SAND2010-1068C

# A Simulator for Large-scale Parallel Computer Architectures

Curtis Janssen, Helgi Adalsteinsson, Scott Cranford, Joseph Kenny, Ali Pinar, David Evensky, Jackson Mayo

Sandia National Laboratories

Livermore, CA

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.

# Designing, developing for, and procuring large-scale system is a multi-faceted problem

**Scale.....**

**Many Cores + Memory** X **Many Many Nodes** X **Many Many Many Threads**

**Complexity.....**

**Multi-Physics Apps**
**Informatics Apps**
X
**Communication Libraries**
**Run-Times**
**OS Effects**
X
**Existing Languages**
**New Languages**

**Constraints.....**

**Performance** **Reliability** **Usability** **Size**

**Cost** **Power** **Cooling** **Risk**

Architecture simulation enables co-design by providing a way to explore this large problem space.

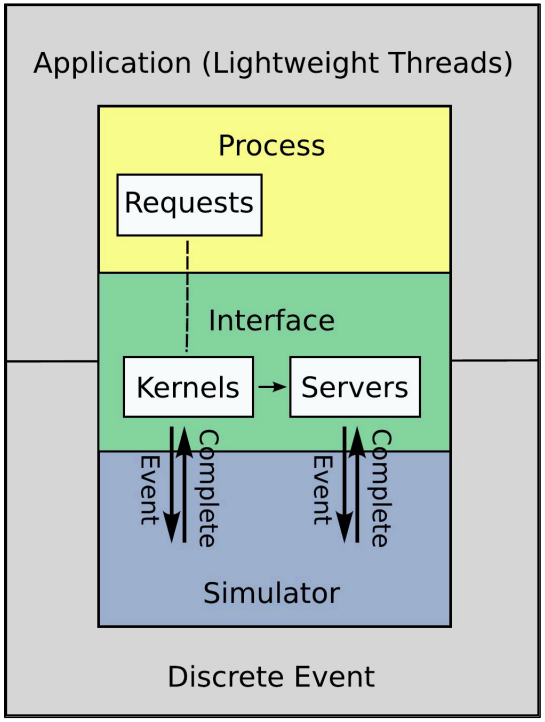
Sandia National Laboratories

# Goals for Structural Simulation Toolkit macroscale (SST/macro) components

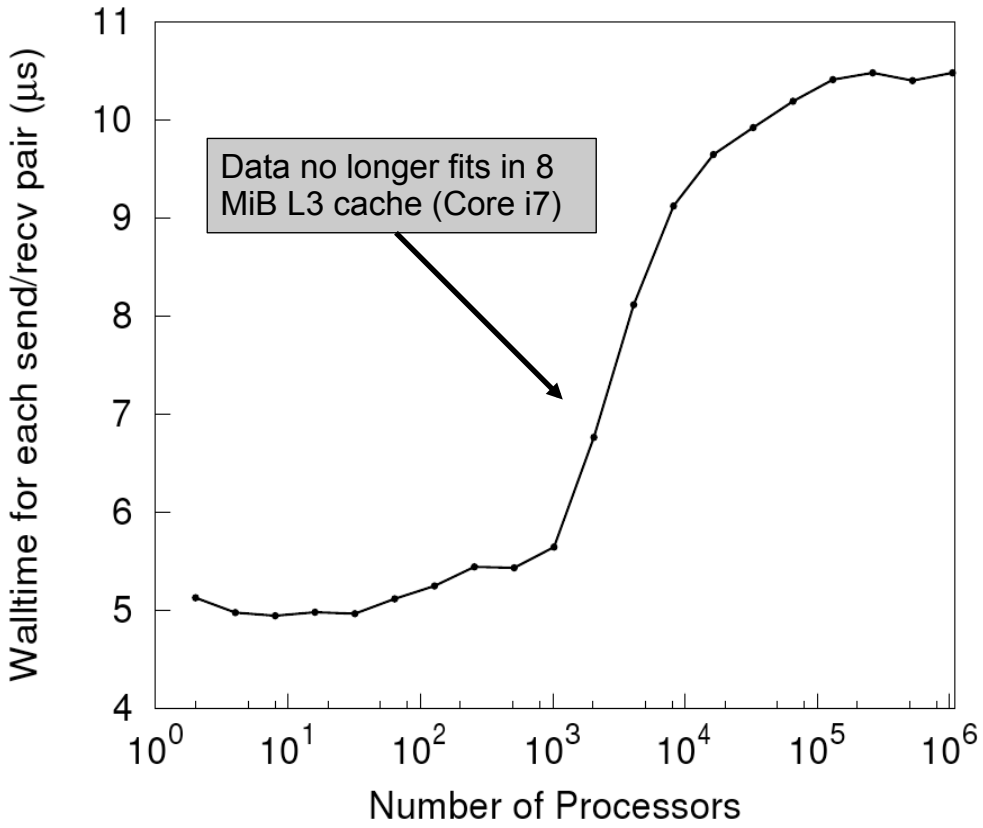
- Take into account the coupling of communication and computation in applications using
  - Trace files that record an actual application run
  - Skeleton applications that mimic application behaviour
- Run very large simulations (complex applications at large scales)
  - 1,000,000's of cores simulated on a single processor
    - Parallelism used for parameter studies
- Allow investigation of effects of
  - Topology, process placement, and interference between jobs
  - Changes in the routing algorithm, network latency, and bandwidth
  - Having many cores share the same network interface
  - Modifications to the MPI layer and the application
  - Incorporation of more detailed models w/multiscale simulation

# SST/macro design

**Generic event interface**: permits integration into the hybrid multi-scale SST simulator framework:




**Extremely lightweight events:** Measured real time to perform MPI ping pong round trips (simulator ran on a single processor):



# Examples of SST usage and impact

Obtain traces for applications and compact applications. Use SST for parameter studies.

Write skeleton applications for extreme scale studies.

Simulate new architectural feature such as extended memory semantics and transactional memory.


Develop acceptance tests and estimate performance before machine is built.

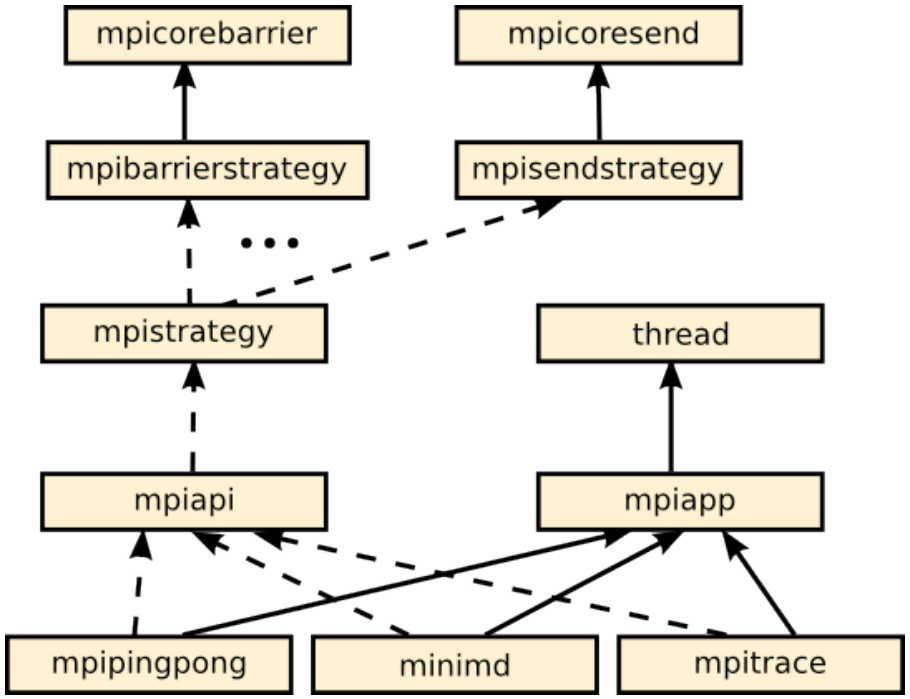
Understand performance and issues for machines several years from deployment.

Allow co-design of advance architectures and applications many years before deployment.

Sandia National Laboratories

# SST/macro has a flexible MPI model

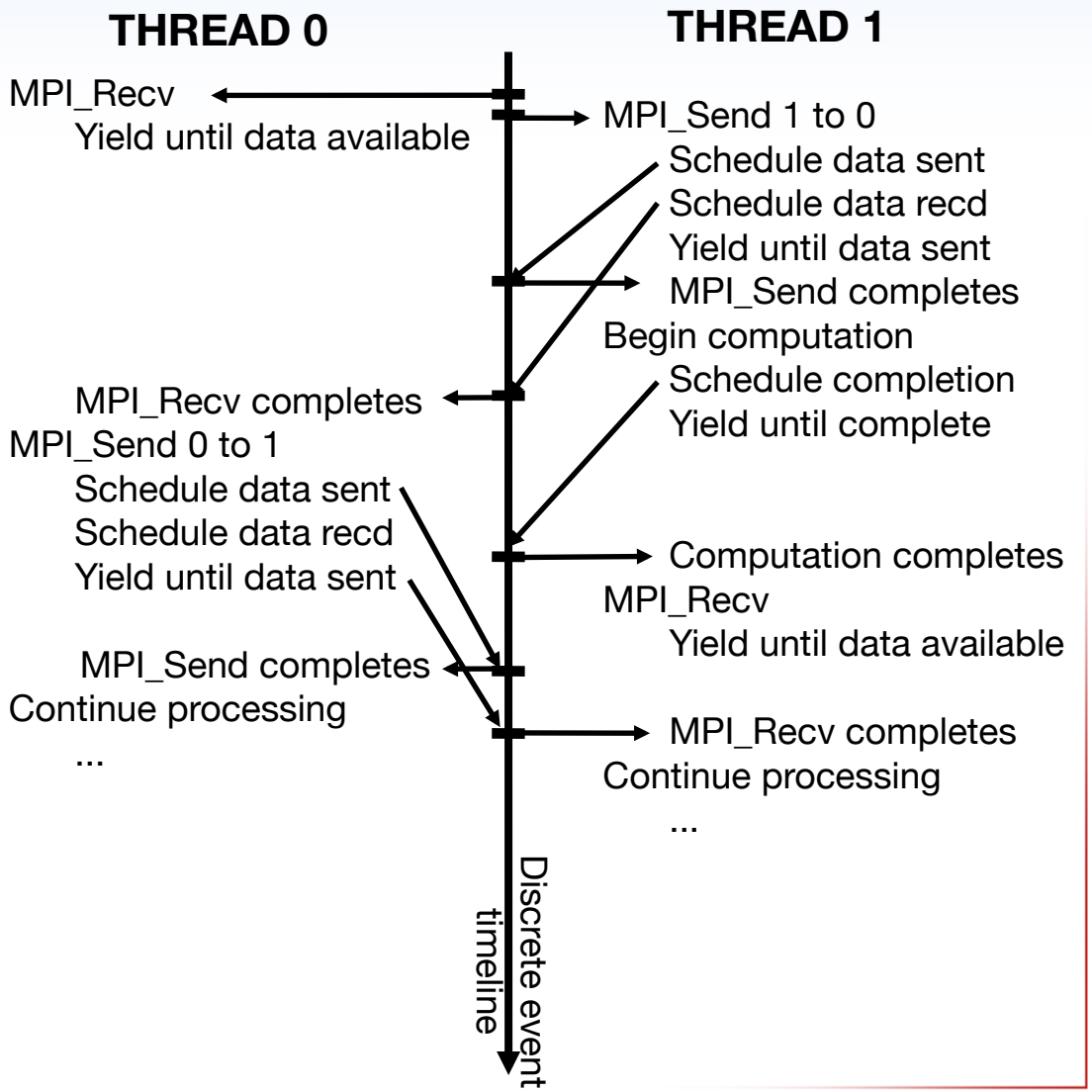
- Traces & skeleton apps record MPI calls—but that can entail a lot
  - MPI model can generate all of the messages that a real MPI library generates …
  - … or a simplified model with less overhead can be used
- Allows details of the MPI implementation to be a part of the design space
- Not just restricted to MPI2
  - Other programming models can be easily added
  - Implemented immediate mode collectives (MPI3 proposal)




Sandia National Laboratories

# Translating an application into discrete events

- MPI ping pong with computation: Simulation time increases going down, events are bars on the time axis, thread 0 executes code on left and process 1 on right
- Each thread inserts events into the discrete event queue until it yields
- Events can cause a process to resume execution
- Single MPI calls can result in multiple events
- Scheduling of data sent/ received events depends on network traffic




Sandia National Laboratories

# Coarse-grain CPU Modeling

- Accurately capture trends due to architecture changes without overhead of cycle-accurate simulation
- Simple models (currently implemented)
  - Time scale factor
  - Performance counters interleaved within dumpi traces, assume average event rates
    - Limitations on simultaneous performance counter collection

$$t_{exe} = \frac{n_{float}}{rate_{float}} + \frac{n_{mem}}{rate_{mem}}$$

- Research directions
  - Compact representations of memory activity (Snavely et. al.)
    - Locality metrics
    - Cache simulation
  - Stochastic approaches (Cook et. al.)

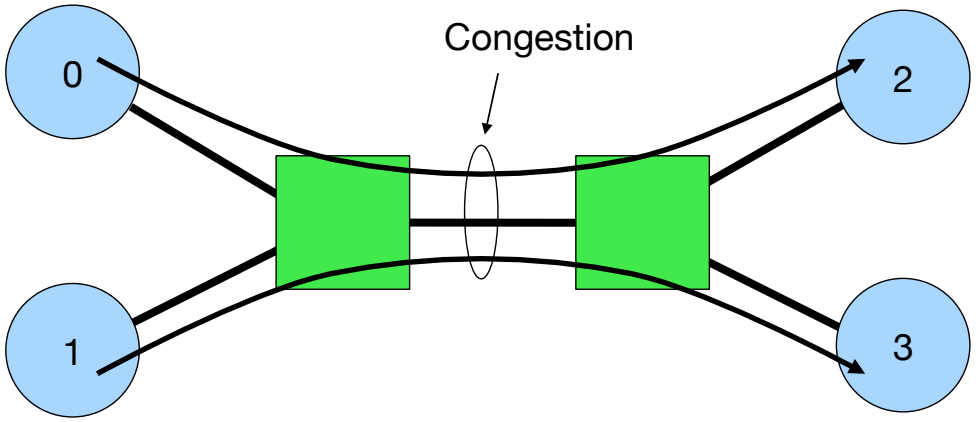
Sandia National Laboratories

# Network Model

- Designed as a separate module keeping  with the focus of the project.
- Allows experimenting with new topologies, routing techniques, network parameterssuch as bandwidth and latency.
- Designed for maximum runtime efficiency.
- Can be replaced with a high-fidelity simulator (e.g.  cycle-by-cycle simulation)
- For a new network, a user has to define
  - the topology,
  - a routing algorithm that returns the links to be used by a specific message
- Congestion is explicitly modeled.
- Currently support topologies high dimensional torus/mesh, fat-tree, hypercube, gamma, clos
- Routing is currently static, dynamic routing is in progress

# The circuit network component: a simple model that includes network congestion effects

Example: Two pairs of nodes try to use the same network link simultaneously:

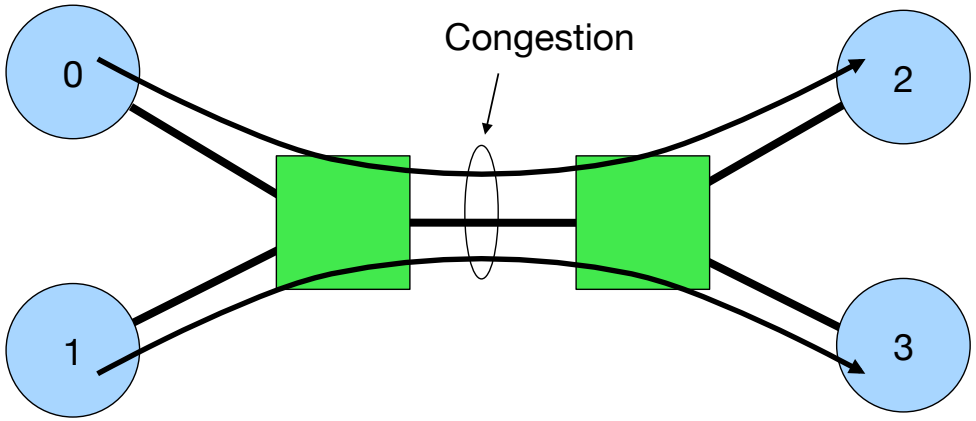



Circuit model handles this case as follows:

- T = 0s: Traffic begins from node 1 to node 3; duration is 4s.
- T = 1s: Attempt to begin traffic from node 0 to 2; duration is 2s. Attempt fails so attempt is rescheduled at T = 4s.
- T = 4s: Traffic from node 1 to node 3 completes.
- T = 4s: Traffic from node 0 to 2 begins; duration is 2s.
- T = 6s: Traffic from node 0 to 2 completes.

# The flower network component: an alternative to model network congestion

Flow based model allows messages share the bandwidth




Flower model handles this case as follows:

- T = 0s: Traffic begins from node 1 to node 3; duration is 4s w/ 100% bandwidth.
- T = 1s: Attempt to begin traffic from node 0 to 2; duration is 2s w/ 100%bw. Bandwidth is split between two messages.
- T = 5s: Traffic from node 0 to node 2 completes. Traffic from node 1 to node 3 receives full bandwidth.
- T = 6s: Traffic from node 1 to 3 completes.

# Methods for driving the SST/macro simulator

## Trace Driven

- Open Trace Format
  - Tools exist to generate trace
  - Visualizers exist for trace
  - Data not complete
- dumpi trace format
  - Custom SST/macro format
  - Records full MPI signature
    - size vectors
    - MPI_Request info
  - Well behaved when application is not
    - Skips irrelevant but resource intensive info (like MPI_Iprobe)

## Skeleton Application

- Programmer writes program that behaves like application
  - Skip heavy computation
- Permits extreme-scale runs

```
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))) {
  mpiid peer(rank ^ 1);
  mpiapi::const_mpistatus_t stat;
  for(int half_cycle = 0;
      half_cycle < 2*iterations_; ++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();
}
```

Sandia National Laboratories

# A more sophisticated skeleton app: miniMD

- Simulator runs a skeletonized molecular dynamics application
- Computation time is derived from measurements with various input parameters
  - The calls to estimate computation time is shown in **blue** type
- MiniMD control logic remains mostly intact

```
void minimd::integrate::run(shared_ptr<atom> atm, shared_ptr<force> frc,
                            shared_ptr<neighbor> nbr, shared_ptr<comm> cmm,
                            shared_ptr<thermo> thm, shared_ptr<timer> tmr)
{
  mpiid rank = mpi_->comm_world().rank();
  for(int n = 0; n < this->ntimes; ++n) {
    env_->compute(this->interpolator_->get("integrate::run", 0));
    if((n+1) % nbr->every) {
      cmm->communicate(atm);
    }
    else {
      cmm->exchange(atm);
      cmm->borders(atm);
      nbr->build(atm);
    }
    frc->compute(atm, nbr);
    env_->compute(this->interpolator_->get("integrate::run", 1));
    if(thm->nstat)
      thm->compute(n+1, atm, nbr, frc);
  }
}
```

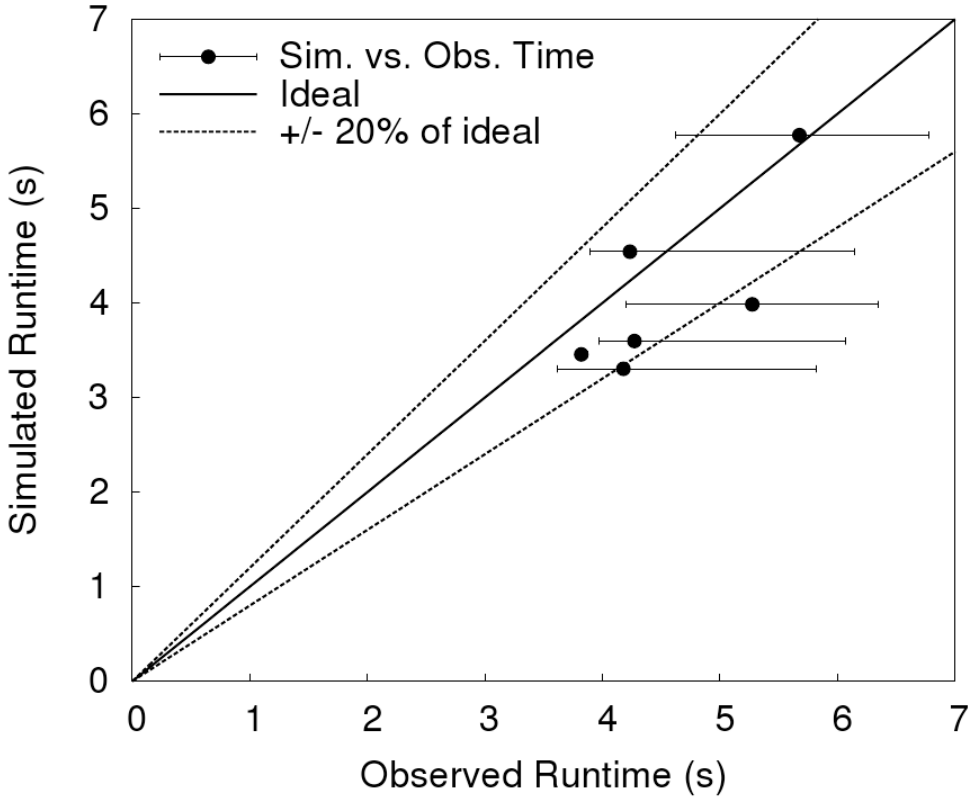
Sandia National Laboratories

# The DUMPI MPI profiling library

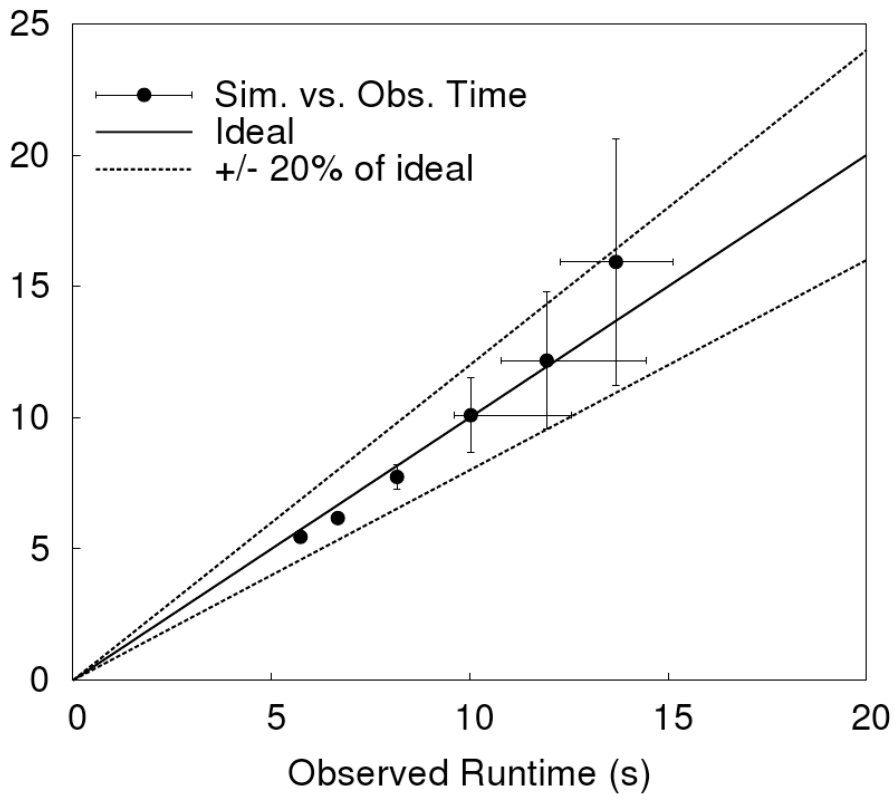
- libdumpi: collects information for detailed simulation of parallel MPI programs
  - Tracks all non-data arguments to MPI functions
  - Time (wall- and CPU time) and PAPI counters on entry and exit for every MPI function
  - Fine-grained control over tracing level
  - Collect reasonably compact (big-endian binary) trace file with a minimal runtime overhead.
- libundumpi: Callback-driven parsing of DUMPI trace files.
  - Mapping of implementation-defined values (e.g. type sizes, MPI_COMM_WORLD, MPI_CONGRUENT, etc.) to common tokens (DUMPI_COMM_WORLD, etc.).
  - Tested on a number of HPC platforms (Cray CNL, BG/P, Catamount)

# Validation of simulator

- Used AMG2006: part of NNSA ASC/Sequoia acceptance tests
- Collected traces on the Thunderbird machine and played back through simulator




"Narrow" decomposition

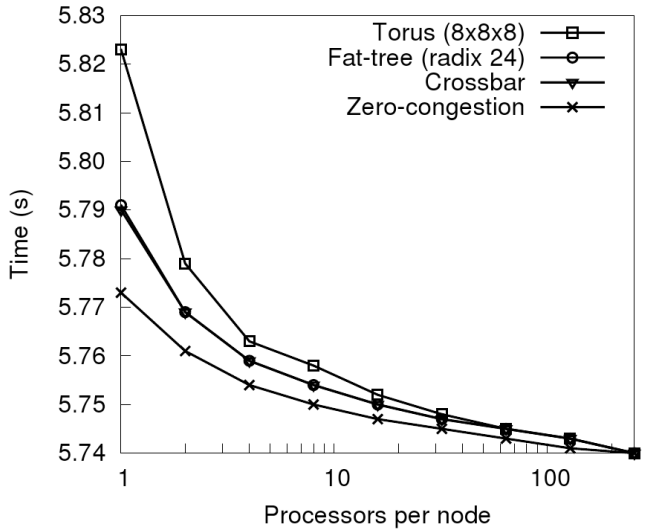


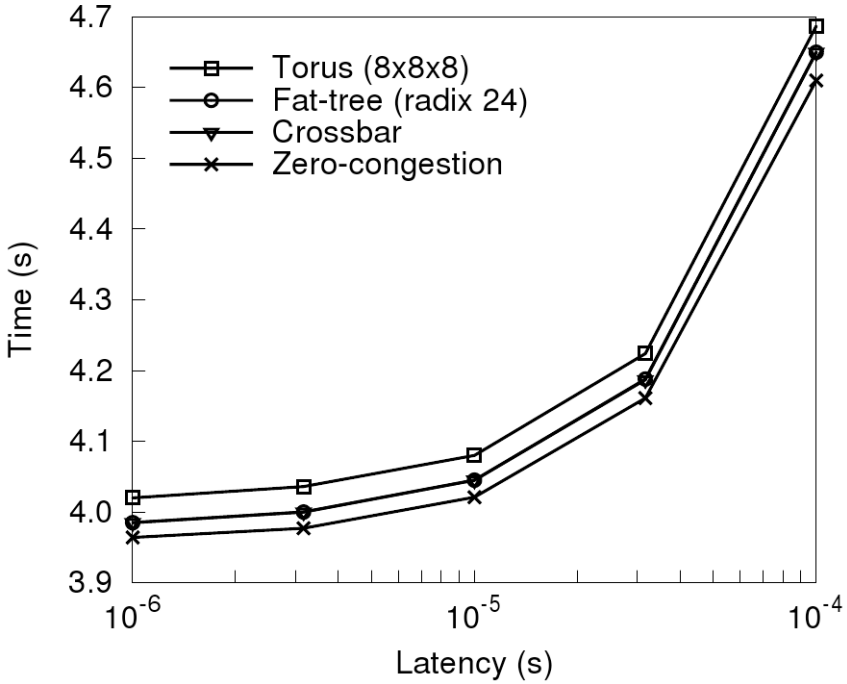

"Fat" decomposition

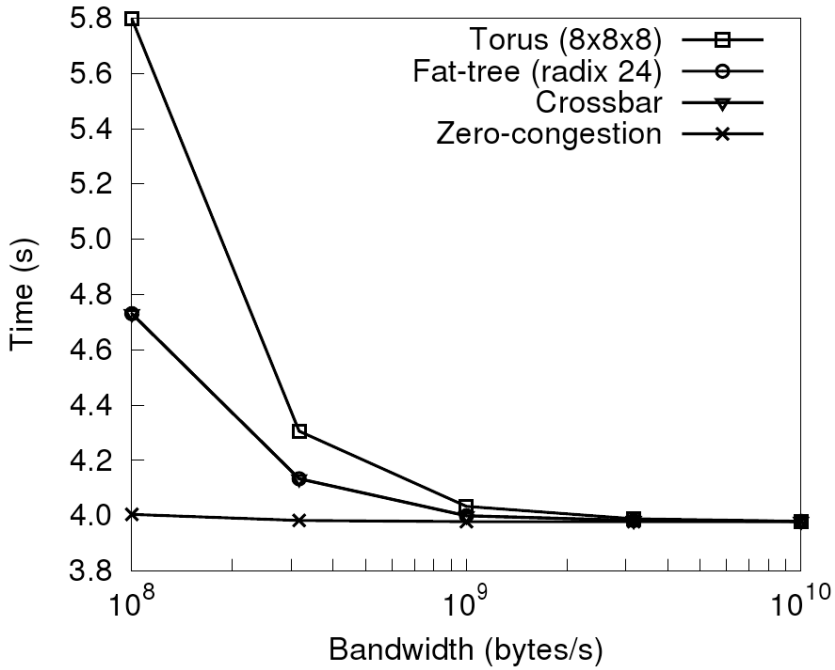
Sandia National Laboratories

# Sensitivity of AMG2006 to architectural parameters

- Examined simulated time to solution as bandwidth, latency, and processors per node are varied for several topologies.



# Current Work

- Extended validation studies
  - Using Red Storm Qualification system
    - Can control process placement
    - Results are highly consistent from run to run
- Advanced routing
  - Dispersive routing
  - Adaptive routing
- Trace file format (dumpi)
  - Finalizing full MPI support
- User interface and visualization
  - Developing a GUI to simplify problem setup
  - Working with others to visualize network congestion, etc.
- Processor models
  - Developing more sophisticated but inexpensive processor models
- Integration into the SST/Core interface

# The more the merrier!

- We welcome collaborations.
- Summer internships are available.
- We are hiring! Please visit http://www.sandia.gov/careers/ for specific job postings.
- Contact Information: Ali Pinar (apinar@sandia.gov)

Sandia National Laboratories