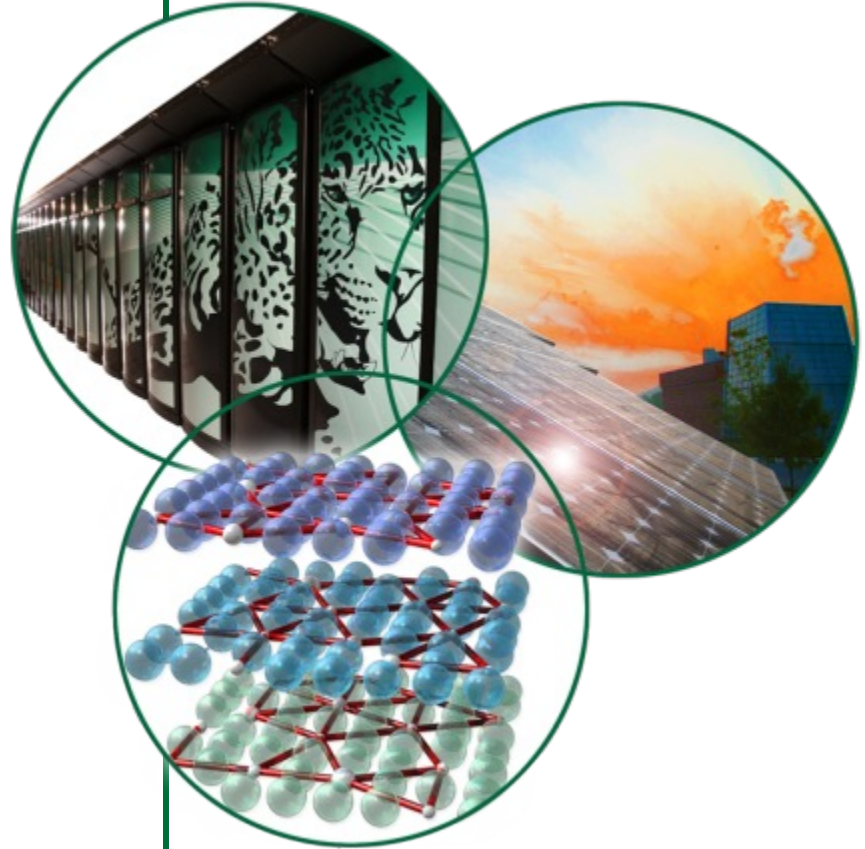


A Hardware/Software Performance/Resilience/ Power Co-Design Tool for Extreme-scale Computing

Christian Engelmann and Thomas Naughton
Oak Ridge National Laboratory

*Workshop on Modeling & Simulation of Exascale
Systems & Applications (ModSim) 2013*

Seattle, WA, September 18-19, 2013

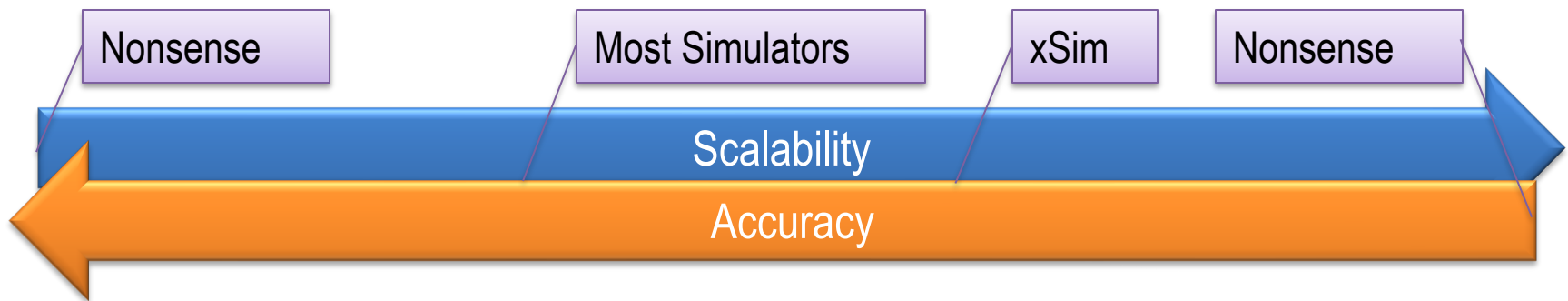


HPC Hardware/Software Co-Design

- Aims at **closing the gap between the peak capabilities of the hardware and the performance realized by applications** (application-architecture performance gap, system efficiency)
- Relies on **hardware prototypes of future HPC architectures at small scale** for performance profiling (typically node level)
- Utilizes **simulation of future HPC architectures at small and large scale** for performance profiling
- Simulation approaches **investigate the impact of different architectural parameters** on parallel application performance
- **Parallel discrete event simulation (PDES)** is often used with cycle accuracy at small scale and less accuracy at large scale

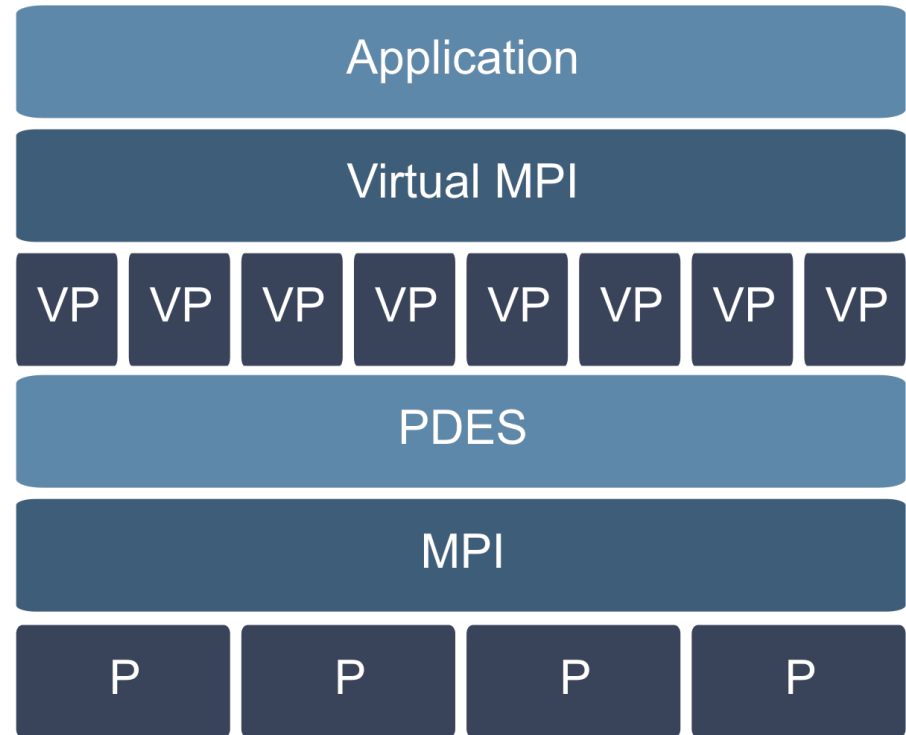
xSim: The Extreme-Scale Simulator

- 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: A highly scalable solution that trades off accuracy



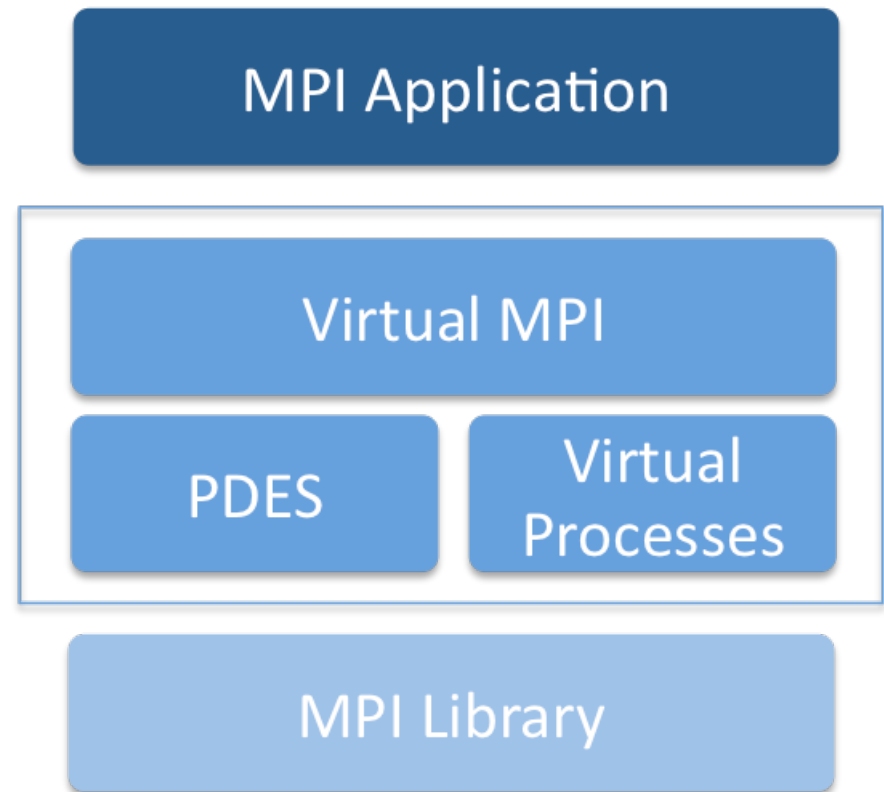
Technical Approach

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



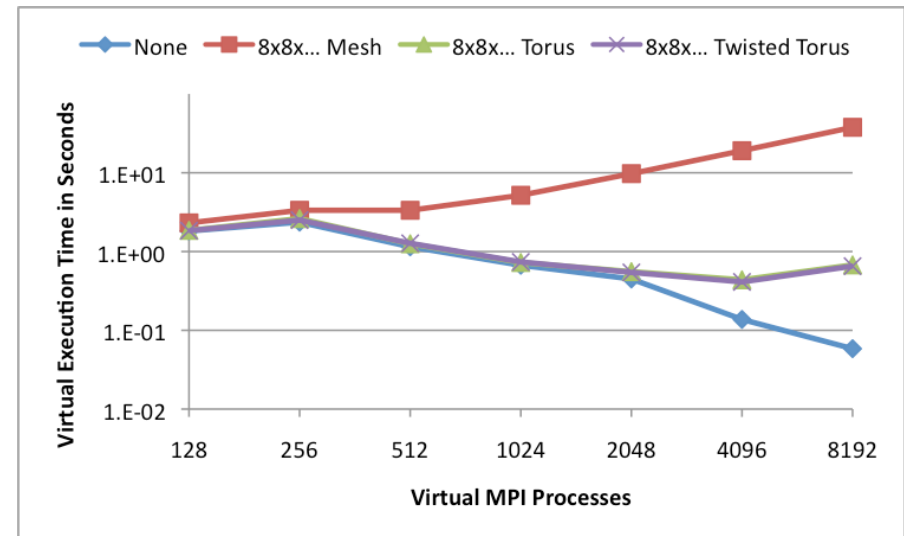
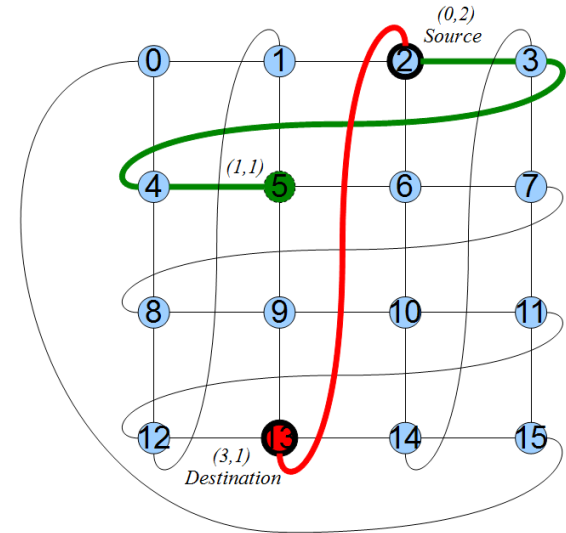
Implementation

- The simulator is a library
- Utilizes PMPI to intercept MPI calls and to hide the PDES
- Implemented in C with 2 threads per native process
- Support for C/Fortran MPI
- Easy to use:
 - Compile with xSim header
 - Link with the xSim library
 - Execute: `mpirun -np <np> <application> -xsim-np <vp>`



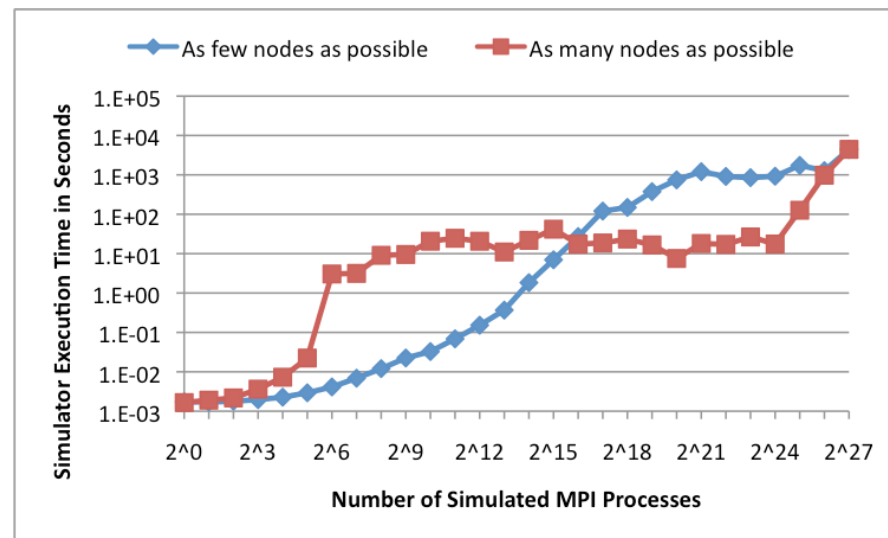
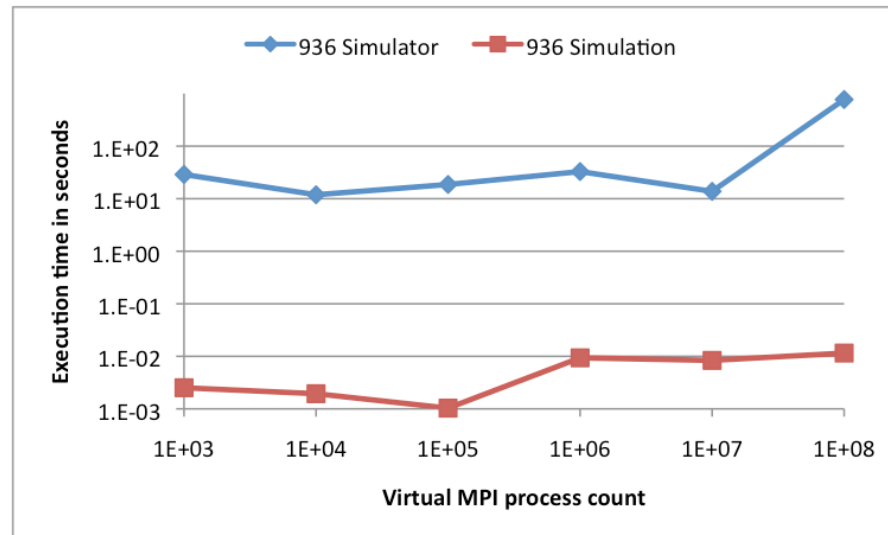
Processor and Network Models

- Scaling processor model
 - Relative to native execution
- Configurable network model
 - Link latency & bandwidth
 - NIC contention and routing
 - Star, ring, mesh, torus, twisted torus, and tree
 - Hierarchical combinations, e.g., on-chip, on-node, & off-node
 - Simulated rendezvous protocol
- Example: NAS MG in a dual-core 3D mesh or twisted torus



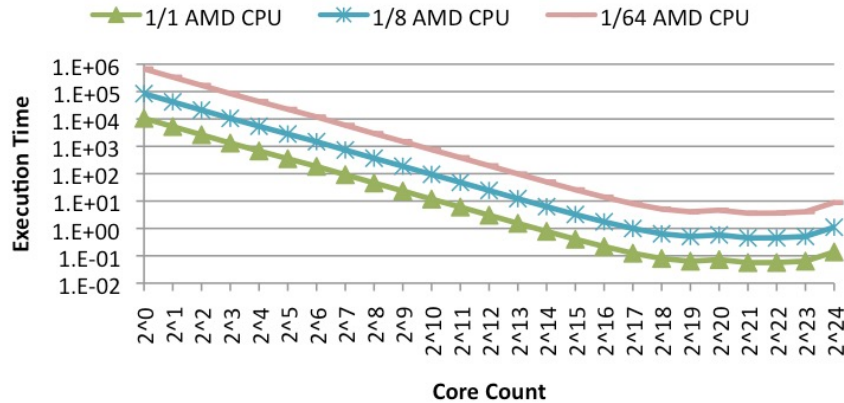
Scaling up by Oversubscribing

- Running on a 960-core Linux cluster with 2.5TB RAM
- Executing 134,217,728 (2^{27}) simulated MPI ranks
 - 1TB total user-space stack
 - 0.5TB total data segment
 - 8kB user-space stack per rank
 - 4kB data segment per rank
- Running MPI hello world
 - Native vs. simulated time
 - Native time using as few or as many nodes as possible

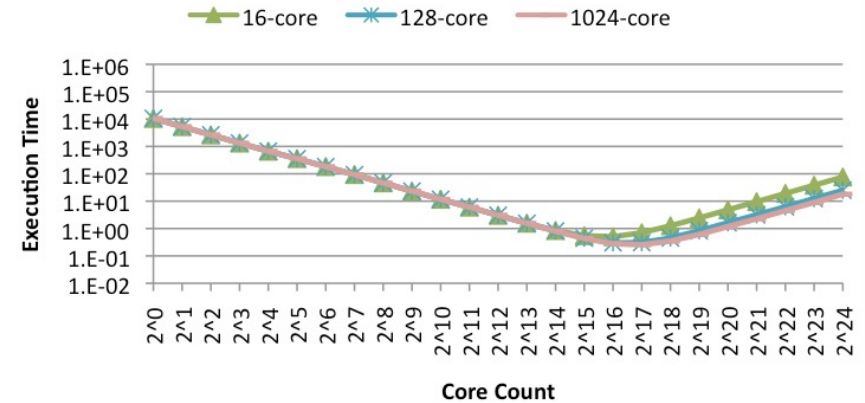


Scaling a Monte Carlo Solver to 2^{24} Ranks

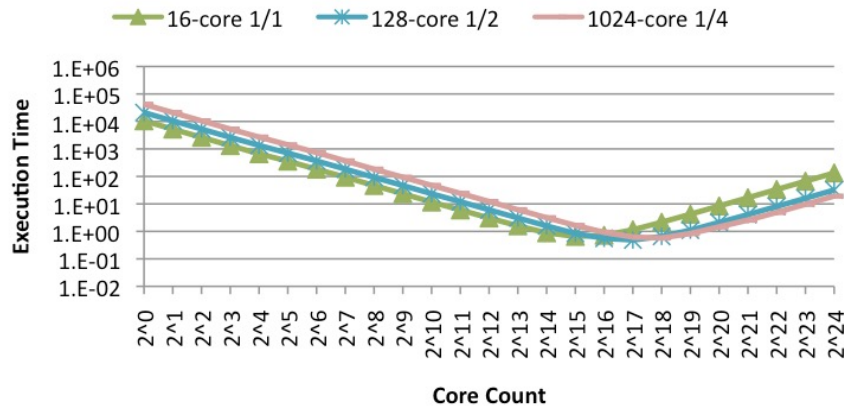
Perfect Network - 128x



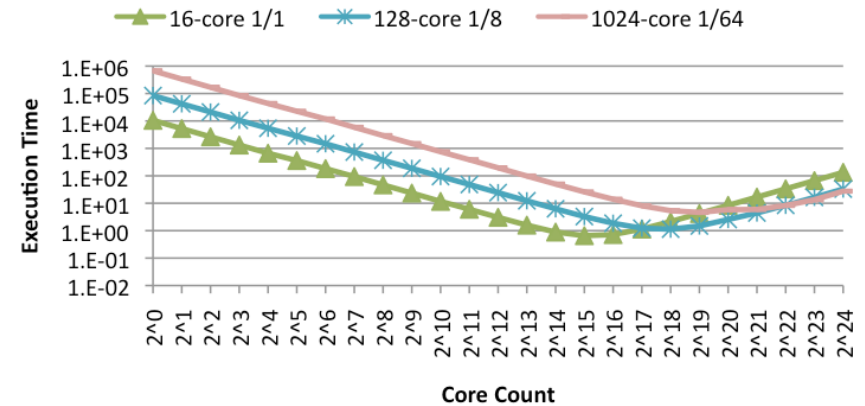
8-8-X Twisted Torus - 128x



16-16-X Twisted Torus - 128x



16-16-X Twisted Torus - 128x



Simulating OS Noise at Extreme Scale

- OS noise injection into a simulated HPC system
 - Part of the processor model
 - Synchronized OS noise
 - Random OS noise
- Experiment: 128x128x128 3-D torus with 1 μ s latency and 32 GB/s bandwidth

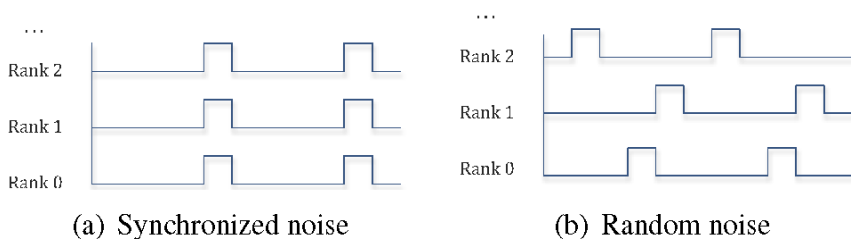
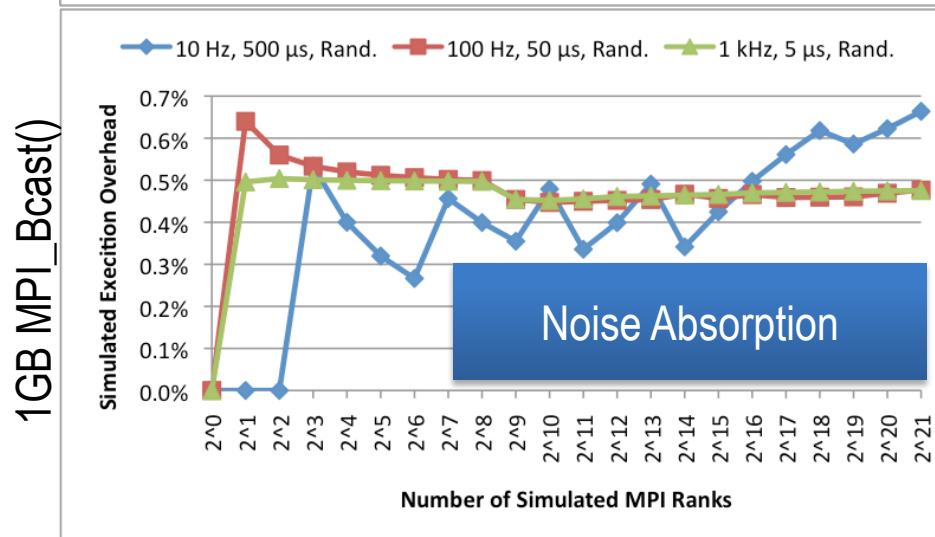
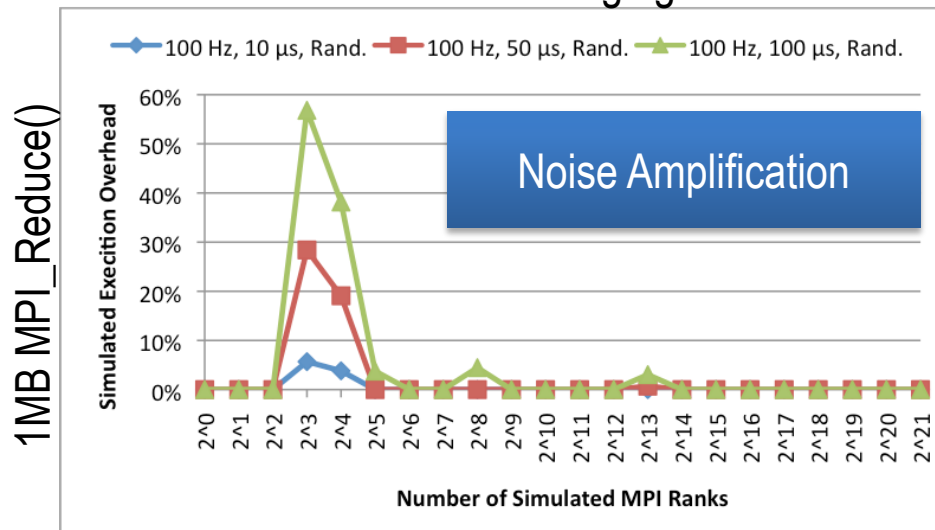


Figure 2. Generated noise injected into the simulation

Random OS Noise with Changing Noise Period



Random OS Noise with Fixed Noise Ratio

Investigating Performance Under Fault

- Parameterized MPI process failure injection/detection
- Parameterized MPI job failure injection/detection and restart
- First performance toolkit supporting checkpoint/restart:

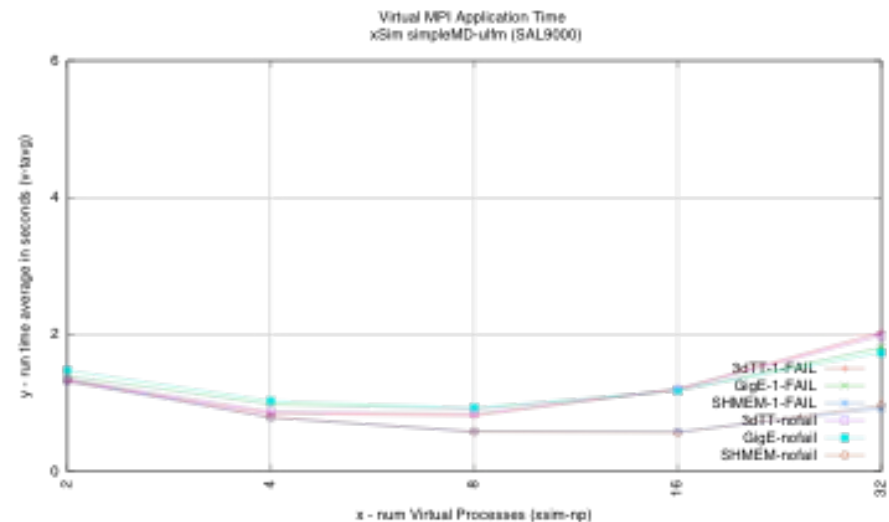
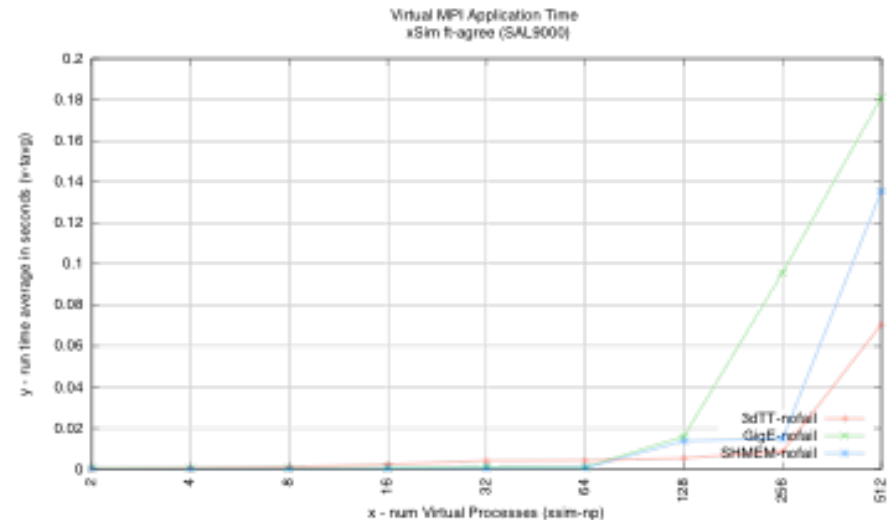
Table 1: Varying the checkpoint interval and system MTTF

$MTTF_S$	C	E_1	E_2	F	$MTTF_A$
—	1,000	5,248 s	—	0	—
6,000 s	500	5,258 s	7,957 s	1	3,978 s
6,000 s	250	6,377 s	7,074 s	1	3,537 s
6,000 s	125	6,601 s	6,750 s	1	3,375 s
3,000 s	500	5,258 s	10,584 s	2	3,528 s
3,000 s	250	6,377 s	8,618 s	2	2,872 s
3,000 s	125	6,601 s	7,948 s	2	2,649 s

- Identified different application failure modes, including the creation of incomplete or corrupted checkpoints

Investigating Performance Under Fault

- User-Level Failure Mitigation
- MPI standard enhancements proposed by the MPI-FTWG
- Permits handling of MPI process faults by applications
 - `MPI_ERR_PROC_FAILED` and `MPI_ERR_PENDING`
 - `MPI_Comm_failure_ack()`
 - `MPI_Comm_failure_get_acked()`
 - `MPI_Comm_revoke()`
 - `MPI_Comm_shrink()`
 - `MPI_Comm_agree()`
- First toolkit supporting ULFM



Future Work

- Add soft error injection (almost done, memory tracking exists)
 - Investigate performance under failure with applications resilient to process failures and soft errors (Small RX-Solvers component)
- Add storage models (almost done, POSIX file I/O API exists)
 - Investigate the impact of different storage architectures on checkpointing
- More targeted features
 - File system fault injection, fault prediction, and full/partial redundancy
 - Tool interface (for tracing, etc.) and generated benchmarks (ScalaBenchGen + xSim)
- Add reliability models (incl. developing a fault model)
 - Investigate fault probability, propagation, and handling efficiency
 - Investigate system and application correctness
- Add power models (potentially using external tools)
 - **Study the performance/resilience/power trade-off**

Conclusion

- The Extreme-scale Simulator (xSim) is a performance investigation toolkit that utilizes a PDES and oversubscription
- It supports a basic processor model and an advanced network model to simulate a future-generation HPC system
- It is the first performance toolkit supporting checkpoint/restart and the proposed MPI FT enhancements
- Future work targets enhancing features and adding reliability and power models
- The ultimate goal is to study the performance/resilience/power trade-off

Sponsored by ORNL LDRD and by DOE ASCR under the IAA program

Questions



C. Engelmann and T. Naughton. A Hardware/Software Performance/Resilience/Power Co-Design Tool for Extreme-scale Computing.