Large Scale Simulations Enabled by BigSim

Dr. Gengbin Zheng and Ehsan Totoni

Parallel Programming Laboratory
University of Illinois at Urbana-Champaign

April 18, 2011

What is BigSim

- A function level simulator for parallel applications on peta-scale machine
- An emulator runs applications at full scale
- A simulator predicts performance by simulating message passing

What's New Last Year

- Improved simulation framework from these aspects
  - Even larger/faster emulation and simulation
  - More accurate prediction of sequential performance
- Apply BigSim to predict Blue Waters machine

Emulation at Large Scale

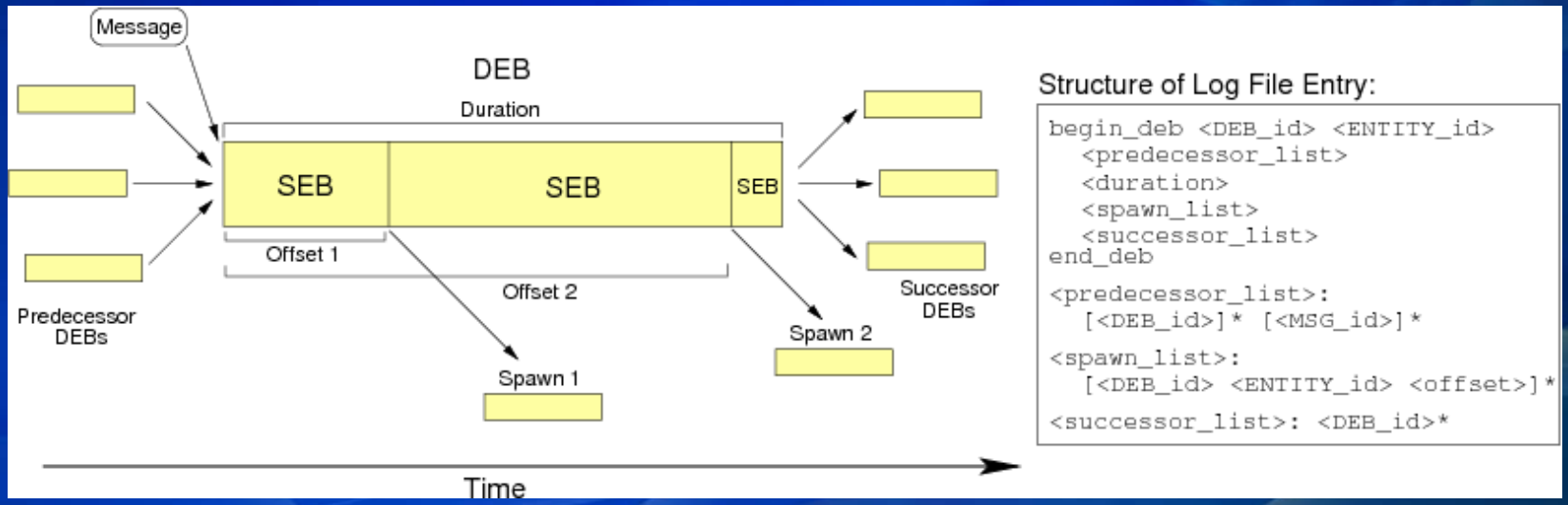
- Using existing (possibly different) clusters for emulation
- Applied memory reduction techniques
- For example, we achieved for NAMD:
  - Emulate 10M-atom benchmark on 256K cores using only 4K cores of Ranger (2GB/core)
  - Emulate 100M-atom benchmark on 1.2M cores using only 48K cores of Jaguar (1.3GB/core)

Accurate Prediction of Sequential Execution Blocks

- Predicting parallel applications can be challenging
  - Accurate prediction of sequential execution blocks
  - Network performance
- This paper explores techniques to use statistical models for sequential execution blocks

Gengbin Zheng, Gagan Gupta, Eric Bohm, Isaac Dooley, and Laxmikant V. Kale, "Simulating Large Scale Parallel Applications using Statistical Models for Sequential Execution Blocks", in the Proceedings of the 16th International Conference on Parallel and Distributed Systems (ICPADS 2010)

Dependent Execution Blocks and Sequential Execution Blocks (SEB)



Structure of Log File Entry:

```
begin_deb <DEB_id> <ENTITY_id>
  <predecessor_list>
  <duration>
  <spawn_list>
  <successor_list>
end_deb

<predecessor_list>:
  [<DEB_id>]* [<MSG_id>]*

<spawn_list>:
  [<DEB_id> <ENTITY_id> <offset>]*

<successor_list>: <DEB_id>*
```

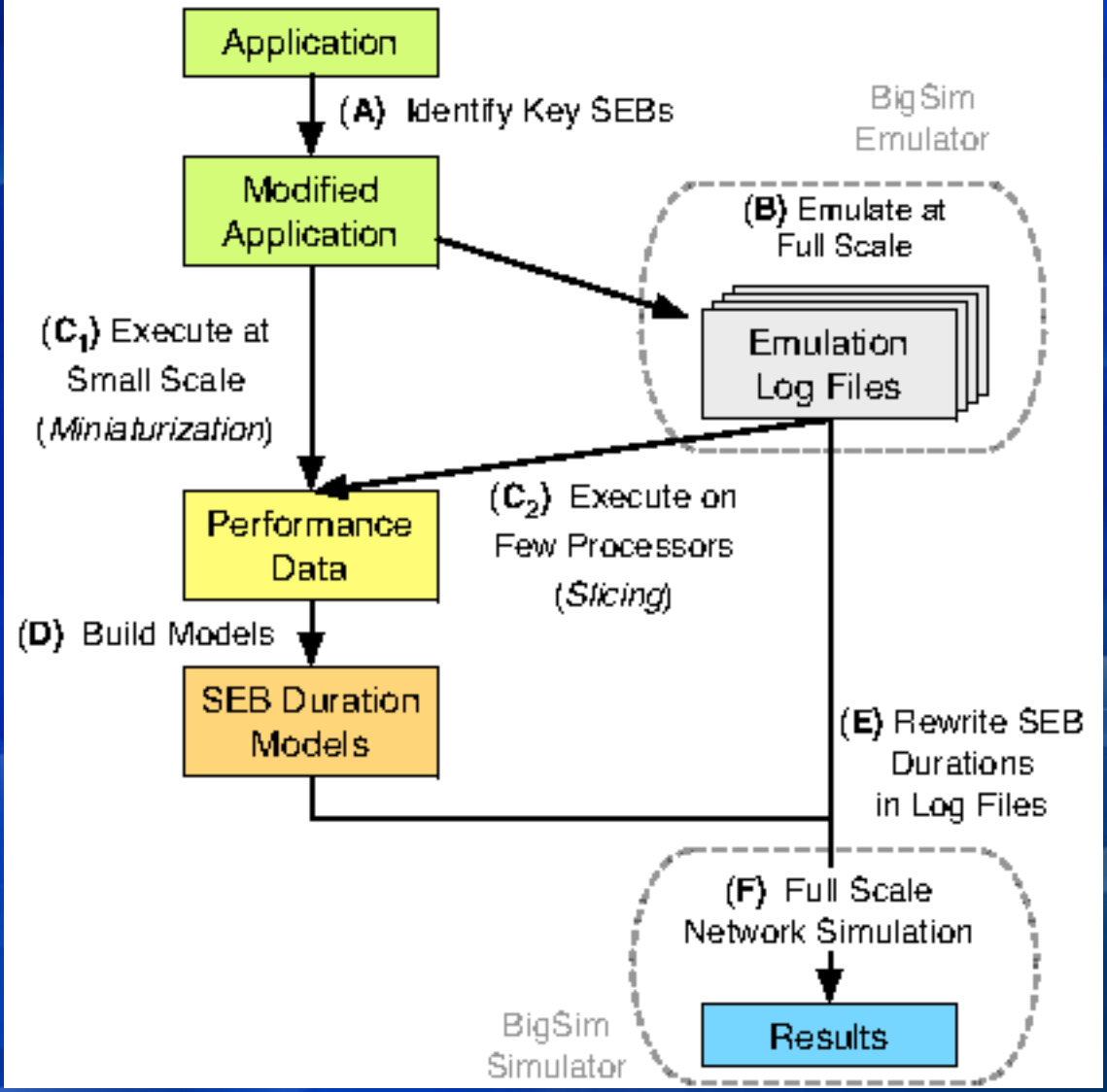
Old Ways to Predict SEB time

- CPU scaling
  - Take execution on one machine to predict another
  - not accurate
- Hardware simulator
  - Cycle accurate timing
  - However it is slow
- Performance counters
  - Very hard, and platform specific
- An efficient and accurate prediction method?

Basic Idea – Machine learning

- Identify parameters that characterize the execution time of an SEB:
  - $x_1, x_2, \ldots, x_n$
- With machine learning, build statistical model of the execution time:
  - $T=f(x_1,x_2,\ldots,x_n)$
- Run SEBs in realistic scenario and collect data points
  - $X_{11},x_{21},\ldots,x_{n1}$ => T1
  - $X_{12},x_{22},\ldots,x_{n2}$ => T2
  - …

SEB Prediction Scheme (6 steps)



Data Collection at Small Scale

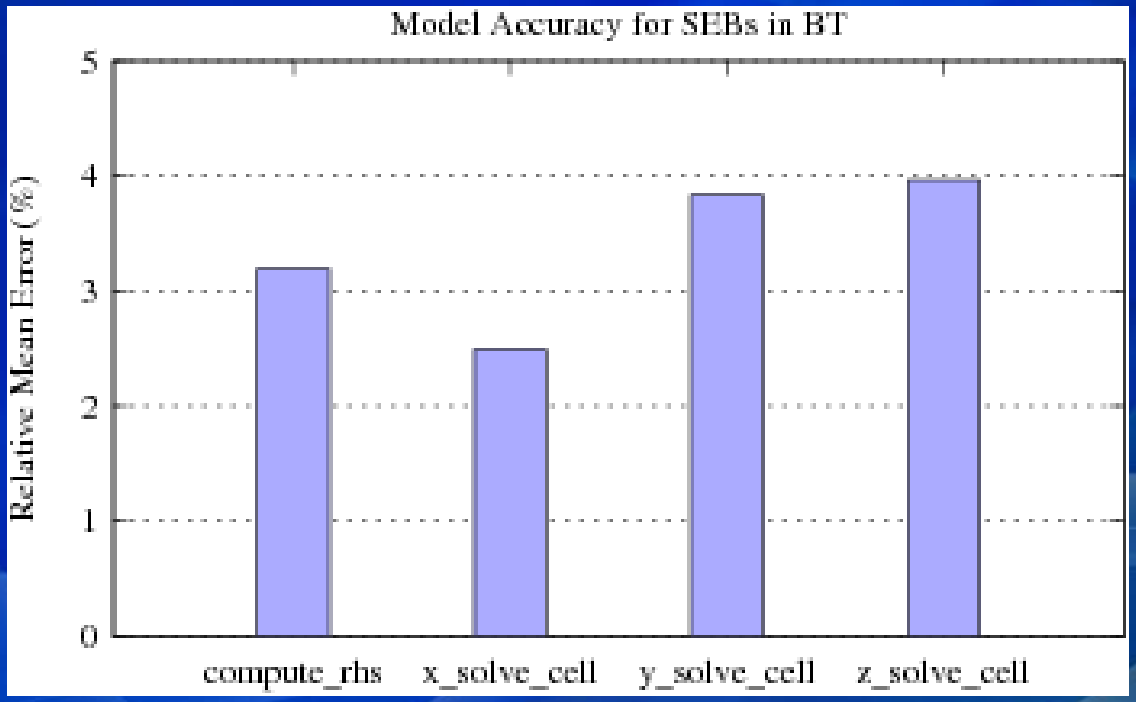
- Slicing (C_2)
  - Record and replay any processor
  - Instrument exact parameters
  - Require binary compatible architecture
- Miniaturization (C_1)
  - Scaled down execution (smaller dataset, smaller number of processors)
  - Not all application can scale down

Build Prediction Models (D)

- Machine learning techniques
  - Linear Regression
  - Least median squared linear regression
  - SVMreg
- Use machine learning software like weka
- As an example:
  - $T(N) = -0.036N^2 + 0.009N^3 + 12.47$

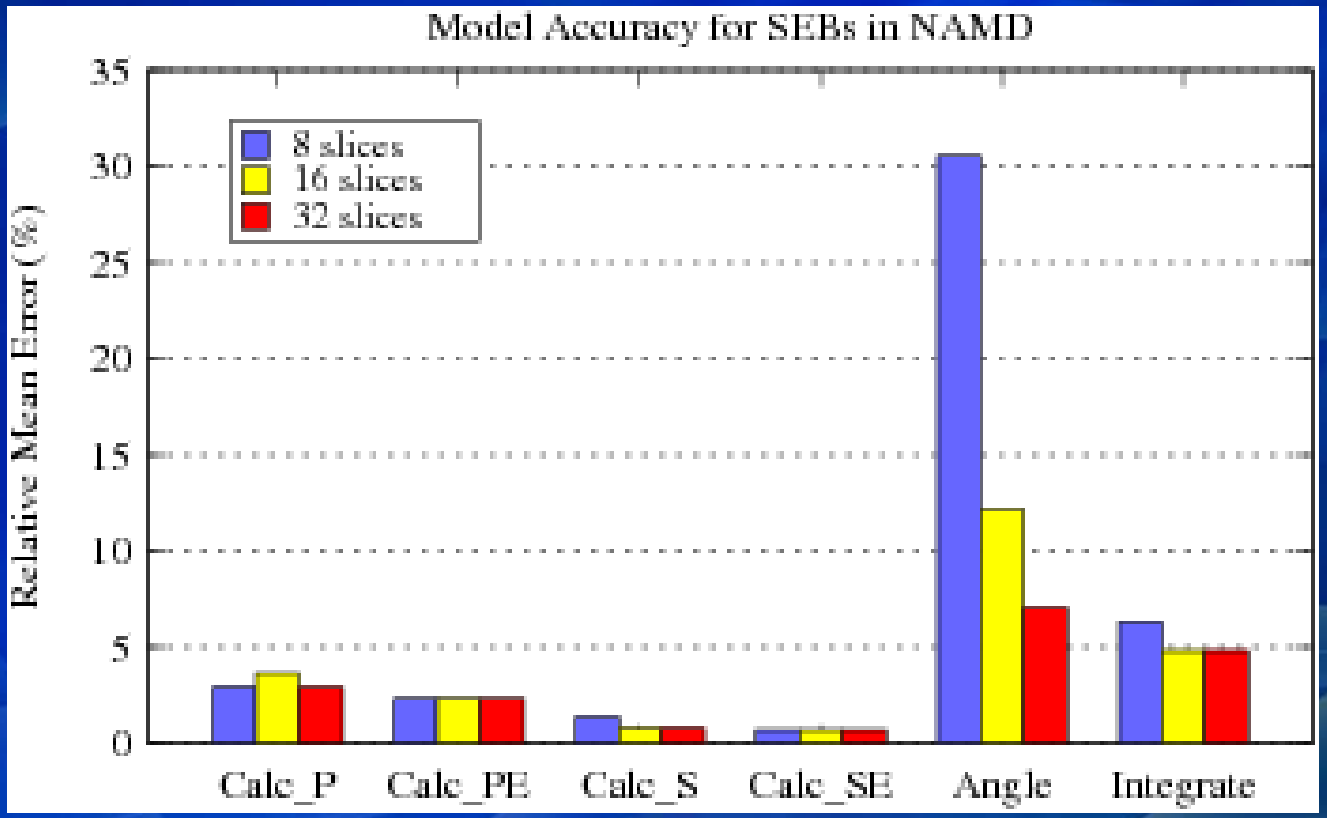
Validation – NAS BT Benchmark

Use Abe to predict Ranger



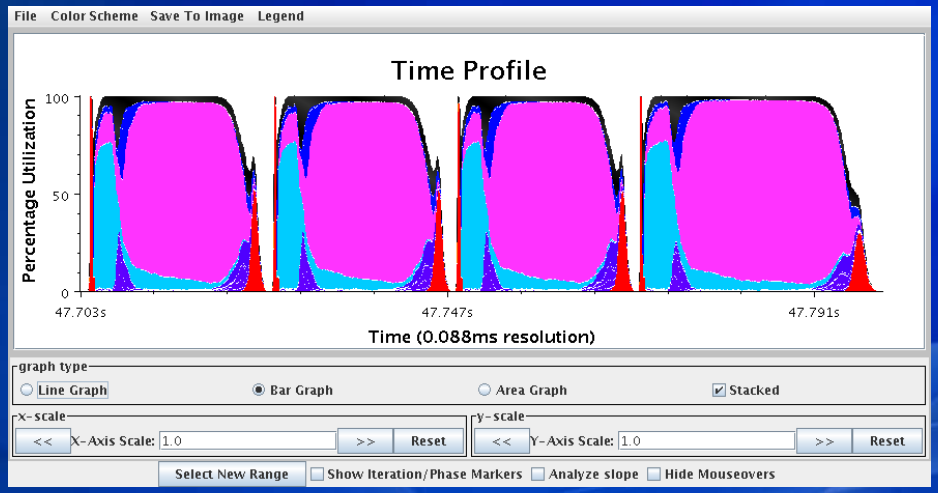
NAS BT Benchmark BT.D.1024
emulate 64 core Abe, recording 8 target cores
replay on Ranger to predict

NAMD Prediction

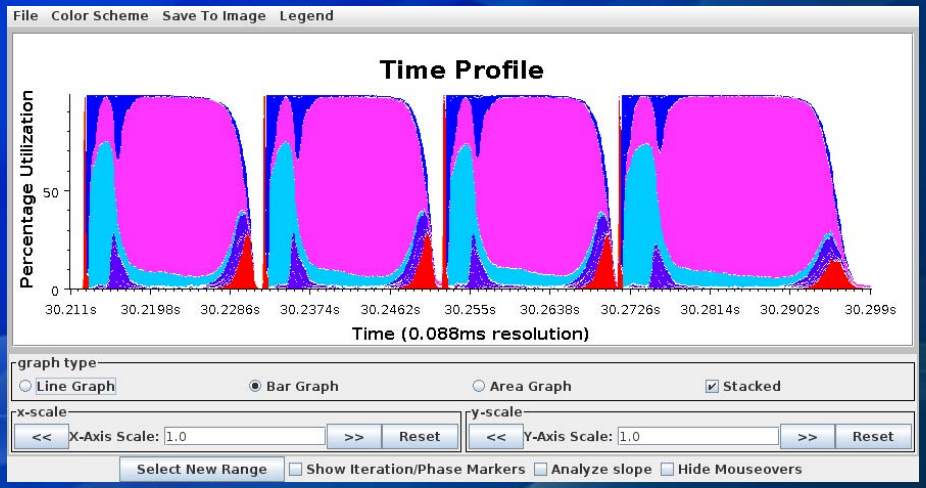


NAMD STMV 1M-atom benchmark
4096 core prediction
Using 512 core of BG/P

NAMD Prediction (STMV 4096 core)



Native



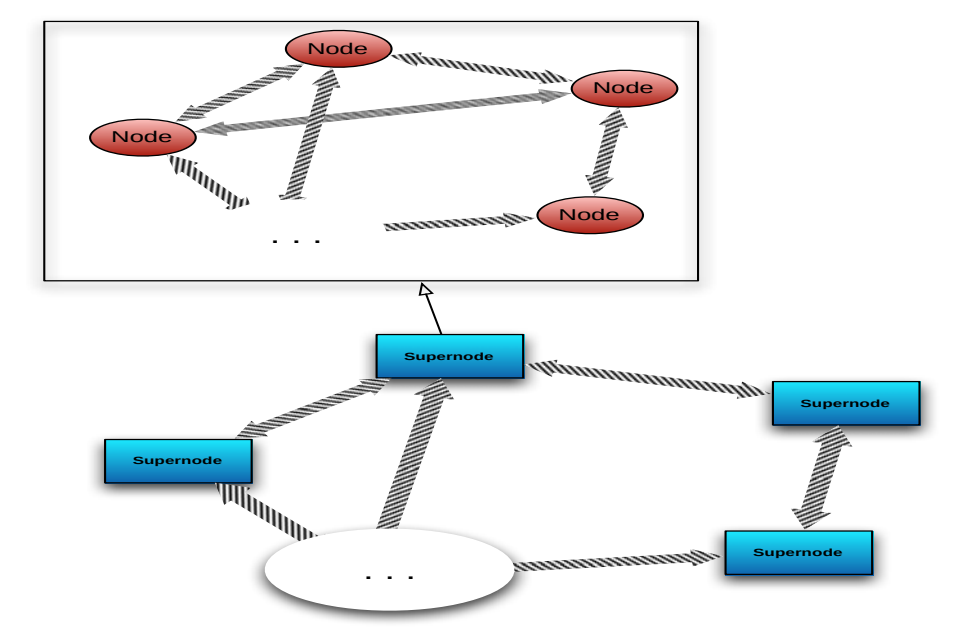
Prediction

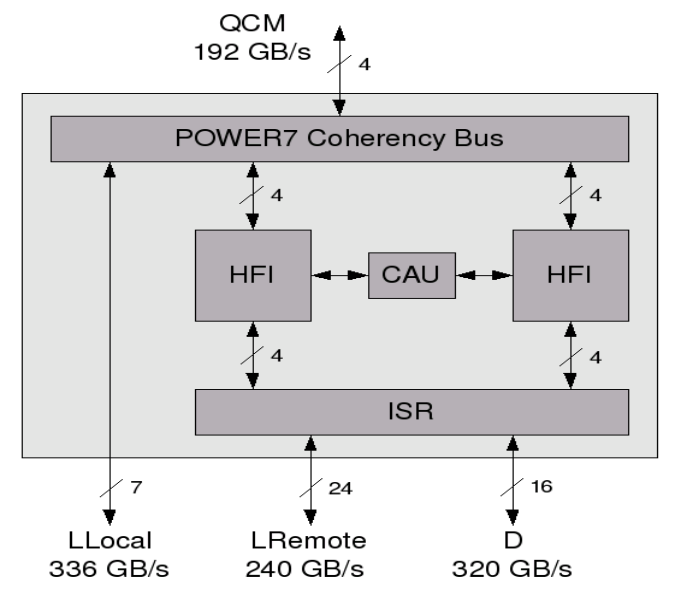
Blue Waters

- Will be here soon
- More than 300K POWER7 Cores
- PERCS Network
  - Unique network not similar to any other!
  - No theory or legacy for it
  - Applications and runtimes tuned for other networks
- NSF Acceptance: Sustained petaFLOPs for real applications
  - Tuning applications typically takes months to years after machines become online

PERCS Network

- Two-level fully-connected
- Supernodes with 32 nodes
- 24 GB/s LLocal (LL) link, 5 GB/s LRemote (LR) link, 10 GB/s D link, 192 GB/s to IBM HUB Chip



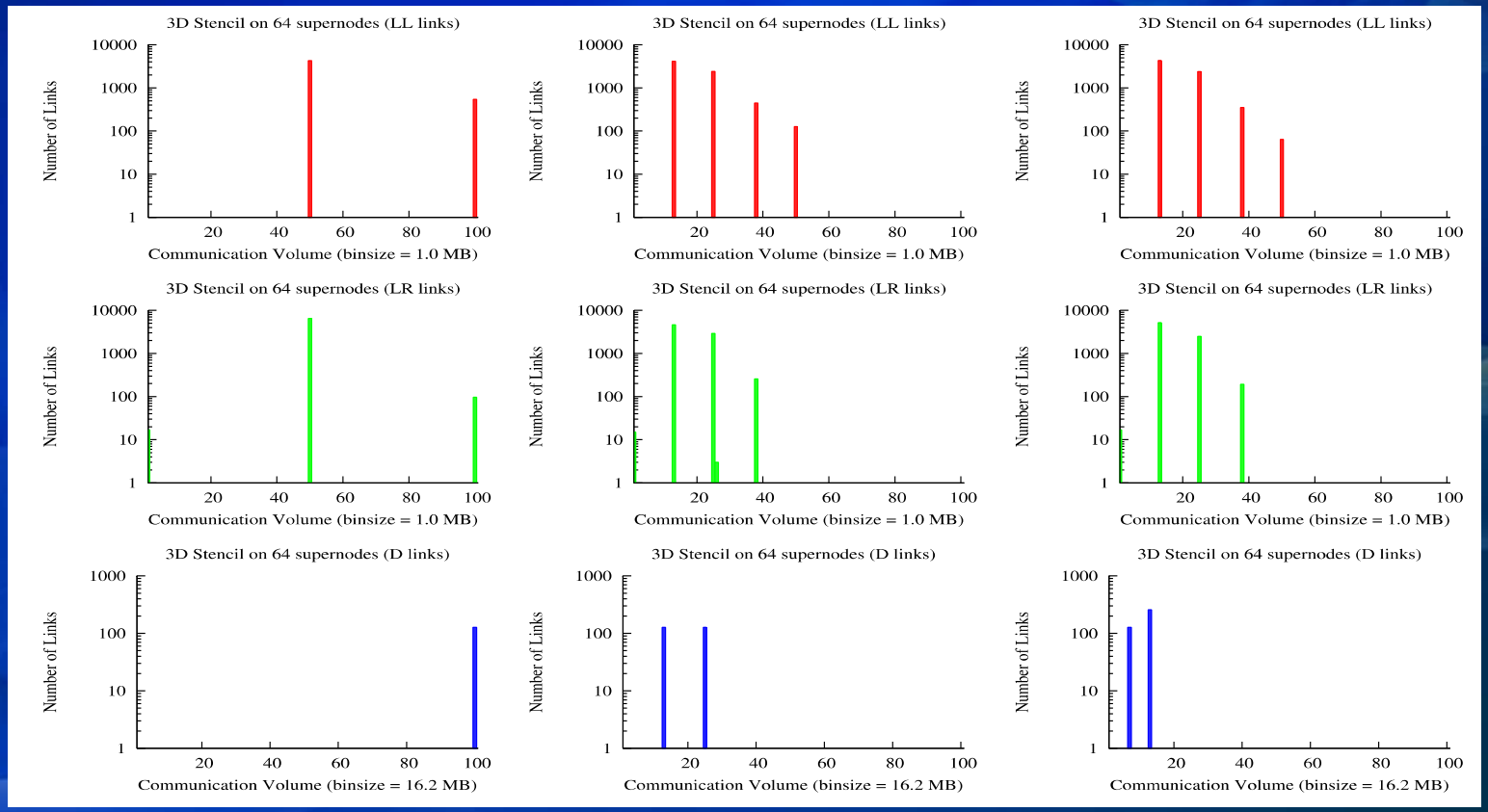


BigSim for Blue Waters

- Detailed packet level network model developed for Blue Waters
- Validated against MERCURY and IH-Drawer
  - Ping Pong within 1.1% of MERCURY
  - Alltoall within 10% of drawer
- Optimizations for that scale
- Different outputs (link statistics, projections…)
- Some example studies: (E. Totoni et al., submitted to SC11)
  - Topology-aware mapping
  - System noise
  - Collective optimization

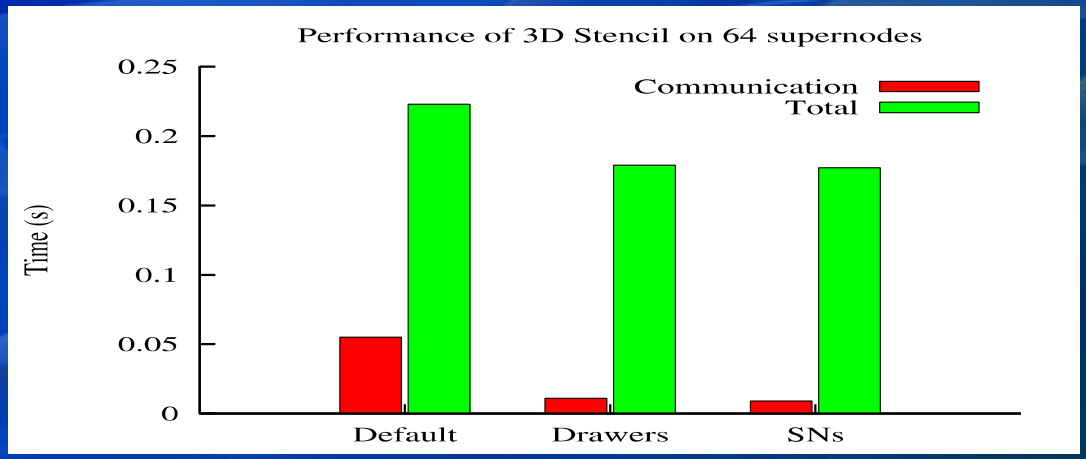
Topology-Aware Mapping

- Apply different mappings in simulation
- Evaluate using output tools (Link Utilization…)



Topology-Aware Mapping

- Example: Mapping of 3D stencil
  - Default mapping: rank ordered mapping of MPI tasks to cores of nodes
  - Drawer mapping: 3D cuboids on nodes and drawers
  - Supernode mapping: 3D cuboids on supernodes also; 80% communication decrease, 20% overall time



System Noise

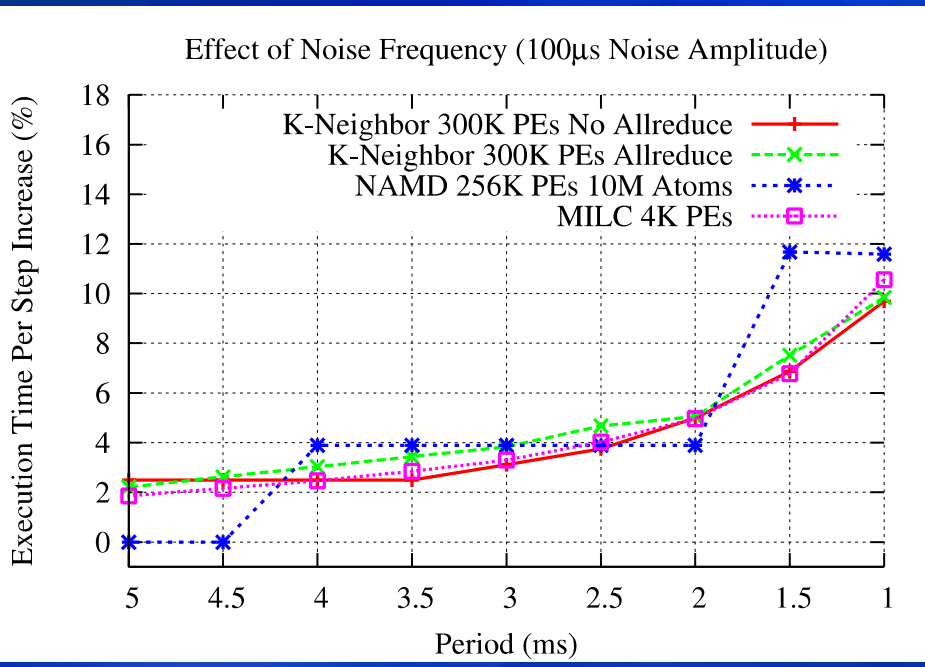
- BigSim's noise injection support
  - Captured noise of a node with the same architecture
    - Replicate on all the target machines nodes with random offsets (not synchronized)
  - Artificial periodic noise patterns for each core of a node
    - With periodicity, amplitude and offset (phase)
    - Replicate on all nodes with random offsets
- Increases length of computations appropriately

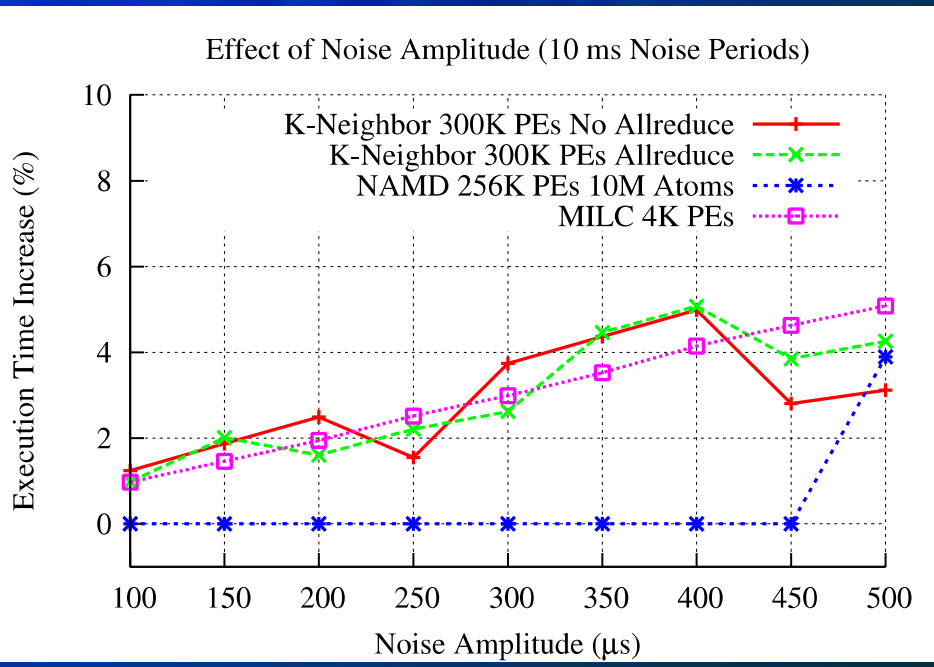
Noise Studies

- Applications:
  - NAMD: 10 Million atom system on 256K cores
  - MILC: su3_rmd on 4K cores
  - K-Neighbor with Allreduce 1 ms sequential computation, 2.4 ms total iteration time
  - K-Neighbor without Allreduce
- Inspecting with different noise frequencies and amplitudes

Noise Studies

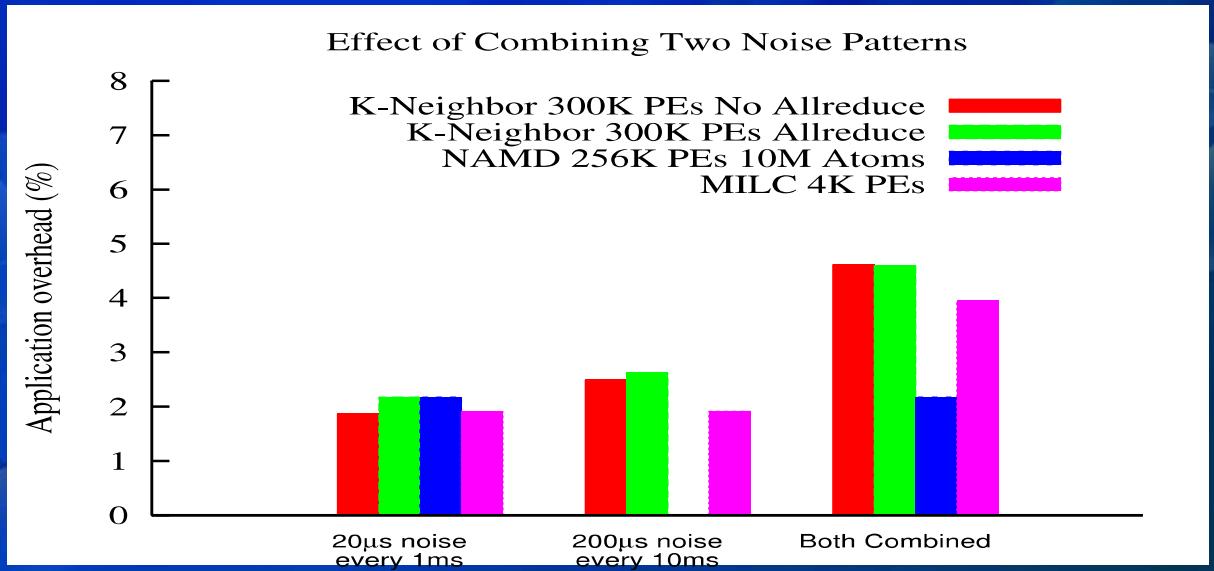
- NAMD is tolerant, but jumps around certain frequencies
- MILC is sensitive even in small jobs (4K cores), almost as much affected as large jobs





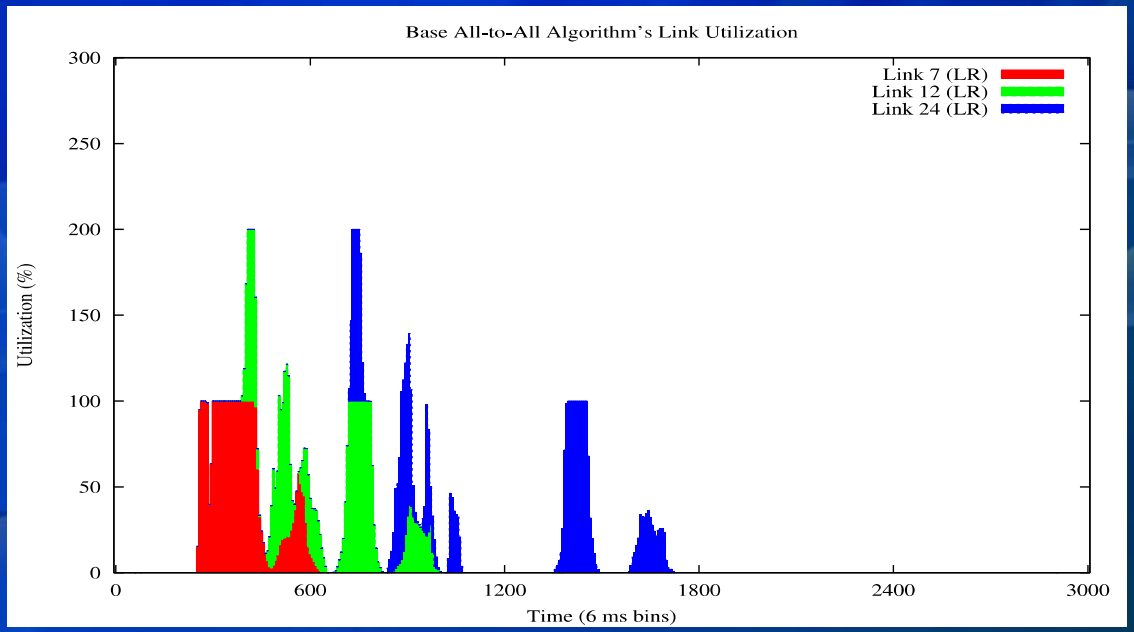
Noise Studies

- Simple models are not enough
- In high frequencies there is chance that small MPI calls for collectives get affected
  - Delaying execution significantly
- Combining noise patterns:



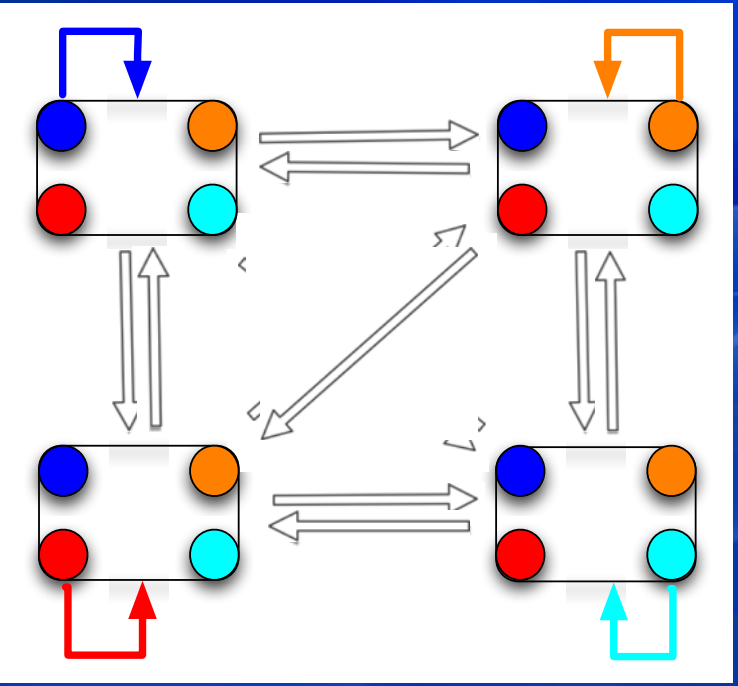
MPI_Alltoall Optimization

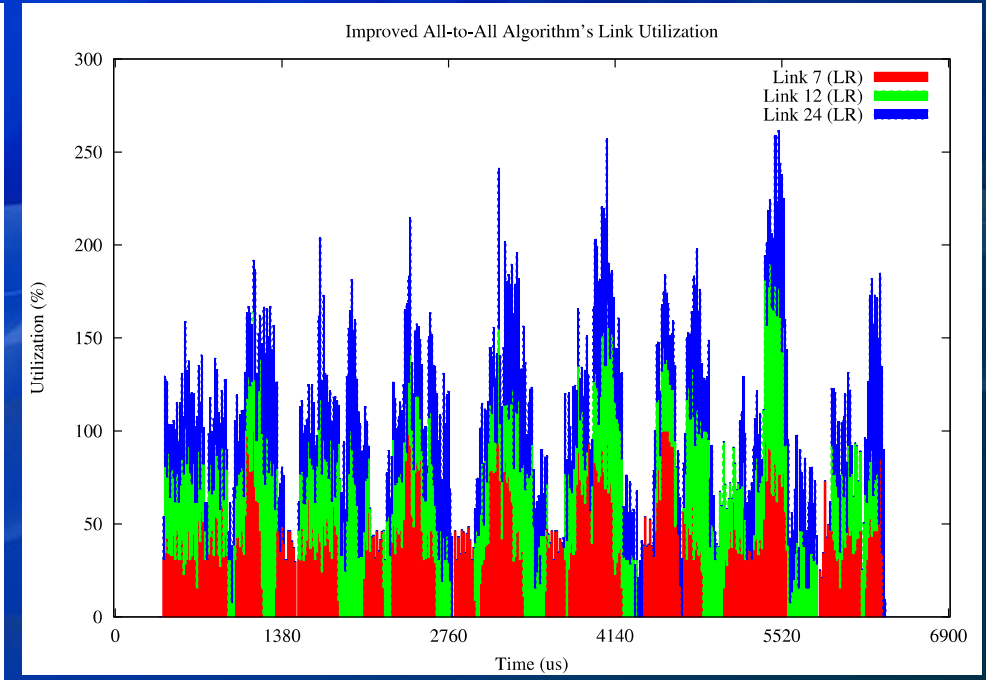
- Optimized Alltoall for large messages in SN
- BigSim's link utilization output gives the idea:
  - Outgoing links of a node are not used effectively at all times during Alltoall
  - Shifted usage of different links



MPI_Alltoall Optimization

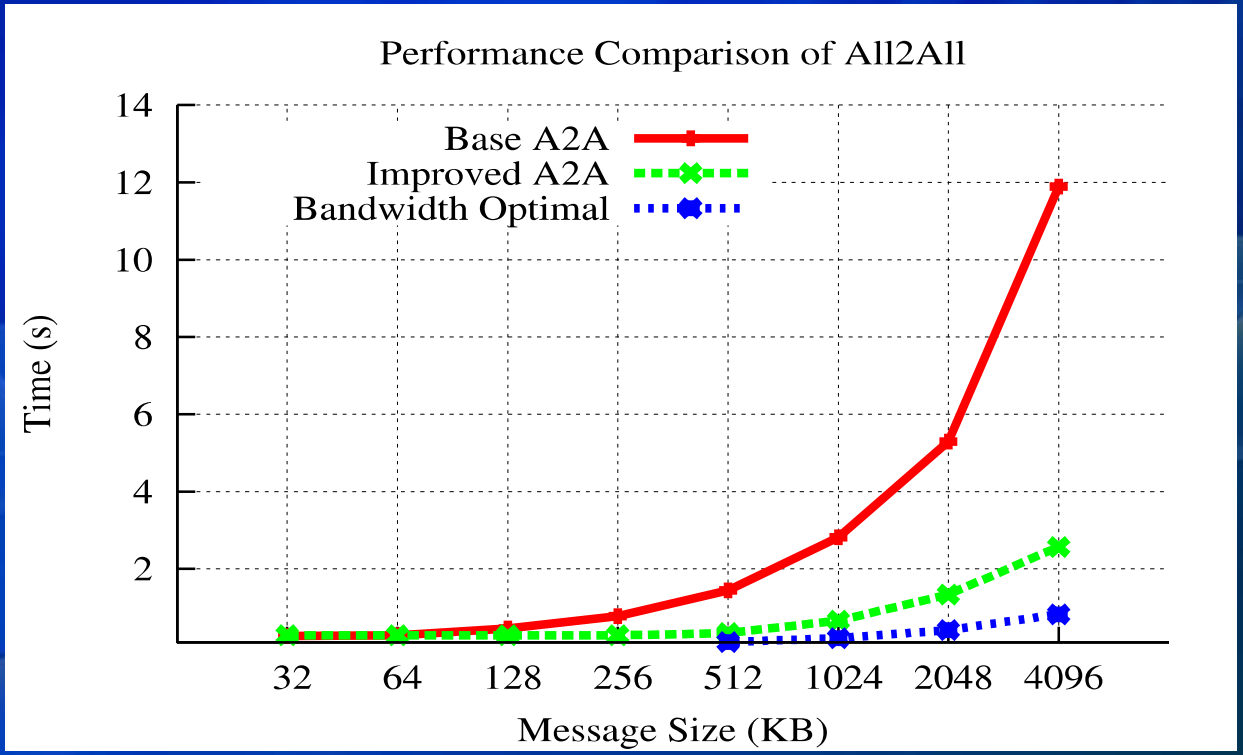
- New algorithm: each core send to different node
  - Use all 31 outgoing links
  - Link utilization now looks better





MPI_Alltoall Optimization

- Up to 5 times faster than existing Pairwise-Exchange scheme
- Much closer to theoretical peak bandwidth of links



Questions?