SAND2010-3351P

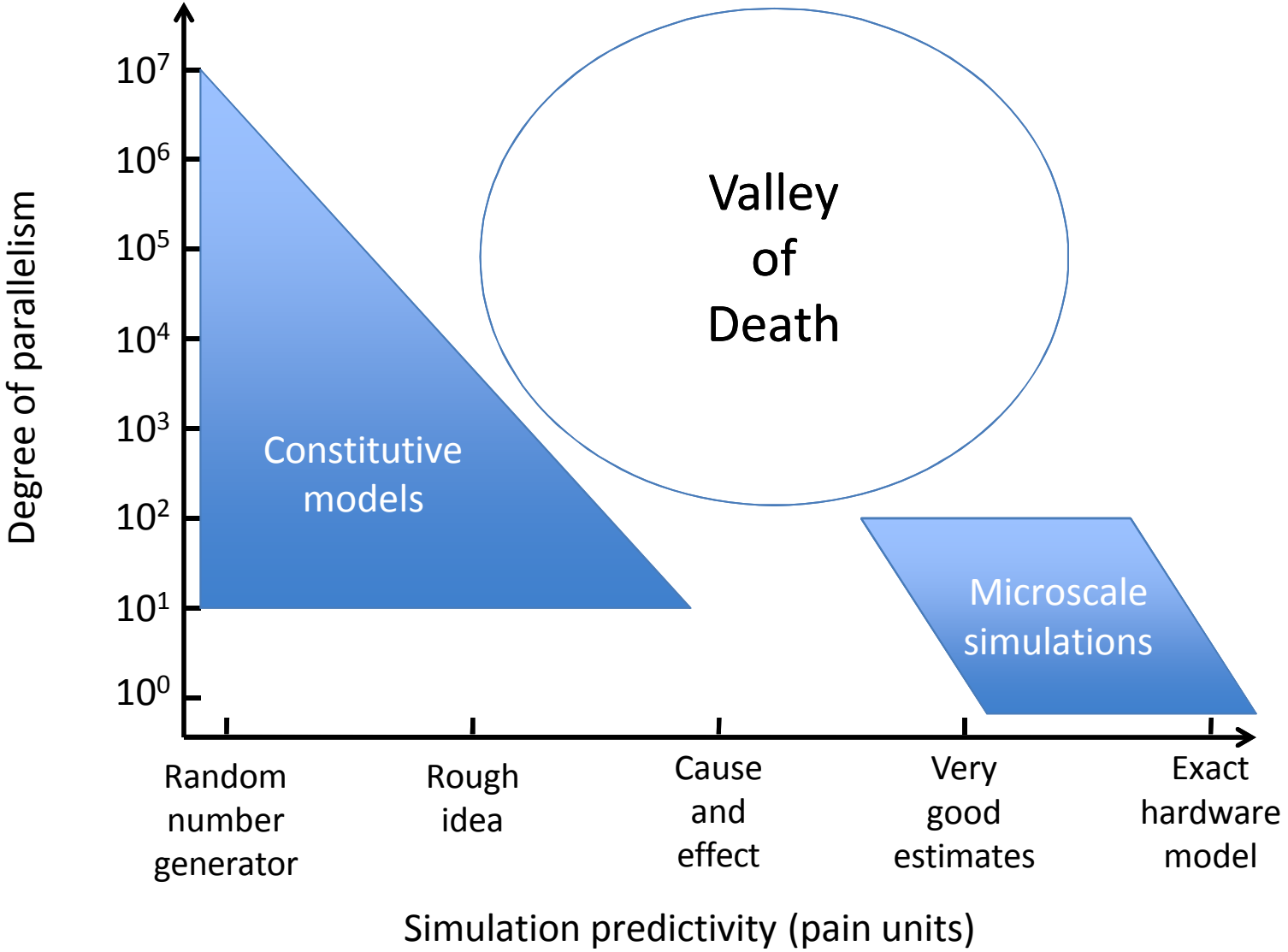
SST/macroscale

The other simulator

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company,
for the United States Department of Energy's National Nuclear Security Administration
under contract DE-AC04-94AL85000.

Sandia National Laboratories

The SST/macroscale target



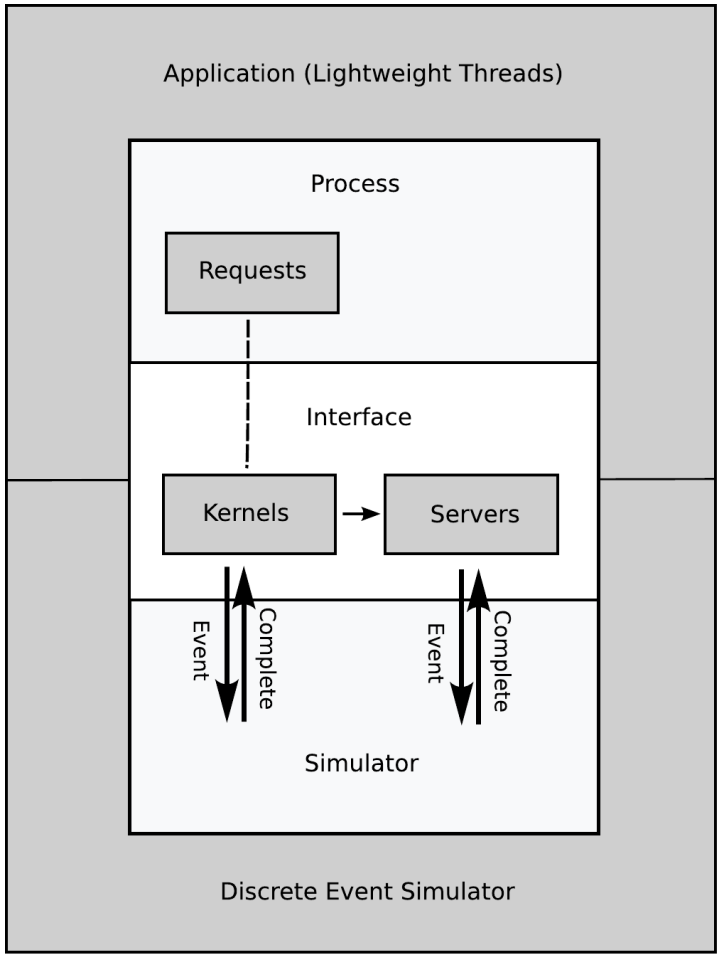
Sandia National Laboratories

More specifically, we want to:

- Identify causal relationships
  - Sensitivity analysis (topology, noise, bandwidth, latency, resource contention, etc.)
- Play “what if” games
  - Interpolating and extrapolating runtime behavior
  - Implementation effects for communication routines
  - Multiple networks, perfect/”magic” operations
- Test changes to application algorithms
  - Reordering, perfect scheduling, etc.
- Test novel programming models
  - Fault-tolerant or fault-oblivious execution models
  - Alternatives to MPI, parallel runtime designs.

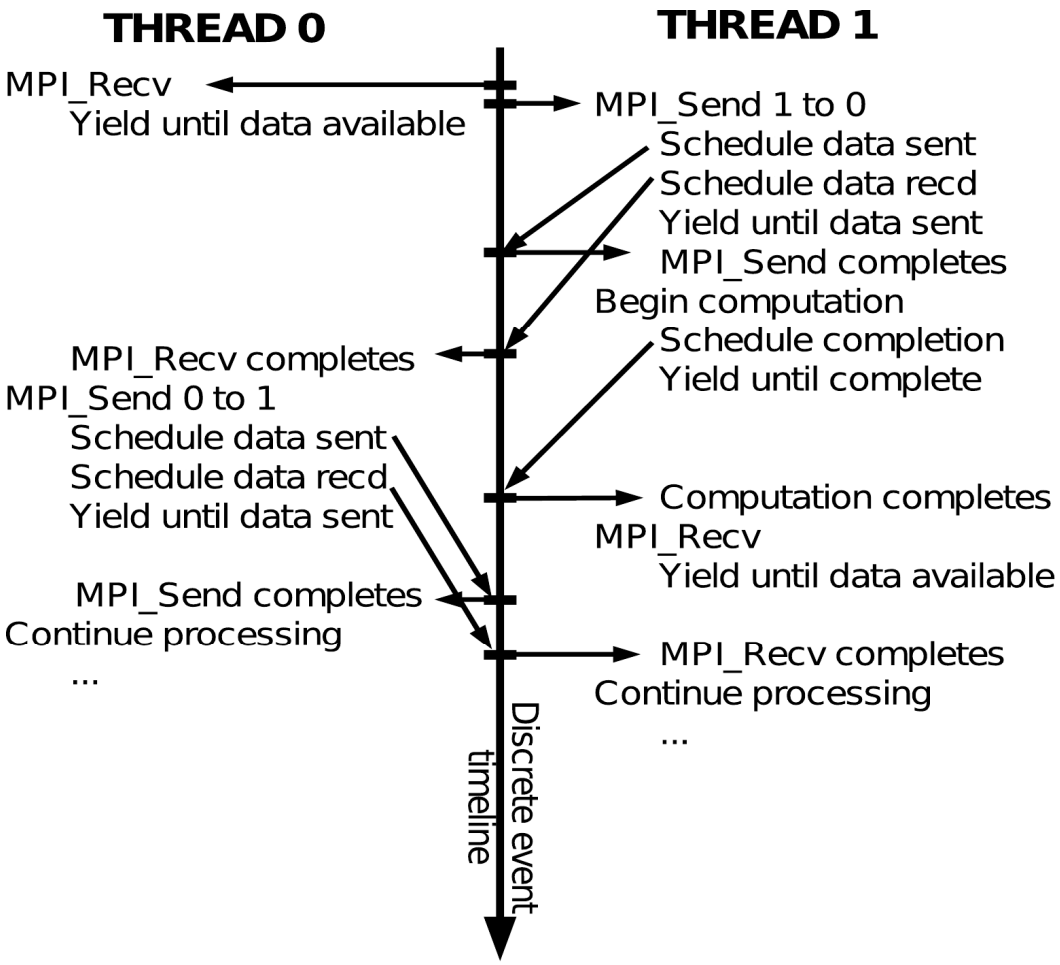
High-level design

- "Process" holds application process state
  - User-space threads
  - Private stacks
- "Interface" bridges threads and DES
  - Context switching for blocking calls
  - Management of active/blocked queues
- "Simulator" handles DES
  - Single-threaded (lock-free)



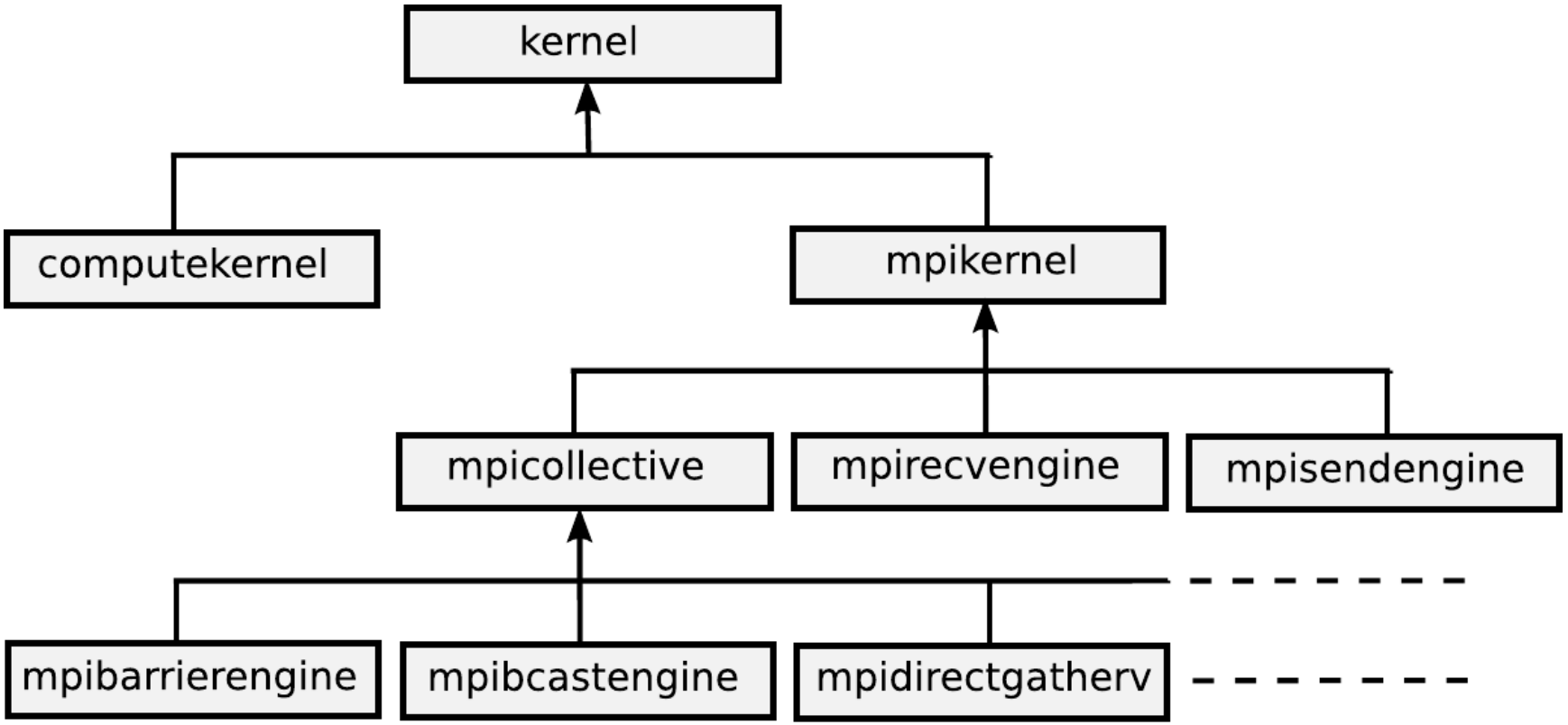
Sandia National Laboratories

Example control flow

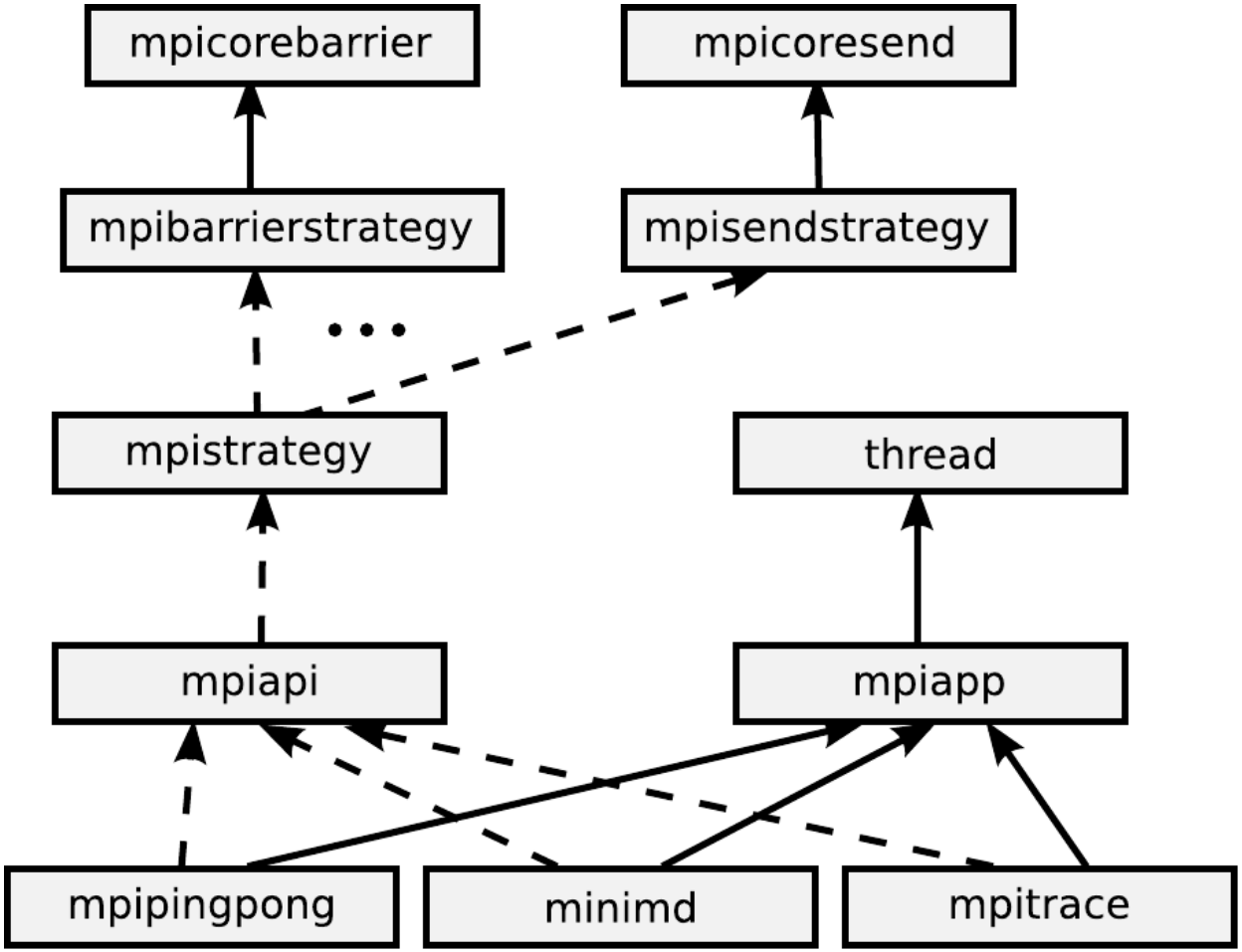


Two simulated processes exchanging data via MPI send/recv pairs

Example inheritance diagram

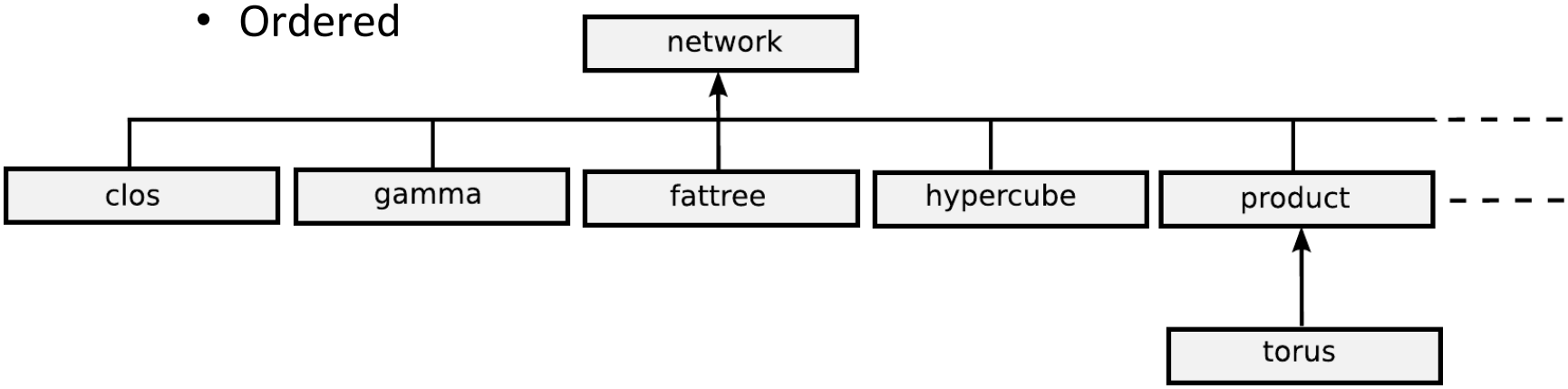


Partial collaboration diagram for MPI



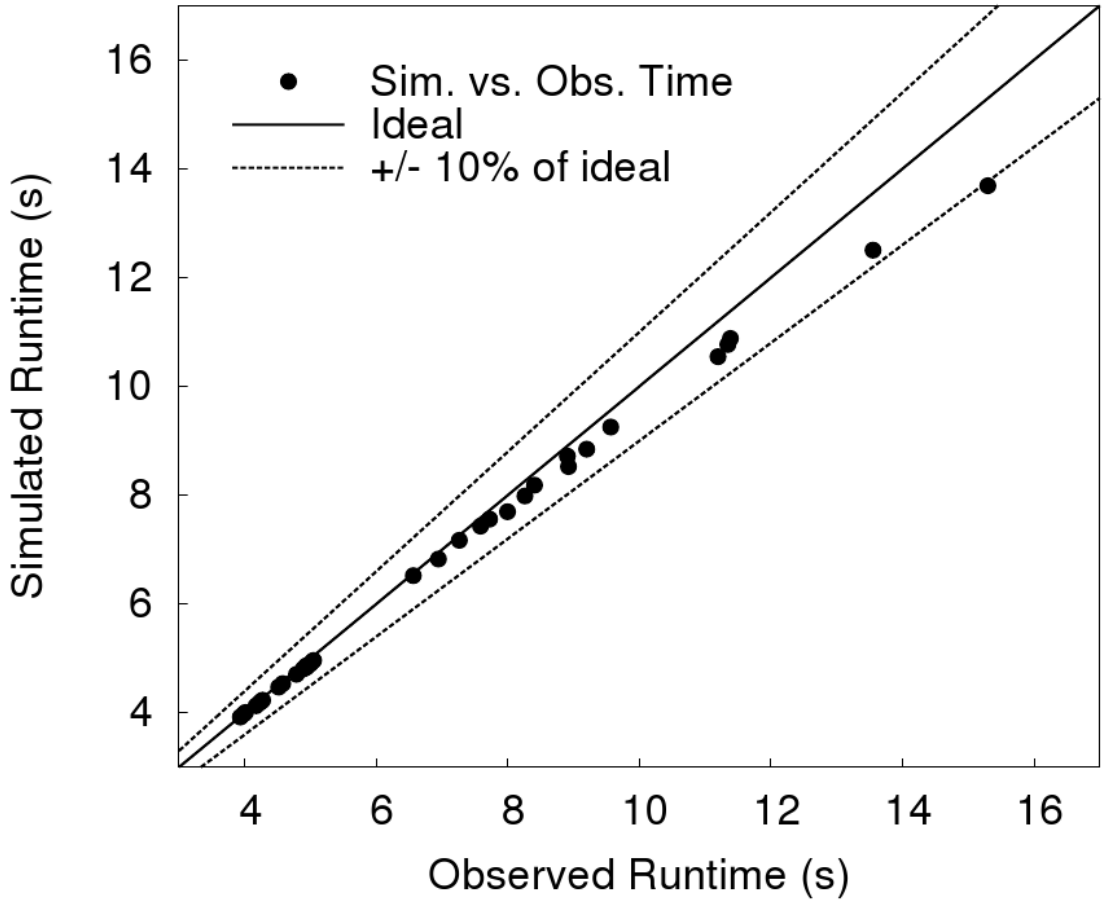
Networks

- Currently support simplified network routing/sharing
  - Contention-free
  - Circuit
  - Fair sharing of concurrent flows
    - Overcommit
    - Even split
    - Throttle overcommits
    - Ordered



Sandia National Laboratories

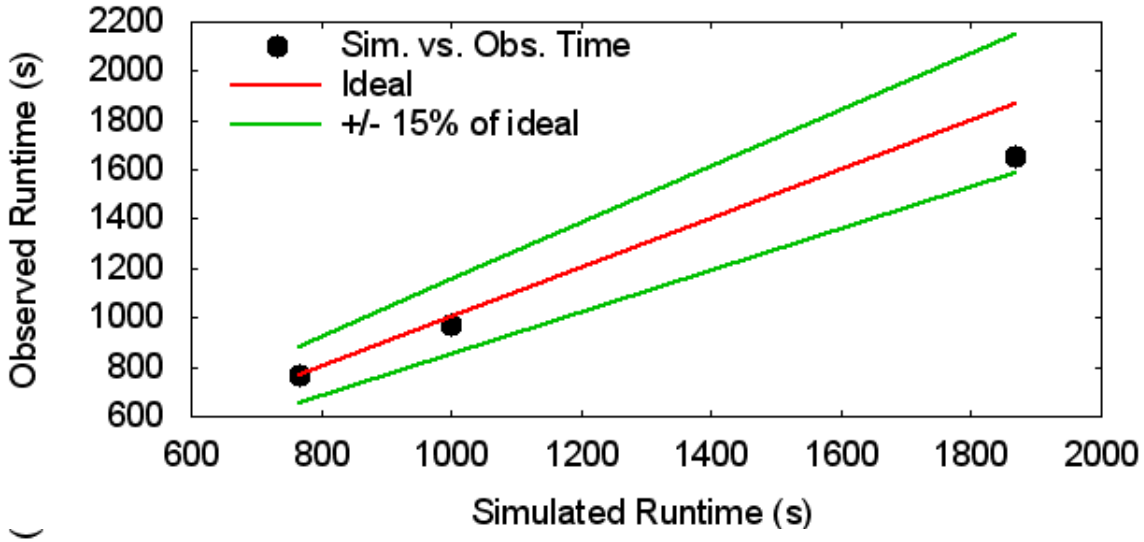
MPI trace example: AMG2006

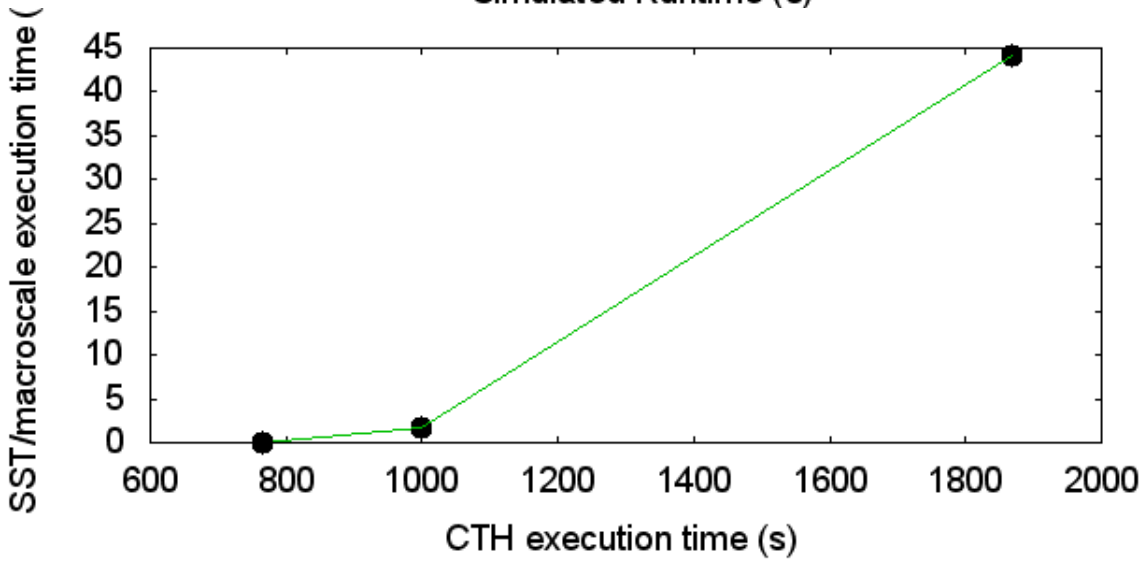


AMG2006 on a variety of node counts and decompositions.

Sandia National Laboratories

MPI trace example: CTH





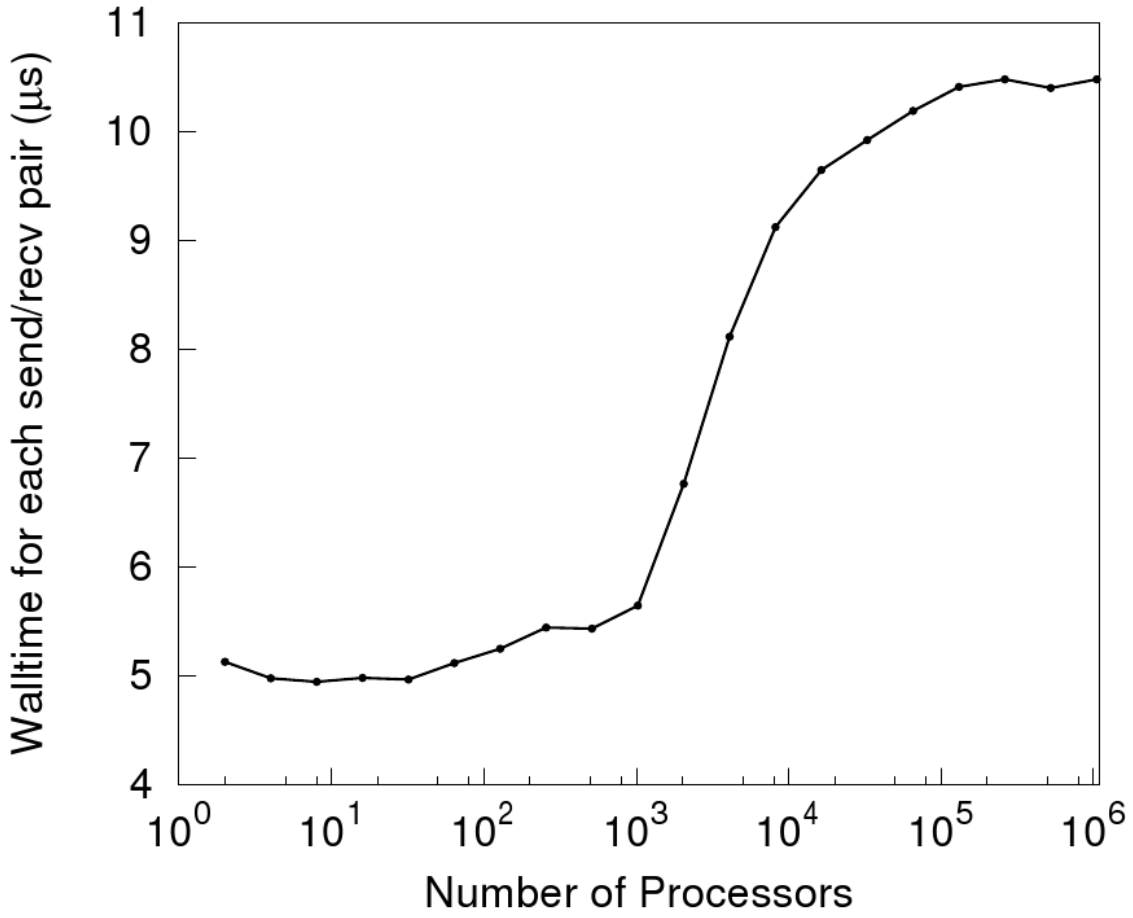
CTH runs on 1, 2, and 16 nodes courtesy of Courtenay Vaughan

Sandia National Laboratories

Skeleton applications

```
void mpipingpong::run() {
    this->mpi_->init();
    mpicomm world = this->mpi_->comm_world();
    mpitype type = mpitype::mpi_double;
    int rank = world.rank().id;
    int size = world.size().id;
    if(! ((size % 2) && (rank+1 >= size))) {
        // With an odd number of nodes, rank (size-1) sits out
        mpiid peer(rank ^ 1); // partner nodes 0<=>1, 2<=>3, etc.
        mpiapi::const_mpistatus_t stat;
        for(int half_cycle = 0; half_cycle < 2*niter; ++half_cycle) {
            if((half_cycle + rank) & 1)
                mpi_->send(count_, type, peer, mpitag(0), world);
            else
                mpi_->recv(count_, type, peer, mpitag(0), world, stat);
        }
    }
    mpi_->finalize();
}
```

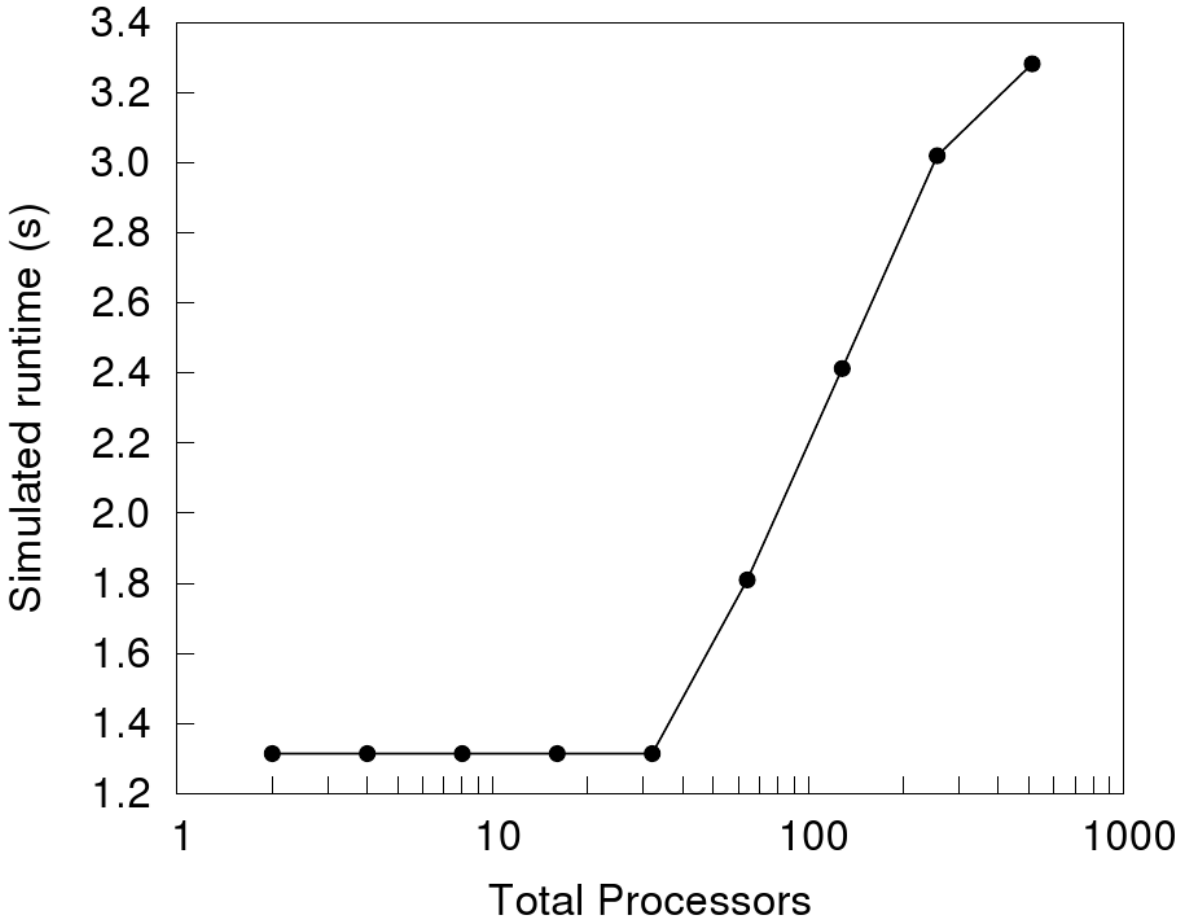
Simulator performance



MPI ping-pong on a contention-free network. Total of 4M messages.

Sandia National Laboratories

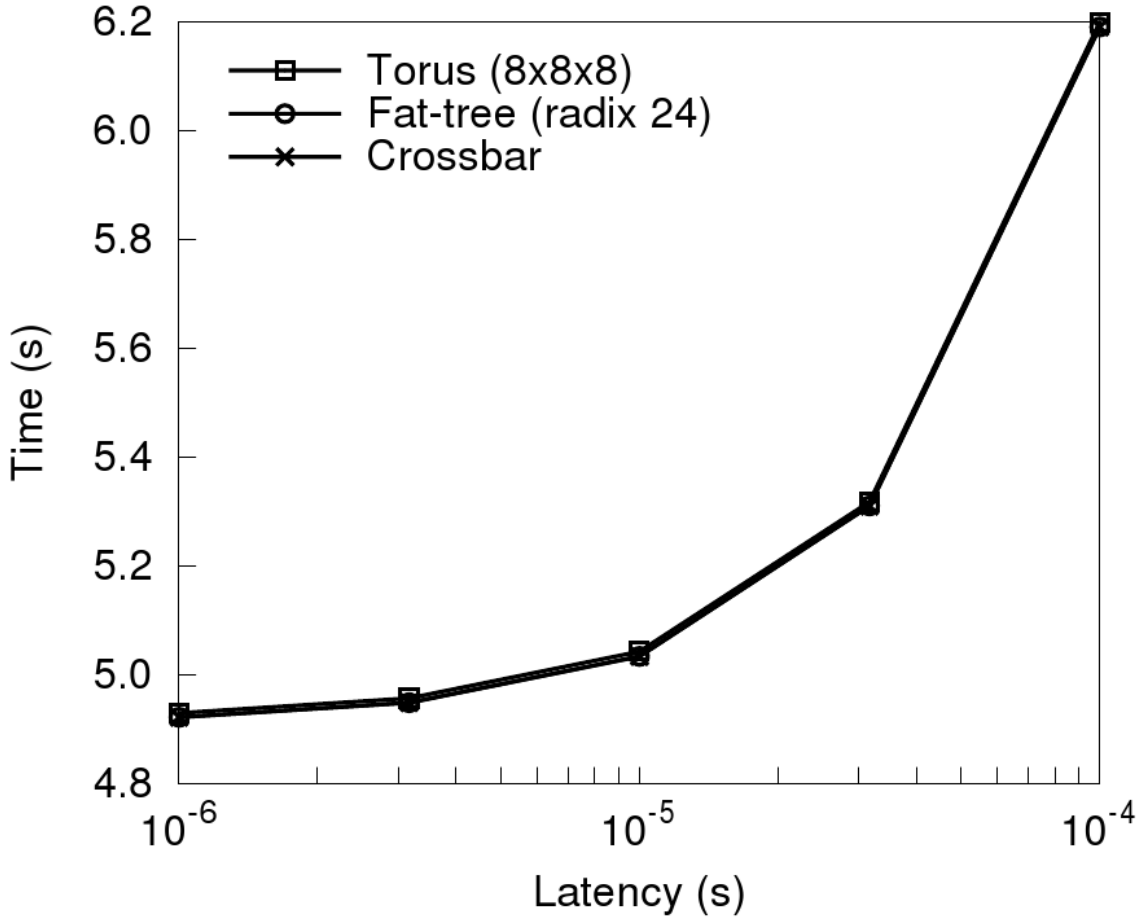
Simulated traffic congestion



MPI ping-pong on fat-tree network. Fixed total of 65536 messages.

Sandia National Laboratories

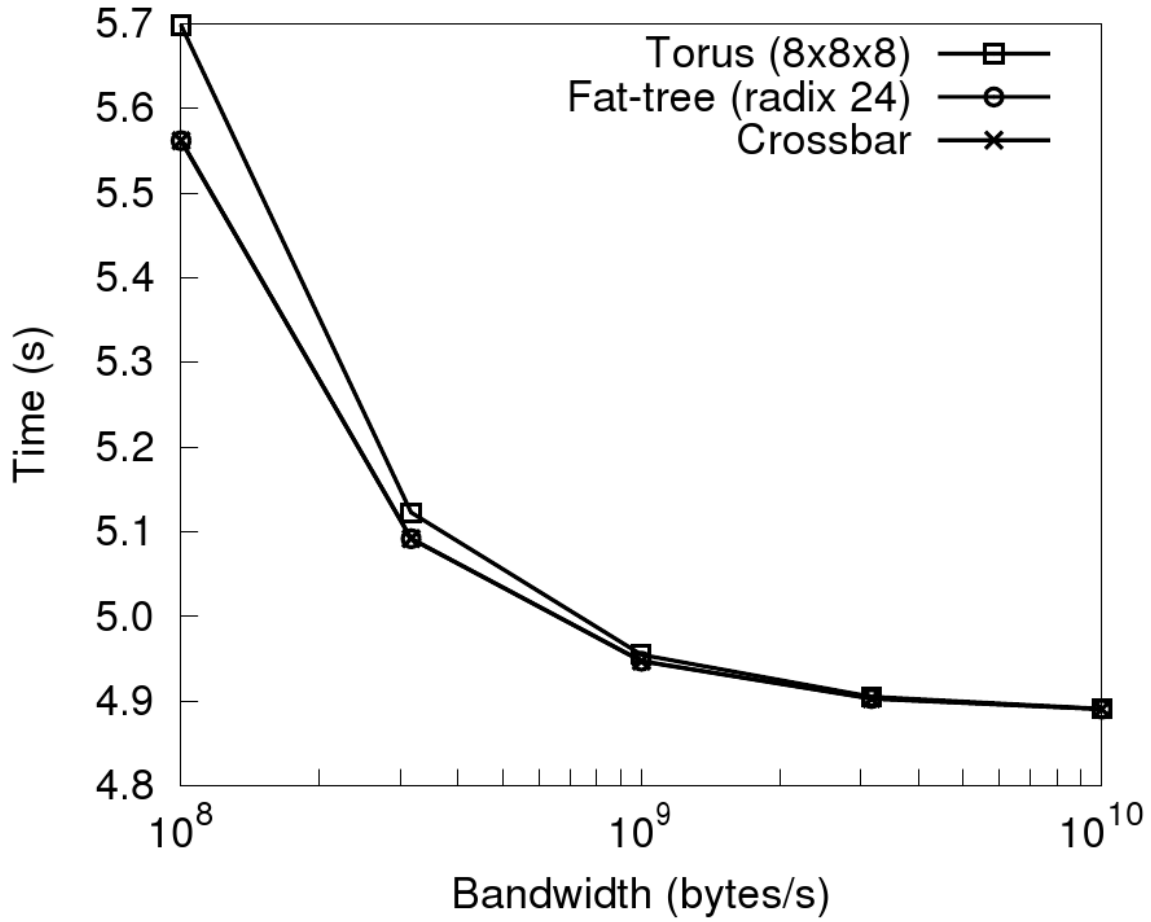
Example parameter sweep: Latency



AMG2006 on 128 nodes. Bandwidth constant at 1 GB/s.

Sandia National Laboratories

Example parameter sweep: Bandwidth



AMG2006 on 128 nodes. Latency constant at 3 µs.

Sandia National Laboratories

DUMPI: The MPI tracer

- PMPI link-time library for trace file generation
- Full fingerprints for all MPI-2 functions
- Writes a (reasonably compact) binary trace file
- Negligible runtime overhead
- Reasonably portable (but unreadable) C code

libdumpi	common	libundumpi
PMPI bindings Type mapping Call tree tracing (gcc/icc)	MPI type identifiers MPI function identifiers Trace file IO Timers Performance counters	Parsing of trace files

Example of data output by DUMPI

(converted using dumpi2ascii)

MPI_Allgatherv entering at walltime 1274314439.744512000, cputime 0.201756000 \
seconds in thread 0.

int commsize=16

int sendcount=1024

MPI_Datatype sendtype=14 (MPI_DOUBLE)

int recvcounts[16]=[1024, 1024, 1024, 1024, 1024, 1024, 1024, 1024, 1024, 1024, 1024, \
1024, 1024, 1024, 1024, 1024]

int displs[16]=[0, 1024, 2048, 3072, 4096, 5120, 6144, 7168, 8192, 9216, 10240, 11264, \
12288, 13312, 14336, 15360]

MPI_Datatype recvtype=14 (MPI_DOUBLE)

MPI_Comm comm=2 (MPI_COMM_WORLD)

MPI_Allgatherv returning at walltime 1274314439.749554000, cputime 0.202159000 \
seconds in thread 0.

Callback-driven parsing of DUMPI files

- Each MPI call is assigned a callback function
- The full set of arguments to the function is passed in as a struct

```
typedef int (*dumpi_allgatherv_call)(const dumpi_allgatherv *prm, uint16_t thread,
                const dumpi_time *cpu, const dumpi_time *wall, const dumpi_perfinfo *perf,
                void *userarg);

typedef struct dumpi_allgatherv {
    /** Not an MPI argument.  Added to index relevant data in the struct. */
    int  commsize;
    /** Argument value before PMPI call */
    int  sendcount;
    /** Argument value before PMPI call */
    dumpi_datatype  sendtype;
    /** Argument value before PMPI call.  Array of length [commsize] */
    int * recvcounts;
    /** Argument value before PMPI call.  Array of length [commsize] */
    int * displs;
    /** Argument value before PMPI call */
    dumpi_datatype  recvtype;
    /** Argument value before PMPI call */
    dumpi_comm  comm;
  } dumpi_allgatherv;
```

Current efforts

- SST/macroscale
  - GUI
  - Redoing MPI bindings (internals)
  - Better processor models
  - Added networks
  - Additional non-MPI programming models
  - Collecting more traces at various scales
  - Various collaboration efforts
- DUMPI
  - Updated callback mechanism
  - DUMPI to OTF converter

Acknowledgments

Contributors

- Curtis Janssen (PI)
- Helgi Adalsteinsson
- Scott Cranford
- Damian Dechev
- David Evensky
- Joseph Kenny
- Jackson Mayo
- Ali Pinar

Friends and neighbors

- Doug Doerfler
- Mike Heroux
- Mahesh Rajan
- Courtenay Vaughan
- Alan Williams

Funding

- ASC/CSSE

Sandia National Laboratories