Resilient Software for ExaScale Computing

Christian Engelmann

Computer Science Research Group
Computer Science and Mathematics Division
Oak Ridge National Laboratory, USA
**Discussed Exascale Road Map**

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

<table>
<thead>
<tr>
<th>Systems</th>
<th>2009</th>
<th>2011</th>
<th>2015</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>days</td>
<td>days</td>
<td>days</td>
<td>O(1 day)</td>
</tr>
<tr>
<td>Power</td>
<td>6 MW</td>
<td>~10MW</td>
<td>~10 MW</td>
<td>~20 MW</td>
</tr>
</tbody>
</table>

C. Engelmann: Resilient Software for ExaScale Computing
Discussed Exascale Road Map

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

<table>
<thead>
<tr>
<th>Systems</th>
<th>2009</th>
<th>2011</th>
<th>2015</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 size (nodes)</td>
<td>5x</td>
<td>5x</td>
<td>2x</td>
<td></td>
</tr>
</tbody>
</table>

System MTTI based on optimistic assumption that node MTTI goes up

<table>
<thead>
<tr>
<th>MTTI</th>
<th>days</th>
<th>days</th>
<th>days</th>
<th>O(1 day)</th>
</tr>
</thead>
</table>

C. Engelmann: Resilient Software for ExaScale Computing
# My Exascale Resilience Scenario

**MTTI Scales with Node Count**

<table>
<thead>
<tr>
<th>Systems</th>
<th>2009</th>
<th>2011</th>
<th>2015</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 size (nodes)</td>
<td>5x</td>
<td>5x</td>
<td>2x</td>
<td></td>
</tr>
</tbody>
</table>

*Vendors are able to maintain current node MTTI*

<table>
<thead>
<tr>
<th>MTTI</th>
<th>4 days</th>
<th>19 h 4 min</th>
<th>3 h 52 min</th>
<th>1 h 56 min</th>
</tr>
</thead>
</table>
### My Scary Scenario

**Current MTTI of 1 Day**

<table>
<thead>
<tr>
<th>Systems</th>
<th>2009</th>
<th>2011</th>
<th>2015</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>
</tbody>
</table>

| System size (nodes)   | 5x     | 5x     | 2x         |

*Current system MTTI is actually lower*

| MTTI                  | 1 day  | 4 h 48 min | 58 min | 29 min |

C. Engelmann: Resilient Software for ExaScale Computing
### My Really Scary Scenario

Component MTTI drops 3% Each Year

<table>
<thead>
<tr>
<th>Systems</th>
<th>2009</th>
<th>2011</th>
<th>2015</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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System size (nodes)</th>
<th>5x</th>
<th>5x</th>
<th>2x</th>
</tr>
</thead>
</table>

Vendors are not able to maintain current node MTTI

<table>
<thead>
<tr>
<th>MTTI</th>
<th>1 day</th>
<th>4 h 31 min</th>
<th>48 min</th>
<th>22 min</th>
</tr>
</thead>
</table>

C. Engelmann: Resilient Software for ExaScale Computing
Discussed Exascale Road Map

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

<table>
<thead>
<tr>
<th>Systems</th>
<th>2009</th>
<th>2011</th>
<th>2015</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>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>

MTTI is still the wrong metric for RESILIENCE!

C. Engelmann: Resilient Software for ExaScale Computing
Why are we only talking about MTTI?

- None of these MTTI numbers are accurate ±O(10)
- Looking at MTTI alone doesn’t make much sense
- For 90% availability with 1M nodes, each none needs:
  - 7 nines

\[
A = \frac{MTTF}{MTTF + MTTR} = \frac{1}{1 + \frac{MTTR}{MTTF}}
\]

<table>
<thead>
<tr>
<th>9s</th>
<th>Availability</th>
<th>Annual Downtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90%</td>
<td>36 days, 12 hours</td>
</tr>
<tr>
<td>2</td>
<td>99%</td>
<td>87 hours, 36 minutes</td>
</tr>
<tr>
<td>3</td>
<td>99.9%</td>
<td>8 hours, 45.6 minutes</td>
</tr>
<tr>
<td>4</td>
<td>99.99%</td>
<td>52 minutes, 33.6 seconds</td>
</tr>
<tr>
<td>5</td>
<td>99.999%</td>
<td>5 minutes, 15.4 seconds</td>
</tr>
<tr>
<td>6</td>
<td>99.9999%</td>
<td>31.5 seconds</td>
</tr>
</tbody>
</table>

C. Engelmann: Resilient Software for ExaScale Computing
Why are we only talking about MTTI?

- None of these MTTI numbers are accurate ±O(10)
- Looking at MTTI alone doesn’t make much sense
- For 90% availability with 1M nodes, each node needs:
  - 7 nines without redundancy
  - 4 nines for DMR
  - 3 nines for dynamic DMR
  - 3 nines for TMR
  - 2 nines for dynamic TMR

\[ A = \frac{MTTF}{MTTF + MTR} = \frac{1}{1 + \frac{MTR}{MTTF}} \]

<table>
<thead>
<tr>
<th>9s</th>
<th>Availability</th>
<th>Annual Downtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90%</td>
<td>36 days, 12 hours</td>
</tr>
<tr>
<td>2</td>
<td>99%</td>
<td>87 hours, 36 minutes</td>
</tr>
<tr>
<td>3</td>
<td>99.9%</td>
<td>8 hours, 45.6 minutes</td>
</tr>
<tr>
<td>4</td>
<td>99.99%</td>
<td>52 minutes, 33.6 seconds</td>
</tr>
<tr>
<td>5</td>
<td>99.999%</td>
<td>5 minutes, 15.4 seconds</td>
</tr>
<tr>
<td>6</td>
<td>99.9999%</td>
<td>31.5 seconds</td>
</tr>
</tbody>
</table>

C. Engelmann: Resilient Software for ExaScale Computing
Why are we only talking about MTTI?

• None of these MTTI numbers are accurate ±O(10)
• Looking at MTTI alone doesn’t make much sense
• For 90% availability with 1M nodes, each none needs:
  - 7 nines without redundancy
  - 4 nines for DMR
  - 3 nines for dynamic DMR
  - 3 nines for TMR
  - 2 nines for dynamic TMR
  - ? nines for checkpoint/restart
  - ? nines for message logging
  - ? nines for algorithm-based fault tolerance

\[ A = \frac{MTTF}{MTTF + MTR} = \frac{1}{1 + \frac{MTR}{MTTF}} \]

<table>
<thead>
<tr>
<th>9s</th>
<th>Availability</th>
<th>Annual Downtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90%</td>
<td>36 days, 12 hours</td>
</tr>
<tr>
<td>2</td>
<td>99%</td>
<td>87 hours, 36 minutes</td>
</tr>
<tr>
<td>3</td>
<td>99.9%</td>
<td>8 hours, 45.6 minutes</td>
</tr>
<tr>
<td>4</td>
<td>99.99%</td>
<td>52 minutes, 33.6 seconds</td>
</tr>
<tr>
<td>5</td>
<td>99.999%</td>
<td>5 minutes, 15.4 seconds</td>
</tr>
<tr>
<td>6</td>
<td>99.9999%</td>
<td>31.5 seconds</td>
</tr>
</tbody>
</table>
Why are we only talking about MTTI?

- None of these MTTI numbers are accurate ±O(10)
- Looking at MTTI alone doesn’t make much sense
- For 90% availability with 1M nodes, each node needs:
  - 7 nines without redundancy
  - 4 nines for DMR
  - 3 nines for dynamic DMR
  - 3 nines for TMR
  - 2 nines for dynamic TMR
  - ? nines for checkpoint/restart
  - ? nines for message logging
  - ? nines for algorithm-based fault tolerance

\[
A = \frac{MTTF}{MTTF + MTRR} = \frac{1}{1 + \frac{MTRR}{MTTF}}
\]

<table>
<thead>
<tr>
<th>9s</th>
<th>Availability</th>
<th>Annual Downtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90%</td>
<td>36 days, 12 hours</td>
</tr>
<tr>
<td>2</td>
<td>99%</td>
<td>87 hours, 36 minutes</td>
</tr>
<tr>
<td>3</td>
<td>99.9%</td>
<td>8 hours, 45.6 minutes</td>
</tr>
<tr>
<td>4</td>
<td>99.99%</td>
<td>52 minutes, 33.6 seconds</td>
</tr>
<tr>
<td>5</td>
<td>99.999%</td>
<td>5 minutes, 15.4 seconds</td>
</tr>
<tr>
<td>6</td>
<td>99.9999%</td>
<td>31.5 seconds</td>
</tr>
</tbody>
</table>

- There is a huge gap in HPC resilience models and metrics
HPC Resilience Solutions

- "Existing" solutions
  - Application-level C/R, system-level C/R, incremental C/R
  - C/R to memory or SSDs in neighbor or I/O nodes
  - Message logging + C/R
  - Proactive fault tolerance (migration-based fault avoidance)
  - Rejuvenation (reboot to clear latent errors)
  - Process-level redundancy
HPC Resilience Solutions

• “Existing” solutions
  - Application-level C/R, system-level C/R, incremental C/R
  - C/R to memory or SSDs in neighbor or I/O nodes
  - Message logging + C/R
  - Proactive fault tolerance (migration-based fault avoidance)
  - Rejuvenation (reboot to clear latent errors)
  - Process-level redundancy

• Only the C/R solutions are ready for production

• None of the advanced solutions are even close to that
HPC Resilience Solutions

• “Existing” solutions
  - Application-level C/R, system-level C/R, incremental C/R
  - C/R to memory or SSDs in neighbor or I/O nodes
  - Message logging + C/R
  - Proactive fault tolerance (migration-based fault avoidance)
  - Rejuvenation (reboot to clear latent errors)
  - Process-level redundancy

• Only the C/R solutions are ready for production

• None of the advanced solutions are even close to that

• There is a huge gap between research and production HPC resilience solutions

C. Engelmann: Resilient Software for ExaScale Computing
What about soft errors?

- Smaller circuit sizes in conjunction with lower circuit voltages will result in higher soft error rates
What about soft errors?

• Smaller circuit sizes in conjunction with lower circuit voltages will result in higher soft error rates

• Missing strategy for silent data/code corruption will cause applications to produce erroneous results, hang or crash

• Initial work focuses on redundancy and online correction
What about soft errors?

- Smaller circuit sizes in conjunction with lower circuit voltages will result in higher soft error rates

- Missing strategy for silent data/code corruption will cause applications to produce erroneous results, hang or crash

- Initial work focuses on redundancy and online correction

- There is a huge gap in understanding and dealing with soft errors
Key Areas for Future Resilience Research, Development, and Standards Work
Proactive Fault Tolerance (PFT) Framework

• Objectives
  – Facilitate fault resilience at extreme scale
  – Offer migration-based fault avoidance for 24x7 RAS

• Impact
  – Permit avoiding the impact of predictable failures
  – Help in understanding failure causes and propagation

• Novelty
  – Establishes a new area of research in HPC resilience
  – System-software approach for anticipating failures
  – Offers process migration and scalable system monitoring
  – Builds theoretical foundations:
    • System and component reliability analysis
    • System logging and monitoring requirements analysis
PFT: Reactive vs. Proactive Fault Tolerance

- **Reactive fault tolerance**
  - Keeps parallel applications alive through recovery from experienced failures
  - Employed mechanisms react to failures
  - Examples: Checkpoint/restart and message logging/replay

- **Proactive fault tolerance**
  - Keeps parallel applications alive by avoiding failures through preventative measures
  - Employed mechanisms anticipate failures
  - Example: Migration and rejuvenation
PFT: Approach

- Relies on a feedback-loop control mechanism
  - Application health is constantly monitored and analyzed
  - Application is reallocated to avoid failures
  - Closed-loop control similar to dynamic load balancing

- Real-time control problem
  - Need to act in time to avoid imminent failures

- No 100% coverage
  - Not all failures can be anticipated
PFT: BLCR Process-Level Migration

- **Single migration overhead**
  - Stop & copy: 0.09-6.00%
  - Live: 0.08-2.98%

- **Single migration duration**
  - Stop & copy: 1.0-1.9s
  - Live: 2.6-6.5s

- **Application downtime**
  - Stop & copy > Live

- **Node eviction time**
  - Stop & copy < Live

NPB runs on 16-node dual-core dual-processor Linux cluster at NCSU with AMD Opteron and Gigabit Ethernet

C. Engelmann: Resilient Software for ExaScale Computing
PFT: System Monitoring Tool using MRNet

- Aggregation of metrics
- Tree-based overlay network
- Fan-in for metric data
- Fan-out for management
- Classification of data on back-end nodes
- In-flight processing on intermediate nodes
- Collection and storing on front-end node

- 1 MB of data in 4 hours
- ≈250 kB/hour
- ≈2 kb/interval
- ≈56x less than Ganglia

C. Engelmann: Resilient Software for ExaScale Computing
PFT: Current Status and Future Work

- Developed prototypes for proactive fault tolerance
  - Core framework that plugs individual components together
  - Interfaces to job and resource manager, system monitoring, message logging and runtime environment (MPI)
  - Process-level migration solution (BLCR)
  - Scalable system monitoring solution (MRNet-based)
  - Statistical analyses of system log and monitoring data
MR-MPI: Process-Level Redundancy for MPI

• Objectives
  – Facilitate hard and soft error resilience at extreme scale
  – Offer process-level redundancy for seamless resilience

• Impact
  – Survive hard and soft errors without the need for recovery
  – Detect and optionally correct silent data corruption

• Novelty
  – Establishes a new area of research in HPC resilience
  – Software-only approach, i.e., no need for expensive hardware
  – Redundancy at the Message Passing Interface (MPI)
  – Transparent: No application modification needed
  – Provides resilience on-demand on a job-by-job basis
  – Only solution to support file I/O under redundancy
MR-MPI: Technical Approach

- Aim at MPI process-level redundancy
- Transparent redundant execution of MPI processes:
  - On the same processor
  - On different processors
  - On different compute nodes
- Input replication and output comparison between the MPI library and the application
- The fault model is fail-stop
- MPI and platform independent

C. Engelmann: Resilient Software for ExaScale Computing
MR-MPI: Design (1/2)

- $r \times m$ native MPI ranks:
  - $r$ ranks visible to the application
  - $m$ is the replication degree
  - $<np> = r \times m$
  - $<vnp> = r$

- Full message replication

- Master-failover for non-determinism, e.g., for MPI_Wtime()

- Partial replication is supported statically
  - For example, execute with a replication degree of 1.5
MR-MPI: Design (2/2)

- Intercepts POSIX file I/O calls from the application
- New file I/O calls employ coordination protocols for
  - Redundancy-oblivious node-local file system access
  - Redundancy-aware shared networked file system access
- File read and write protocols with different features
  - Utilizing parallel I/O and inter-node communication
Acknowledgements

• Investigators at Oak Ridge National Laboratory (ORNL):

• Investigators at Louisiana Tech University:
  – C. Leangsuksun, N. Naksinehaboon, R. Nassar, M. Paun

• Investigators at North Carolina State University:
  – F. Mueller, A. Nagarajan, J. Varma, X. Ma, F. Meng

• Investigators at Virginia Tech
  – M. Li, A. Butt

• Funding sources:
  – U.S. Department of Energy, Office of Science, FASTOS 2
  – U.S. National Science Foundation, CCF-0937827, CCF-0746832, CCF 0621470, CCF-0937690
  – ORNL LDRD and NCSU/ORNL Joint Appointment programs
Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA)

- @ SC’10, SC’11
- Topics of interest include, but are not limited to:
  - Novel scientific algorithms that improve performance, scalability, resilience and power efficiency
  - Porting algorithms to many-core and heterogeneous architectures
  - Performance and resilience limitations of algorithms at scale
  - Crosscutting approaches (system software and applications)
  - Scientific algorithms that can exploit extreme concurrency
  - Naturally fault tolerant, self-healing or fault oblivious algorithms
  - Programming models & system software for scalability/resilience
- Chairs: V. Alexandrov (BSC), J. Dongarra (UT), A. Geist (ORNL)
- URL: http://www.csm.ornl.gov/srt/conferences/Scala
Workshop on Resiliency in HPC (Resilience) in Clusters, Clouds, and Grids

- @ Euro-Par’11, CCGrid’10, HPDC’09, CCGrid’08
- Topics of interest include, but are not limited to:
  - Reports on current HPC system and application resiliency
  - HPC resiliency metrics, standards, and analysis
  - HPC system and application-level fault handling and anticipation
  - HPC system and application health monitoring
  - Resiliency for HPC file and storage systems
  - Algorithm-based resiliency fundamentals for HPC (not Hadoop)
  - Fault tolerant MPI concepts and solutions
  - Soft error detection and recovery in HPC systems
- Chairs: S. Scott (TTU/ORNL) and C. Leangsuksun (LA Tech)
- URL: http://xcr.cenit.latech.edu/resilience20XX/

C. Engelmann: Resilient Software for ExaScale Computing