Reliability, Availability, and Serviceability (RAS) for Petascale High-End Computing and Beyond

Christian Engelmann and Stephen L. Scott

Computer Science and Mathematics Division
Oak Ridge National Laboratory

www.fastos.org/ras
Motivation

• Large-scale PFlop/s systems have arrived
  – #1 ORNL Jaguar XT5: 1.759 PFlop/s LINPACK, 224,162 cores
  – #2 NSCS Nebulae: 1.271 PFlop/s LINPACK, 120,640 cores
  – #3 LANL Roadrunner: 1.042 PFlop/s LINPACK, 122,400 cores

• Other large-scale systems exist
  – #4 NICS Kraken XT5: 0.831 PFlop/s LINPACK, 98,928 cores
  – #5 Juelich JUGENE: 0.825 PFlop/s LINPACK, 294,912 cores
  – #6 NASA Pleiades: 0.773 PFlop/s LINPACK, 81,920 cores

• The trend is toward even larger-scale systems
### Toward Exascale Computing (My Roadmap)

*Based on proposed DOE roadmap with MTTI adjusted to scale linearly*

<table>
<thead>
<tr>
<th>Systems</th>
<th>2009</th>
<th>2011</th>
<th>2015</th>
<th>2018</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>200 GF</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>4 days</td>
<td>19 h 4 min</td>
<td>3 h 52 min</td>
<td>1 h 56 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>

C. Engelmann and S.L. Scott: Reliability, Availability, and Serviceability (RAS) for Petascale High-End Computing and Beyond
Factors Driving up the Error Rate

• Significant growth in component count (up to 50x nodes) results in respectively higher system error rate

• Smaller circuit sizes and lower voltages increase soft error vulnerability (bit flips caused by thermal and voltage variations as well as radiation)

• Power management cycling decreases component lifetimes due to thermal and mechanical stresses

• Hardware fault detection and recovery is limited by power consumption requirements and costs

• Heterogeneous architectures (CPU & GPU cores) add more complexity to fault detection and recovery
Risks of the Business as Usual Approach

- Increased error rate requires more frequent checkpoint/restart, thus lowering efficiency (application progress)
- Current application-level checkpoint/restart to a parallel file system is becoming less efficient and soon obsolete
- Memory to I/O ratio (dump time) improves from 25 min to 8.3 min, but concurrency for coordination and I/O scheduling increases significantly (50x nodes, 444x cores)
- Missing strategy for silent data/code corruption will cause applications to produce erroneous results or just hang
Objectives

- Develop scalable system software technologies to achieve high-level RAS for next-generation petascale scientific high-end computing resources
- Provide for non-stop scientific computing on a 24x7 basis without interruption with virtualized adaptation, reconfiguration, and preemptive measures
- Address the RAS research challenges outlined by DOE’s Forum to Address Scalable Technology for Runtime and Operating Systems (FAST-OS)
Approach

• Leverage virtualization for transparent fault tolerance on extreme scale computing systems
• Perform reliability analysis to enable failure prediction
• Investigate proactive fault tolerance using migration away from components that are “about to fail”
• Develop reactive fault tolerance enhancements, such as checkpoint interval and placement adaption
• Offer holistic fault tolerance through combination of adaptive proactive and reactive fault tolerance
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
Proactive Fault Tolerance using Migration

- Relies on a feedback-loop control mechanism
  - Application health is constantly monitored and analyzed
  - Application is reallocated to improve 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, such as random bit flips
VM-level Migration with Xen

- Type 1 system setup
  - Xen VMM on entire system
  - Host OS for management
  - Guest OS for computation
  - Spares without Guest OS
  - Monitoring in Host OS
  - Decentralized scheduler/load balancer w/ Ganglia

- Deteriorating node health
  - Ganglia threshold trigger
  - Migrate guest OS to spare
  - Utilize Xen migration
VM-level Migration Performance Impact

- **Single migration overhead**
  - Live: 0.5-5.0%

- **Double migration overhead**
  - Live: 2.0-8.0%

- **Migration duration**
  - Stop & copy: 13-14s
  - Live: 14-24s

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

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

- LAM/MPI with Berkeley Lab Checkpoint/Restart (BLCR)
- Per-node health monitoring
- New decentralized scheduler/load balancer in LAM
- New process migration facility in BLCR (stop&copy and live)
- Deteriorating node health
  - Simple threshold trigger
  - Migrate process to spare
- Available through BLCR distribution

C. Engelmann and S.L. Scott: Reliability, Availability, and Serviceability (RAS) for Petascale High-End Computing and Beyond
Process-Level Migration Performance Impact

- 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
Proactive Fault Tolerance Framework

- Central MySQL database
- Environmental monitoring
  - OpenIPMI and Ganglia
- Event logging and analysis
  - Syslog forwarding
- Job & resource monitoring
  - Torque (epilogue/prologue)
- Migration mechanism
  - Process-level with BLCR
System Monitoring with Ganglia and Syslog

Experiment #1:
• 32-node Linux cluster
• 30 second interval
• 40 Ganglia metrics
• \(\approx 20 \text{ GB of data in 27 days}\)
• \(\approx 33 \text{ MB/hour}\)
• \(\approx 275 \text{ kb/interval}\)

Experiment #2:
• 32-node Linux cluster
• 30 second interval
• 40 Ganglia metrics
• No measurable impact on NAS benchmarks

<table>
<thead>
<tr>
<th>Class C NPB on 32 nodes</th>
<th>CG</th>
<th>FT</th>
<th>LU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average time in seconds</td>
<td>264</td>
<td>235</td>
<td>261</td>
</tr>
<tr>
<td>Average time under load in seconds</td>
<td>264</td>
<td>236</td>
<td>260</td>
</tr>
</tbody>
</table>

Table 2. NPB test results (averages over 10 runs)
MRNet-based System Monitoring

- 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
Incremental Checkpointing with BLCR

- Recent enhancement for Berkeley Lab Checkpoint/Restart (BLCR)
- Track differences with dirty bit at PTE
- Hybrid: 1 full and $k$ incremental checkpoints
- Available through BLCR distribution
Simulation of Fault Tolerance Policies

- DES with actual system logs
- Evaluation of policies
  - Reactive only
  - Proactive only
  - Reactive/proactive combination
- Evaluation of parameters
  - Checkpoint interval
  - Prediction accuracy
- Customizable simulation
  - # of active/spare nodes
  - Checkpoint and migration overheads
Combining Proactive & Reactive Approaches

- Optimum for the given logs:
  - Prediction accuracy > 60%
  - Checkpoint interval 16-32h

- Results for higher accuracies and very low intervals are worse than only proactive or only reactive

<table>
<thead>
<tr>
<th>Number of processes</th>
<th>125</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active/Spare nodes</td>
<td>125/12</td>
</tr>
<tr>
<td>Checkpoint overhead</td>
<td>50min</td>
</tr>
<tr>
<td>Migration overhead</td>
<td>1 min</td>
</tr>
</tbody>
</table>

Simulation based on ASCI White logs (nodes 1-125 and 500-512)
Research in Reliability Modeling

- Application MTTI estimation
  - Monitoring & recording of application & system health
  - Reliability analysis on recorded data
  - Adaptation of checkpoint interval to system health

- Finding failure patterns
  - Additional recording of application interrupts
  - Reliability analysis on recent and historical data
Accomplishments

• Select developed software
  – BLCR-based process migration
  – BLCR-based incremental/adaptive checkpointing
  – MRNet-based scalable system monitoring
  – Fully integrated RAS framework for Linux clusters
  – Simulation framework for fault tolerance policies
  – Xen-based virtual machine migration framework

• Select developed theoretical foundations:
  – Analyzed system and component reliability
  – Identified needs for more accurate failure reporting

• Numerous high-level publications
  – ICS’06, ICS’07, IPDPS’07, SC’08, …
Ongoing FAST-OS Work

• Extending the MRNet-based system monitoring
  – In-flight statistical processing of monitoring data
  – In-flight pattern matching of syslog data (for reduction)
• Advanced statistical analysis for anomalous detection
• Compute-node rejuvenation, e.g., reboot between jobs
• Checkpointing GPUs using CUDA streams
• Next HA-OSCAR release:
  – Extends the OSCAR Linux cluster installation and management suite with the developed RAS mechanisms
Acknowledgements

• Investigators at Oak Ridge National Laboratory:
  – *Stephen L. Scott [Lead PI]*, Christian Engelmann, Geoffroy Vallée, Thomas Naughton, Anand Tikotekar, George Ostrouchov

• Investigators at Louisiana Tech University:
  – *Chokchai (Box) Leangsuksun [Lead PI]*, Nichamon Naksinehaboon, Raja Nassar, Mihaela Paun

• Investigators at North Carolina State University:
  – *Frank Mueller [Lead PI]*, Chao Wang, Arun Nagarajan, Jyothish Varma

• Funding sources:
  – U.S. Department of Energy, Office of Science, FASTOS 2