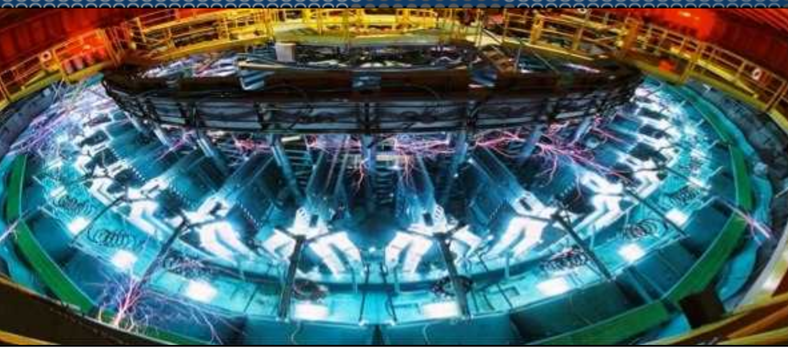
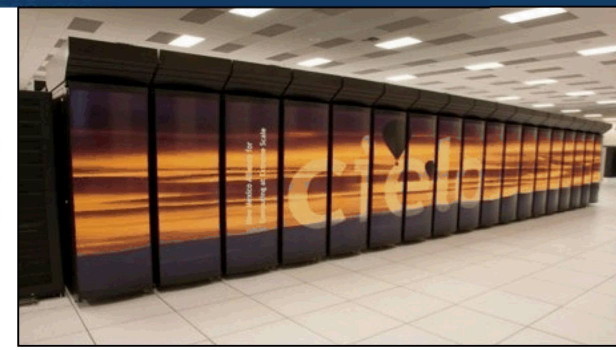
SAND2016-2827PE

Exceptional service in the national interest

Sandia National Laboratories





Building a Simulation Community for Solving Exascale Problems: The Structural Simulation Toolkit (SST)

Jeremiah Wilke, Joseph Kenny, David Hollman
Scalable Modeling and Analysis
Sandia National Labs, Livermore CA

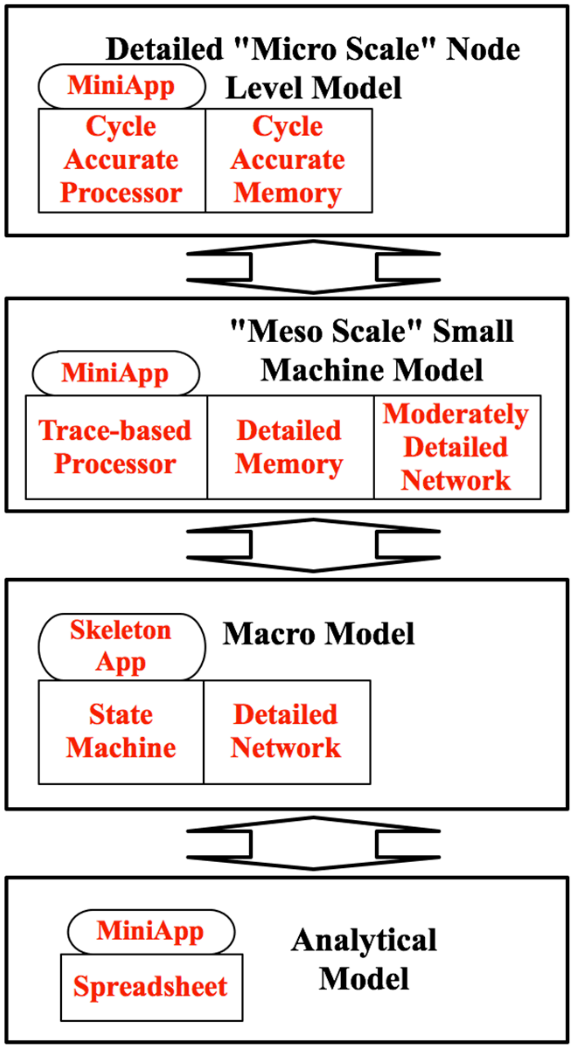
Simon Hammond, K. Scott Hemmert, Arun Rodrigues, Gwendolyn Voskuilen, et al.
Scalable Computer Architectures,
Sandia National Labs, Albuquerque, NM

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

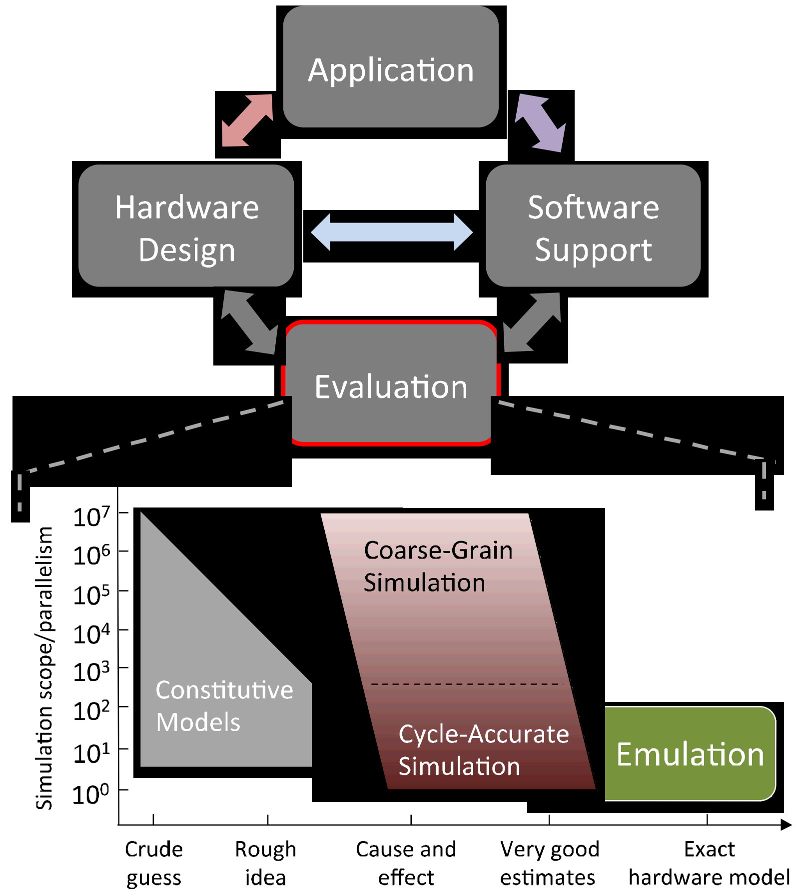
Simulating exascale systems faces conflict between accuracy/cost



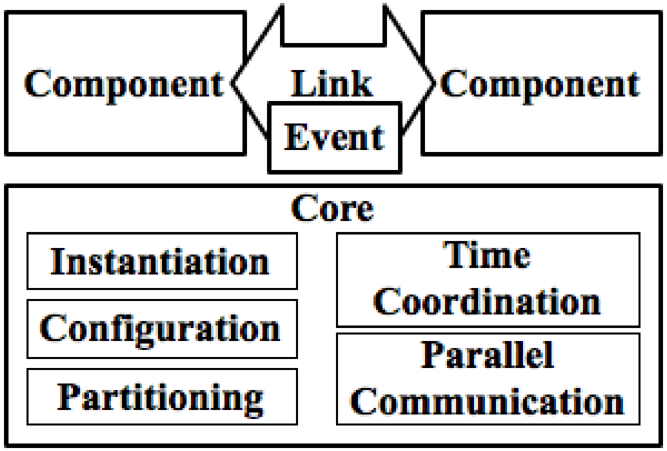
- Need to understand details of individual processors/memories
- But then need to understand how memories/processes interact on-node in parallel program…
- But then need to understand how multiple nodes interact to produce scaling behavior…
- But then how do we explore parameter/design space to assess best overall machine configuration…
- For our goals, *structural simulation* provides good balance between flexibility, accuracy, cost

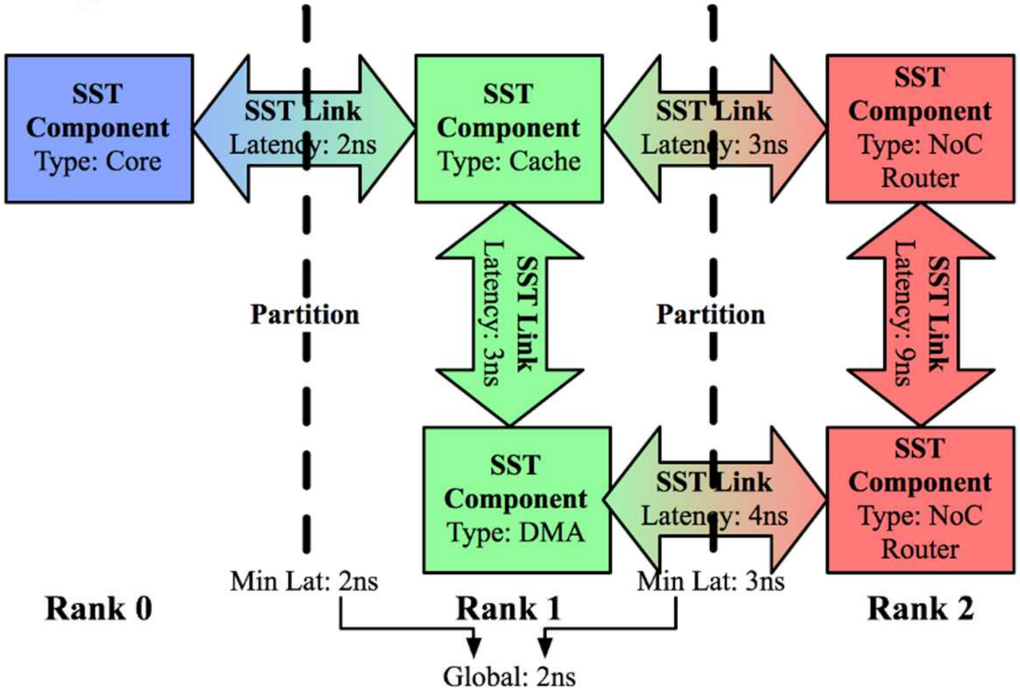
Structural simulation of exascale system requires numerous different components

- Endpoint (application) model
  - Instruction-based
  - Trace-based
  - Skeleton apps
  - State machine
- Compute/memory model
  - Detailed emulation
  - Coarse-grained structural simulation
  - Analytic models
- NIC/network model
  - Analytic models (messages only)
  - Coarse-grained packet models
  - Flit-level



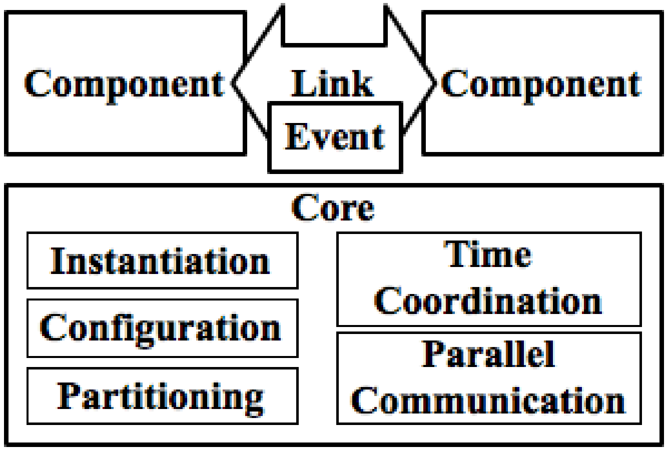
Scale simulations to match system scale: Couple models through common PDES core

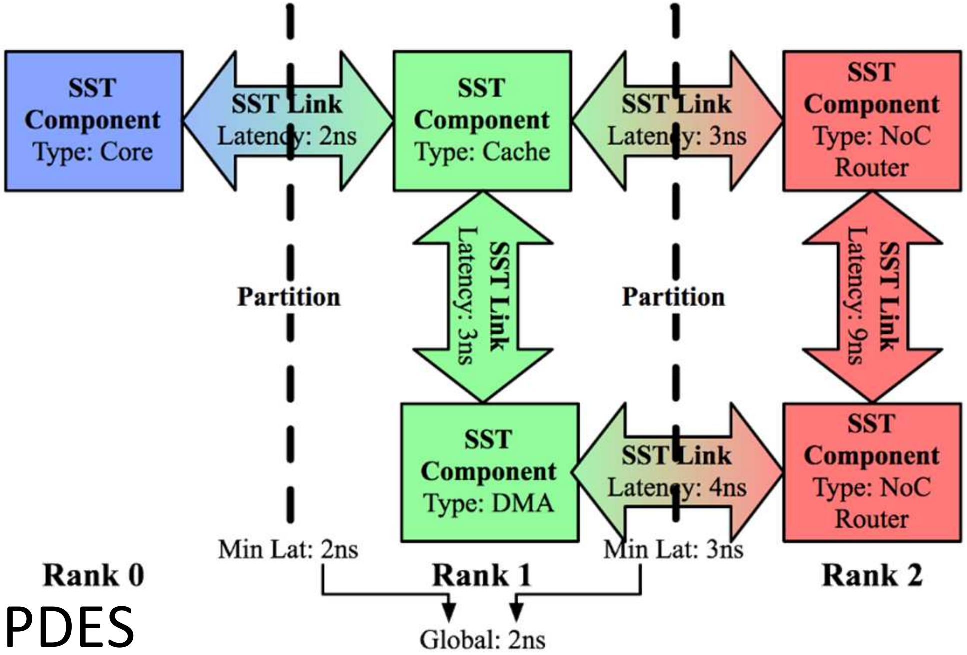




- If supercomputers are going to scale up, simulation tools need to scale with them
- Parallel discrete event simulation is a challenging problem
- Partially-ordered parallelism always balancing optimistic parallelism against rollback costs

PDES core should be solved problem: Researchers should focus on models





- Several steps in deploying PDES
  - Optimal partitioning of elements
  - Delivering events across distributed memory
  - Synchronizing many, many components
- This work is *universal* to all parallel simulators
- Modularity and composability of simulation tools almost impossible without unified PDES core

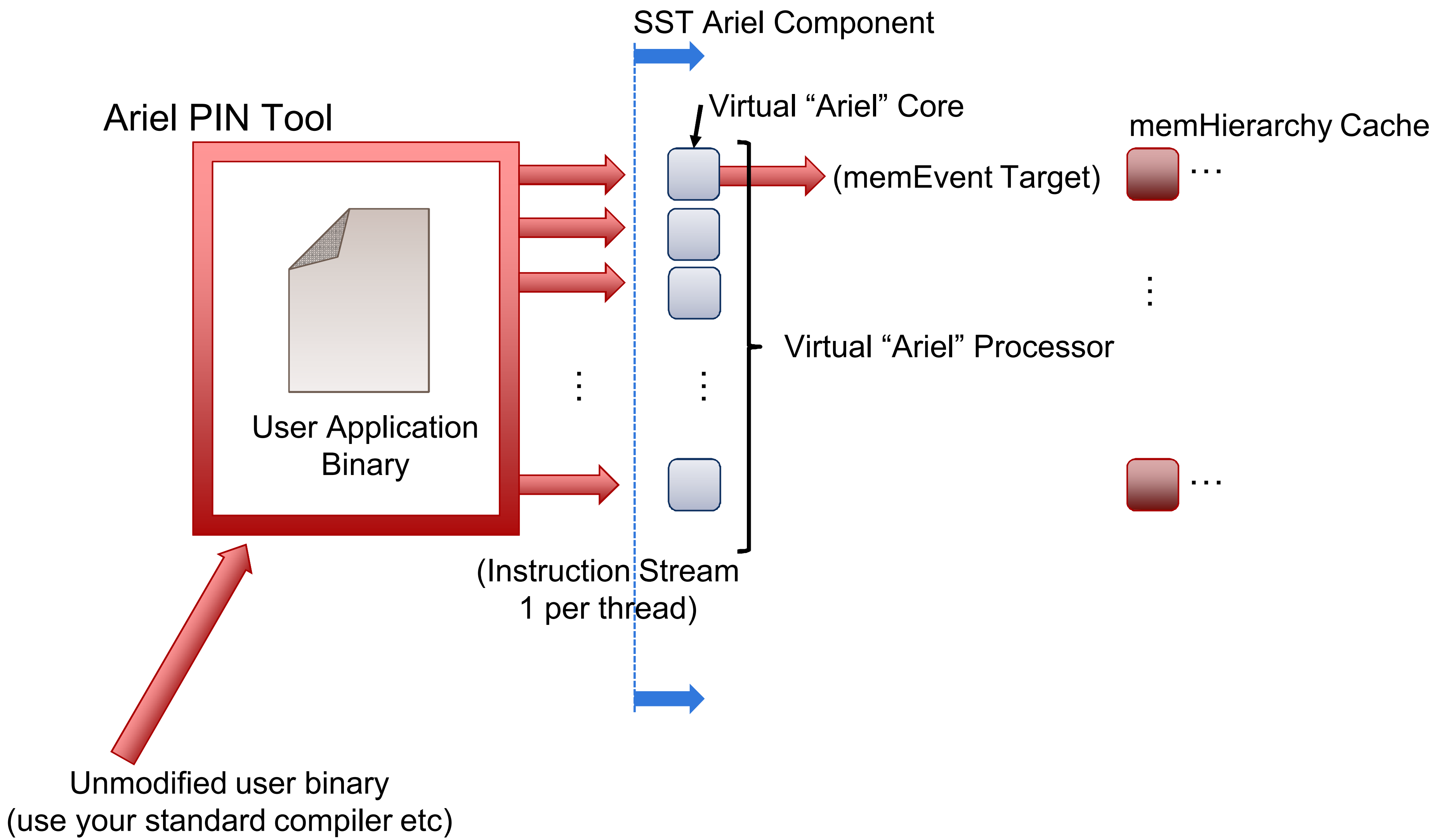
SST defines interfaces for coupling models

- SimpleMem:
  - Issue arbitrary memory requests to memory model
  - Register arbitrary callbacks for request notifications
  - Designed as lightweight wrapper around any memory simulation tool
- SimpleNetwork:
  - Issue arbitrary network requests (at the packet level)
  - Register arbitrary callbacks for packet send/recv notifications
  - Designed as lightweight wrapper around any network simulation tool
- FOSSIL:
  - On-line simulation, directly compile MPI/pThread code into simulator
  - "Interface" for writing endpoint models is just valid C/C++ code
- Under development:
  - Message-layer interface (have tool called Hermes for packetization/de-packetization of messages)

Diverse set of processor models covering accuracy/cost spectrum

- Ariel: PIN-based processor
  - Uses Intel’s PIN tools and XED decoders to analyze binaries (x86, x86-64, SSE/AVX, etc. compiled binaries)
  - Passes information to virtual core in SST
  - Virtual cores implement *abstract memory interface*
- Prospero: Traced-based processor
  - Reads memory ops from file and passes to simulated memory system
  - “Single core” but can emulate parallel applications with multiple traces
- Miranda: Pattern-based processor
  - Light-weight processor model for specific memory address patterns
  - Currently implements random access, STREAM, GUPS, stencils, and more

Ariel advertisement



Diverse memory models covering cost/accuracy spectrum with MemHierarchy components

- SimpleMem interface for issuing memory requesting, register callback notifications for completed requests
- Can have arbitrary ports/connections
- Supports many different backends
  - SimpleMem – basic read/write with associated latencies
  - DRAMSim2 – DRAM (external)
  - NVDIMMSim – Non-volatile memory (e.g., Flash) (external)
  - HybridSim – non-volatile memory with a DRAM cache (external)
  - VaultSimC – stacked DRAM

Diverse memory models covering cost/accuracy spectrum with MemHierarchy components

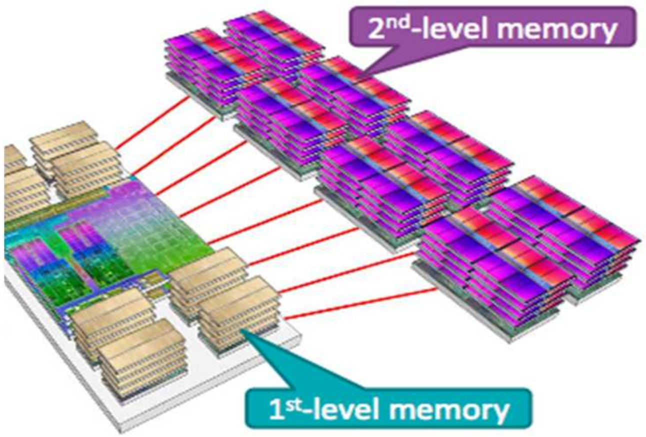
CacheController
- Routes incoming events to handlers
- Manages retry of buffered events in the MSHRs
- Manages cache allocations and evictions

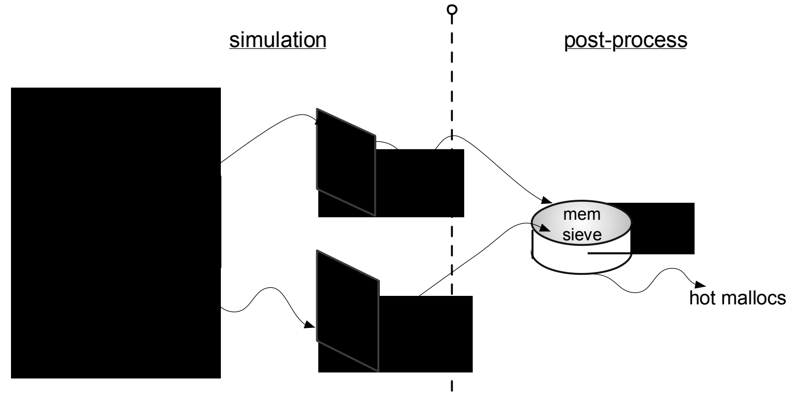
CacheArray
- Stores cache lines – data and coherence state
- Replacements via the replacement policy manager

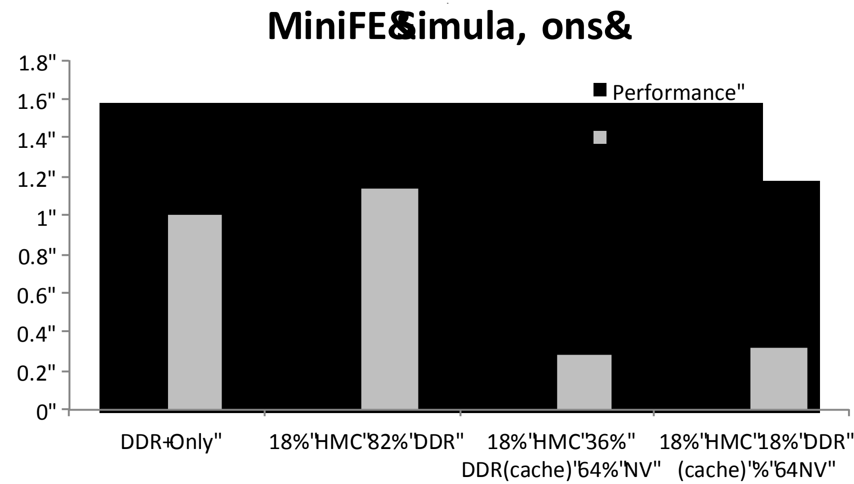
MSHRs
- Buffers stalled and blocked events

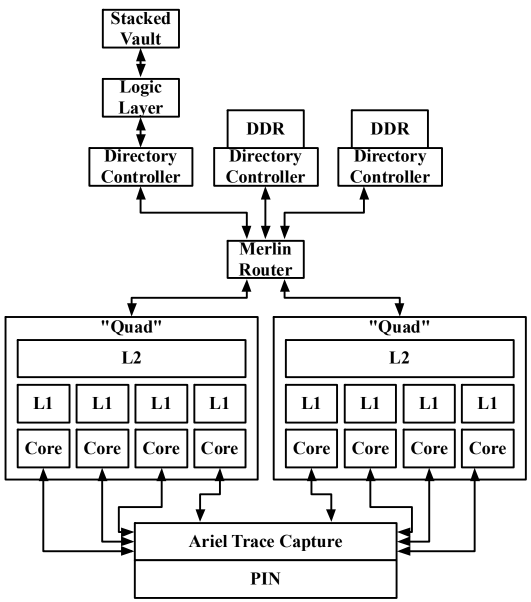
CoherenceController
- Manages coherence state
- Receives events from CacheController
- Sends outgoing events
  - Forwarded requests, responses, etc.
- Decides when events need to stall

Multi-level memory advertisement







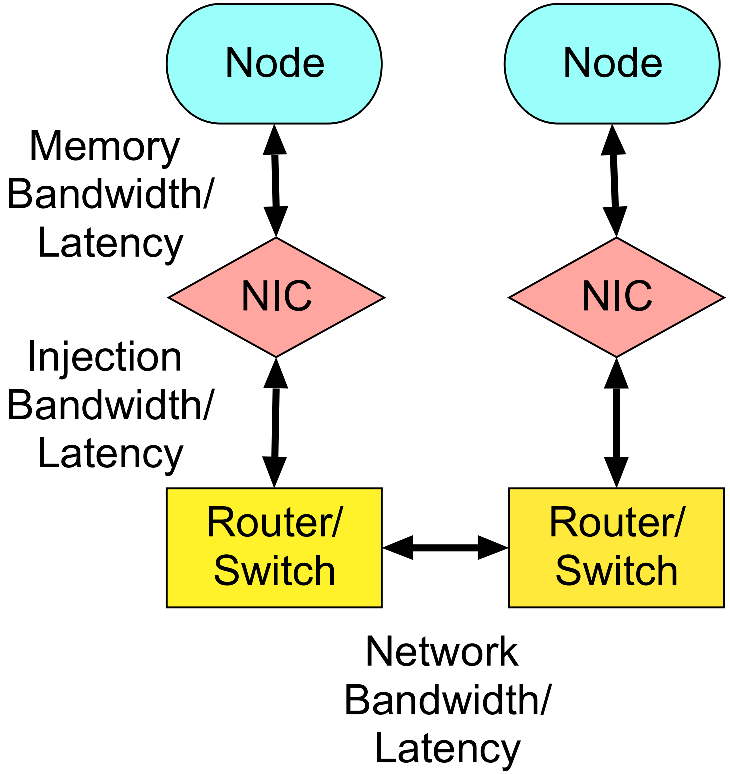


Siam Conference on Parallel Processing
Friday, April 15 2:40 PM

Diverse set of network models covering cost/accuracy spectrum

- PISCES: Coarse-grained packet simulator (formerly in SST/macro)
  - Packet-Flow Interconnect Simulation for Congestion at Extreme-Scale
  - Performs routing and flow-control at level of packets
  - Uses flow approximations to account for flit-based/cut-through routing
  - Supports arbitrarily complex routing strategies (factory interface – more details available)
  - Currently have minimal, minimal-adaptive, Valiant, and UGAL
- MACRELS: Approximate, analytic models (formerly in SST/macro)
  - MTL for AnalytiC REally Lightweight Simulation
  - Rapid prototyping of design space without detailed congestion
  - Designed to support algorithm/software stack explorations
- Merlin: Cycle-based packet simulator

MACRELS: simple network supporting application/runtime development



Modeling with simple delay formula similar to LogGOPSim

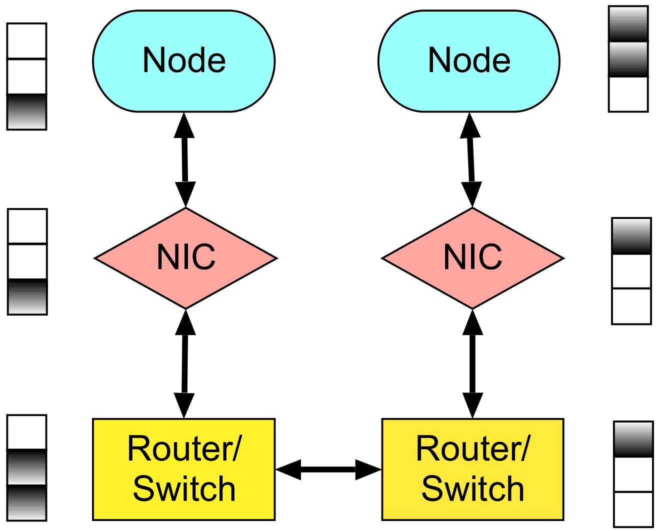
α = Latency

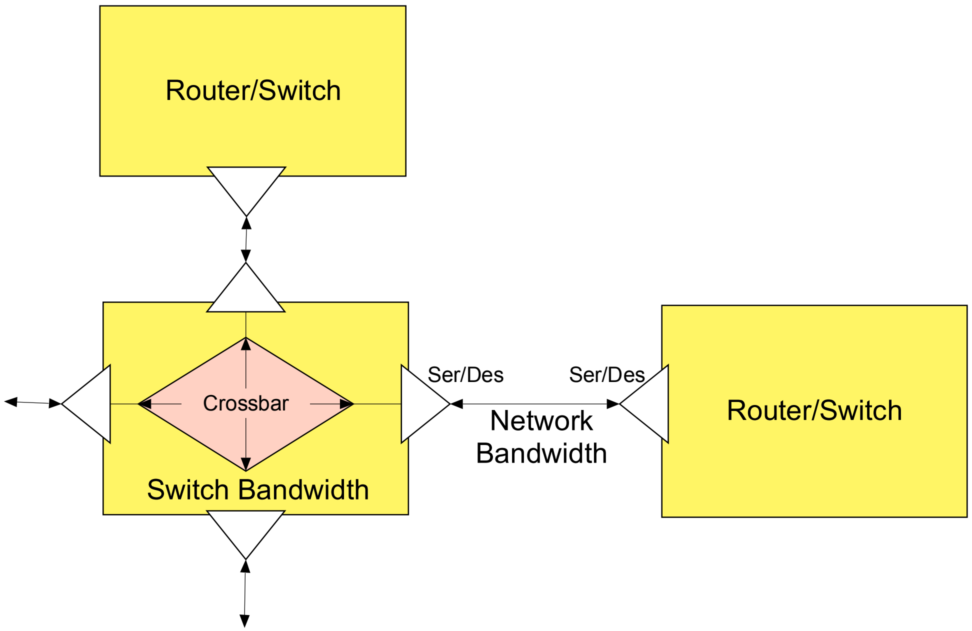
β = Inverse bandwidth

N =Message size

$$\Delta T = \alpha + \beta N$$

PISCES: packet-level for prototyping topologies/routing/congestion management



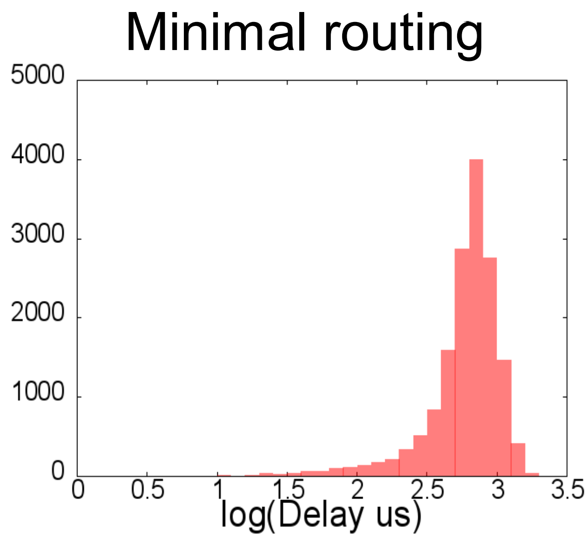


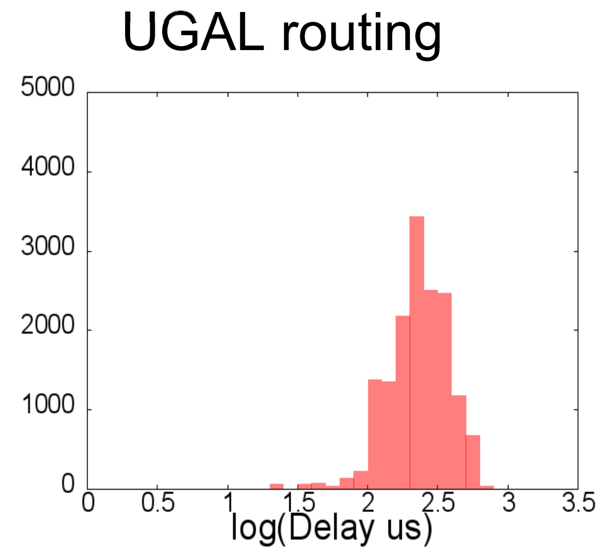
Congestion models based on arbitration/credits for buffers in memory/NIC/switches

Most common usage simulates congestion separately on crossbar and network links (ser/des)

Congestion statistics collected with PISCES

Bin counts for packet latency



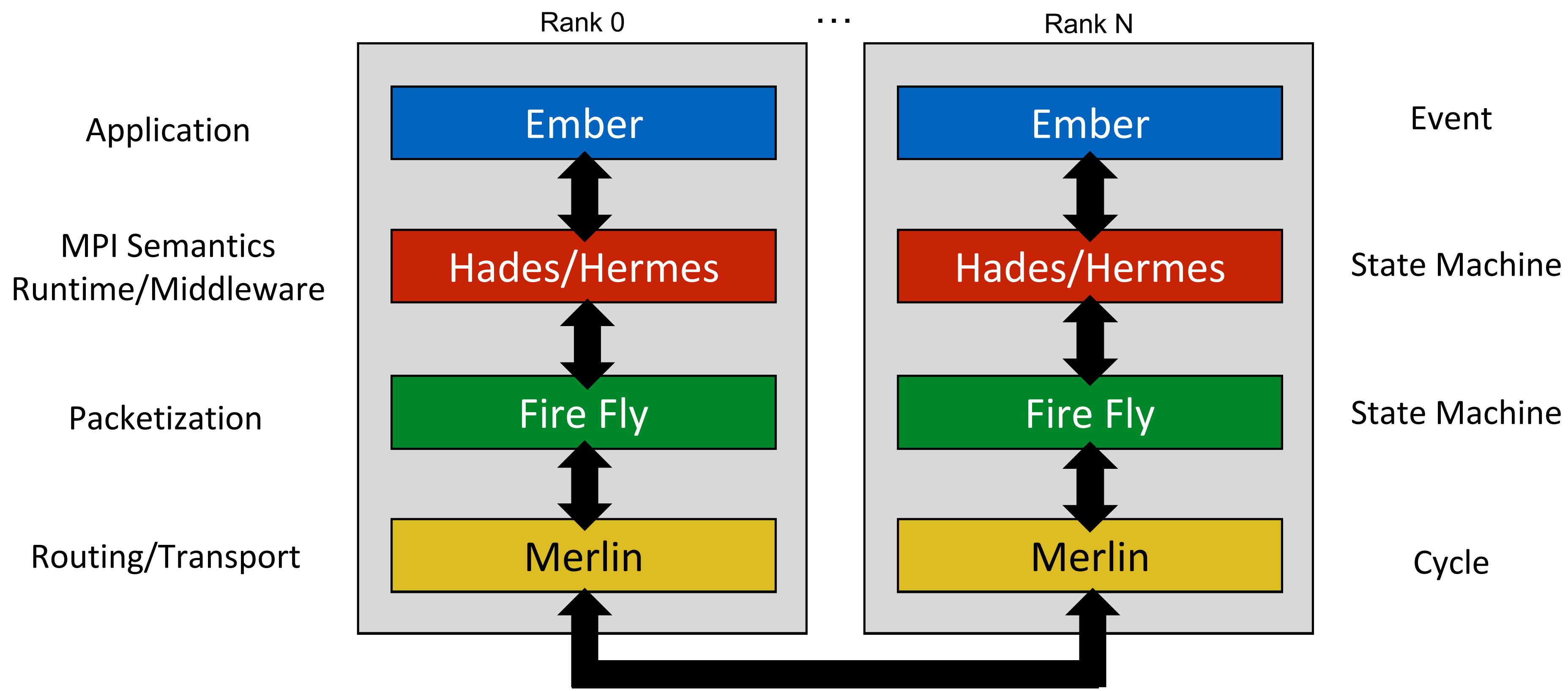


- Adversarial traffic pattern (tornado on 2D torus)
- UGAL takes minimal path until hitting congestion – then Valiant
- Major reduction in packet latencies

Diverse set of network endpoints covering cost/accuracy spectrum

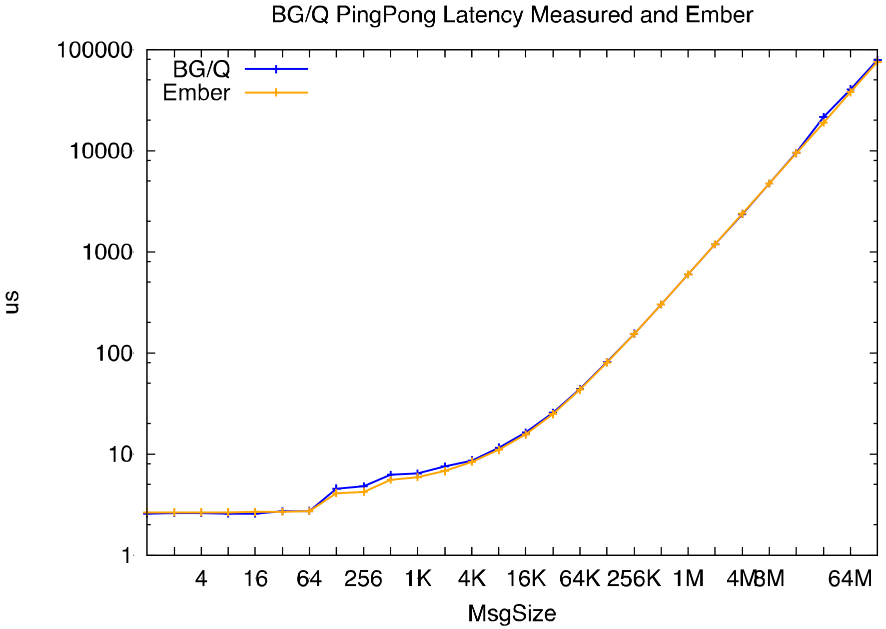
- MPI trace replay
  - DUMPI tool previously in SST/macro
  - Collect MPI calls with timestamps/performance counters
  - Works well with time-scaling. Needs better support for performance counter convolution.
- FOSSIL: Skeleton applications (previously in SST/macro)
  - Framework for Online Software-stack Simulation with Imitated Libraries
  - Compile and link C/C++ code directly into simulator (intercept MPI and compute calls; OSSIfy, operating system symbol interception tool)
  - Can run in emulate mode (execute all instructions/deliver message payloads) or simulate mode (estimate compute/communicate times only)
- Ember
  - State machine model of application
  - Based on common communication/computation motifs or patterns

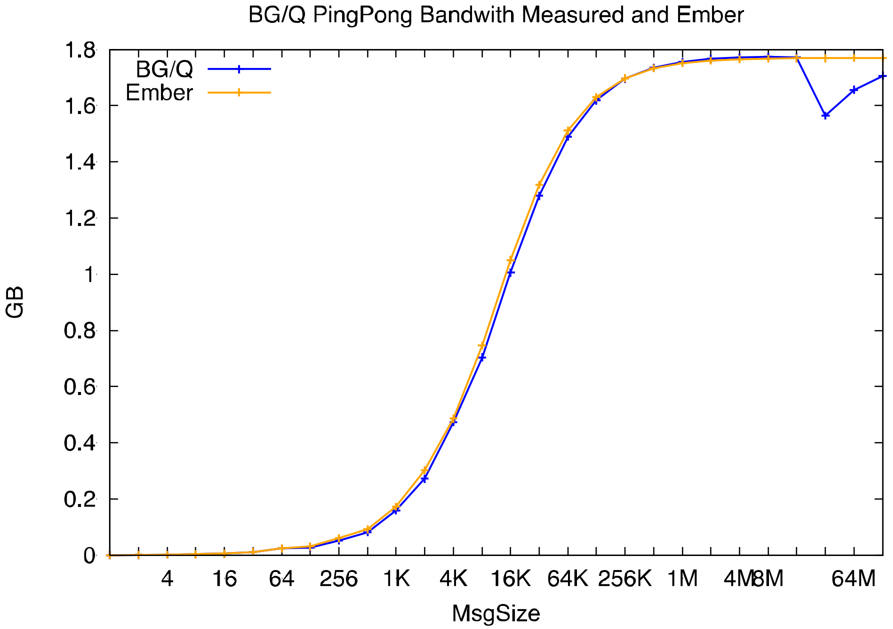
Ember: state machine application models



- Simulated application, middleware, routing and transport
- Completely modular, don’t like it – go swap some out
- Mixed fidelity – pay for the timing complexity you *need*

Ember advertisement





- DOE can not make detailed hardware models (vendors best able)
- DOE needs to provide workloads for simulation components
- DOE can develop basic models, co-design abstract interfaces

FOSSIL: simulation with real software stacks

```
int main(int argc, char** argv){
  MPI_Init(&argc, &argv);
  /* ... Initialization code */
  MPI_Request reqs[NUM_REQUESTS];
  for (int iter=0; iter < niter; ++iter){
    MPI_Isend(...);
    MPI_Isend(...);
    MPI_Irecv(...);
    MPI_Irecv(...);
    dgemm(...);
  }
  /* ... finalize code */
  MPI_Finalize();
  return 0;
}
```

- Compiler wrapper modifies symbols and intercepts calls
- Time delay of MPI calls simulated on virtual network
- -lsstmac_blas intercepts BLAS call, sends to compute model instead of executing natively

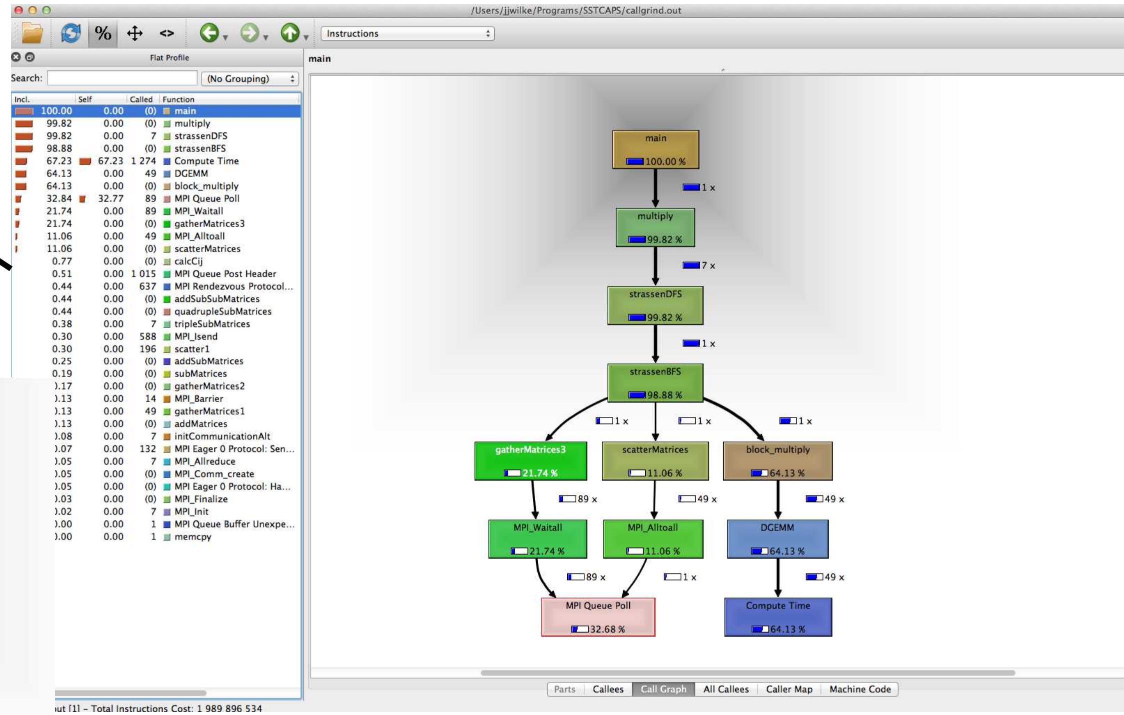
```
:samples jjwilke$ sst++ main.cc -o run -lsstmac_blas
```

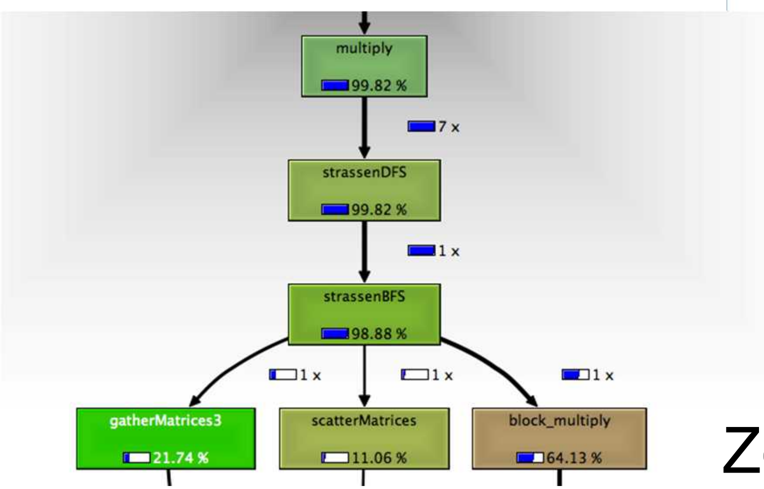
Each process runs simulated as independent user-space threads

	Sim Thread	Process 0	Process 1
$t = 0\mu s$	0)Launch proc 0 2)Launch proc 1	1)Block until send complete	3)Post recv to NIC; block
$t = 1\mu s$	4)Send done; unblock proc 0 6)Deliver msg to NIC 1 ($1\mu s$)	5)Wait for ack; block	
$t = 2\mu s$	7)Recv at NIC 1; unblock proc 1		8)Send ack for recv ($1\mu s$); block
$t = 3\mu s$	9)Deliver ack to NIC 0 ($1\mu s$) 10)Send done; unblock proc 1		11)Continue execution...
$t = 4\mu s$	12)Recv at NIC 0; unblock proc 0	13)Continue execution...	

Call graphs collected with SoftServe

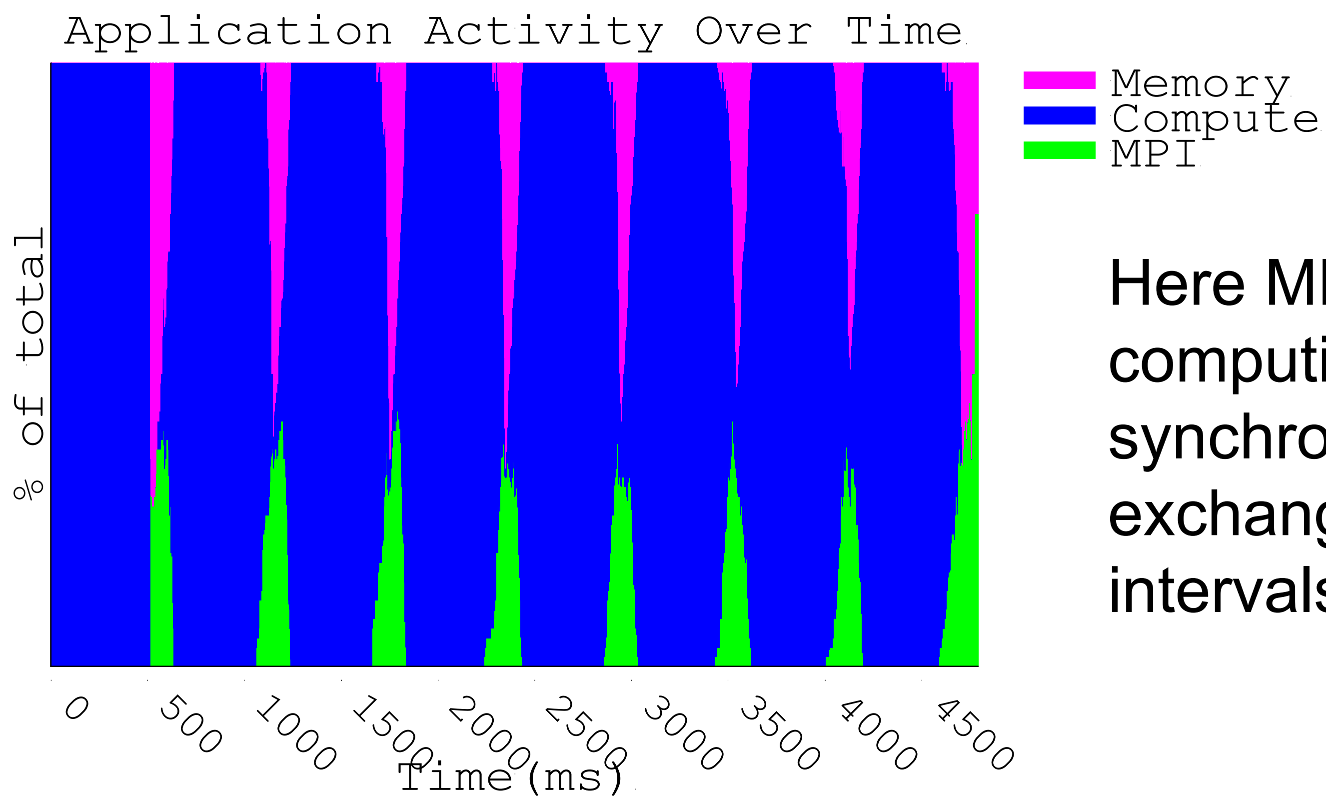
List of all functions and time spent





Zoom in to see functions and what percentage of time was spent there

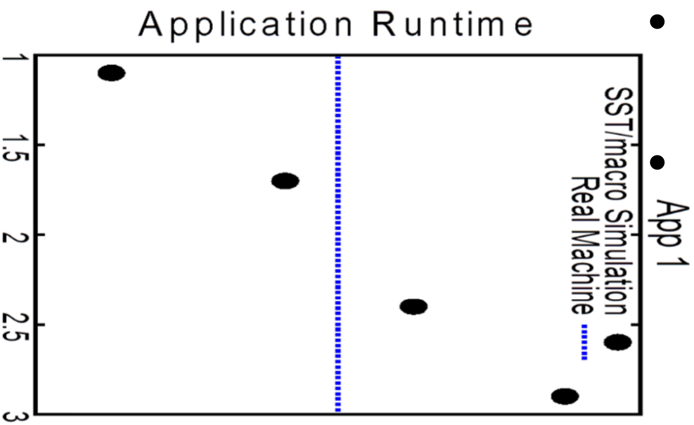
Activity timelines collected with FOSSIL

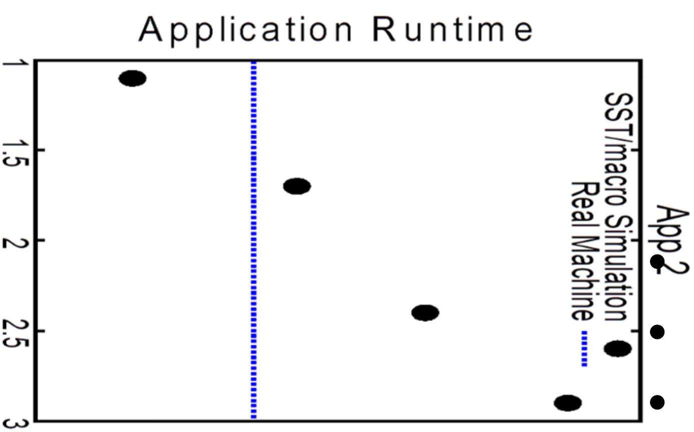


Here MPI stops computing during synchronous exchange at regular intervals

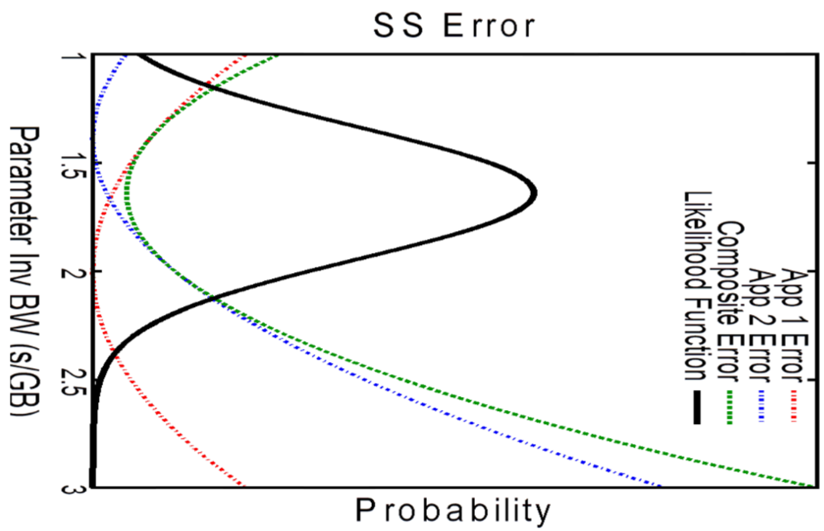
Uncertainty quantification

Sandia National Laboratories



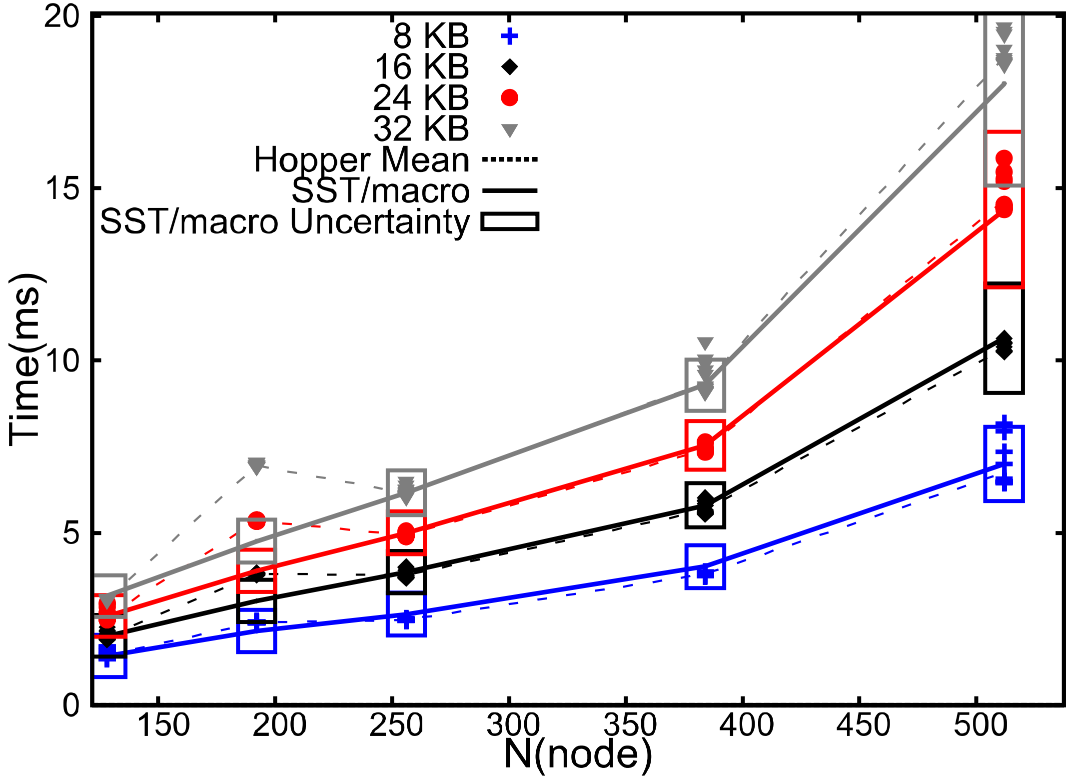


- Cheap approximations are great for rapidly exploring design space, but…
- How much can we actually learn from them?



- App 1 simulation suggests 2.0 s/GB
- App 2 simulations suggests 1.5 s/GB
- No single bandwidth is exact, knowledge of "correct" parameter is probability distribution
- "Most likely" parameter is ~1.6

Collective simulations with error bars



Validation Data

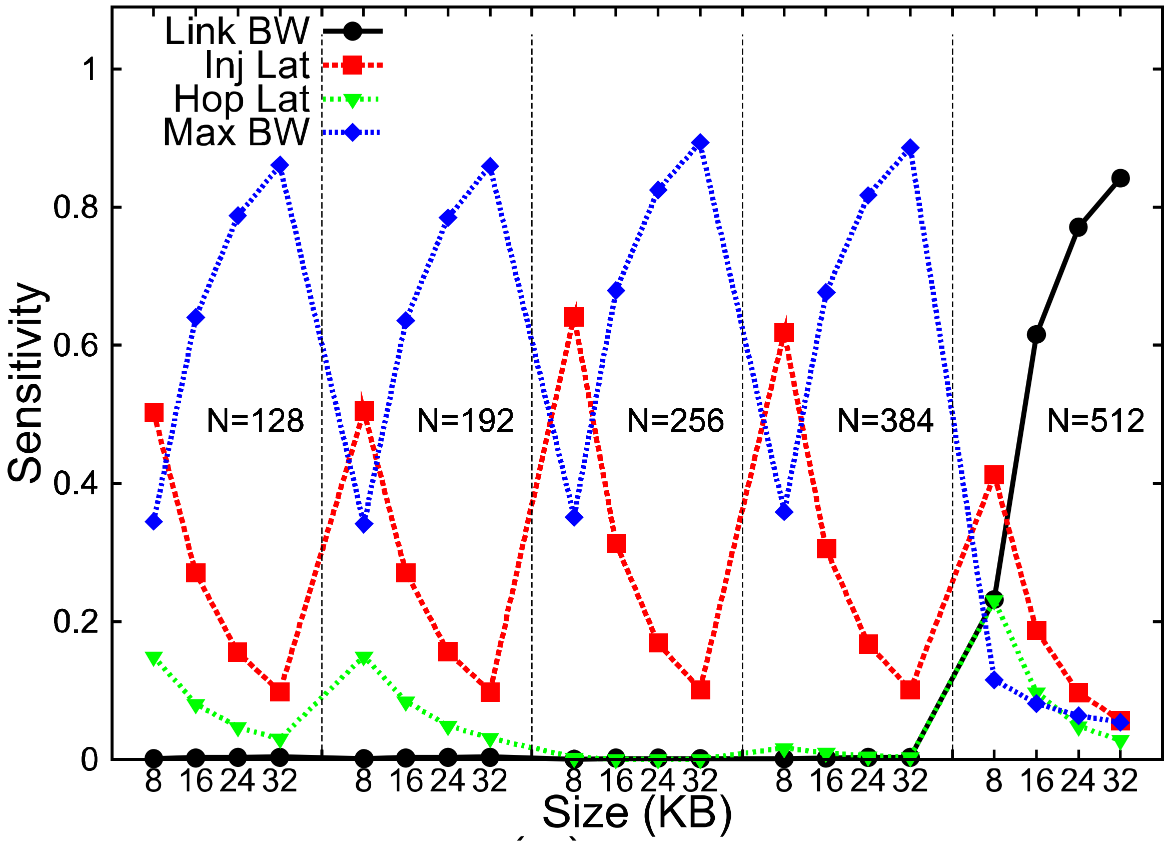
Bayesian Inference

Adaptive Markov Chain Monte Carlo

Results with error bars for all-gather collective collected on Hopper Cray XE6

Error bars determined by scatter in validation data and intrinsic accuracy of coarse-grained models

Collective simulation with sensitivity analysis



Validation Data

Bayesian Inference

Adaptive Markov Chain Monte Carlo

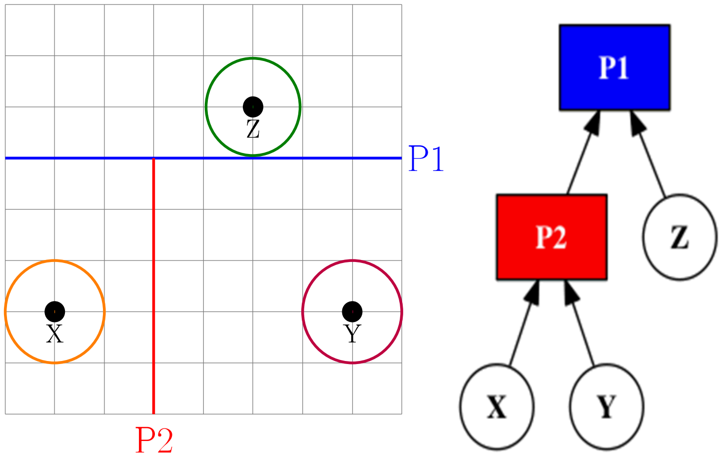
All-gather analysis matches intuition! As message size increases, bandwidth/congestion becomes more important

Black = Congestion parameter
Red = Injection latency
Green = Network hop latency
Blue = Single message bandwidth

Future directions: multiscale simulation

Offline Database	Online Database
Collect all microscale results a priori with brute force parameter scan ☐ Oversample, infeasible for many dimensions ☐ Must select grid/fitting and samples a priori ☑ Workflow integration of micro/macro scales ☑ Embarrassingly parallel, local database	Compute fine-scale results, retrieve pre-computed results if you can ☐ Mixed micro/macro scales in simulation ☐ Results distributed, parallel indexing ☑ Compute many fewer microscale results ☑ No a priori assumptions

Future directions: multiscale simulation



S = Speedup over full microscale
N = Micro kernels w/out reuse
n = Micro kernels with reuse
W = Cost of full kernel
w = Cost of indexing/reuse

$$S = \frac{NW}{nW + Nw}$$

- How much faster can I make my simulation than a "full microscale" simulation by memoizing results?
- How does it affect parallel performance (relative to offline model building with parameter sweep)
- Can my sampling/parallel efficiency be great enough to make multiscale the most cost-efficient option?

SST is about us benefiting from other people’s work, not the other way around

Sandia National Laboratories

- Ultimately simulation work is about building/procuring machines for Department of Energy workloads
- We want to use other people’s models!
  - Years of work goes into debugging/calibrating tools
  - Not enough staff or hours to build all the modeling infrastructure we want
  - Don’t have access to proprietary details in many cases
- We want to provide a specific path for integrating models
  - Unified PDES core
  - SST integration is lightweight and (mostly) non-intrusive
  - gem5 and SST/macro can as stand-alone or part of integrated core
  - Co-designing flexible abstract interfaces for tying independent components together

Acknowledgments

Sandia National Laboratories

This work was supported by the U.S. Department of Energy (DOE) National Nuclear Security Administration (NNSA) Advanced Simulation and Computing program and the DOE Office of Advanced Scientific Computing Research. SNL is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the DOE NNSA under contract DE-AC04-94AL85000.

DEPARTMENT OF ENERGY • UNITED STATES OF AMERICA •

NNSA

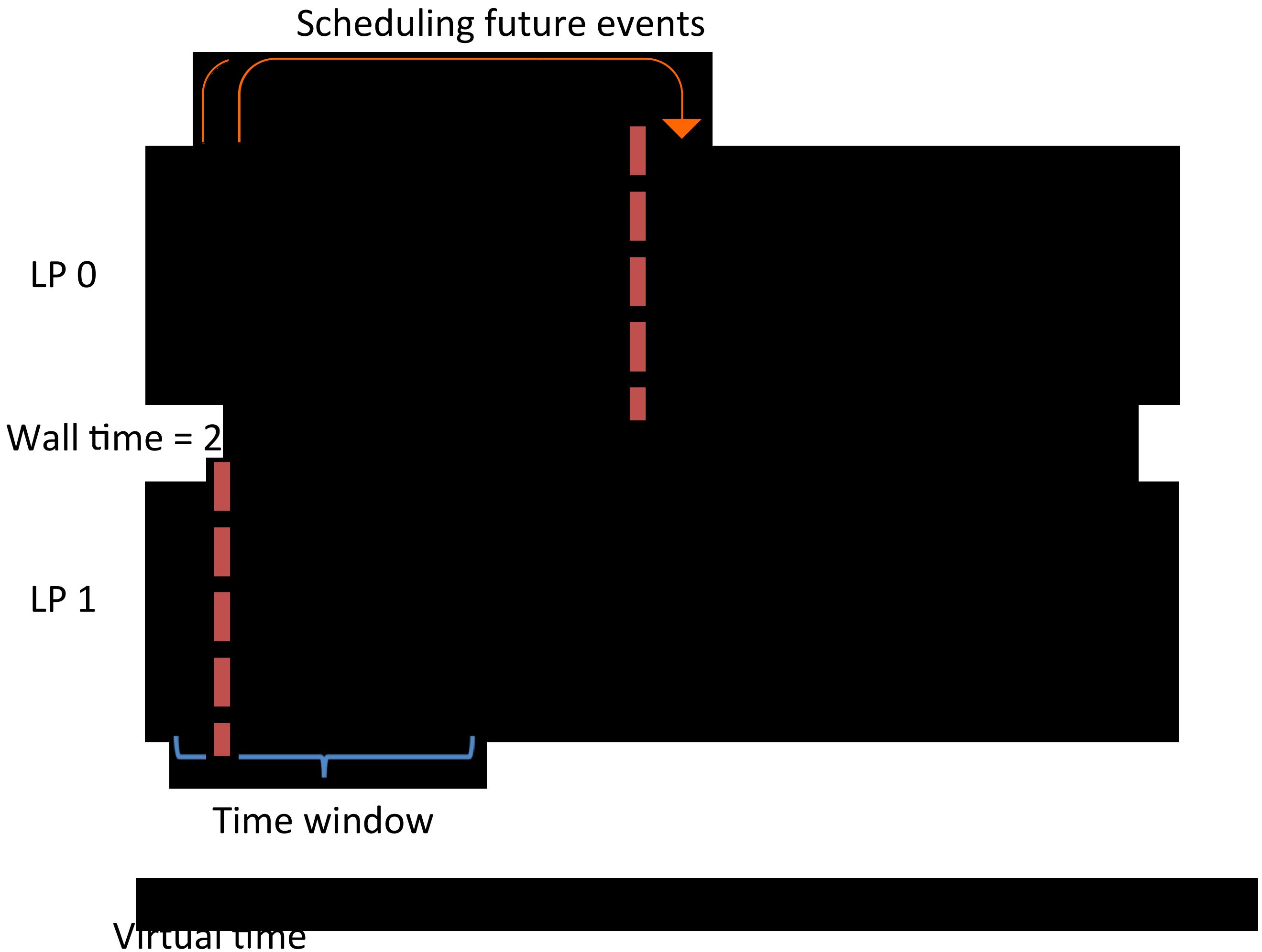
National Nuclear Security Administration

Sandia National Laboratories

Sandia National Laboratories

Exceptional service in the national interest

http://sst-simulator.org


Scheduling future events
LP 0
Wall time = 2
LP 1
Time window
Virtual time