Function Documentation

void **PE_clearMsgBuffer**(*PE* *p)

Function PE_dec_iter

- Defined in *File CWrapper.h*

Function Documentation

void **PE_dec_iter**(*PE* *p)

Function PE_findTaskFromMsg

- Defined in *File CWrapper.h*

Function Documentation

int **PE_findTaskFromMsg**(*PE* *p, *MsgID* *msgId)

Function PE_get_currentTask

- Defined in *File CWrapper.h*

Function Documentation

int **PE_get_currentTask**(*PE* *p)

Function PE_get_iter

- Defined in *File CWrapper.h*

Function Documentation

int **PE_get_iter** (*PE* *p)

Function PE_get_myEmPE

- Defined in *File CWrapper.h*

Function Documentation

int **PE_get_myEmPE** (*PE* *p)

Function PE_get_myNum

- Defined in *File CWrapper.h*

Function Documentation

int **PE_get_myNum** (*PE* *p)

Function PE_get_numWorkThreads

- Defined in *File CWrapper.h*

Function Documentation

int **PE_get_numWorkThreads** (*PE* *p)

Function PE_get_taskDone

- Defined in *File CWrapper.h*

Function Documentation

bool **PE_get_taskDone** (*PE* *p, int, int *tInd*)

Function PE_get_tasksCount

- Defined in *File CWrapper.h*

Function Documentation

int **PE_get_tasksCount** (*PE* *p)

Function PE_get_totalTasksCount

- Defined in *File CWrapper.h*

Function Documentation

int **PE_get_totalTasksCount** (*PE* *p)

Function PE_getBufferSize

- Defined in *File CWrapper.h*

Function Documentation

int **PE_getBufferSize** (*PE* *p)

Function PE_getFirstTask

- Defined in *File CWrapper.h*

Function Documentation

int **PE_getFirstTask** (*PE* *p)

Function PE_getNextBuffedMsg

- Defined in *File CWrapper.h*

Function Documentation

TaskPair **PE_getNextBuffedMsg** (*PE* *p)

Function PE_getTaskExecTime

- Defined in *File CWrapper.h*

Function Documentation

double **PE_getTaskExecTime** (*PE* *p, int *tInd*)

Function PE_inc_iter

- Defined in *File CWrapper.h*

Function Documentation

void **PE_inc_iter**(*PE* *p)

Function PE_invertMsgPe

- Defined in *File CWrapper.h*

Function Documentation

void **PE_invertMsgPe**(*PE* *p, int, int *tInd*)

Function PE_is_busy

- Defined in *File CWrapper.h*

Function Documentation

bool **PE_is_busy**(*PE* *p)

Function PE_isEndEvent

- Defined in *File CWrapper.h*

Function Documentation

bool **PE_isEndEvent**(*PE* *p, int *task_id*)

Function PE_isLoopEvent

- Defined in *File CWrapper.h*

Function Documentation

bool **PE_isLoopEvent**(*PE* *p, int *task_id*)

Function PE_mark_all_done

- Defined in *File CWrapper.h*

Function Documentation

void **PE_mark_all_done**(*PE* *p, int *iter*, int *task_id*)

Function PE_noMsgDep

- Defined in *File CWrapper.h*

Function Documentation

bool **PE_noMsgDep**(*PE* *p, int, int *tInd*)

Function PE_noUnsatDep

- Defined in *File CWrapper.h*

Function Documentation

bool **PE_noUnsatDep**(*PE* *p, int, int *tInd*)

Function PE_printStat

- Defined in *File CWrapper.h*

Function Documentation

void **PE_printStat**(*PE* *p)

Function PE_removeFromBuffer

- Defined in *File CWrapper.h*

Function Documentation

void **PE_removeFromBuffer**(*PE* *p, *TaskPair* *task_id)

Function PE_resizeBuffer

- Defined in *File CWrapper.h*

Function Documentation

void **PE_resizeBuffer**(*PE* *p, int *num_elems_to_remove*)

Function PE_set_busy

- Defined in *File CWrapper.h*

Function Documentation

void **PE_set_busy**(*PE* *p, bool *b*)

Function PE_set_currentTask

- Defined in *File CWrapper.h*

Function Documentation

void **PE_set_currentTask**(*PE* *p, int *tInd*)

Function PE_set_taskDone

- Defined in *File CWrapper.h*

Function Documentation

void **PE_set_taskDone**(*PE* *p, int, int *tInd*, bool *b*)

Function pe_to_job

- Defined in *File tracer-driver.h*

Function Documentation

int **pe_to_job**(int *pe*)

Function pe_to_lpid

- Defined in *File tracer-driver.h*

Function Documentation

int **pe_to_lpid**(int *pe*, int *job*)

Function perform_a2a

- Defined in *File 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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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-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_event

- Defined in *File 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-driver.h*

Function Documentation

void **proc_finalize**(*proc_state* *ns, tw_lp *lp)

Function proc_init

- Defined in *File tracer-driver.h*

Function Documentation

void **proc_init** (*proc_state* *ns, tw_lp *lp)

Function proc_rev_event

- Defined in *File 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-driver.h*

Function Documentation

tw_stime **s_to_ns** (tw_stime *ns*)

Function send_coll_comp

- Defined in *File 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-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-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 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-driver.h*

Variable Documentation

double **copy_per_byte**

Variable eager_limit

- Defined in *File tracer-driver.h*

Variable Documentation

double **eager_limit**

Variable jobs

- Defined in *File tracer-driver.h*

Variable Documentation

JobInf ***jobs**

Variable net_id

- Defined in *File tracer-driver.h*

Variable Documentation

int **net_id**

Variable nic_delay

- Defined in *File tracer-driver.h*

Variable Documentation

tw_stime **nic_delay**

Variable print_frequency

- Defined in *File tracer-driver.h*

Variable Documentation

unsigned int **print_frequency**

Variable rdma_delay

- Defined in *File tracer-driver.h*

Variable Documentation

tw_stime **rdma_delay**

Variable soft_delay_mpi

- Defined in *File tracer-driver.h*

Variable Documentation

tw_stime **soft_delay_mpi**

4.3.5 Defines

Define BCAST_DEGREE

- Defined in *File tracer-driver.h*

Define Documentation

BCAST_DEGREE

Define MPI_INTERNAL_DELAY

- Defined in *File tracer-driver.h*

Define Documentation

MPI_INTERNAL_DELAY

Define REDUCE_DEGREE

- Defined in *File tracer-driver.h*

Define Documentation

REDUCE_DEGREE

Define TIME_MULT

- Defined in *File Task.h*

Define Documentation

TIME_MULT

Define TRACER_A2A_ALG_CUTOFF

- Defined in *File tracer-driver.h*

Define Documentation

TRACER_A2A_ALG_CUTOFF

Define TRACER_ALLGATHER_ALG_CUTOFF

- Defined in *File tracer-driver.h*

Define Documentation

TRACER_ALLGATHER_ALG_CUTOFF

Define TRACER_BLOCK_SIZE

- Defined in *File tracer-driver.h*

Define Documentation

TRACER_BLOCK_SIZE

Define TRACER_SCATTER_ALG_CUTOFF

- Defined in *File tracer-driver.h*

Define Documentation

TRACER_SCATTER_ALG_CUTOFF

4.3.6 Typedefs

Typedef CollKeyType

- Defined in *File PE.h*

Typedef Documentation

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

Typedef CoreInf

- Defined in *File tracer-driver.h*

Typedef Documentation

typedef struct *CoreInf* **CoreInf**

Typedef JobInf

- Defined in *File datatypes.h*

Typedef Documentation

typedef struct *JobInf* **JobInf**

Typedef KeyType

- Defined in *File PE.h*

Typedef Documentation

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

Typedef MsgEntry

- Defined in *File CWrapper.h*

Typedef Documentation

typedef struct *MsgEntry* **MsgEntry**

Typedef MsgID

- Defined in *File CWrapper.h*

Typedef Documentation

`typedef struct` *MsgID* `MsgID`

Typedef PE

- Defined in *File CWrapper.h*

Typedef Documentation

`typedef struct` *PE* `PE`

Typedef TaskPair

- Defined in *File datatypes.h*

Typedef Documentation

`typedef struct` *TaskPair* `TaskPair`

4.3.7 Directories

Directory tracer

Directory path: `tracer`

Subdirectories

- *Directory elements*
- *Directory reader*

Files

- *File tracer-driver.h*

Directory elements

Parent directory (`tracer`)

Directory path: `tracer/elements`

Files

- *File MsgEntry.h*
- *File PE.h*
- *File Task.h*

Directory reader

Parent directory (`tracer`)

Directory path: `tracer/reader`

Files

- *File CWrapper.h*
- *File datatypes.h*
- *File otf2_reader.h*
- *File TraceReader.h*

4.3.8 Files

File CWrapper.h

Parent directory (`tracer/reader`)

Contents

- *Definition (`tracer/reader/CWrapper.h`)*
- *Includes*
- *Functions*
- *Typedefs*

Definition (`tracer/reader/CWrapper.h`)

Program Listing for File CWrapper.h

Return to documentation for file (`tracer/reader/CWrapper.h`)

```
// Copyright (c) 2015, Lawrence Livermore National Security, LLC.
// Produced at the Lawrence Livermore National Laboratory.
//
// Written by:
//     Nikhil Jain <nikhil.jain@acm.org>
//     Bilge Acun <acun2@illinois.edu>
//     Abhinav Bhatele <bhatele@llnl.gov>
```

(continues on next page)

(continued from previous page)

```
//
// LLNL-CODE-740483. All rights reserved.
//
// This file is part of TraceR. For details, see:
// https://github.com/LLNL/TraceR
// Please also read the LICENSE file for the MIT License notice.

#ifndef __CWRAPPER_H
#define __CWRAPPER_H

#include <ross.h>
#include "datatypes.h"

//MsgID
typedef struct MsgID MsgID;
MsgID* newMsgID(int size, int pe, int id);
int MsgID_getSize(MsgID* m);
int MsgID_getID(MsgID* m);
int MsgID_getPE(MsgID* m);

//MsgEntry
typedef struct MsgEntry MsgEntry;
MsgEntry* newMsgEntry();
int MsgEntry_getSize(MsgEntry* m);
int MsgEntry_getID(MsgEntry* m);
int MsgEntry_getPE(MsgEntry* m);
int MsgEntry_getNode(MsgEntry* m);
int MsgEntry_getThread(MsgEntry* m);

//PE
typedef struct PE PE;
void PE_set_busy(PE* p, bool b);
bool PE_is_busy(PE* p);
bool PE_noUnsatDep(PE* p, int, int tInd);
bool PE_noMsgDep(PE* p, int, int tInd);
int PE_get_iter(PE* p);
void PE_inc_iter(PE* p);
void PE_dec_iter(PE* p);
double PE_getTaskExecTime(PE* p, int tInd);
void PE_addTaskExecTime(PE* p, int tInd, double time);
#if TRACER_BIGSIM_TRACES
int PE_getTaskMsgEntryCount(PE* p, int tInd);
MsgEntry** PE_getTaskMsgEntries(PE* p, int tInd);
MsgEntry* PE_getTaskMsgEntry(PE* p, int tInd, int mInd);
void PE_execPrintEvt(tw_lp * lp, PE* p, int tInd, double stime);
#endif
void PE_set_taskDone(PE* p, int, int tInd, bool b);
void PE_mark_all_done(PE *p, int iter, int task_id);
bool PE_get_taskDone(PE* p, int, int tInd);
#if TRACER_BIGSIM_TRACES
int* PE_getTaskFwdDep(PE* p, int tInd);
int PE_getTaskFwdDepSize(PE* p, int tInd);
void PE_undone_fwd_deps(PE* p, int iter, int tInd);
#endif
void PE_set_currentTask(PE* p, int tInd);
int PE_get_currentTask(PE* p);
int PE_get_myEmPE(PE* p);
```

(continues on next page)

(continued from previous page)

```
int PE_get_myNum(PE* p);
int PE_getFirstTask(PE* p);
bool PE_isEndEvent(PE *p, int task_id);
bool PE_isLoopEvent(PE *p, int task_id);

int PE_getBufferSize(PE* p);
void PE_clearMsgBuffer(PE* p);
void PE_addToBuffer(PE* p, TaskPair *task_id);
void PE_addToFrontBuffer(PE* p, TaskPair *task_id);
void PE_removeFromBuffer(PE* p, TaskPair *task_id);
void PE_resizeBuffer(PE* p, int num_elems_to_remove);
TaskPair PE_getNextBuffedMsg(PE* p);

int PE_findTaskFromMsg(PE* p, MsgID* msgId);
void PE_invertMsgPe(PE* p, int, int tInd);
int PE_get_tasksCount(PE* p);
int PE_get_totalTasksCount(PE* p);
void PE_printStat(PE* p);
int PE_get_numWorkThreads(PE* p);

#if TRACER_BIGSIM_TRACES
//TraceReader
typedef struct TraceReader TraceReader;
TraceReader* newTraceReader(char*);
void TraceReader_loadTraceSummary(TraceReader* t);
void TraceReader_loadOffsets(TraceReader* t);
int* TraceReader_getOffsets(TraceReader* t);
void TraceReader_setOffsets(TraceReader* t, int** offsets);
void TraceReader_readTrace(TraceReader* t, int* tot, int* numnodes, int* empes,
    int* nwth, PE* pe, int penum, int jobnum, double* startTime);
int TraceReader_totalWorkerProcs(TraceReader* t);
#endif
void addEventSub(int job, char *key, double val, int numjobs);
void addMsgSizeSub(int job, int64_t key, int64_t val, int numjobs);

bool isPEonThisRank(int jobID, int i);
void TraceReader_readOTF2Trace(PE* pe, int my_pe_num, int my_job, double *startTime);
#endif
```

Includes

- `datatypes.h` (*File datatypes.h*)
- `ross.h`

Functions

- *Function addEventSub*
- *Function addMsgSizeSub*
- *Function isPEonThisRank*
- *Function MsgEntry_getID*
- *Function MsgEntry_getNode*

- *Function MsgEntry_getPE*
- *Function MsgEntry_getSize*
- *Function MsgEntry_getThread*
- *Function MsgID_getID*
- *Function MsgID_getPE*
- *Function MsgID_getSize*
- *Function newMsgEntry*
- *Function newMsgID*
- *Function PE_addTaskExecTime*
- *Function PE_addToBuffer*
- *Function PE_addToFrontBuffer*
- *Function PE_clearMsgBuffer*
- *Function PE_dec_iter*
- *Function PE_findTaskFromMsg*
- *Function PE_get_currentTask*
- *Function PE_get_iter*
- *Function PE_get_myEmPE*
- *Function PE_get_myNum*
- *Function PE_get_numWorkThreads*
- *Function PE_get_taskDone*
- *Function PE_get_tasksCount*
- *Function PE_get_totalTasksCount*
- *Function PE_getBufferSize*
- *Function PE_getFirstTask*
- *Function PE_getNextBuffedMsg*
- *Function PE_getTaskExecTime*
- *Function PE_inc_iter*
- *Function PE_invertMsgPe*
- *Function PE_is_busy*
- *Function PE_isEndEvent*
- *Function PE_isLoopEvent*
- *Function PE_mark_all_done*
- *Function PE_noMsgDep*
- *Function PE_noUnsatDep*
- *Function PE_printStat*
- *Function PE_removeFromBuffer*

- *Function PE_resizeBuffer*
- *Function PE_set_busy*
- *Function PE_set_currentTask*
- *Function PE_set_taskDone*
- *Function TraceReader_readOTF2Trace*

Typedefs

- *Typedef MsgEntry*
- *Typedef MsgID*
- *Typedef PE*

File datatypes.h

Parent directory (`tracer/reader`)

Contents

- *Definition (`tracer/reader/datatypes.h`)*
- *Includes*
- *Included By*
- *Classes*
- *Typedefs*

Definition (`tracer/reader/datatypes.h`)

Program Listing for File datatypes.h

Return to documentation for file (`tracer/reader/datatypes.h`)

```
// Copyright (c) 2015, Lawrence Livermore National Security, LLC.
// Produced at the Lawrence Livermore National Laboratory.
//
// Written by:
//     Nikhil Jain <nikhil.jain@acm.org>
//     Bilge Acun <acun2@illinois.edu>
//     Abhinav Bhatele <bhatele@llnl.gov>
//
// LLNL-CODE-740483. All rights reserved.
//
// This file is part of TraceR. For details, see:
// https://github.com/LLNL/TraceR
// Please also read the LICENSE file for the MIT License notice.

#ifndef _DATATYPES_H_
```

(continues on next page)

(continued from previous page)

```
#define _DATATYPES_H_

#if TRACER_OTF_TRACES
#include "otf2_reader.h"
#endif

#include <map>
#include <list>

struct TaskPair {
  int iter;
  int taskid;

#ifdef __cplusplus
  TaskPair(int a, int b) {
    iter = a;
    taskid = b;
  }

  TaskPair() {
    iter = -1;
    taskid = -1;
  }

  TaskPair(const TaskPair &t) {
    iter = t.iter;
    taskid = t.taskid;
  }

  inline bool operator==(const TaskPair &t) {
    return iter == t.iter && taskid == t.taskid;
  }
#endif
};

typedef struct TaskPair TaskPair;

typedef struct JobInf {
    int numRanks;
    char traceDir[256];
    char map_file[256];
    int *rankMap;
    int *offsets;
    int skipMsgId;
    int numIters;
#if TRACER_OTF_TRACES
    AllData *allData;
    OTF2_Reader *reader;
    bool localDefs;
#endif
} JobInf;

#endif
```

Includes

- `list`
- `map`

Included By

- *File PE.h*
- *File CWrapper.h*

Classes

- *Struct JobInf*
- *Struct TaskPair*

Typedefs

- *Typedef JobInf*
- *Typedef TaskPair*

File MsgEntry.h

Parent directory (`tracer/elements`)

Contents

- *Definition (`tracer/elements/MsgEntry.h`)*
- *Includes*
- *Included By*
- *Classes*

Definition (`tracer/elements/MsgEntry.h`)

Program Listing for File MsgEntry.h

Return to documentation for file (`tracer/elements/MsgEntry.h`)

```
// Copyright (c) 2015, Lawrence Livermore National Security, LLC.
// Produced at the Lawrence Livermore National Laboratory.
//
// Written by:
//     Nikhil Jain <nikhil.jain@acm.org>
//     Bilge Acun <acun2@illinois.edu>
//     Abhinav Bhatele <bhatele@llnl.gov>
```

(continues on next page)

(continued from previous page)

```
//
// LLNL-CODE-740483. All rights reserved.
//
// This file is part of TraceR. For details, see:
// https://github.com/LLNL/TraceR
// Please also read the LICENSE file for the MIT License notice.

#ifndef MSGENTRY_H_
#define MSGENTRY_H_

#ifdef __cplusplus
#include <climits>
#endif
#include <stdint.h>

struct MsgID {
    int pe;
    int id;
    uint64_t size;
#ifdef __cplusplus
    MsgID() : pe(INT_MIN), id(0), size(0) {}
    MsgID(int size_) : pe(INT_MIN), id(0), size(size_) {}
    MsgID(int size_, int pe_, int id_) : pe(pe_), id(id_), size(size_) {};
#endif
#if TRACER_OTF_TRACES
    int comm, coll_type;
    int64_t seq;
#endif
};

struct MsgEntry {
#ifdef __cplusplus
    MsgEntry();
#endif
    int node;   // node number in global order
    int thread;
    MsgID msgId;
};

#endif /* MSGENTRY_H_ */
```

Includes

- stdint.h

Included By

- *File PE.h*
- *File Task.h*

Classes

- *Struct MsgEntry*
- *Struct MsgID*

File otf2_reader.h

Parent directory (`tracer/reader`)

Contents

- *Definition (`tracer/reader/otf2_reader.h`)*

Definition (`tracer/reader/otf2_reader.h`)

Program Listing for File otf2_reader.h

Return to documentation for file (`tracer/reader/otf2_reader.h`)

```
// Copyright (c) 2015, Lawrence Livermore National Security, LLC.
// Produced at the Lawrence Livermore National Laboratory.
//
// Written by:
//     Nikhil Jain <nikhil.jain@acm.org>
//     Bilge Acun <acun2@illinois.edu>
//     Abhinav Bhatele <bhatele@llnl.gov>
//
// LLNL-CODE-740483. All rights reserved.
//
// This file is part of TraceR. For details, see:
// https://github.com/LLNL/TraceR
// Please also read the LICENSE file for the MIT License notice.

#ifndef _OTF2_READER_H_
#define _OTF2_READER_H_
#if TRACER_OTF_TRACES

#include <stdlib.h>
#include <stdio.h>
#include <inttypes.h>
#include <mpi.h>
#include <otf2/otf2.h>
#include <vector>
#include <map>
#include <string>
#include "elements/Task.h"

#if MPI_VERSION < 3
#define OTF2_MPI_UINT64_T MPI_UNSIGNED_LONG
#define OTF2_MPI_INT64_T  MPI_LONG
#endif
#include <otf2/OTF2_MPI_Collectives.h>
```

(continues on next page)

(continued from previous page)

```
enum Tracer_evt_type {
  TRACER_USER_EVT = -1,
  TRACER_PRINT_EVT = -2,
  TRACER_SEND_EVT = -3,
  TRACER_RECV_EVT = -4,
  TRACER_COLL_EVT = -5,
  TRACER_SEND_COMP_EVT = -6,
  TRACER_RECV_POST_EVT = -7,
  TRACER_RECV_COMP_EVT = -8,
  TRACER_LOOP_EVT = -9
};

struct ClockProperties {
  uint64_t ticks_per_second;
  double ticksToSecond;
  uint64_t time_offset;
};

struct Region {
  OTF2_StringRef name;
  OTF2_RegionRole role;
  OTF2_Paradigm paradigm;
  bool isTracerPrintEvt;
  bool isLoopEvt;
  bool isCommunication;
};

struct Group {
  OTF2_GroupType type;
  std::vector<uint64_t> members;
  std::map<int, int> rmembers;
};

struct LocationData {
  uint64_t lastLogTime;
  bool firstEnter;
  std::vector<Task> tasks;
};

struct AllData {
  ClockProperties clockProperties;
  std::vector<uint64_t> locations;
  std::map<uint64_t, std::string> strings;
  std::map<uint64_t,uint64_t> communicators;
  std::map<uint64_t,Group> groups;
  std::map<uint64_t,Region> regions;
  LocationData *ld;
  std::map<int, int> matchRecvIds;//temp space
};

OTF2_Reader * readGlobalDefinitions(int jobID, char* tracefileName,
  AllData *allData);

void readLocationTasks(int jobID, OTF2_Reader *reader, AllData *allData,
  uint32_t loc, LocationData* ld);
```

(continues on next page)

(continued from previous page)

```
void closeReader(OTF2_Reader *reader);
#endif
#endif
```

File PE.h

Parent directory (`tracer/elements`)

Contents

- *Definition (tracer/elements/PE.h)*
- *Includes*
- *Included By*
- *Classes*
- *Typedefs*

Definition (`tracer/elements/PE.h`)

Program Listing for File PE.h

Return to documentation for file (`tracer/elements/PE.h`)

```
// Copyright (c) 2015, Lawrence Livermore National Security, LLC.
// Produced at the Lawrence Livermore National Laboratory.
//
// Written by:
//     Nikhil Jain <nikhil.jain@acm.org>
//     Bilge Acun <acun2@illinois.edu>
//     Abhinav Bhatele <bhatele@llnl.gov>
//
// LLNL-CODE-740483. All rights reserved.
//
// This file is part of TraceR. For details, see:
// https://github.com/LLNL/TraceR
// Please also read the LICENSE file for the MIT License notice.

#ifndef PE_H_
#define PE_H_

#include "MsgEntry.h"
#include <cstring>
#include "Task.h"
#include <list>
#include <map>
#include <vector>
#include "reader/datatypes.h"

class Task;
```

(continues on next page)

(continued from previous page)

```
class MsgKey {
  public:
  uint32_t rank, comm, tag;
  int64_t seq;
  MsgKey(uint32_t _rank, uint32_t _tag, uint32_t _comm, int64_t _seq) {
    rank = _rank; tag = _tag; comm = _comm; seq = _seq;
  }
  bool operator< (const MsgKey &rhs) const {
    if(rank != rhs.rank) return rank < rhs.rank;
    else if(tag != rhs.tag) return tag < rhs.tag;
    else return comm < rhs.comm;
    //else if(comm != rhs.comm) return comm < rhs.comm;
    //else return seq < rhs.seq;
  }
  ~MsgKey() { }
};
typedef std::map< MsgKey, std::list<int> > KeyType;

class CollMsgKey {
  public:
  uint32_t rank, comm;
  int64_t seq;
  CollMsgKey(uint32_t _rank, uint32_t _comm, int64_t _seq) {
    rank = _rank; comm = _comm; seq = _seq;
  }
  bool operator< (const CollMsgKey &rhs) const {
    if(rank != rhs.rank) return rank < rhs.rank;
    else if(comm != rhs.comm) return comm < rhs.comm;
    else return seq < rhs.seq;
  }
  ~CollMsgKey() { }
};
typedef std::map< CollMsgKey, std::list<int> > CollKeyType;

class PE {
  public:
    PE();
    ~PE();
    std::list<TaskPair> msgBuffer;
    Task* myTasks;  // all tasks of this PE
    bool **taskStatus, **taskExecuted;
    bool **msgStatus;
    bool *allMarked;
    double currTime;
    bool busy;
    int beforeTask, totalTasksCount;
    int myNum, myEmPE, jobNum;
    int tasksCount; //total number of tasks
    int currentTask; // index of first not-executed task (helps searching messages)
    int firstTask;
    int currIter;
    int loop_start_task;

    bool noUnsatDep(int iter, int tInd);    // there is no unsatisfied dependency for␣
↪task
    void mark_all_done(int iter, int tInd);
    double taskExecTime(int tInd);
```

(continues on next page)

(continued from previous page)

```
    void printStat();
    void check();
    void printState();

    void invertMsgPe(int iter, int tInd);
    double getTaskExecTime(int tInd);
    void addTaskExecTime(int tInd, double time);
    std::map<int, int>* msgDestLogs;
    int findTaskFromMsg(MsgID* msg);
    int numWth, numEmPes;

    KeyType pendingMsgs;
    KeyType pendingRMsgs;
    int64_t *sendSeq, *recvSeq;
    std::map<int, int> pendingReqs;
    std::map<int, int64_t> pendingRReqs;

    //handling collectives
    std::vector<int64_t> collectiveSeq;
    std::map<int64_t, std::map<int64_t, std::map<int, int> > > pendingCollMsgs;
    CollKeyType pendingRCollMsgs;
    int64_t currentCollComm, currentCollSeq, currentCollTask, currentCollMsgSize;
    int currentCollRank, currentCollPartner, currentCollSize;
    int currentCollSendCount, currentCollRecvCount;
};

#endif /* PE_H_ */
```

Includes

- `MsgEntry.h` (*File MsgEntry.h*)
- `Task.h` (*File Task.h*)
- `cstring`
- `list`
- `map`
- `reader/datatypes.h` (*File datatypes.h*)
- `vector`

Included By

- *File TraceReader.h*

Classes

- *Class CollMsgKey*
- *Class MsgKey*
- *Class PE*

Typedefs

- *Typedef CollKeyType*
- *Typedef KeyType*

File Task.h

Parent directory (`tracer/elements`)

Contents

- *Definition (`tracer/elements/Task.h`)*
- *Includes*
- *Included By*
- *Classes*
- *Defines*

Definition (`tracer/elements/Task.h`)

Program Listing for File Task.h

Return to documentation for file (`tracer/elements/Task.h`)

```
// Copyright (c) 2015, Lawrence Livermore National Security, LLC.
// Produced at the Lawrence Livermore National Laboratory.
//
// Written by:
//     Nikhil Jain <nikhil.jain@acm.org>
//     Bilge Acun <acun2@illinois.edu>
//     Abhinav Bhatele <bhatele@llnl.gov>
//
// LLNL-CODE-740483. All rights reserved.
//
// This file is part of TraceR. For details, see:
// https://github.com/LLNL/TraceR
// Please also read the LICENSE file for the MIT License notice.

#ifndef TASK_H_
#define TASK_H_
#include "MsgEntry.h"
#include <cstdlib>
#include <cstdio>
#if TRACER_BIGSIM_TRACES
#include <mpi.h>
#include <ross.h>
#endif

class MsgEntry;
#include <cstring>
```

(continues on next page)

(continued from previous page)

```
#define TIME_MULT 1000000000

#if TRACER_BIGSIM_TRACES
class BgPrint{
  public:
    void print(tw_lp * lp, double startTime, int PEno, int jobNo)
    {
      char str[1000];
      strcpy(str, "[%d %d : %s] ");
      strcat(str, msg);
      tw_output(lp, str, jobNo, PEno, taskName, startTime/((double)TIME_MULT));
    }
    char* msg;
    double time;
    char taskName[50];
};
#endif

// represents each DEP ~ SEB
class Task {
  public:
    Task();
    ~Task();
#if TRACER_BIGSIM_TRACES
    void printEvt(tw_lp * lp, double startTime, int PEno, int jobNo);
    int msgEntCount; // number of msg entries
    MsgEntry* myEntries; // outgoing messages of task
    int* forwardDep; //backward dependent tasks
    int forwDepSize;     // size of forwardDep array

    int* backwardDep;    //forward dependent tasks
    int backwDepSize;    // size of backwDep array
    int bgPrintCount;
    BgPrint* myBgPrints;
#elif TRACER_OTF_TRACES
    int64_t event_id;
    int64_t req_id;
    bool isNonBlocking;
    MsgEntry myEntry;
    bool beginEvent;
#else
#error Either TRACER_BIGSIM_TRACES or TRACER_OTF_TRACES should be 1
#endif
    bool endEvent;
    bool loopEvent, loopStartEvent;
    double execTime;     //execution time of the task
};

#endif /* TASK_H_ */
```

Includes

- `MsgEntry.h` (*File MsgEntry.h*)
- `cstdio`

- cstdlib
- cstring

Included By

- *File PE.h*
- *File TraceReader.h*

Classes

- *Class Task*

Defines

- *Define TIME_MULT*

File tracer-driver.h

Parent directory (tracer)

Contents

- *Definition (tracer/tracer-driver.h)*
- *Detailed Description*
- *Includes*
- *Classes*
- *Enums*
- *Functions*
- *Defines*
- *Typedefs*
- *Variables*

Definition (tracer/tracer-driver.h)

Program Listing for File tracer-driver.h

Return to documentation for file (tracer/tracer-driver.h)

```
#ifndef _TRACER_DRIVER_H_
#define _TRACER_DRIVER_H_

#include "reader/datatypes.h"
```

(continues on next page)

(continued from previous page)

```
#include "reader/CWrapper.h"
#include "elements/MsgEntry.h"
#include "elements/PE.h"

#if TRACER_OTF_TRACES
#include "reader/otf2_reader.h"
#endif

#define BCAST_DEGREE  2
#define REDUCE_DEGREE  2

#define TRACER_A2A_ALG_CUTOFF 512
#define TRACER_ALLGATHER_ALG_CUTOFF 163840
#define TRACER_BLOCK_SIZE 32
#define MPI_INTERNAL_DELAY 10
#define TRACER_SCATTER_ALG_CUTOFF 0

/* stores mapping of core to job ID and process ID */
typedef struct CoreInf {
    int mapsTo, jobID;
} CoreInf;

/* ROSS level state information for each core */
struct proc_state
{
    tw_stime start_ts;  /* time when first event is processed */
    tw_stime end_ts;    /* time when last event is processed */
    PE* my_pe;          /* stores all core information */
#if TRACER_BIGSIM_TRACES
    TraceReader* trace_reader; /* for reading the bigsim traces */
#endif
    clock_t sim_start;  /* clock time when simulation starts */
    int my_pe_num, my_job;
};

extern JobInf *jobs;
extern tw_stime soft_delay_mpi;
extern tw_stime nic_delay;
extern tw_stime rdma_delay;

extern int net_id;
extern unsigned int print_frequency;
extern double copy_per_byte;
extern double eager_limit;


/* types of events that will constitute ROSS event requests */
enum proc_event
{
    KICKOFF=1,          /* initial event */
    LOCAL,              /* local event */
    RECV_MSG,           /* receive a message */
    BCAST,              /* broadcast --> to be deprecated */
    EXEC_COMPLETE,      /* marks completion of task */
    SEND_COMP,          /* send completed */
    RECV_POST,          /* Message from receiver that the recv is posted */
    COLL_BCAST,         /* Collective impl for bcast */
```

(continues on next page)

(continued from previous page)

```
    COLL_REDUCTION,      /* Collective impl for reduction */
    COLL_A2A,            /* Collective impl for a2a */
    COLL_A2A_SEND_DONE,
    COLL_ALLGATHER,      /* Collective impl for allgather */
    COLL_ALLGATHER_SEND_DONE,
    COLL_BRUCK,          /* event used by Bruck implementation */
    COLL_BRUCK_SEND_DONE,
    COLL_A2A_BLOCKED,    /* event used by blocked A2A implementation */
    COLL_A2A_BLOCKED_SEND_DONE,
    COLL_SCATTER_SMALL, /* scatter event for small messages */
    COLL_SCATTER,        /* scatter event */
    COLL_SCATTER_SEND_DONE,
    RECV_COLL_POST,      /* Message from receiver that a recv for collective is posted␣
↪*/
    COLL_COMPLETE        /* collective completion event */
};

/* Tracer's part of the ROSS message */
struct proc_msg
{
    enum proc_event proc_event_type;
    tw_lpid src;             /* source of this event */
    int iteration;           /* iteration number when repeating traces */
    TaskPair executed;       /* task related to this event */
    int fwd_dep_count;       /* number of tasks dependent on the source task */
    int saved_task;          /* which task was acted on (for REV_HDL) */
    MsgID msgId;             /* message ID */
    bool incremented_flag;   /* core status (for REV_HDL) */
    int model_net_calls;     /* number of model_net calls (for REV_HDL) */
    unsigned int coll_info, coll_info_2; /* collective info */
};

/* Collective routine type */
enum tracer_coll_type
{
  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
};


/* pairs up a local and remote event for collective */
struct Coll_lookup {
  proc_event remote_event, local_event;
};

/* core info to/from ROSS LP */
int pe_to_lpid(int pe, int job);
int pe_to_job(int pe);
int lpid_to_pe(int lp_gid);
int lpid_to_job(int lp_gid);
```

(continues on next page)

(continued from previous page)

```
/* change of units for time */
tw_stime ns_to_s(tw_stime ns);
tw_stime s_to_ns(tw_stime ns);

void proc_init(
    proc_state * ns,
    tw_lp * lp);
void proc_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);
void proc_rev_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);
void proc_finalize(
    proc_state * ns,
    tw_lp * lp);

//event handler declarations
void handle_kickoff_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);
void handle_local_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
   tw_lp * lp);
void handle_recv_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
   tw_lp * lp);
void handle_bcast_event( /* to be deprecated */
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
   tw_lp * lp);
void handle_exec_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
   tw_lp * lp);
void handle_send_comp_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
   tw_lp * lp);
void handle_a2a_send_comp_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
```

(continues on next page)

(continued from previous page)

```
   tw_lp * lp);
void handle_allgather_send_comp_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
   tw_lp * lp);
void handle_bruck_send_comp_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
   tw_lp * lp);
void handle_a2a_blocked_send_comp_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
   tw_lp * lp);
void handle_scatter_send_comp_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
   tw_lp * lp);
void handle_recv_post_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
   tw_lp * lp);

//reverse event handler declarations
void handle_kickoff_rev_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);
void handle_local_rev_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
   tw_lp * lp);
void handle_recv_rev_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);
void handle_bcast_rev_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);
void handle_exec_rev_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);
void handle_send_comp_rev_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
```

(continues on next page)

(continued from previous page)

```
    tw_lp * lp);
void handle_a2a_send_comp_rev_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);
void handle_allgather_send_comp_rev_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);
void handle_bruck_send_comp_rev_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);
void handle_a2a_blocked_send_comp_rev_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);
void handle_scatter_send_comp_rev_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);
void handle_recv_post_rev_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);

tw_stime exec_task(
    proc_state * ns,
    TaskPair task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf *b);

void exec_task_rev(
    proc_state * ns,
    TaskPair task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf *b);

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,
```

(continues on next page)

(continued from previous page)

```
    int64_t size2 = 0);

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);

void delegate_send_msg(
    proc_state *ns,
    tw_lp * lp,
    proc_msg * m,
    tw_bf * b,
    Task * t,
    int taskid,
    tw_stime delay);

int bcast_msg(
    proc_state * ns,
    int size,
    int iter,
    MsgID *msgId,
    tw_stime timeOffset,
    tw_stime copyTime,
    tw_lp * lp,
    proc_msg *m);

int exec_comp(
    proc_state * ns,
    int iter,
    int task_id,
    int comm_id,
    tw_stime sendOffset,
    int recv,
    tw_lp * lp);

void perform_collective(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b);

void perform_bcast(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);
```

(continues on next page)

(continued from previous page)

```
void perform_reduction(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void perform_a2a(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void perform_allreduce(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void perform_allgather(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void perform_bruck(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void perform_a2a_blocked(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void perform_scatter_small(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void perform_scatter(
```

(continues on next page)

(continued from previous page)

```
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void handle_coll_recv_post_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);

void handle_coll_complete_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);

int send_coll_comp(
    proc_state * ns,
    tw_stime sendOffset,
    int collType,
    tw_lp * lp,
    int isEvent,
    proc_msg * m);

void perform_collective_rev(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b);

void perform_bcast_rev(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void perform_reduction_rev(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void perform_a2a_rev(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);
```

(continues on next page)

(continued from previous page)

```
void perform_allreduce_rev(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void perform_allgather_rev(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void perform_bruck_rev(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void perform_a2a_blocked_rev(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void perform_scatter_small_rev(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void perform_scatter_rev(
    proc_state * ns,
    int task_id,
    tw_lp * lp,
    proc_msg *m,
    tw_bf * b,
    int isEvent);

void handle_coll_recv_post_rev_event(
    proc_state * ns,
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);

void handle_coll_complete_rev_event(
    proc_state * ns,
```

(continues on next page)

(continued from previous page)

```
    tw_bf * b,
    proc_msg * m,
    tw_lp * lp);

int send_coll_comp_rev(
    proc_state * ns,
    tw_stime sendOffset,
    int collType,
    tw_lp * lp,
    int isEvent,
    proc_msg * m);


#endif
```

Detailed Description

Includes

- `elements/MsgEntry.h` (*File MsgEntry.h*)
- `elements/PE.h` (*File PE.h*)
- `reader/CWrapper.h` (*File CWrapper.h*)
- `reader/datatypes.h` (*File datatypes.h*)

Classes

- *Struct Coll_lookup*
- *Struct CoreInf*
- *Struct proc_msg*
- *Struct proc_state*

Enums

- *Enum proc_event*
- *Enum tracer_coll_type*

Functions

- *Function bcast_msg*

- *Function delegate_send_msg*
- *Function enqueue_msg*
- *Function exec_comp*
- *Function exec_task*
- *Function exec_task_rev*
- *Function handle_a2a_blocked_send_comp_event*
- *Function handle_a2a_blocked_send_comp_rev_event*
- *Function handle_a2a_send_comp_event*
- *Function handle_a2a_send_comp_rev_event*
- *Function handle_allgather_send_comp_event*
- *Function handle_allgather_send_comp_rev_event*
- *Function handle_bcast_event*
- *Function handle_bcast_rev_event*
- *Function handle_bruck_send_comp_event*
- *Function handle_bruck_send_comp_rev_event*
- *Function handle_coll_complete_event*
- *Function handle_coll_complete_rev_event*
- *Function handle_coll_recv_post_event*
- *Function handle_coll_recv_post_rev_event*
- *Function handle_exec_event*
- *Function handle_exec_rev_event*
- *Function handle_kickoff_event*
- *Function handle_kickoff_rev_event*
- *Function handle_local_event*
- *Function handle_local_rev_event*
- *Function handle_recv_event*
- *Function handle_recv_post_event*
- *Function handle_recv_post_rev_event*
- *Function handle_recv_rev_event*
- *Function handle_scatter_send_comp_event*
- *Function handle_scatter_send_comp_rev_event*
- *Function handle_send_comp_event*
- *Function handle_send_comp_rev_event*
- *Function lpid_to_job*
- *Function lpid_to_pe*
- *Function ns_to_s*

- *Function pe_to_job*
- *Function pe_to_lpid*
- *Function perform_a2a*
- *Function perform_a2a_blocked*
- *Function perform_a2a_blocked_rev*
- *Function perform_a2a_rev*
- *Function perform_allgather*
- *Function perform_allgather_rev*
- *Function perform_allreduce*
- *Function perform_allreduce_rev*
- *Function perform_bcast*
- *Function perform_bcast_rev*
- *Function perform_bruck*
- *Function perform_bruck_rev*
- *Function perform_collective*
- *Function perform_collective_rev*
- *Function perform_reduction*
- *Function perform_reduction_rev*
- *Function perform_scatter*
- *Function perform_scatter_rev*
- *Function perform_scatter_small*
- *Function perform_scatter_small_rev*
- *Function proc_event*
- *Function proc_finalize*
- *Function proc_init*
- *Function proc_rev_event*
- *Function s_to_ns*
- *Function send_coll_comp*
- *Function send_coll_comp_rev*
- *Function send_msg*

Defines

- *Define BCAST_DEGREE*
- *Define MPI_INTERNAL_DELAY*
- *Define REDUCE_DEGREE*
- *Define TRACER_A2A_ALG_CUTOFF*

- *Define TRACER_ALLGATHER_ALG_CUTOFF*
- *Define TRACER_BLOCK_SIZE*
- *Define TRACER_SCATTER_ALG_CUTOFF*

Typedefs

- *Typedef CoreInf*

Variables

- *Variable copy_per_byte*
- *Variable eager_limit*
- *Variable jobs*
- *Variable net_id*
- *Variable nic_delay*
- *Variable print_frequency*
- *Variable rdma_delay*
- *Variable soft_delay_mpi*

File TraceReader.h

Parent directory (`tracer/reader`)

Contents

- *Definition (`tracer/reader/TraceReader.h`)*
- *Includes*
- *Classes*

Definition (`tracer/reader/TraceReader.h`)

Program Listing for File TraceReader.h

Return to documentation for file (`tracer/reader/TraceReader.h`)

```
// Copyright (c) 2015, Lawrence Livermore National Security, LLC.
// Produced at the Lawrence Livermore National Laboratory.
//
// Written by:
//     Nikhil Jain <nikhil.jain@acm.org>
//     Bilge Acun <acun2@illinois.edu>
//     Abhinav Bhatele <bhatele@llnl.gov>
//
```

(continues on next page)

(continued from previous page)

```
// LLNL-CODE-740483. All rights reserved.
//
// This file is part of TraceR. For details, see:
// https://github.com/LLNL/TraceR
// Please also read the LICENSE file for the MIT License notice.

#ifndef TRACEFILEREADER_H_
#define TRACEFILEREADER_H_
#include "assert.h"
#if TRACER_BIGSIM_TRACES
#include "blue.h"
#include "blue_impl.h"
#endif
#include "elements/PE.h"
#include "elements/Task.h"
class PE;
class Node;
class Task;

class TraceReader {
public:
    TraceReader(char *);
    ~TraceReader();
#if TRACER_BIGSIM_TRACES
    void loadOffsets();
    void loadTraceSummary();
    void readTrace(int* tot, int* numnodes, int* empes, int* nwth, PE* pe,
        int penum, int jobnum, double* startTime);
    void setTaskFromLog(Task *t, BgTimeLog* bglog, int taskPE, int emPE, int␣
↪jobPEindex, PE* pe, int, bool, double);
#endif

    int numEmPes;   // number of emulation PEs, there is a trace file for each of them
    int totalWorkerProcs;
    int totalNodes;
    int numWth; //Working PEs per node
    int* allNodeOffsets;
    char tracePath[256];

    int fileLoc; // each worker needs separate file offset
    int firstLog; // first log of window to read for each worker
    int totalTlineLength; // apparently totalTlineLength should be kept for each PE!
};

#endif /* TRACEFILEREADER_H_ */
```

Includes

- assert.h
- elements/PE.h (*File PE.h*)
- elements/Task.h (*File Task.h*)

Classes

- *Class TraceReader*

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

A

addEventSub (*C++ function*), 19
addMsgSizeSub (*C++ function*), 19

B

BCAST (*C++ enumerator*), 18
BCAST_DEGREE (*C macro*), 41
bcast_msg (*C++ function*), 20

C

COLL_A2A (*C++ enumerator*), 18
COLL_A2A_BLOCKED (*C++ enumerator*), 18
COLL_A2A_BLOCKED_SEND_DONE (*C++ enumerator*), 18
COLL_A2A_SEND_DONE (*C++ enumerator*), 18
COLL_ALLGATHER (*C++ enumerator*), 18
COLL_ALLGATHER_SEND_DONE (*C++ enumerator*), 18
COLL_BCAST (*C++ enumerator*), 18
COLL_BRUCK (*C++ enumerator*), 18
COLL_BRUCK_SEND_DONE (*C++ enumerator*), 18
COLL_COMPLETE (*C++ enumerator*), 19
Coll_lookup (*C++ class*), 11
Coll_lookup::local_event (*C++ member*), 11
Coll_lookup::remote_event (*C++ member*), 11
COLL_REDUCTION (*C++ enumerator*), 18
COLL_SCATTER (*C++ enumerator*), 19
COLL_SCATTER_SEND_DONE (*C++ enumerator*), 19
COLL_SCATTER_SMALL (*C++ enumerator*), 18
CollKeyType (*C++ type*), 43
CollMsgKey (*C++ class*), 14
CollMsgKey::~CollMsgKey (*C++ function*), 14
CollMsgKey::CollMsgKey (*C++ function*), 14
CollMsgKey::comm (*C++ member*), 14
CollMsgKey::operator< (*C++ function*), 14
CollMsgKey::rank (*C++ member*), 14
CollMsgKey::seq (*C++ member*), 14
copy_per_byte (*C++ member*), 40
CoreInf (*C++ class*), 11
CoreInf (*C++ type*), 43
CoreInf::jobID (*C++ member*), 11
CoreInf::mapsTo (*C++ member*), 11

D

delegate_send_msg (*C++ function*), 20

E

eager_limit (*C++ member*), 40
enqueue_msg (*C++ function*), 20
exec_comp (*C++ function*), 20
EXEC_COMPLETE (*C++ enumerator*), 18
exec_task (*C++ function*), 20
exec_task_rev (*C++ function*), 21

H

handle_a2a_blocked_send_comp_event (*C++ function*), 21
handle_a2a_blocked_send_comp_rev_event (*C++ function*), 21
handle_a2a_send_comp_event (*C++ function*), 21
handle_a2a_send_comp_rev_event (*C++ function*), 21
handle_allgather_send_comp_event (*C++ function*), 21
handle_allgather_send_comp_rev_event (*C++ function*), 22
handle_bcast_event (*C++ function*), 22
handle_bcast_rev_event (*C++ function*), 22
handle_bruck_send_comp_event (*C++ function*), 22
handle_bruck_send_comp_rev_event (*C++ function*), 22
handle_coll_complete_event (*C++ function*), 23
handle_coll_complete_rev_event (*C++ function*), 23
handle_coll_recv_post_event (*C++ function*), 23
handle_coll_recv_post_rev_event (*C++ function*), 23
handle_exec_event (*C++ function*), 23
handle_exec_rev_event (*C++ function*), 23
handle_kickoff_event (*C++ function*), 24
handle_kickoff_rev_event (*C++ function*), 24

handle_local_event (C++ function), 24
handle_local_rev_event (C++ function), 24
handle_recv_event (C++ function), 24
handle_recv_post_event (C++ function), 25
handle_recv_post_rev_event (C++ function), 25
handle_recv_rev_event (C++ function), 25
handle_scatter_send_comp_event (C++ function), 25
handle_scatter_send_comp_rev_event (C++ function), 25
handle_send_comp_event (C++ function), 25
handle_send_comp_rev_event (C++ function), 26

I

isPEonThisRank (C++ function), 26

J

JobInf (C++ class), 12
JobInf (C++ type), 43
JobInf::map_file (C++ member), 12
JobInf::numIters (C++ member), 12
JobInf::numRanks (C++ member), 12
JobInf::offsets (C++ member), 12
JobInf::rankMap (C++ member), 12
JobInf::skipMsgId (C++ member), 12
JobInf::traceDir (C++ member), 12
jobs (C++ member), 40

K

KeyType (C++ type), 43
KICKOFF (C++ enumerator), 18

L

LOCAL (C++ enumerator), 18
lpid_to_job (C++ function), 26
lpid_to_pe (C++ function), 26

M

MPI_INTERNAL_DELAY (C macro), 41
MsgEntry (C++ class), 12
MsgEntry (C++ type), 43
MsgEntry::msgId (C++ member), 12
MsgEntry::node (C++ member), 12
MsgEntry::thread (C++ member), 12
MsgEntry_getID (C++ function), 26
MsgEntry_getNode (C++ function), 27
MsgEntry_getPE (C++ function), 27
MsgEntry_getSize (C++ function), 27
MsgEntry_getThread (C++ function), 27
MsgID (C++ class), 12
MsgID (C++ type), 44
MsgID::id (C++ member), 13
MsgID::pe (C++ member), 13
MsgID::size (C++ member), 13
MsgID_getID (C++ function), 27
MsgID_getPE (C++ function), 27
MsgID_getSize (C++ function), 28
MsgKey (C++ class), 14
MsgKey::~MsgKey (C++ function), 15
MsgKey::comm (C++ member), 15
MsgKey::MsgKey (C++ function), 15
MsgKey::operator< (C++ function), 15
MsgKey::rank (C++ member), 15
MsgKey::seq (C++ member), 15
MsgKey::tag (C++ member), 15

N

net_id (C++ member), 40
newMsgEntry (C++ function), 28
newMsgID (C++ function), 28
nic_delay (C++ member), 41
ns_to_s (C++ function), 28

P

PE (C++ class), 15
PE (C++ type), 44
PE::~PE (C++ function), 15
PE::addTaskExecTime (C++ function), 15
PE::allMarked (C++ member), 16
PE::beforeTask (C++ member), 16
PE::busy (C++ member), 16
PE::check (C++ function), 15
PE::collectiveSeq (C++ member), 16
PE::currentCollComm (C++ member), 16
PE::currentCollMsgSize (C++ member), 16
PE::currentCollPartner (C++ member), 16
PE::currentCollRank (C++ member), 16
PE::currentCollRecvCount (C++ member), 17
PE::currentCollSendCount (C++ member), 17
PE::currentCollSeq (C++ member), 16
PE::currentCollSize (C++ member), 17
PE::currentCollTask (C++ member), 16
PE::currentTask (C++ member), 16
PE::currIter (C++ member), 16
PE::currTime (C++ member), 16
PE::findTaskFromMsg (C++ function), 15
PE::firstTask (C++ member), 16
PE::getTaskExecTime (C++ function), 15
PE::invertMsgPe (C++ function), 15
PE::jobNum (C++ member), 16
PE::loop_start_task (C++ member), 16
PE::mark_all_done (C++ function), 15
PE::msgBuffer (C++ member), 16
PE::msgDestLogs (C++ member), 16
PE::msgStatus (C++ member), 16

PE::myEmPE (C++ member), 16
PE::myNum (C++ member), 16
PE::myTasks (C++ member), 16
PE::noUnsatDep (C++ function), 15
PE::numEmPes (C++ member), 16
PE::numWth (C++ member), 16
PE::PE (C++ function), 15
PE::pendingCollMsgs (C++ member), 16
PE::pendingMsgs (C++ member), 16
PE::pendingRCollMsgs (C++ member), 16
PE::pendingReqs (C++ member), 16
PE::pendingRMsgs (C++ member), 16
PE::pendingRReqs (C++ member), 16
PE::printStat (C++ function), 15
PE::printState (C++ function), 15
PE::recvSeq (C++ member), 16
PE::sendSeq (C++ member), 16
PE::taskExecTime (C++ function), 15
PE::taskExecuted (C++ member), 16
PE::tasksCount (C++ member), 16
PE::taskStatus (C++ member), 16
PE::totalTasksCount (C++ member), 16
PE_addTaskExecTime (C++ function), 28
PE_addToBuffer (C++ function), 29
PE_addToFrontBuffer (C++ function), 29
PE_clearMsgBuffer (C++ function), 29
PE_dec_iter (C++ function), 29
PE_findTaskFromMsg (C++ function), 29
PE_get_currentTask (C++ function), 29
PE_get_iter (C++ function), 30
PE_get_myEmPE (C++ function), 30
PE_get_myNum (C++ function), 30
PE_get_numWorkThreads (C++ function), 30
PE_get_taskDone (C++ function), 30
PE_get_tasksCount (C++ function), 31
PE_get_totalTasksCount (C++ function), 31
PE_getBufferSize (C++ function), 31
PE_getFirstTask (C++ function), 31
PE_getNextBuffedMsg (C++ function), 31
PE_getTaskExecTime (C++ function), 31
PE_inc_iter (C++ function), 32
PE_invertMsgPe (C++ function), 32
PE_is_busy (C++ function), 32
PE_isEndEvent (C++ function), 32
PE_isLoopEvent (C++ function), 32
PE_mark_all_done (C++ function), 33
PE_noMsgDep (C++ function), 33
PE_noUnsatDep (C++ function), 33
PE_printStat (C++ function), 33
PE_removeFromBuffer (C++ function), 33
PE_resizeBuffer (C++ function), 33
PE_set_busy (C++ function), 34
PE_set_currentTask (C++ function), 34
PE_set_taskDone (C++ function), 34
pe_to_job (C++ function), 34
pe_to_lpid (C++ function), 34
perform_a2a (C++ function), 35
perform_a2a_blocked (C++ function), 35
perform_a2a_blocked_rev (C++ function), 35
perform_a2a_rev (C++ function), 35
perform_allgather (C++ function), 35
perform_allgather_rev (C++ function), 35
perform_allreduce (C++ function), 36
perform_allreduce_rev (C++ function), 36
perform_bcast (C++ function), 36
perform_bcast_rev (C++ function), 36
perform_bruck (C++ function), 36
perform_bruck_rev (C++ function), 37
perform_collective (C++ function), 37
perform_collective_rev (C++ function), 37
perform_reduction (C++ function), 37
perform_reduction_rev (C++ function), 37
perform_scatter (C++ function), 37
perform_scatter_rev (C++ function), 38
perform_scatter_small (C++ function), 38
perform_scatter_small_rev (C++ function), 38
print_frequency (C++ member), 41
proc_event (C++ enum), 18
proc_event (C++ function), 38
proc_finalize (C++ function), 38
proc_init (C++ function), 39
proc_msg (C++ class), 13
proc_msg::coll_info (C++ member), 13
proc_msg::coll_info_2 (C++ member), 13
proc_msg::executed (C++ member), 13
proc_msg::fwd_dep_count (C++ member), 13
proc_msg::incremented_flag (C++ member), 13
proc_msg::iteration (C++ member), 13
proc_msg::model_net_calls (C++ member), 13
proc_msg::msgId (C++ member), 13
proc_msg::proc_event_type (C++ member), 13
proc_msg::saved_task (C++ member), 13
proc_msg::src (C++ member), 13
proc_rev_event (C++ function), 39
proc_state (C++ class), 13
proc_state::end_ts (C++ member), 13
proc_state::my_job (C++ member), 14
proc_state::my_pe (C++ member), 13
proc_state::my_pe_num (C++ member), 13
proc_state::sim_start (C++ member), 13
proc_state::start_ts (C++ member), 13

R

rdma_delay (C++ member), 41
RECV_COLL_POST (C++ enumerator), 19
RECV_MSG (C++ enumerator), 18
RECV_POST (C++ enumerator), 18

REDUCE_DEGREE (*C macro*), 42

S

s_to_ns (*C++ function*), 39
send_coll_comp (*C++ function*), 39
send_coll_comp_rev (*C++ function*), 39
SEND_COMP (*C++ enumerator*), 18
send_msg (*C++ function*), 39
soft_delay_mpi (*C++ member*), 41

T

Task (*C++ class*), 17
Task::~Task (*C++ function*), 17
Task::endEvent (*C++ member*), 17
Task::execTime (*C++ member*), 17
Task::loopEvent (*C++ member*), 17
Task::loopStartEvent (*C++ member*), 17
Task::Task (*C++ function*), 17
TaskPair (*C++ class*), 14
TaskPair (*C++ type*), 44
TaskPair::iter (*C++ member*), 14
TaskPair::taskid (*C++ member*), 14
TIME_MULT (*C macro*), 42
TRACER_A2A_ALG_CUTOFF (*C macro*), 42
TRACER_ALLGATHER_ALG_CUTOFF (*C macro*), 42
TRACER_BLOCK_SIZE (*C macro*), 42
tracer_coll_type (*C++ enum*), 19
TRACER_COLLECTIVE_ALL_BRUCK (*C++ enumerator*), 19
TRACER_COLLECTIVE_ALLGATHER_LARGE (*C++ enumerator*), 19
TRACER_COLLECTIVE_ALLTOALL_BLOCKED (*C++ enumerator*), 19
TRACER_COLLECTIVE_ALLTOALL_LARGE (*C++ enumerator*), 19
TRACER_COLLECTIVE_BARRIER (*C++ enumerator*), 19
TRACER_COLLECTIVE_BCAST (*C++ enumerator*), 19
TRACER_COLLECTIVE_REDUCE (*C++ enumerator*), 19
TRACER_COLLECTIVE_SCATTER (*C++ enumerator*), 19
TRACER_COLLECTIVE_SCATTER_SMALL (*C++ enumerator*), 19
TRACER_SCATTER_ALG_CUTOFF (*C macro*), 43
TraceReader (*C++ class*), 17
TraceReader::~TraceReader (*C++ function*), 17
TraceReader::allNodeOffsets (*C++ member*), 18
TraceReader::fileLoc (*C++ member*), 18
TraceReader::firstLog (*C++ member*), 18
TraceReader::numEmPes (*C++ member*), 18
TraceReader::numWth (*C++ member*), 18
TraceReader::totalNodes (*C++ member*), 18
TraceReader::totalTlineLength (*C++ member*), 18
TraceReader::totalWorkerProcs (*C++ member*), 18
TraceReader::tracePath (*C++ member*), 18
TraceReader::TraceReader (*C++ function*), 17
TraceReader_readOTF2Trace (*C++ function*), 40