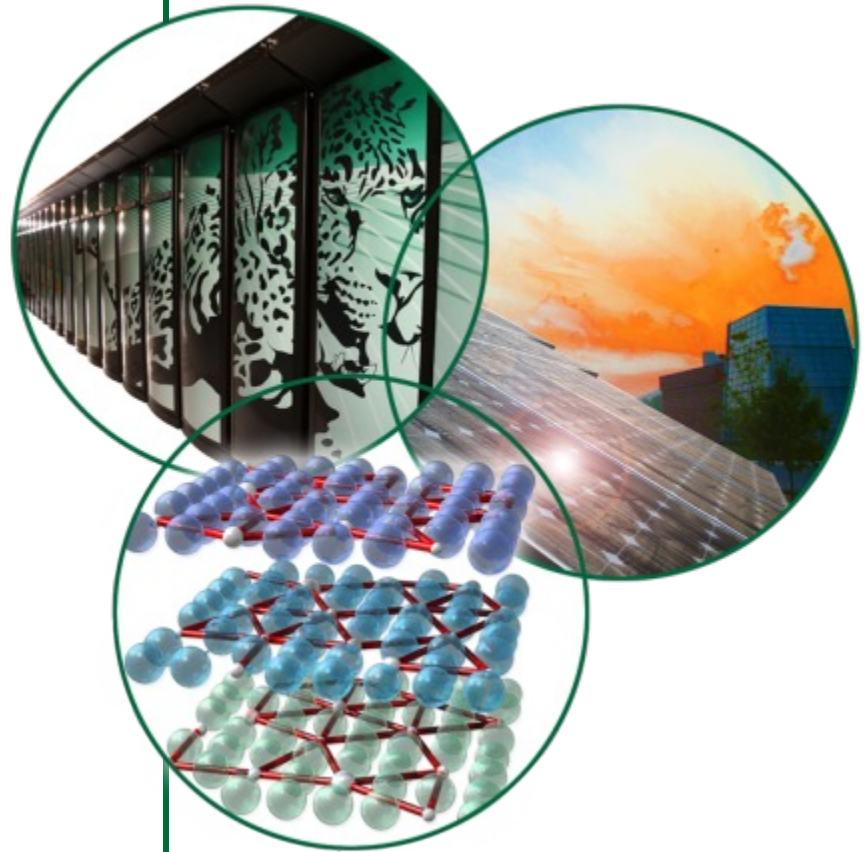


Supporting the Development of Resilient Message Passing Applications using Simulation

Thomas Naughton, Christian Engelmann,
Geoffroy Vallée, and Swen Böhm
Oak Ridge National Laboratory

*22nd Euromicro International Conference on
Parallel, Distributed, and network-based
Processing (PDP) 2014*



The Road to Exascale

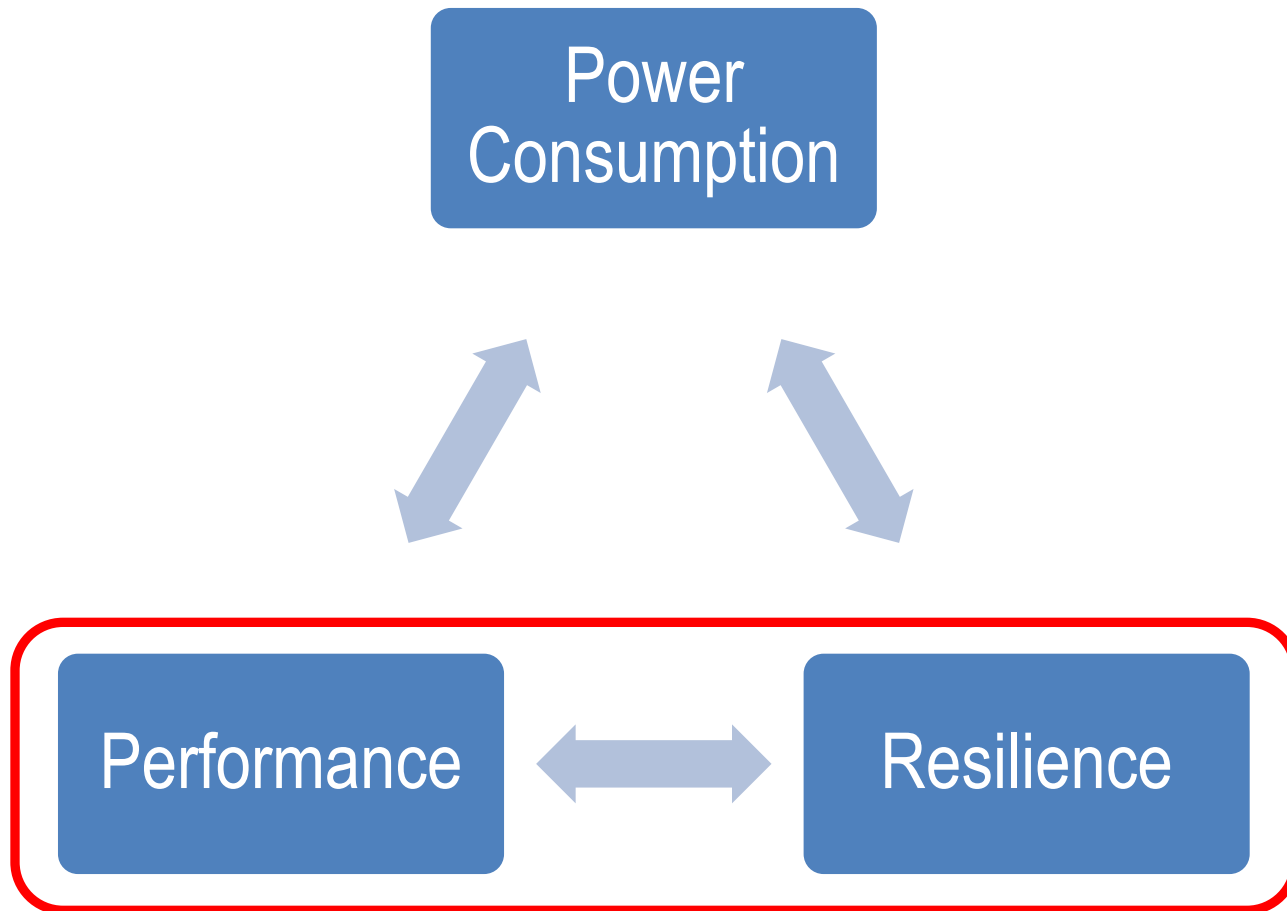
- Current top systems are at ~1-15 PFlops:
 - #1: NUDT, NUDT, 3,120,000 cores, 33.9 PFlops, 62% Eff.
 - #2: ORNL, Cray XK7, 560,640 cores, 17.6 PFlops, 65% Eff.
 - #3: LLNL, IBM BG/Q, 1,572,864 cores, 16.3 PFlops, 81% Eff.
- Need 30-60 times performance increase in the next 9 years
- Major challenges:
 - *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

Discussed Exascale Road Map (2011)

Many design factors are driven by the power ceiling (op. costs)

Systems	2009	2012	2016	2020
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	1-4 days	5-19 hours	50-230 min	22-120 min
Power	6 MW	~10MW	~10 MW	~20 MW

Trade-offs on the Road to Exascale



Examples: ECC memory, checkpoint storage, data redundancy, computational redundancy, algorithmic resilience

HPC Hardware/Software Co-Design

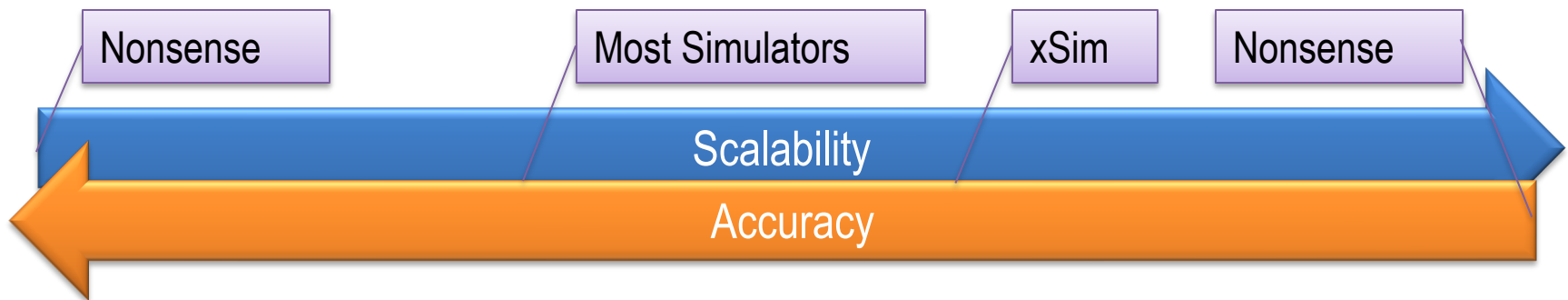
- Aims at closing the gap between the peak capabilities of the hardware and the performance realized by applications (application-architecture performance gap, system efficiency)
- Relies on hardware prototypes of future HPC architectures at small scale for performance profiling (typically node level)
- 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

Objectives

- Develop an HPC resilience co-design toolkit with corresponding definitions, metrics, and methods
- Evaluate the performance, resilience, and power cost/benefit trade-off of resilience solutions
- Help to coordinate interfaces and responsibilities of individual hardware and software components
- Provide the tools and data needed to decide on future architectures using the key design factors: performance, resilience, and power consumption
- Enable feedback to vendors and application developers

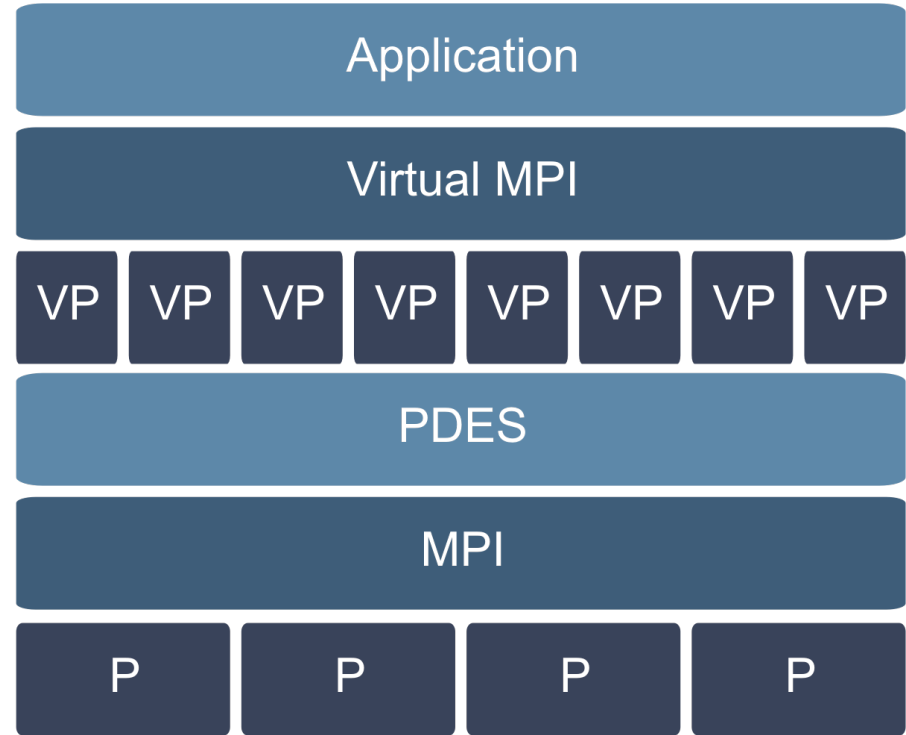
xSim: The Extreme-Scale Simulator

- 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



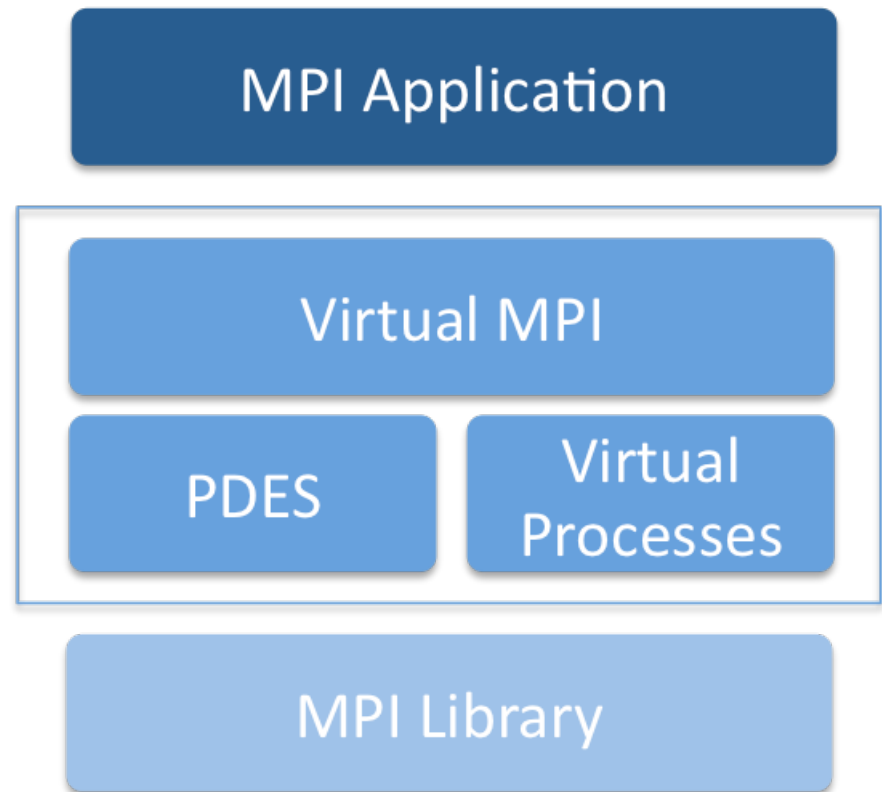
xSim: Technical Approach

- 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



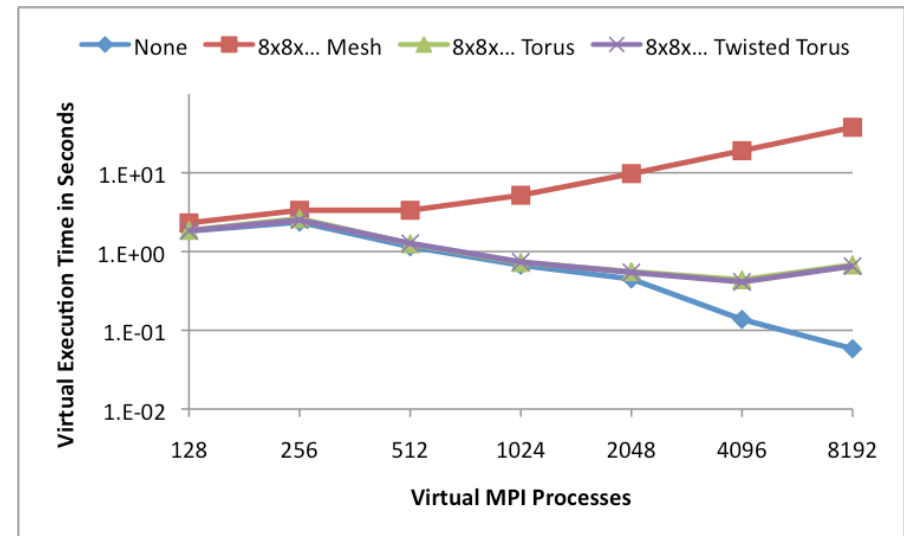
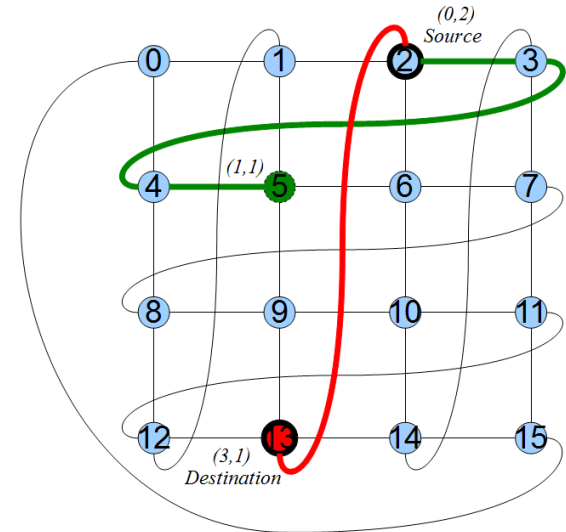
xSim: Design

- 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>`



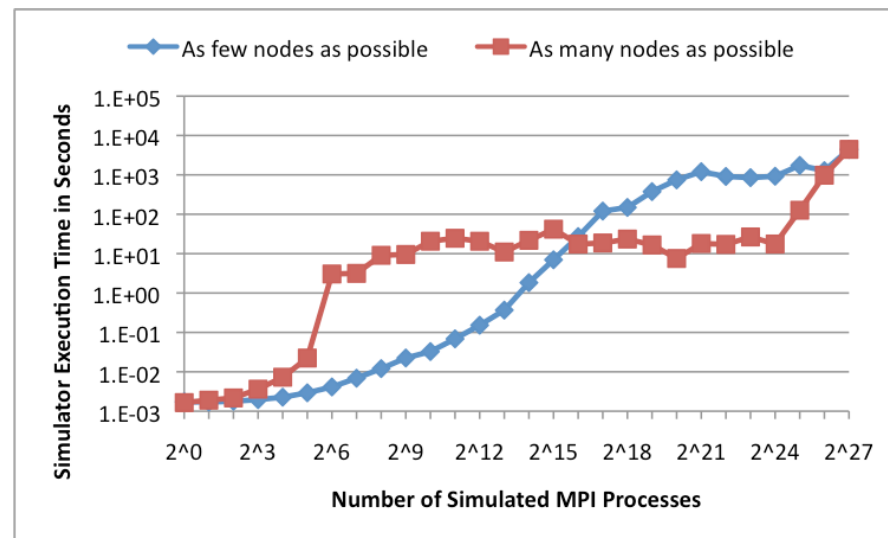
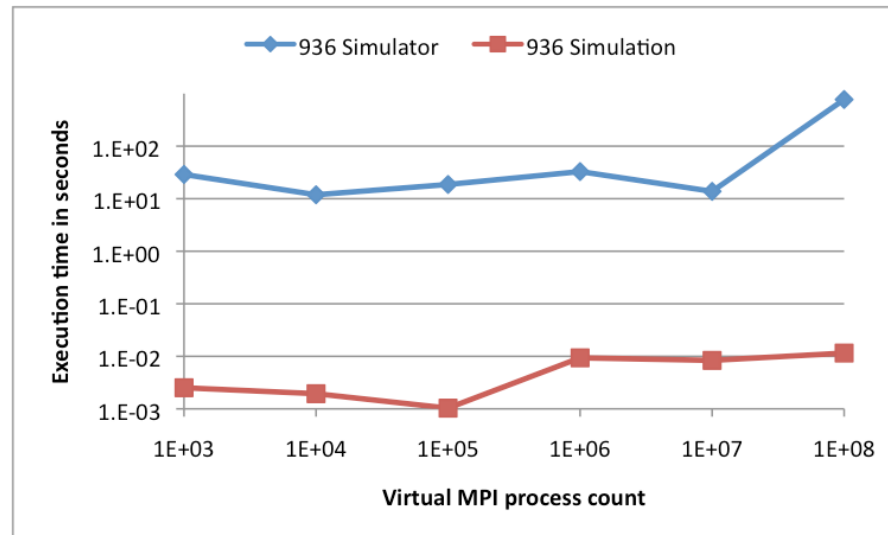
Processor and Network Models

- 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



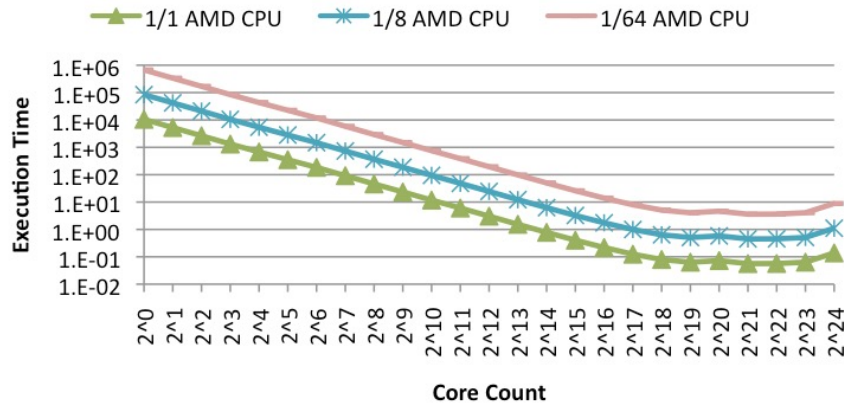
Scaling up by Oversubscribing

- 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

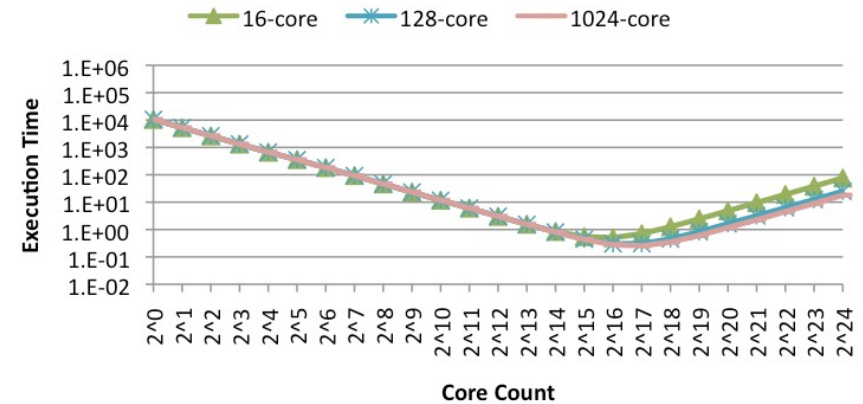


Scaling a Monte Carlo Solver to 2^{24} Ranks

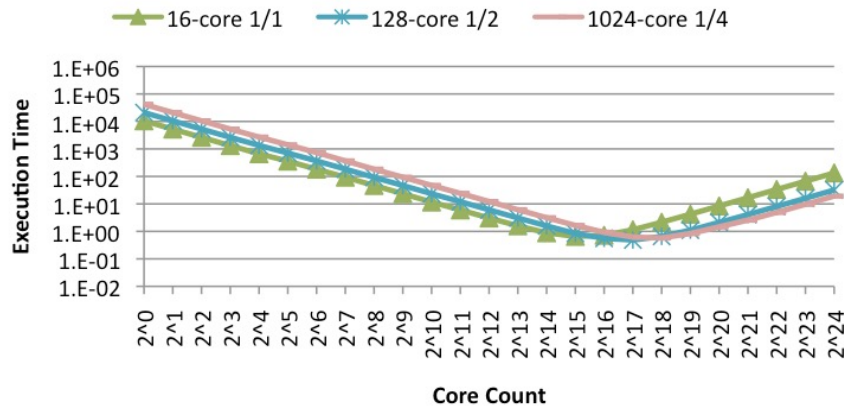
Perfect Network - 128x



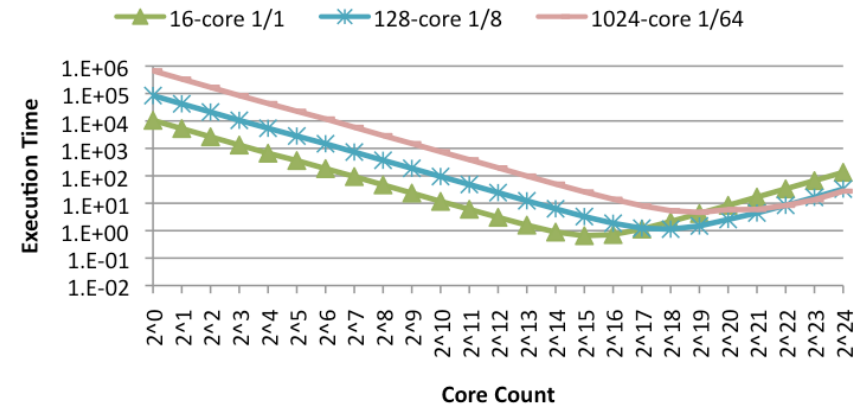
8-8-X Twisted Torus - 128x



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

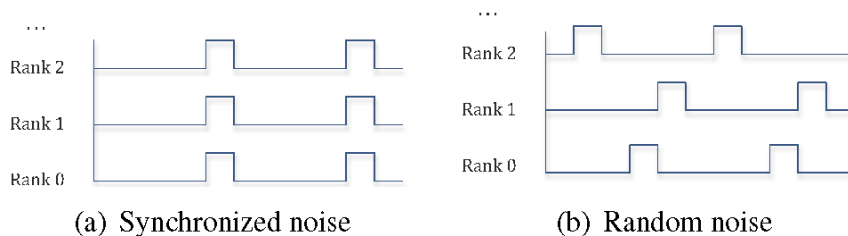
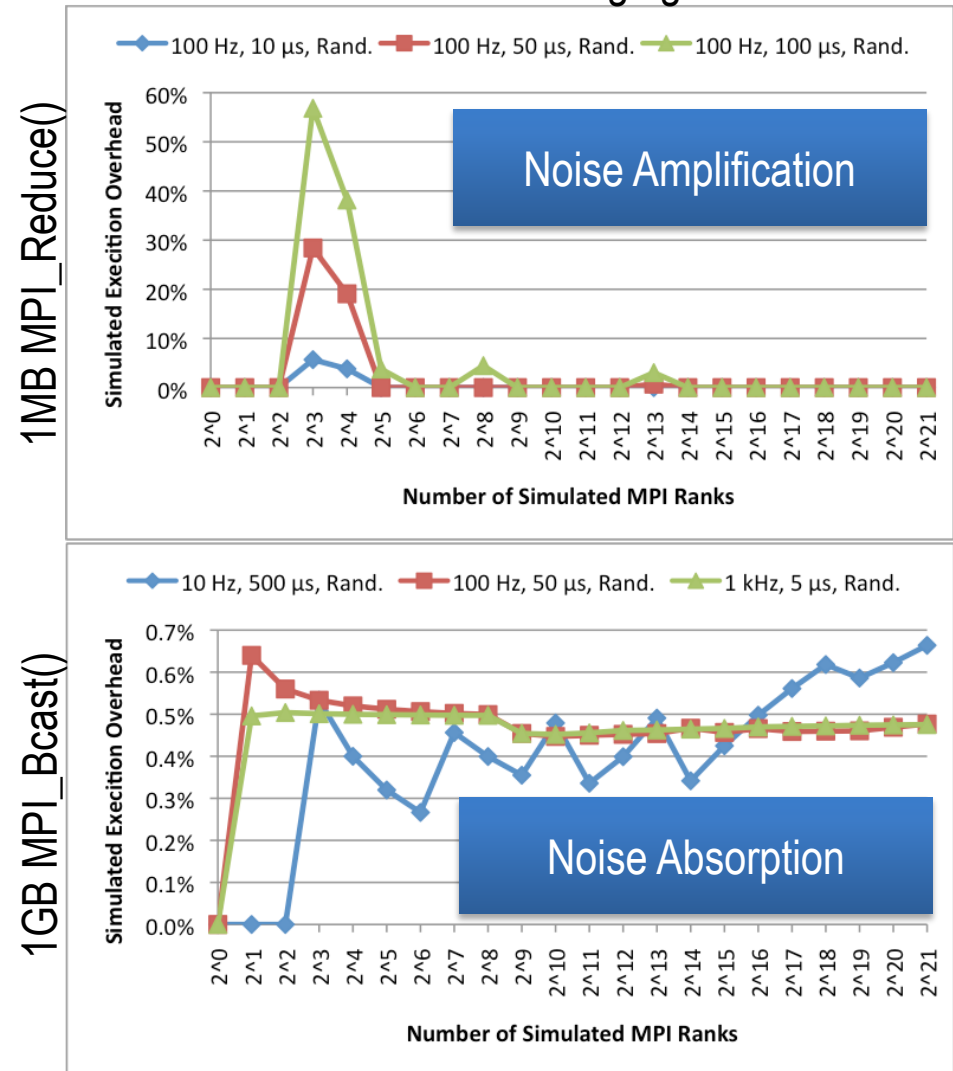


Figure 2. Generated noise injected into the simulation

Random OS Noise with Changing Noise Period



Random OS Noise with Fixed Noise Ratio

New Resilience Simulation Features

- Simulate MPI process failure injection to study their impact
- Simulate MPI process failure detection based on the simulated architecture to study failure notification propagation
- Simulate MPI abort after a simulated MPI process failure to study the current MPI fault model
- Provide full support for MPI-FT to study the runtime and performance impact of MPI process failures on applications

Simulated MPI Rank Execution in xSim

- User-space threading: 1 pthread stack per native MPI rank, split across a subset of the simulated MPI ranks
- Each simulated MPI rank has its own full thread context (CPU registers, stack, heap, and global variables)
- Always one simulated MPI rank per native MPI rank at a time
- Simulated MPI rank yields to xSim when receiving an MPI message or performing a simulator-internal function
- Context switches occur between simulated MPI ranks on the same native MPI rank upon receiving a message or termination
- Execution of simulated MPI ranks is sequentialized and interleaved at each native MPI rank

Simulated MPI Process Failure Injection

- A failure is injected by scheduling it at the simulated MPI rank (by the user via the command line or by the application or xSim via a call)
- Each simulated MPI rank has its own failure time (default: never)
- A failure is activated when a simulated MPI rank is executing and its simulated process clock reaches or exceeds the failure time
- Since xSim needs to regain control from the failing simulated MPI rank to fail it, the failure time is the time of regaining control
- A failed simulated MPI rank stops executing and all messages directed to it are deleted by xSim upon receiving
- xSim prints out an informational message on the command line to let the user know of the time and location (rank) of the failure
- A simulator-internal broadcast notifies all simulated MPI ranks
- Each simulated MPI rank maintains its own failure list

Simulated MPI Process Failure Detection

- The simulator-internal broadcast notifies all simulated MPI ranks
- It releases and fails any unmatched message receive requests from the failed simulated MPI rank
- Each simulated MPI rank adjusts the time of failure based on the network model to simulate a network communication timeout
- The simulator-internal synchronization mechanism releases and fails unmatched MPI_ANY_SOURCE receive requests
- The simulator-internal synchronization mechanism also assures that existing send requests fail at the correct time
- Future send requests and receive requests fail based on the simulated MPI rank's own failure list

Simulated MPI Abort

- Once an MPI rank failure is detected, `MPI_Abort()` is invoked if the error handler of the communicator is set to `MPI_ERRORS_ARE_FATAL`
- Sim does support other error handlers, such as `MPI_ERRORS_RETURN` and user-defined error handlers
- xSim prints out an informational message on the command line to let the user know of the time and location (rank) of the abort
- A simulator-internal broadcast notifies all simulated MPI ranks of the abort
- Similar to the simulated MPI rank failure detection, any comm. requests are failed, including `MPI_ANY_SOURCE`, with the correct failure time
- Similar to the simulated MPI rank failure activation, an abort is activated when a simulated MPI rank is executing and its simulated process clock reaches/exceeds the abort time
- All aborted simulated MPI ranks stop executing and xSim terminates

MPI-FT: User-level Failure Mitigation

- Proposed extension to the MPI 4 standard:

```
int MPI_Comm_agree                (MPI_Comm comm,  
                                   int        *flag);  
  
int MPI_Comm_iagree              (MPI_Comm comm,  
                                   int        *flag,  
                                   MPI_Request *request);  
  
int MPI_Comm_failure_ack         (MPI_Comm comm);  
  
int MPI_Comm_failure_get_acked   (MPI_Comm comm,  
                                   MPI_Group *failedgrp);  
  
int MPI_Comm_revoke              (MPI_Comm comm);  
  
int MPI_Comm_shrink              (MPI_Comm comm,  
                                   MPI_Comm  *newcomm);  
  
MPI_ERR_PROC_FAILED  
MPI_ERR_PROC_FAILED_PENDING  
MPI_ERR_REVOKED
```

Simulated User-level Failure Mitigation

- MPI-FT permits handling of MPI process faults by applications
- Implemented the MPI-FT functions in xSim:
 - To evaluate applications that handle MPI process faults
 - To evaluate the underlying algorithms of MPI-FT
- Implementation accurately reflects the execution timing of the MPI-FT functions using xSim's network model
 - Fully supports MPI-FT error reporting
 - Accurately reflects error and revocation notification timing

User-level Failure Mitigation in xSim

- `MPI_Comm_revoke()`
 - Linear broadcast of failure notification through the simulated runtime
 - No matching receive – Releases any waited on send or receive request with `MPI_ERR_PROC_FAILED | MPI_ERR_PROC_FAILED_PENDING`
- `MPI_Comm_shrink()`
 - Two-phase commit protocol to establish the list of failed MPI ranks
 - Fault-tolerant linear reduction and broadcast operations
- `MPI_Comm_agree()`
 - Agreement on a single value (logical AND operation by live members)
 - Fault-tolerant linear `MPI_Allreduce()` implementation
- `MPI_Comm_failure_ack()` and `MPI_Comm_failure_get_acked()`
 - Failure registry per rank and per communicator with low memory overhead (bit arrays)

Experimental Setup

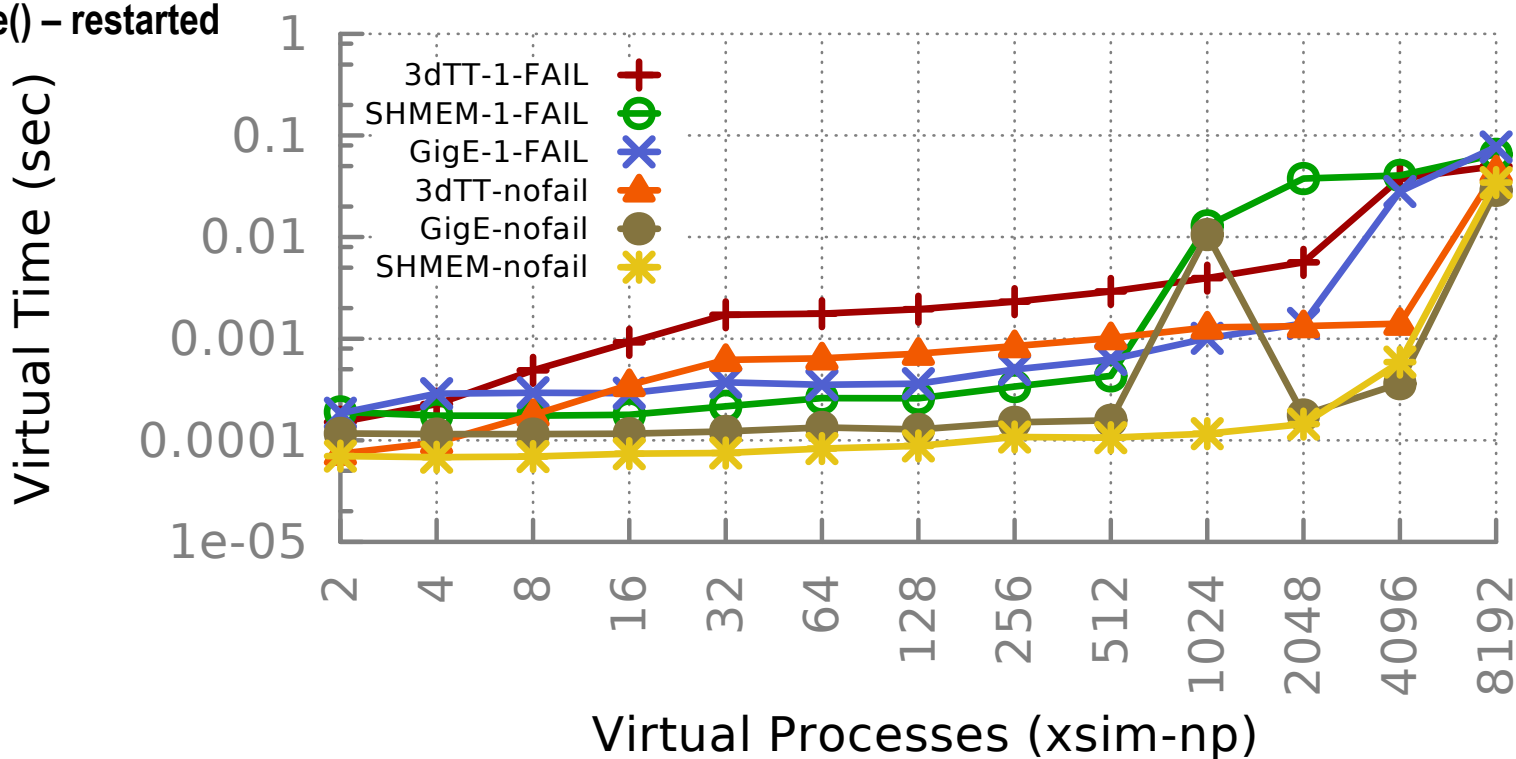
- System setup
 - 960-core Linux cluster with 40 compute nodes and 2.5 TB RAM in total
 - Bonded dual non-blocking 1 Gbps Ethernet interconnect
 - Ubuntu 12.04 LTS, Open MPI 1.6.4, and GCC 4.6
- Targeted applications
 - Microbenchmarks
 - SimpleMD: A molecular dynamics application
- Simulated systems
 - Shared memory (i.e. a many-core node)
 - Gigabit Ethernet (i.e. a Linux cluster)
 - 3D twisted torus (i.e. a MPP supercomputer, like a Cray XT5)

Micro Benchmark: ft-shrink-barrier-xsim

Benchmark executes:

- MPI_Reduce() – with/without failure
- MPI_Comm_revoked()
- MPI_Comm_shrink()
- MPI_Reduce() – restarted

Virtual MPI Application Time
xSim: ft-shrink-barrier (SAL9000)

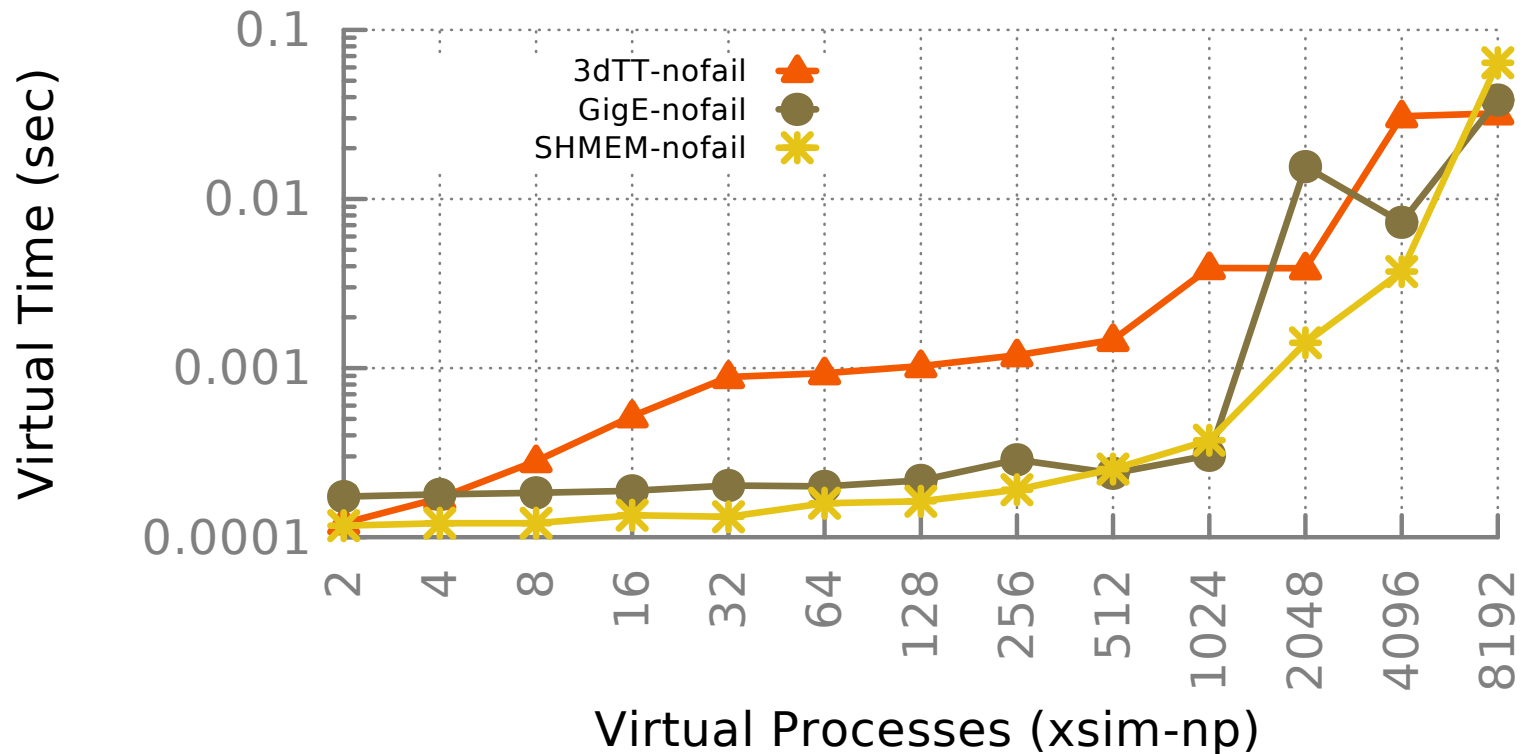


Micro Benchmark: ft-agree-xsim

Benchmark executes:

- MPI_Comm_agree() – without failure

Virtual MPI Application Time
xSim: ft-agree (SAL9000)

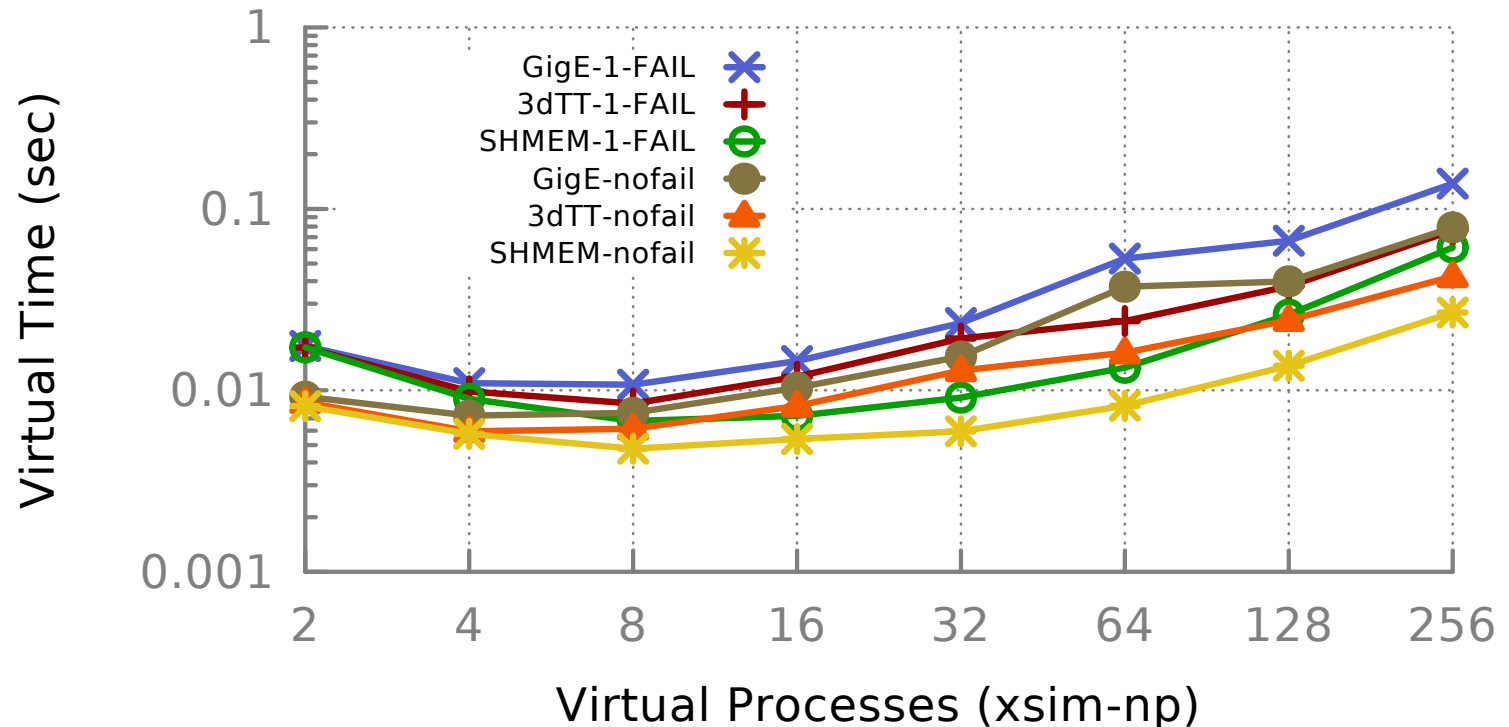


Simple MD: A Molecular Dynamics App

Benchmark executes:

- MD application with 4 iterations
- Checkpoint on NFS with each iteration
- Revoke-shrink-rollback on failure

Virtual MPI Application Time
xSim simpleMD-ulfm (SAL9000)



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 MPI process failure injection, checkpoint/restart, and MPI-FT
- Future work targets enhancing features, such as adding soft error injection
- The ultimate goal is to study the performance/resilience/power trade-off

Questions



T. Naughton, C. Engelmann, G. Vallée, S. Böhm. Supporting the Development of Resilient Message Passing Applications using Simulation. International Conference on Parallel, Distributed, and network-based Processing (PDP). Turin, Italy, Feb 12-24, 2014.