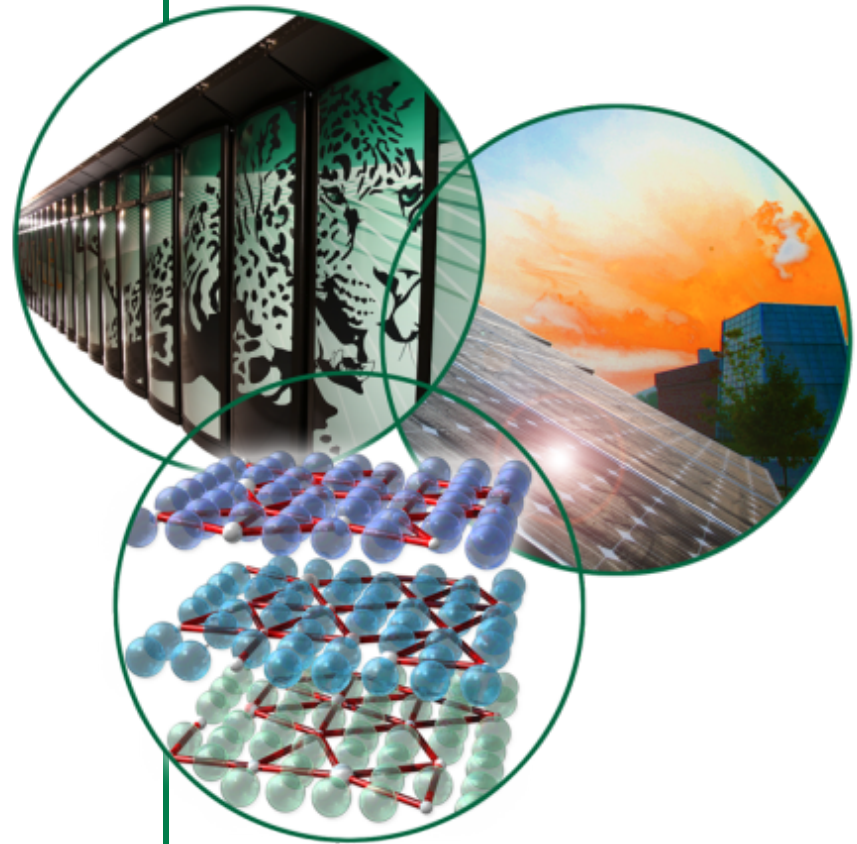


Investigating Operating System Noise in Extreme-Scale High-Performance Computing Systems using Simulation

Christian Engelmann
Oak Ridge National Laboratory

*11th IASTED International Conference on
Parallel and Distributed Computing and
Networks (PCDN) 2013*



The Road to Exascale

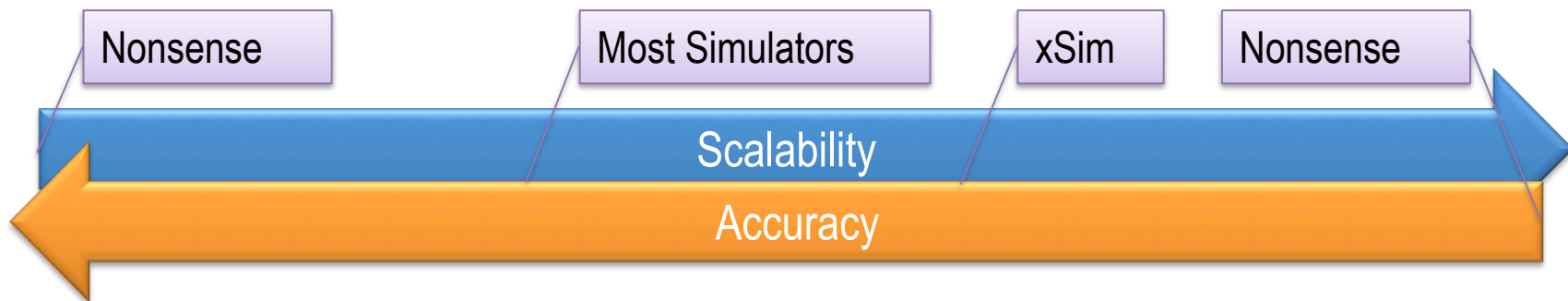
- Current top systems are at ~1-15 PFlops:
 - #1: ORNL, Cray XK7, 560,640 cores, 17.6 PFlops LINPACK, 65% Eff.
 - #2: LLNL, IBM BG/Q, 1,572,864 cores, 16.3 Pflops LINPACK, 81% Eff.
 - #3: AICS, K Computer, 705,024 cores, 10.5 Pflops LINPACK, 92% Eff.
- Need 100-times performance increase in the next 8-10 years
- Major challenges:
 - *Power consumption*: Envelope of ~20MW (drives everything else)
 - *Programmability*: Accelerators and PIM-like architectures
 - *Performance*: Extreme-scale parallelism (up to 1B)
 - *Data movement*: Complex memory hierarchy, locality
 - *Data management*: Too much data to track and store
 - *Resilience*: Faults will occur continuously

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 to reduce costs for prototyping
- Simulation approaches investigate the impact of different architectural parameters on parallel application performance
- Parallel discrete event simulation (PDES) is often employed with cycle accuracy at small scale and less accuracy at large scale

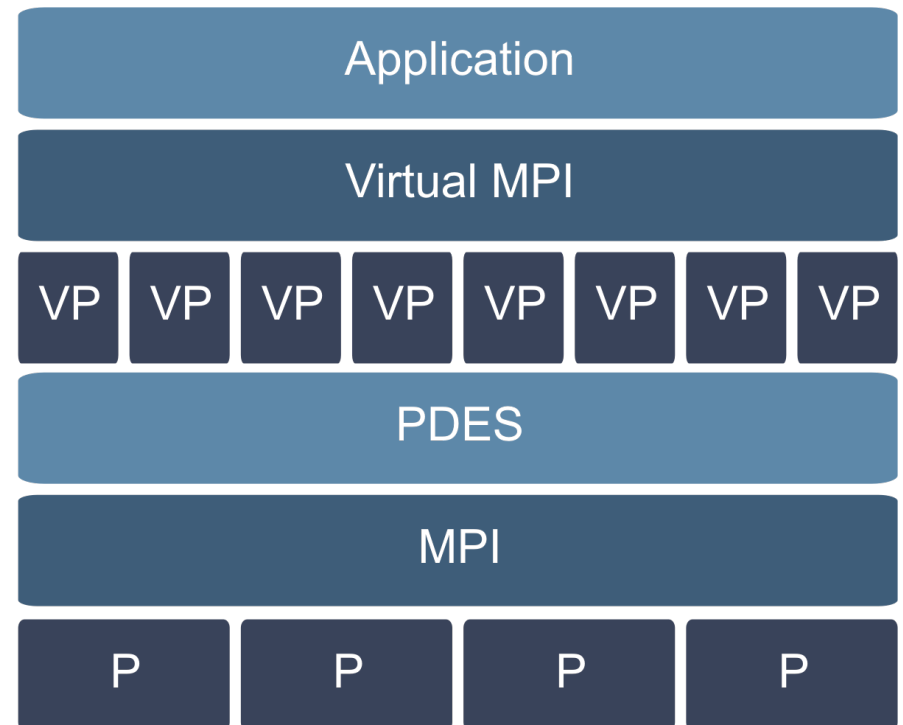
xSim: The Extreme-Scale Simulator

- 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



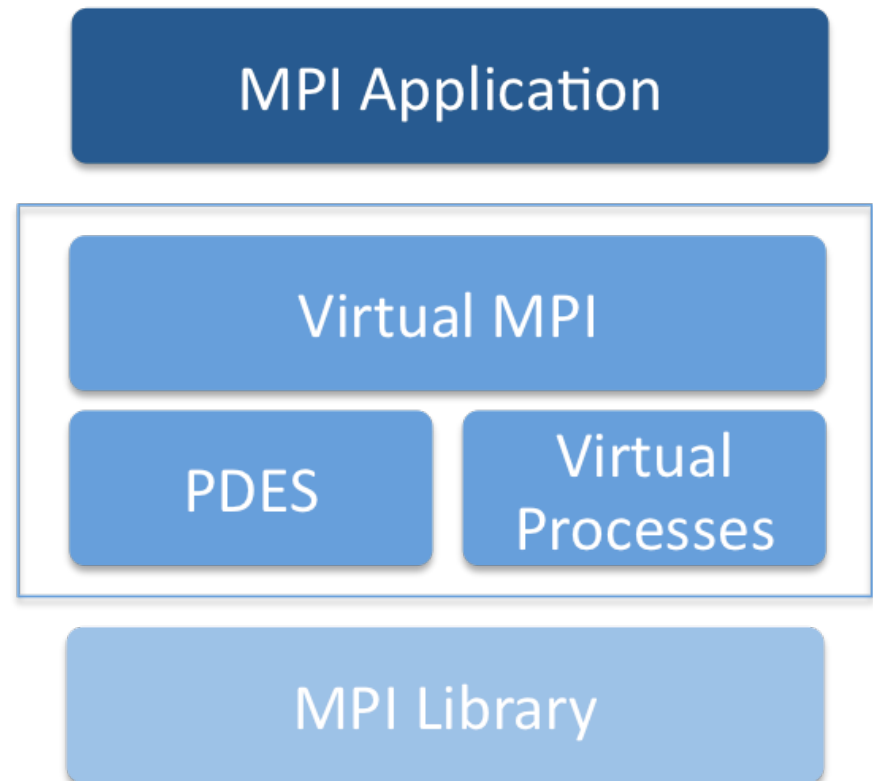
xSim: Technical Approach

- 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



xSim: Design

- 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>`



Operating System (OS) Noise

- OS interferes with application performance for I/O and resource management
 - Cache misses, page faults, hardware interrupts, multi-processing, networking
- Some OS noise is application dependent, e.g., cache misses, page faults and some HW interrupts
- Some OS noise is system dependent, e.g., timer update and pre-emption

Source	Magnitude	Example
Cache miss	100 ns	Accessing next row of a C array
TLB miss	100 ns	Accessing rarely used variable
HW interrupt	1 μ s	Network packet arrives
Timer update	1 μ s	Process scheduler runs
Page fault	10 μ s	Modifying a variable after fork()
Pre-emption	10 ms	Another process runs

Table 1. Overview of typical detours [2]

Impact of OS Noise on MPI Collectives

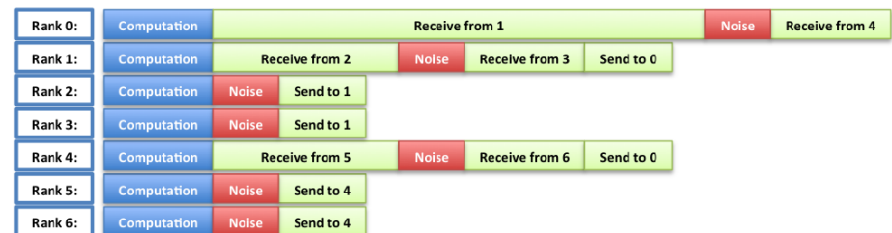
- No impact in noise-free environment
- Constant overhead with synchronized OS noise
 - Available in few HPC systems
- Variable overhead with random OS noise
 - Noise amplification
 - Noise absorption
 - Default in many HPC systems



(a) Noise free



(b) Synchronized noise



(c) Random noise

Figure 1. Example: Impact of OS noise on MPI_Reduce()

Simulating OS Noise at Extreme Scale

- OS noise frequency (periodic recurrence) and period (duration of each occurrence) abstraction
- OS noise injection into a simulated HPC system
 - Synchronized OS noise
 - Random OS noise
- Added OS noise injection to xSim's processor model
 - Simple and scalable solution
 - Regularly injected waste time

Noise Frequency	Noise Period	Noise Ratio
10 Hz	100 μ s	0.1 %
100 Hz	10 μ s	0.1 %
1 kHz	1 μ s	0.1 %
10 Hz	500 μ s	0.5 %
100 Hz	50 μ s	0.5 %
1 kHz	5 μ s	0.5 %
10 Hz	1 ms	1.0 %
100 Hz	100 μ s	1.0 %
1 kHz	10 μ s	1.0 %

Table 2. Noise ratios derived from noise frequency/period

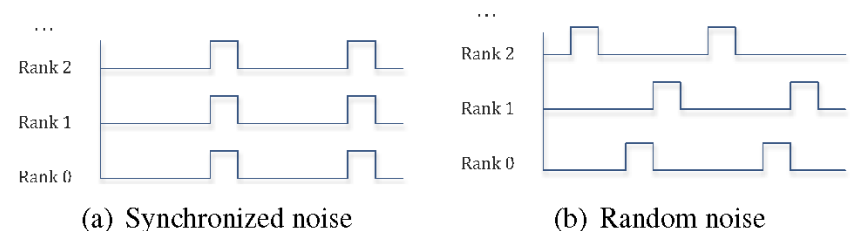
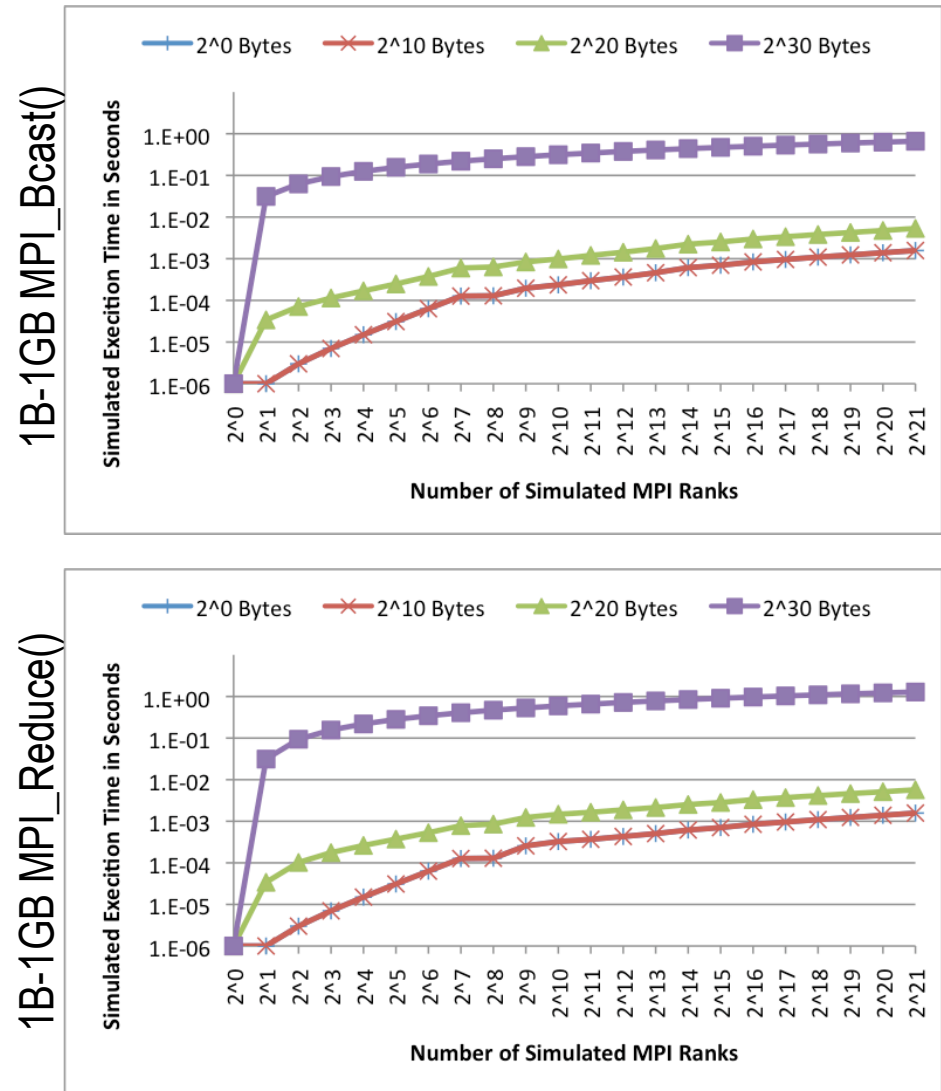


Figure 2. Generated noise injected into the simulation

Simulation Results (1/4)

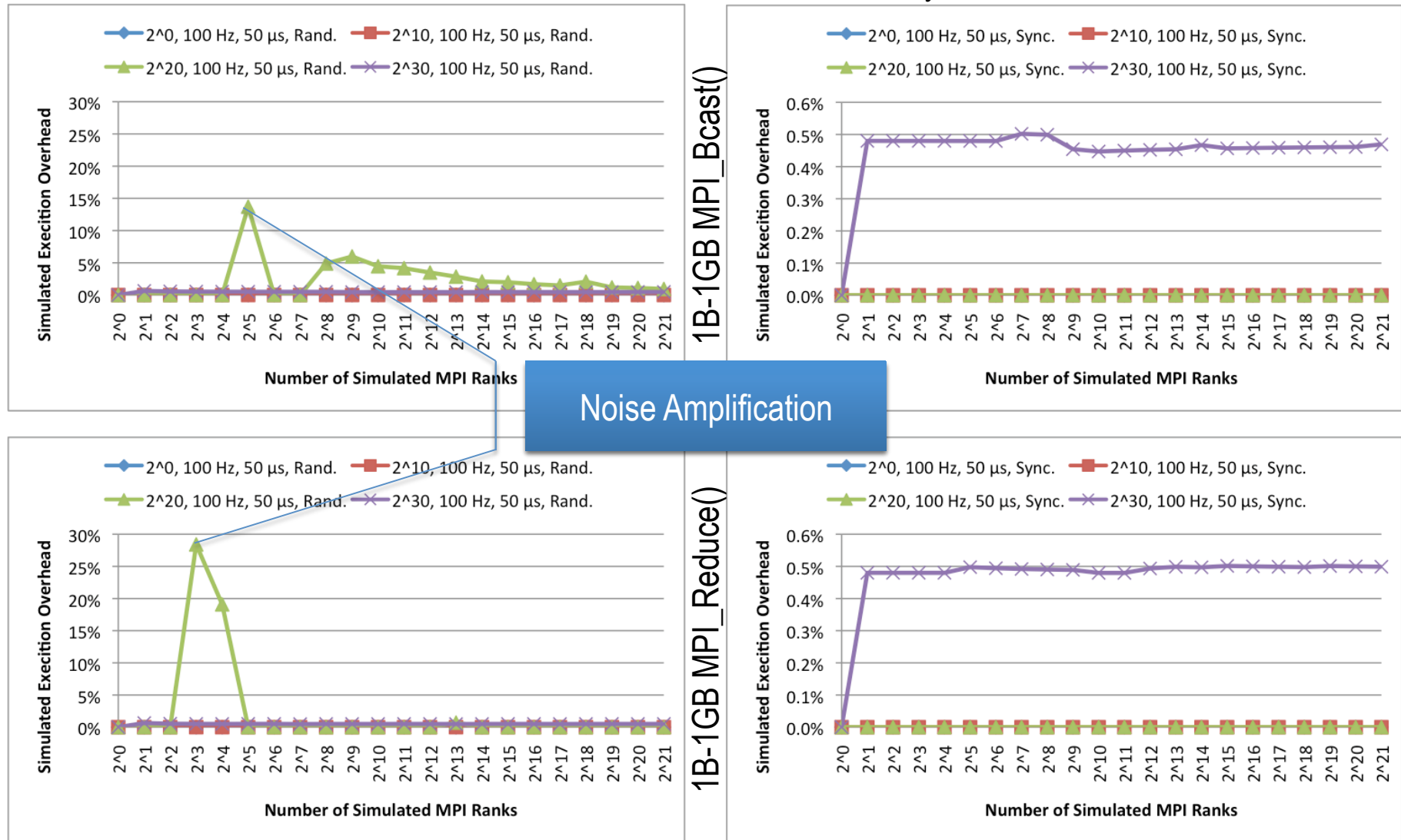
- MPI_Bcast() & MPI_Reduce() on a future HPC system
- Simulated system has 2,097,152 (2^{21}) nodes
- Organized in a $128 \times 128 \times 128$ 3-D torus with 1 μ s latency and 32 GB/s bandwidth, based on estimates for a future-generation system
- Eager threshold of 256 kB
- MPI+X programming model, i.e., 1 MPI process per node



Simulation Results (2/4)

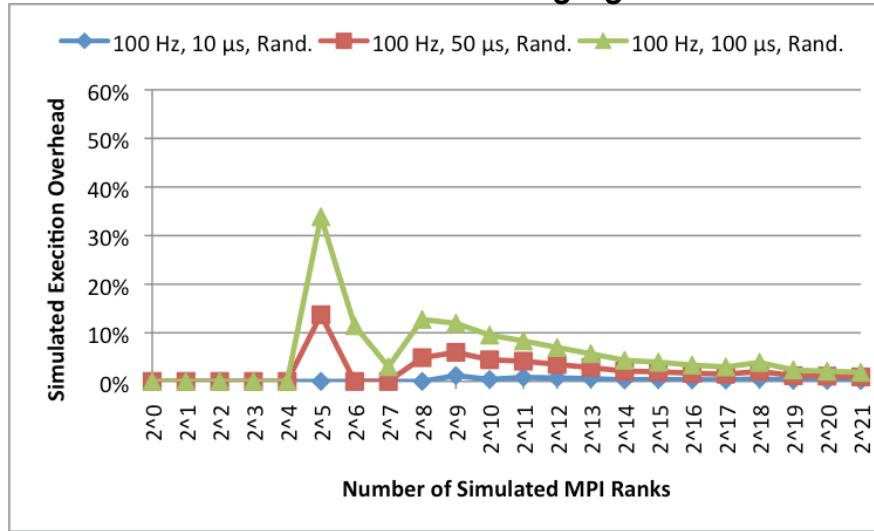
Random OS Noise

Synchronized OS Noise

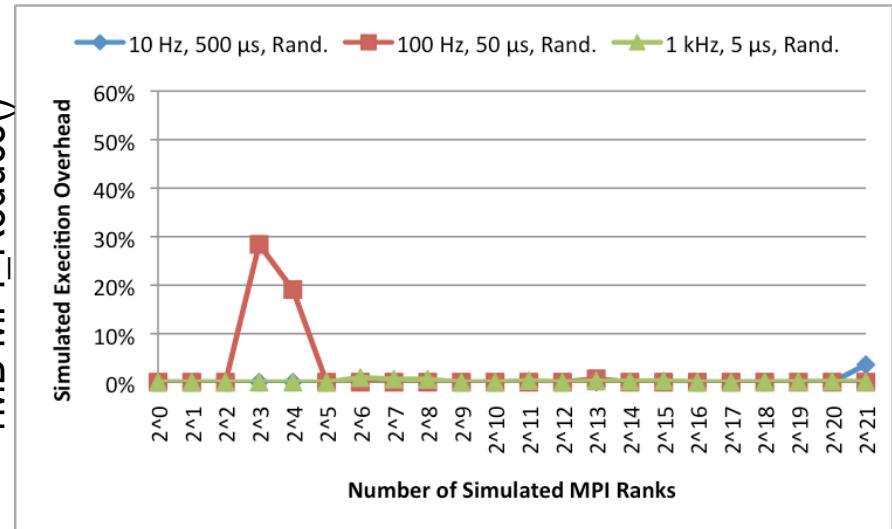
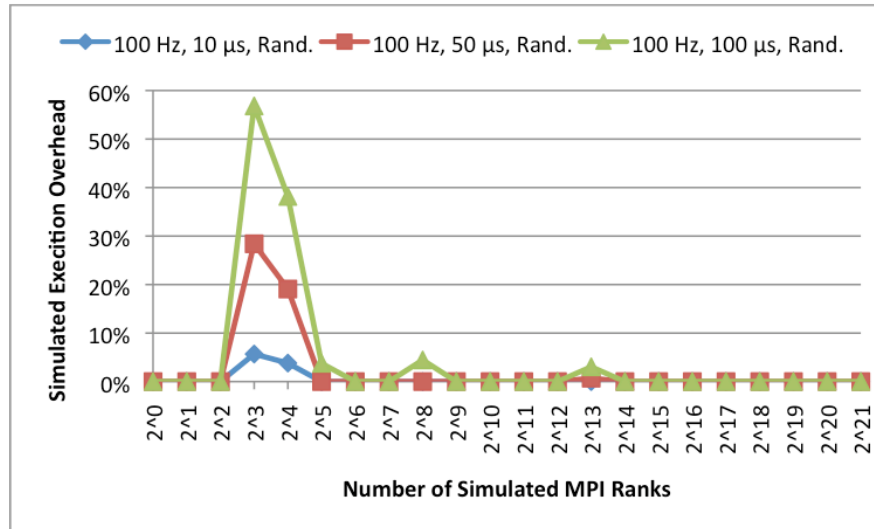
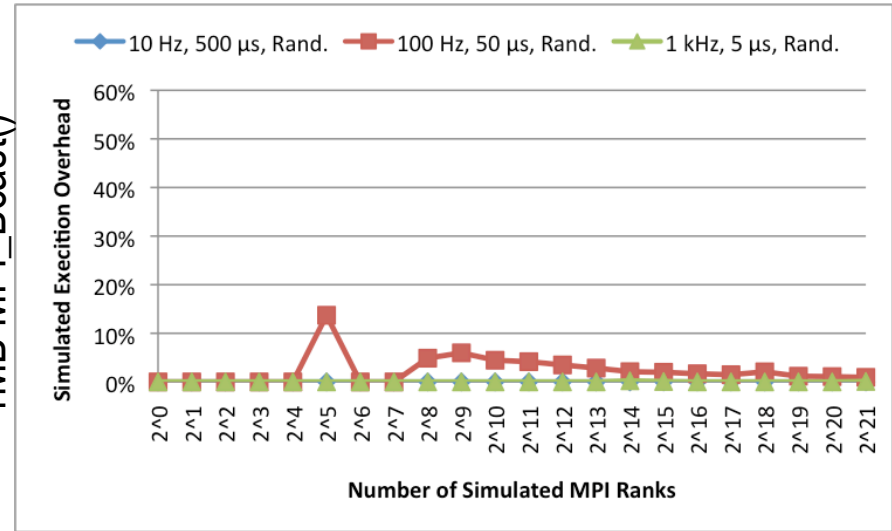


Simulation Results (3/4)

Random OS Noise with Changing Noise Period

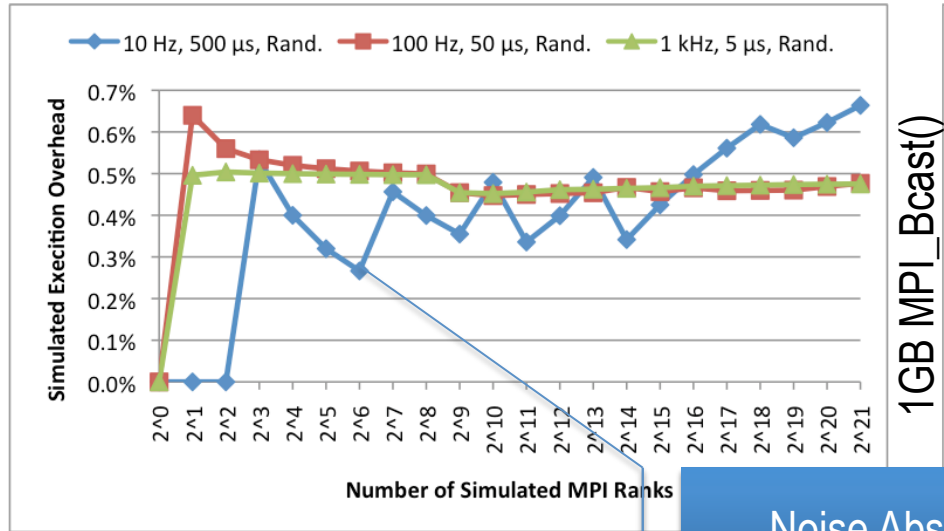


Random OS Noise with Fixed Noise Ratio

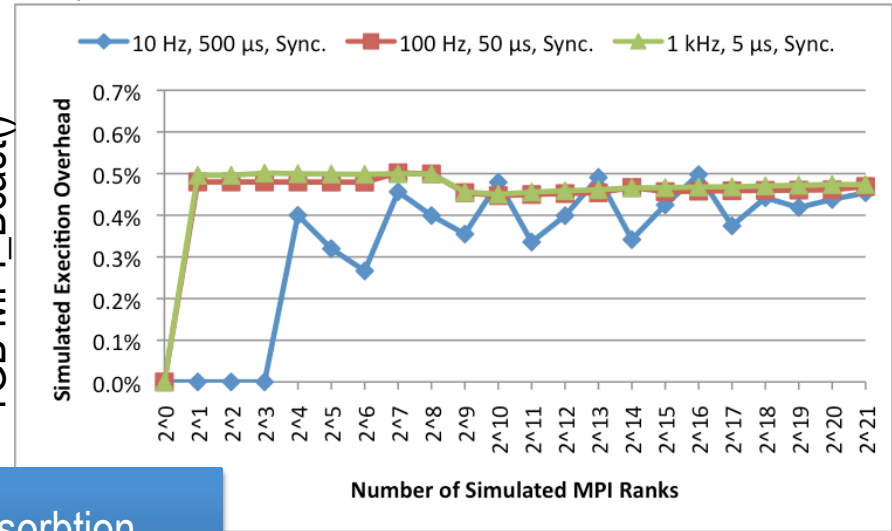


Simulation Results (4/4)

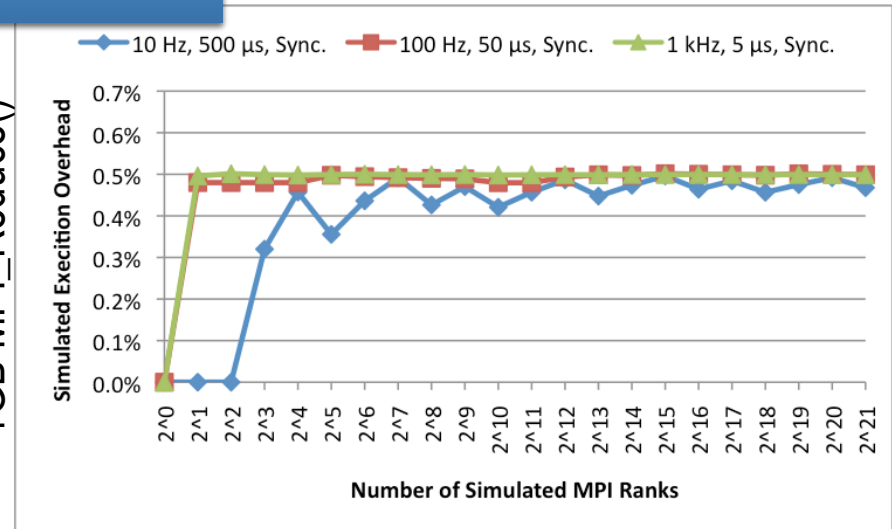
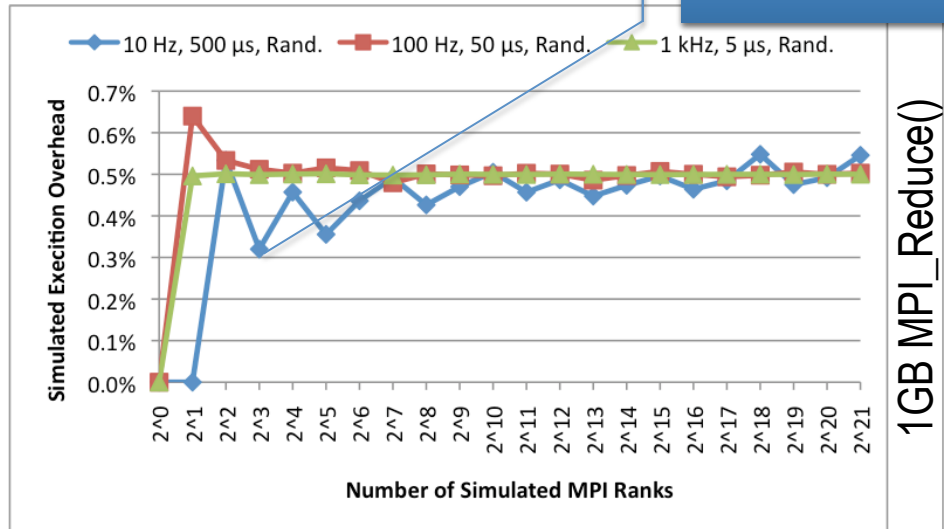
Random OS Noise with Fixed Noise Ratio



Synchronized OS Noise with Fixed Noise Ratio



Noise Absorption



Conclusions

- First step in including OS noise in HPC hardware/software co-design by adding an OS noise injection to a co-design toolkit
- Using the abstraction of OS noise with frequency and period
- Experiments show OS noise amplification, absorption, and sweet spots
- Future work targets:
 - Different OS noise frequencies/periods to more closely simulate real system behavior
 - Different OS noise patterns for each MPI process to simulate noise cores and different OS kernels within a compute node
 - Investigating the impact of OS noise on real applications using proxy/mini applications