

Simulation of Large-Scale HPC Architectures

Ian S. Jones and Christian Engelmann

Computer Science Research Group
Computer Science and Mathematics Division
Oak Ridge National Laboratory, USA

Trends in HPC System Design

- Ongoing trends in HPC system design:
 - Increasing core counts (in total and per processor)
 - Increasing node counts (OS instances)
 - Heterogeneity (CPU+GPGPU at large scale)
- Emerging technology influencing HPC system design:
 - Stacked memory (3D chip layering)
 - Non-volatile memory (SSD and phase change memory)
 - Network-on-chip
- Additional forces influencing HPC system design:
 - Power consumption ceiling (overall and per-chip)
- *How to design HPC systems to fit application needs?*
- *How to design applications to efficiently use HPC systems?*

Current-Generation HPC Systems

- Large-scale 1 PFlop/s systems are here
 - #1 RIKEN K: 8.162 PFlop/s, 548,352 cores, 93%
 - #2 NSCT Tianhe-1A: 2.566 PFlop/s, 186,368 cores, 55%
 - #3 *ORNL Jaguar XT5: 1.759 PFlop/s, 224,162 cores, 75%*
 - #4 NSCS Nebulae: 1.271 PFlop/s, 120,640 cores, 43%
 - #5 GSIC Tsubame 2.0: 1.192 PFlop/s, 73,278 cores, 61%
 - #5 LANL Cielo: 1.110 PFlop/s, 142,272 cores, 81%
 - #6 NASA Pleiades: 1.088 PFlop/s, 111,104 cores, 81%
 - #7 LBNL Hopper: 1.054 PFlop/s, 153,408 cores, 82%
- The trend is toward even larger-scale systems
 - End of processor frequency scaling → Node/core scaling

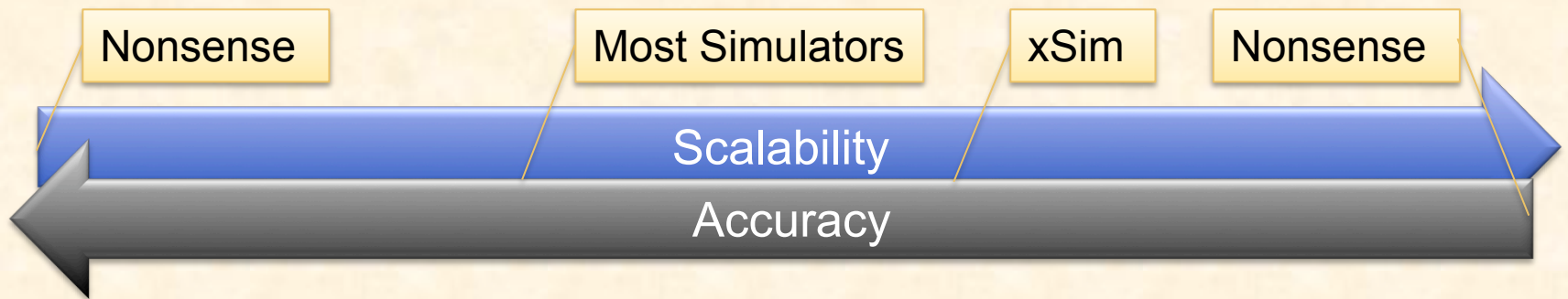
Exascale Road Map (Constantly Changing)

Many design factors are driven by the power ceiling of 20MW

Systems	2009	2011	2015	2018
System peak	2 Peta	20 Peta	100-200 Peta	1 Exa
System memory	0.3 PB	1.6 PB	5 PB	10 PB
Node performance	125 GF	200GF	200-400 GF	1-10TF
Node memory BW	25 GB/s	40 GB/s	100 GB/s	200-400 GB/s
Node concurrency	12	32	O(100)	O(1000)
Interconnect BW	1.5 GB/s	22 GB/s	25 GB/s	50 GB/s
System size (nodes)	18,700	100,000	500,000	O(million)
Total concurrency	225,000	3,200,000	O(50,000,000)	O(billion)
Storage	15 PB	30 PB	150 PB	300 PB
IO	0.2 TB/s	2 TB/s	10 TB/s	20 TB/s
MTTI	days	days	days	O(1 day)
Power	6 MW	~10MW	~10 MW	~20 MW

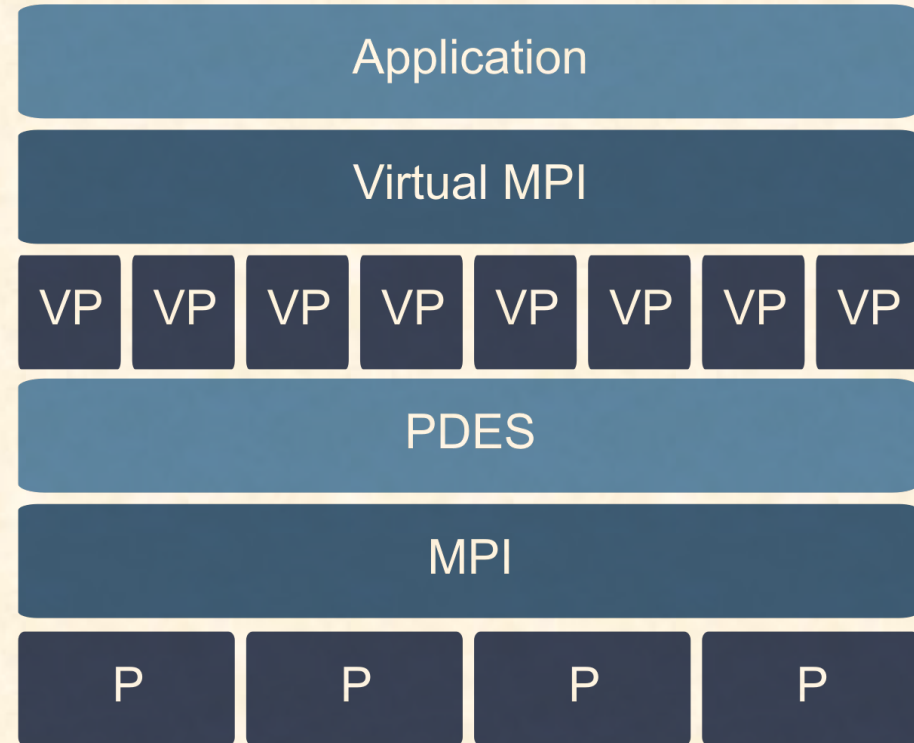
Facilitating HPC Hardware/Software Co-Design Through Simulation

- Parallel discrete event simulation (PDES) to emulate the behavior of future architecture choices
- Execution of real applications, algorithms or their models atop a simulated HPC environment for:
 - Performance evaluation, including identification of resource contention and underutilization issues
 - Investigation at extreme scale, beyond the capabilities of existing simulation efforts
- xSim: Highly scalable solution that trades off accuracy



xSim – The Extreme-scale Simulator

- Combining highly oversubscribed execution, a virtual MPI, and a time-accurate PDES
- PDES uses the native MPI and simulates virtual processors
- The virtual processors expose a virtual MPI to applications
- Applications run within the context of virtual processors:
 - Global and local virtual time
 - Execution on native proc.
 - Local or native MPI comm.
 - Processor/network model



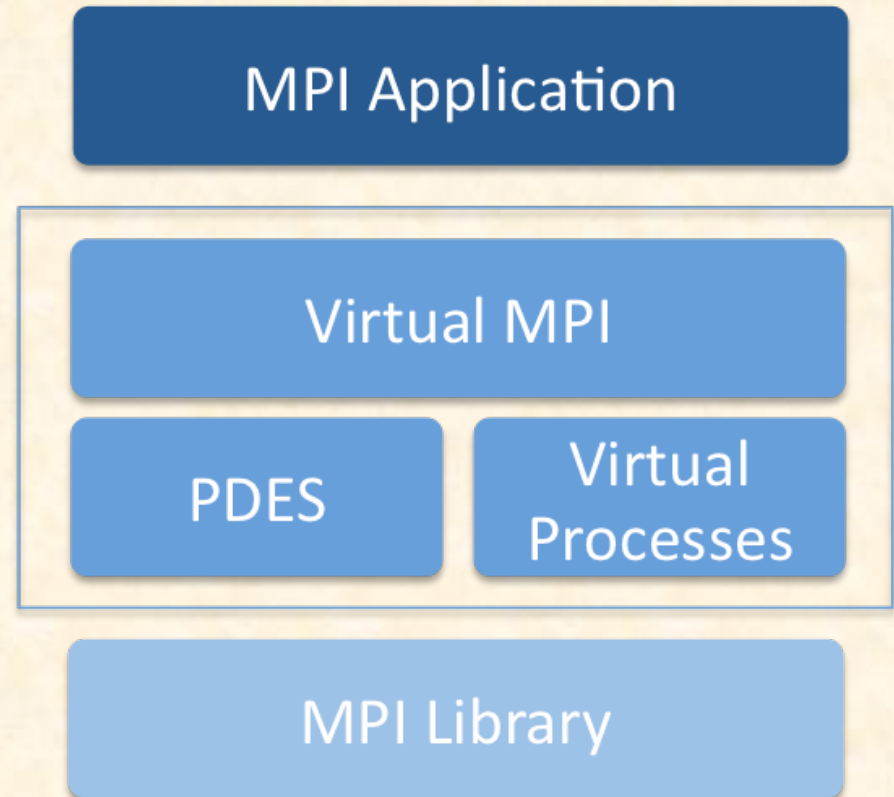
© 2010 IEEE Cluster Co-Design Workshop

xSim Implementation: Overview

- The simulator is a library
- Utilizes PMPI to intercept MPI calls and to hide the PDES
- Easy to use:
 - Include the xSim header
 - Compile and link with the xSim library
 - Run the MPI program:

`mpirun -np <np> <prog> -xsim-np <vp>`

- C with 2 threads/native proc.
- Support for C and Fortran MPI applications



© 2010 IEEE Cluster Co-Design Workshop

Implementation Details (1/3)

- **PDES**

- Maintains virtual time for each VP equivalent to execution time and scaled by the processor model
- Virtual MPI message latency/bandwidth defined by a network model and maintained by the PDES
- PDES bootstrap sends a message to each VP to invoke the programs main
- Conservative execution without deadlock detection (not needed at this point)

- **Virtual Processes**

- Encapsulated in a user-space thread for efficient execution at extreme scale
- User-space (pthread) stack frame management
- User-space global variables management (.data heap)
- User-space stack/globals context switch
- Customizable stack size (via command line option)
- Intel 32/64-bit (x86/x86_64)
- Risk of stack overflow

Implementation Details (2/3)

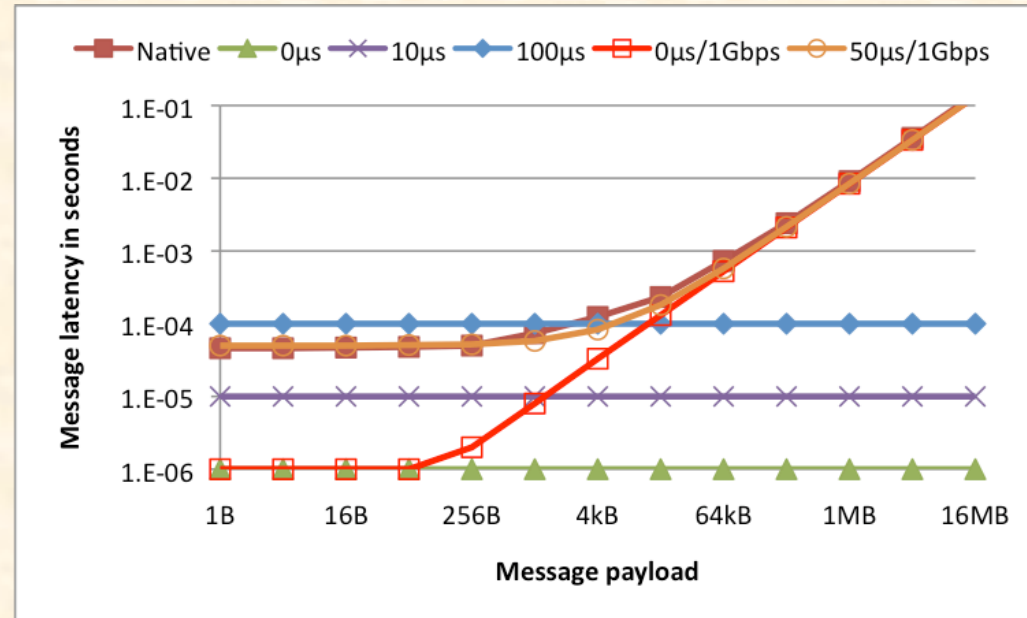
- **Virtual MPI**
 - Reimplementation of MPI atop VP P2P messaging
 - Extensive MPI functionality
 - Full MPI group and communicator support
 - Full MPI collective communication support
 - MPI_Wtime and MPI_Wtick
 - Repeatability issues with:
 - MPI_Waitsome/Testsome
 - MPI_ANY_SOURCE
- **Virtual MPI & PDES**
 - Virtual clock is off during virtual MPI calls
 - Message sends occur immediately (add to queue for local VP or native MPI_Send for remote VP)
 - Message receive time is defined by network model
 - Queued messages are ordered by receive time
 - Virtual MPI msg. receive performs a context switch to the addressee of the first message in the queue

Implementation Details (3/3)

- **Network model**
 - Static p2p message latency and bandwidth
 - No consideration of contention at this time
- **Global variables**
 - ELF binary scan for text segments
 - Copy out/in corresponding memory during context switch
- **Virtual MPI Groups & Comms.**
 - Virtual MPI group registry
 - Virtual MPI comm. registry
 - Pre- and user-defined virt. MPI groups and comms.
 - Virtual MPI group ops. (incl., excl, union, ..)
 - Virtual MPI comm ops. (create, dup, split, ..)
- **Virtual MPI collectives**
 - Taken from MR-MPI/redMPI
 - P2P-based implementation

Initially Developed Basic Network Model

- Model for basic star network with configurable:
 - Bandwidth
 - Latency
 - $0\mu\text{s}/\infty\text{Gbps}$ for baseline
 - $25\mu\text{s}/1\text{Gbps}$ for Gigabit Ethernet in a cluster
- Model does not support advanced architectures
- Model does not support network-on-node and network-on-chip
- ***A more advanced network model is needed!***



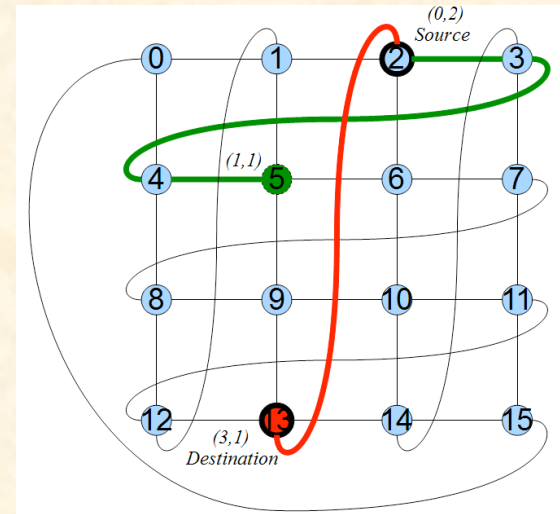
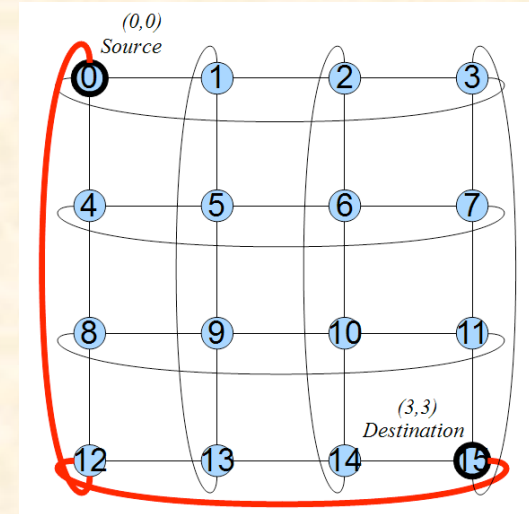
© 2010 IEEE Cluster Co-Design Workshop

Developed Advanced Network Model

- **Simulated network parameters**
 - **Bandwidth:** MPI P2P large-message bandwidth for a link
 - **Latency:** MPI P2P small-message latency for a link
 - **Topology:** Star, tree, ring, mesh, torus, twisted torus
 - **Topology-specific configuration:** Dimensions, loops, ...
- **Hierarchical combinations for multi-level networks**
 - **For example:** Node → on-node → on-chip
- **No congestion modeling (at this point)**
 - **Accuracy/scalability trade-off**
 - **Simulation results establish upper performance bound for applications on a given architecture**

Advanced Network Model: Simulated P2P Routing for Calculating Message Latency

- Message latency $L=(h*l)+(s/b)$
 - Hop count h
 - Link latency l
 - Message size s
 - Link bandwidth b
- Message path is traversed from source to destination within/across networks
 - Star: Hop count $h=2$
 - Tree: Up-down traverse
 - Ring/mesh/torus: Traverse along the dimensions
 - Twisted torus: Depth first search with torus limits



Results: Environment and Parameters

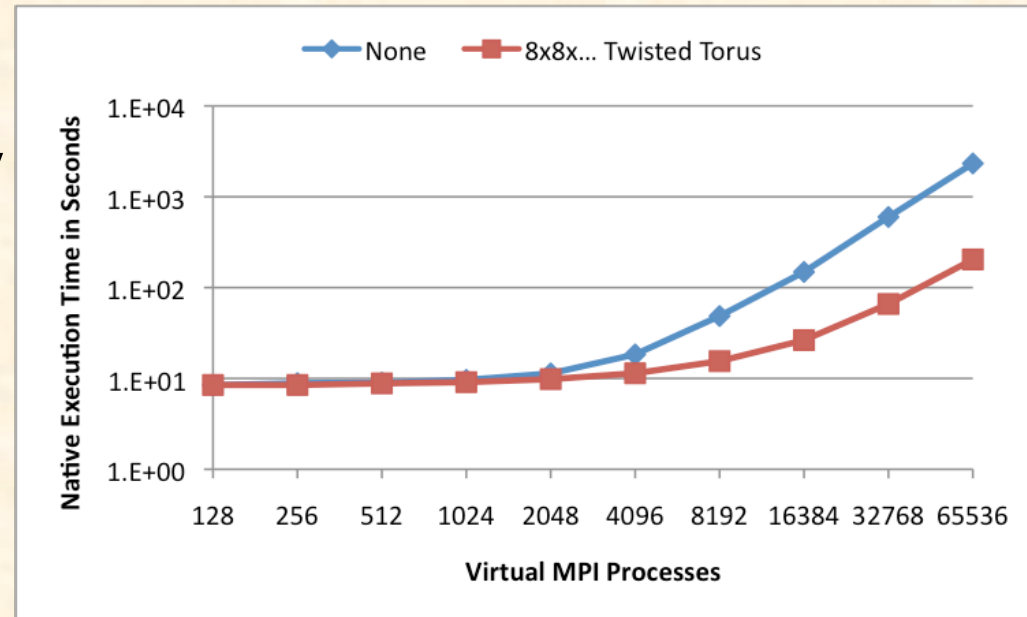
- **16-node Linux cluster**
 - **2 processors per node**
 - **4 cores per processor**
 - **1GB RAM per core**
 - **Gigabit Ethernet**
- **Simulated processor**
 - **1:1 for native-to-virtual execution ratio**
- **Simulated network parameters**
 - **Baseline: No network model, i.e., $0\mu\text{s}/\infty\text{Gbps}$**
 - **Ethernet: Measured Gigabit Ethernet P2P performance**
 - **Mesh/(Twisted) Torus: Measured Cray XT5 P2P performance**

Table I
PARAMETERS OF THE SIMULATED NETWORKS

Network Name	Network Parameters
None	No network model
Ethernet	Star, 1Gbps, $25\mu\text{s}$
<i>x</i> -core Ethernet (multi-core)	On processor: star, 12Gbps, $0.32\mu\text{s}$ Off processor: star, 1Gbps, $25\mu\text{s}$
<i>Yx... Mesh</i>	2D mesh, 8.8Gbps, $7\mu\text{s}$
<i>YxX... Mesh</i> (dual-core)	On processor: star, 9.6Gbps, $0.32\mu\text{s}$ Off processor: 3D mesh, 8.8Gbps, $7\mu\text{s}$
<i>YxX... Torus</i> (dual-core)	On processor: star, 9.6Gbps, $0.32\mu\text{s}$ Off processor: 3D torus, 8.8Gbps, $7\mu\text{s}$
<i>YxX... Twisted Torus</i> (dual-core)	On processor: star, 9.6Gbps, $0.32\mu\text{s}$ Off processor: 3D twisted torus, 8.8Gbps, $7\mu\text{s}$

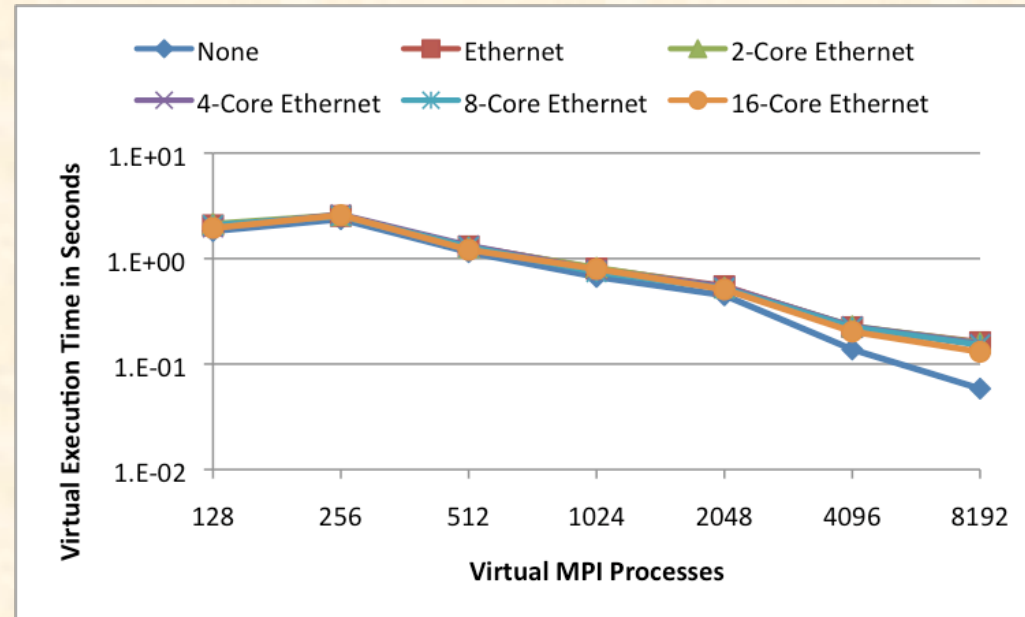
Results: Simulator Performance Running NAS EP without Network Model and with 3D Torus

- Network model should introduce overhead
- Simulator performs actually better with 3D torus
- Same message #: 720,885 for 65,536 procs.
- Similar context switch #: 458,751 vs. 458,906
- *Different message order!*
- Different overhead for message processing based on receive time within the simulator



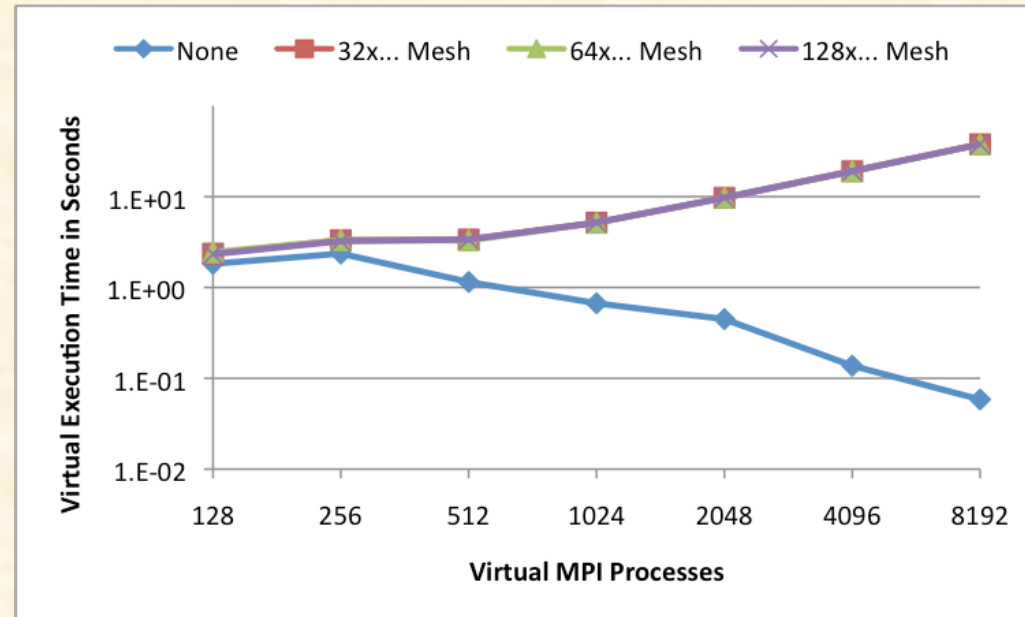
Results: Simulation Performance of NAS MG in a Multi-core Gigabit Ethernet

- Strong scaling: 128 - 8,192 (None = baseline)
- MG scales to some extent
- The impact of the Gigabit Ethernet network becomes visible at 4,096 procs.
- Not much difference between the various multi-core configurations
- The Gigabit Ethernet network is the main bottleneck



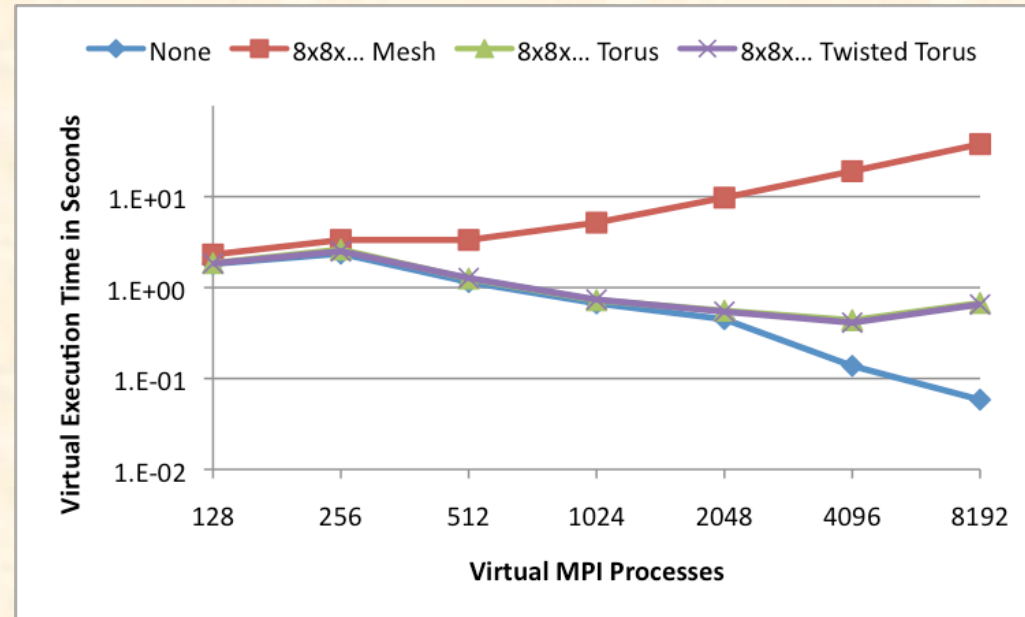
Results: Simulation Performance of NAS MG in a 2D Mesh

- Strong scaling: 128 - 8,192 (None = baseline)
- MG simply does not scale in this network setup
- Practically no difference in using a 32-, 64- or 128-process wide 2D mesh network

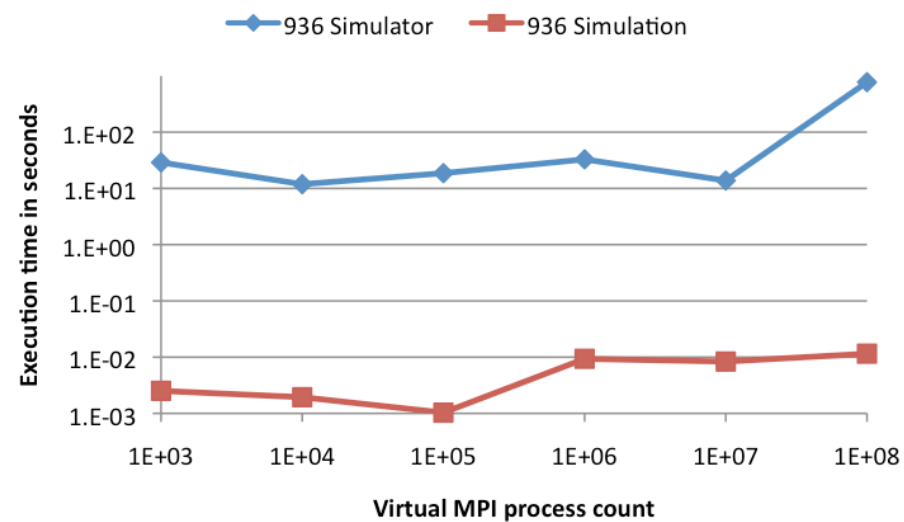
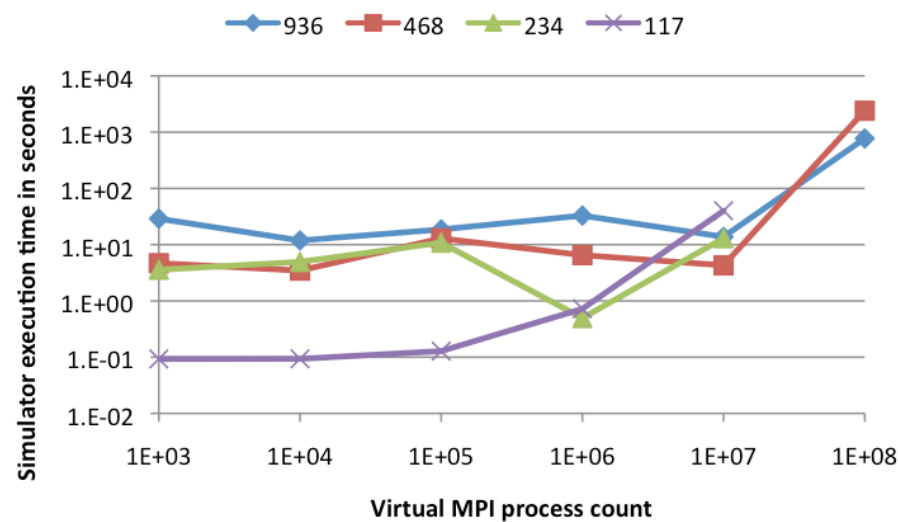


Results: Simulation Performance of NAS MG in a dual-core 3D Mesh/(Twisted) Torus

- Strong scaling: 128 - 8,192 (None = baseline)
- Again, MG does not scale in a mesh network
- It does scale in a torus or twisted torus until 4,096
- There is no difference in performance between torus and twisted torus



Results: 100,000,000 Hello Worlds Stunt (with muted output – not in paper)



- **Scaling MPI hello world with muted output from 1000 to 100,000,000 cores on simulated system**
- **Native system : 12-core 2-processor 39-node Gig. Ethernet**
- **Simulated system : 100,000,000 processor Gigabit Ethernet**
- **xSim runs on up to 936 AMD Opteron cores and 2.5 TB RAM**
- **468 or 936 cores needed for 100,000,000 simulated processes**
- **100,000,000 x 8 kB = 800 GB in virtual MPI process stack**

Conclusion and Future Work

- **Developed a new advanced network model for the xSim toolkit**
- **It offers star, tree, ring, mesh, torus, twisted torus simulations**
- **It further supports hierarchical combinations of networks**
- **It is limited to link latency and bandwidth**
- **Future work will focus on modeling congestion in a scalable fashion**

Questions?

I. S. Jones and C. Engelmann. Simulation of Large-Scale HPC Architectures.
PSTI 2011, Taipei, Taiwan, Sep. 2011.

OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY