Improving the Performance of the Extreme-scale Simulator

Christian Engelmann and Thomas Naughton
Oak Ridge National Laboratory

18th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications (DS-RT) 2014
Overview

• Motivation
  • Co-design for exascale computing

• Background
  • xSim - The Extreme-scale Simulator

• Discovered problems
  • Performance issues with xSim

• Developed solutions
  • New deadlock resolution protocol
  • New simulated MPI message matching algorithm
Motivation

• At the forefront of extreme-scale scientific computing
  • Titan at ORNL: Currently 2\textsuperscript{nd} fastest supercomputer in the world
  • 560,640 cores (AMD Opteron + NVIDIA Kepler GPUs, 17.6 PFlops)
• We are on road to exascale computing: 1,000 Pflop/s by 2023
  • Billions of cores (see next slides)
• There are several major challenges:
  • \textit{Power consumption}: Envelope of \~20-40 MW (drives everything else)
  • \textit{Programmability}: Accelerators and PIM-like architectures
  • \textit{Performance}: Extreme-scale parallelism (up to 1B hardware threads)
  • \textit{Data movement}: Complex memory hierarchy and locality
  • \textit{Data management}: Too much data to track and store
  • \textit{Resilience}: Faults will occur continuously
Discussed Exascale Road Map (2011)
Many design factors are driven by the power ceiling (op. costs)

<table>
<thead>
<tr>
<th>Systems</th>
<th>2009</th>
<th>2012</th>
<th>2016</th>
<th>2020</th>
</tr>
</thead>
<tbody>
<tr>
<td>System peak</td>
<td>2 Peta</td>
<td>20 Peta</td>
<td>100-200 Peta</td>
<td>1 Exa</td>
</tr>
<tr>
<td>System memory</td>
<td>0.3 PB</td>
<td>1.6 PB</td>
<td>5 PB</td>
<td>10 PB</td>
</tr>
<tr>
<td>Node performance</td>
<td>125 GF</td>
<td>200GF</td>
<td>200-400 GF</td>
<td>1-10TF</td>
</tr>
<tr>
<td>Node memory BW</td>
<td>25 GB/s</td>
<td>40 GB/s</td>
<td>100 GB/s</td>
<td>200-400 GB/s</td>
</tr>
<tr>
<td>Node concurrency</td>
<td>12</td>
<td>32</td>
<td>O(100)</td>
<td>O(1000)</td>
</tr>
<tr>
<td>Interconnect BW</td>
<td>1.5 GB/s</td>
<td>22 GB/s</td>
<td>25 GB/s</td>
<td>50 GB/s</td>
</tr>
<tr>
<td>System size (nodes)</td>
<td>18,700</td>
<td>100,000</td>
<td>500,000</td>
<td>O(million)</td>
</tr>
<tr>
<td>Total concurrency</td>
<td>225,000</td>
<td>3,200,000</td>
<td>O(50,000,000)</td>
<td>O(billion)</td>
</tr>
<tr>
<td>Storage</td>
<td>15 PB</td>
<td>30 PB</td>
<td>150 PB</td>
<td>300 PB</td>
</tr>
<tr>
<td>IO</td>
<td>0.2 TB/s</td>
<td>2 TB/s</td>
<td>10 TB/s</td>
<td>20 TB/s</td>
</tr>
<tr>
<td>MTTI</td>
<td>1-4 days</td>
<td>5-19 hours</td>
<td>50-230 min</td>
<td>22-120 min</td>
</tr>
<tr>
<td>Power</td>
<td>6 MW</td>
<td>~10MW</td>
<td>~10 MW</td>
<td>~20 MW</td>
</tr>
</tbody>
</table>
## Discussed Exascale Road Map (2014)

Remember, this road map is a moving target.

<table>
<thead>
<tr>
<th>Systems</th>
<th>2009</th>
<th>2012</th>
<th>2017/18</th>
<th>2023</th>
</tr>
</thead>
<tbody>
<tr>
<td>System peak</td>
<td>2 Peta</td>
<td>20 Peta</td>
<td>100-200 Peta</td>
<td>1 Exa</td>
</tr>
<tr>
<td>System memory</td>
<td>0.3 PB</td>
<td>1.6 PB</td>
<td>5 PB (NVRAM?)</td>
<td>10 PB ?</td>
</tr>
<tr>
<td>Node performance</td>
<td>125 GF</td>
<td>200GF</td>
<td>200-400 GF ?</td>
<td>1-10TF ?</td>
</tr>
<tr>
<td>Node memory BW</td>
<td>25 GB/s</td>
<td>40 GB/s</td>
<td>100 GB/s ?</td>
<td>200-400 GB/s ?</td>
</tr>
<tr>
<td>Node concurrency</td>
<td>12</td>
<td>32</td>
<td>O(100) ?</td>
<td>O(1000) ?</td>
</tr>
<tr>
<td>Interconnect BW</td>
<td>1.5 GB/s</td>
<td>22 GB/s</td>
<td>25 GB/s</td>
<td>50 GB/s</td>
</tr>
<tr>
<td>System size (nodes)</td>
<td>18,700</td>
<td>100,000</td>
<td>500,000 ?</td>
<td>O(million) ?</td>
</tr>
<tr>
<td>Total concurrency</td>
<td>225,000</td>
<td>3,200,000</td>
<td>O(50,000,000)</td>
<td>O(billion) ?</td>
</tr>
<tr>
<td>Storage</td>
<td>15 PB</td>
<td>30 PB</td>
<td>150 PB ?</td>
<td>300 PB ?</td>
</tr>
<tr>
<td>IO</td>
<td>0.2 TB/s</td>
<td>2 TB/s</td>
<td>10 TB/s ?</td>
<td>20 TB/s ?</td>
</tr>
<tr>
<td>MTTI</td>
<td>1-4 days</td>
<td>5-19 hours</td>
<td>50-230 min ?</td>
<td>22-120 min ?</td>
</tr>
<tr>
<td>Power</td>
<td>6 MW</td>
<td>~10MW</td>
<td>~10 MW ?</td>
<td>~20 MW ?</td>
</tr>
</tbody>
</table>

HPC Hardware/Software Co-Design

• Helps closing the system-peak vs. application performance gap
• Develops HPC systems and applications jointly to deal with architecture limitations and application needs
• Employs hardware prototypes of future HPC architectures for performance evaluation at small scale
• Utilizes software simulation of future HPC architectures for performance evaluation at small and large scale
• Simulations 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

Objectives

• Develop a HPC performance/resilience/power co-design toolkit with corresponding definitions, metrics, and methods

• Evaluate the performance, resilience, and power cost/benefit trade-off of different architectures at scale

• 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
Overall Approach

- Execution of real applications, algorithms, or their models atop a simulated HPC environment at scale 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
Simulator 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

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 μs latency and 32 GB/s bandwidth

![Random OS Noise with Changing Noise Period](image1)

![Random OS Noise with Fixed Noise Ratio](image2)

Figure 2. Generated noise injected into the simulation

Resilience Simulation Features

• Simulated MPI process failure
  • Accurate fault injection, propagation and detection based on modeled architecture

• Simulated MPI application checkpoint, abort, and restart
  • Support for checkpoint/restart cycles until completion

• Simulated fault tolerant MPI
  • Support for resilient solvers using proposed fault tolerant MPI extensions

Table 1: Varying the checkpoint interval and system MTTF

<table>
<thead>
<tr>
<th>(MTTF_s)</th>
<th>(C)</th>
<th>(E_1)</th>
<th>(E_2)</th>
<th>(F)</th>
<th>(MTTF_A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6,000 s</td>
<td>500</td>
<td>5,258 s</td>
<td>7,957 s</td>
<td>1</td>
<td>3,978 s</td>
</tr>
<tr>
<td>6,000 s</td>
<td>250</td>
<td>6,377 s</td>
<td>7,074 s</td>
<td>1</td>
<td>3,537 s</td>
</tr>
<tr>
<td>6,000 s</td>
<td>125</td>
<td>6,601 s</td>
<td>6,750 s</td>
<td>1</td>
<td>3,375 s</td>
</tr>
<tr>
<td>3,000 s</td>
<td>500</td>
<td>5,258 s</td>
<td>10,584 s</td>
<td>2</td>
<td>3,528 s</td>
</tr>
<tr>
<td>3,000 s</td>
<td>250</td>
<td>6,377 s</td>
<td>8,618 s</td>
<td>2</td>
<td>2,872 s</td>
</tr>
<tr>
<td>3,000 s</td>
<td>125</td>
<td>6,601 s</td>
<td>7,948 s</td>
<td>2</td>
<td>2,649 s</td>
</tr>
</tbody>
</table>

Performance issues with xSim

• While xSim has shown impressive scalability, the PDES does inflict performance overheads that can be significant.

• A computationally-intensive application may experience an up to ~100% performance overhead.

• A communication-intensive application may experience an up to ~1000% performance overhead.

• With accurate process failure simulation, these overheads can increase to ~3,300% and 37,500%, respectively.

• This overhead grows with scale and limits xSim’s usability.
  • Some of the presented experiments took several days to complete.
  • Some experiments failed due to out-of-memory error issues.
Identified PDES Design Problems

• Deadlock resolution protocol
  • Way too many local virtual time update messages are communicated, resulting in a huge message processing and memory usage overhead
  • Distributed deadlock resolution is sequential, i.e., potentially moves from one deadlock to the next, thus slowing down the simulation
  • Each simulated MPI_Isend() operation becomes a potential global synchronization point when accurately simulating MPI process failures

• Simulated MPI message matching algorithm
  • Switches into the context of the destination thread before it is clear that the message can actually be matched and processed
  • Fair scheduling results in context switches despite the fact that the active process can actually match and process a message in the queue
Solved PDES Design Problems

• New deadlock resolution protocol
  • Virtual time update messages are only communicated in case of a potential deadlock, reducing overall message load
  • The local virtual time is advanced to a non-deadlocked simulated timeline, i.e., deadlocks are resolved by skipping all known deadlocks
  • The resolution of a deadlock does not immediately cause virtual time update messages to be communicated, reducing overall message load

• New simulated MPI message matching algorithm
  • A message is first evaluated if it can be matched and processed, before switching into the context of the receiving simulated MPI process
  • Message processing is batched up for the each simulated MPI process, avoiding unnecessary context switches and accelerating the processing of MPI collectives

Experimental Setup

• System setup
  • 128-core Linux cluster with 16 compute nodes
    • 2 processors per node, 4 cores processor (2.4 GHz AMD Opteron 2378)
    • 8 GB of RAM per node, 1 GB of RAM per core
  • Bonded dual non-blocking 1 Gbps Ethernet interconnect
  • Ubuntu 12.04 LTS, Open MPI 1.6.4, and GCC 4.6

• Targeted applications
  • NAS Parallel Benchmark Suite
    • 128 physical cores, 128 physical MPI processes, 128 simulated MPI processes
  • Sweep3D trace replay using ScalaBenchGen
    • 128 physical cores, 32-256 physical MPI processes, 256 simulated MPI processes

• Simulated system: 1728-core/108-node InfiniBand Linux Cluster

NAS Parallel Benchmark (NPB) Results

- No oversubscription
  - 128 physical cores
  - 128 simulated MPI processes
  - SDR: static deadlock resolution
  - DDR: dyn. deadlock resolution

- Reduced execution overhead
  - From 1,020% to 238% CG
  - From 102% to 0% for EP

- Improvements are due the new deadlock resolution protocol only

NPB Results with Accurate Failures

- No oversubscription
  - 128 physical cores
  - 128 simulated MPI processes
  - SDR: static deadlock resolution
  - DDR: dyn. deadlock resolution
- Reduced execution overhead
  - From 37,511% to 12,871% CG
  - From 3,332% to 105% for EP
- Improvements are due the new deadlock resolution protocol only

NPB Results: Number of Messages

- Top-left: Number of application messages (constant)
- Top-right: Number of simulator messages
- Bottom-right: Number of simulator messages with accurate failures

Sweep3D Trace Replay Results

- **Oversubscription**
  - 128 physical cores
  - 32-256 physical MPI processes
  - 256 simulated MPI processes

- **Application mode (top)**
  - Message payloads are sent

- **Model mode (bottom)**
  - Message payloads are empty

- **Reduced execution overhead**
  - AM@128: 37% of original
  - MM@128: 5% of original

Trace Replay with Accurate Failures

- Oversubscription
  - 128 physical cores
  - 32-256 physical MPI processes
  - 256 simulated MPI processes

- Application mode (top)
  - Message payloads are sent

- Model mode (bottom)
  - Message payloads are empty

- Reduced execution overhead
  - AM@128: 36% of original
  - MM@128: 79% of original

Trace Replay: Context Switches

- Oversubscription
  - 128 physical cores
  - 32-256 physical MPI processes
  - 256 simulated MPI processes

- Application mode (top)
  - Message payloads are sent

- Model mode (bottom)
  - Message payloads are empty

- Reduced context switches
  - AM@128: 160M to 1.6M
  - MM@128: 160M to 1.1M

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

• Two different performance issues were addressed by redesigning core components of xSim’s PDES
  • The deadlock resolution protocol
  • The simulated MPI message matching algorithm

• The results show significant improvements
Questions