Advanced Fault Tolerance Solutions for High Performance Computing

Christian Engelmann

Oak Ridge National Laboratory, Oak Ridge, USA
The University of Reading, Reading, UK
Largest Multipurpose Science Laboratory within the U.S. Department of Energy

- Privately managed for US DOE
- $1.08 billion budget
- 4000+ employees total
  - 1500 scientists and engineers
- 3,000 research guests annually
- 30,000 visitors each year
- Total land area 58mi² (150km²)

- Nation’s largest energy laboratory
- Nation’s largest science facility:
  - The $1.4 billion Spallation Neutron Source
- Nation’s largest concentration of open source materials research
- Nation’s largest open scientific computing facility
National Center for Computational Sciences

- 40,000 ft² (3700 m²) computer center:
  - 36-in (~1m) raised floor, 18 ft (5.5 m) deck-to-deck
  - 12 MW of power with 4,800 t of redundant cooling
  - High-ceiling area for visualization lab:
    - 35 MPixel PowerWall, Access Grid, etc.

- 2 systems in the Top 500 List of Supercomputer Sites:
  - Phoenix: 32? Cray X1E, Vector with 1014 Processors ⇒ 18 TFlop.
At Forefront in Scientific Computing and Simulation

- Leading partnership in developing the National Leadership Computing Facility
  - Leadership-class scientific computing capability
  - 100 TFlop/s in 2006/7 (recently installed)
  - 250 TFlop/s in 2007/8 (commitment made)
  - 1 PFlop/s in 2008/9 (proposed)

- Attacking key computational challenges
  - Climate change
  - Nuclear astrophysics
  - Fusion energy
  - Materials sciences
  - Biology

- Providing access to computational resources through high-speed networking (10Gbps)
Talk Outline

- High performance computing system architectures
- Fault tolerance solutions for head & service nodes:
  - Active/standby with shared storage
  - Active/standby replication
  - Asymmetric active/active replication
  - Symmetric active/active replication
- Fault tolerance solutions for compute nodes:
  - Reactive: Checkpoint/restart and message logging
  - Proactive: Preemptive migration
  - Algorithmic approaches
Advanced Fault Tolerance Solutions for High Performance Computing

HPC System Architectures
Beowulf Cluster Computing Architecture

• Single head node manages entire HPC system
• System-wide services are provided by head node:
  • Job & resource management, networked file system, …
• Local services are provided by compute nodes
  • Message passing (MPI, PVM), …
Massively Parallel Computing Architecture

- Single point of failure
- Single point of control
- Single points of failure
- Single points of control

- Single head node and additional service nodes manage the entire HPC system
- System-wide services are provided by head node and are offloaded to service nodes, e.g., networked file system
- Local services are provided by service nodes and compute nodes, e.g., message passing
Partitioned MPP Computing Architecture

- Single head node manages entire HPC system
- Service nodes manage and support compute nodes belonging to their partitions
**Typical Failure Causes in HPC Systems**

- **Overheating !!!**

- **Memory and network errors (bit flips)**

- **Hardware failures due to wear/age of:**
  - Hard drives, memory modules, network cards, processors

- **Software failures due to bugs in:**
  - Operating system, middleware, applications

- **Different scale requires different solutions:**
  - Compute nodes (up to 150,000)
  - Front-end, service, and I/O nodes (1 to 150)
### Availability Measured by the Nines

http://info.nccs.gov/resources - HPC system status at Oak Ridge National Laboratory

<table>
<thead>
<tr>
<th>9's</th>
<th>Availability</th>
<th>Downtime/Year</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90.0%</td>
<td>36 days, 12 hours</td>
<td>Personal Computers</td>
</tr>
<tr>
<td>2</td>
<td>99.0%</td>
<td>87 hours, 36 min</td>
<td>Entry Level Business</td>
</tr>
<tr>
<td>3</td>
<td>99.9%</td>
<td>8 hours, 45.6 min</td>
<td>ISPs, Mainstream Business</td>
</tr>
<tr>
<td>4</td>
<td>99.99%</td>
<td>52 min, 33.6 sec</td>
<td>Data Centers</td>
</tr>
<tr>
<td>5</td>
<td>99.999%</td>
<td>5 min, 15.4 sec</td>
<td>Banking, Medical</td>
</tr>
<tr>
<td>6</td>
<td>99.9999%</td>
<td>31.5 seconds</td>
<td>Military Defense</td>
</tr>
</tbody>
</table>

- Enterprise-class hardware + Stable Linux kernel = 5+
- Substandard hardware + Good high availability package = 2-3
- Today’s supercomputers = 1-2
- My desktop = 1-2
Fault Tolerance & High Availability Goals

- Provide high-level Reliability, Availability, and Serviceability (RAS) capabilities
- Eliminate many of the numerous single-points of failure and control in HPC systems

- Development of techniques to enable HPC systems to run computational jobs 24x7 without interruption
- Development of proof-of-concept implementations as blueprint for production-type RAS solutions
Advanced Fault Tolerance Solutions for High Performance Computing

Head and Service Nodes
Single Head/Service Node Problem

- Single point of failure
- Compute nodes sit idle while head node is down
- $A = \frac{MTTF}{MTTF + MTTR}$
- MTTF depends on head node hardware/software quality
- MTTR depends on the time it takes to repair/replace node
- $\Rightarrow MTTR = 0 \Rightarrow A = 1.00 (100\%)$ continuous availability
Active/Standy with Shared Storage

- Single active head node
- Backup to shared storage
- Simple checkpoint/restart
- Fail-over to standby node
- Possible corruption of backup state when failing during backup
- Introduction of a new single point of failure
- Correctness and availability are NOT ALWAYS guaranteed

→ SLURM, metadata servers of PVFS and Lustre
Active/Standby Redundancy

- Single active head node
- Backup to standby node
- Simple checkpoint/restart
- Fail-over to standby node
- Idle standby head node
- Rollback to backup
- Service interruption for fail-over and restore-over
  - Torque on Cray XT
  - HA-OSCAR prototype
Asymmetric Active/Active Redundancy

- Many active head nodes
- Work load distribution
- Optional fail-over to standby head node(s) \((n+1)\) or \(n+m\)
- No coordination between active head nodes
- Service interruption for fail-over and restore-over
- Loss of state w/o standby
- Limited use cases, such as high-throughput computing

Prototype based on HA-OSCAR
Symmetric Active/Active Redundancy

- Many active head nodes
- Work load distribution
- Symmetric replication between head nodes
- Continuous service
- Always up-to-date
- No fail-over, no restore-over
- Virtual synchrony model
- Complex algorithms
- JOSHUA prototype for Torque
- PVFS metadata server
JOSHUA: Symmetric Active/Active Replication for PBS Torque

Head Node Fails

To Outside World
To Compute Nodes

Schedule Job A
Schedule Job B
Schedule Job C
Launch Job A
Schedule Job D
Schedule Job E
Launch Job B
Launch Job C

No Single Point of Failure
No Single Point of Control
Symmetric Active/Active Replication

- **Output Unification**
- **Virtually Synchronous Processing**
- **Input Replication**
PVFS MDS Performance and Overhead

Writing Throughput

Throughput (Requests/sec)

Number of Clients

0 20 40 60 80 100 120

1 2 4 8 16 32

PVFS
A/A 1
A/A 2
A/A 4
PVFS MDS Performance and Overhead

Reading Throughput

Throughput (Requests/sec)

Number of Clients

PVFS MDS Performance and Overhead

Advanced Fault Tolerance Solutions for High Performance Computing
June 4, 2007
Workshop on Trends, Technologies and Collaborative Opportunities in High Performance and Grid Computing
Symmetric Active/Active Availability

- $A_{\text{component}} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}$
- $A_{\text{system}} = 1 - (1 - A_{\text{component}})^n$
- $T_{\text{down}} = 8760 \text{ hours} \times (1 - A)$
- Single node MTTF: 5000 hours
- Single node MTTR: 72 hours

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Availability</th>
<th>Est. Annual Downtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>98.58%</td>
<td>5d 4h 21m</td>
</tr>
<tr>
<td>2</td>
<td>99.97%</td>
<td>1h 45m</td>
</tr>
<tr>
<td>3</td>
<td>99.9997%</td>
<td>1m 30s</td>
</tr>
<tr>
<td>4</td>
<td>99.999995%</td>
<td>1s</td>
</tr>
</tbody>
</table>

Single-site redundancy for 7 nines does not mask catastrophic events.
Advanced Fault Tolerance Solutions for High Performance Computing

Compute Nodes
Reactive vs. Proactive Fault Tolerance

- **Reactive fault tolerance:**
  - State saving during failure-free operation
  - State recovery **after failure**
  - Assured quality of service, but limited scalability

- **Proactive fault tolerance:**
  - System health monitoring and online reliability modeling
  - Failure anticipation and prevention through prediction and reconfiguration **before failure**
  - Highly scalable, but not all failures can be anticipated

- **Ideal solution:** Matching combination of both
Reactive Fault Tolerance Techniques (1/2)

- **Checkpoint/restart:**
  - Application state from all processors is saved regularly on stable storage, such as local disk or networked file system
  - On failure, application is restarted using saved state
  - Checkpoint always involves data movement (local/network)
  - Restart always involves a rollback, i.e., lost computation
  - Example: Berkeley Lab Checkpoint/Restart (Linux mod.)
  - May be used in combination with message logging to avoid rollback (see next slide)
Reactive Fault Tolerance Techniques (2/2)

- Message logging:
  - All messages sent between application processes are logged to a central server
  - On failure, only the failed application part is restarted and replayed with saved messages
  - Doubles the number of messages
  - Message replay involves no rollback
  - Example: MPICH-VCL (MPI-based Chandy/Lamport alg.)
  - Combination with checkpoint/restart:
    - No rollback / shorter replay time, even higher overhead
Proactive Fault Tolerance Techniques

- Preemptive migration:
  - System health status is constantly monitored and evaluated
  - Monitoring data is processed by a filtering mechanism and/or an online reliability analysis
  - Pre-failure indicators are used to predict failures based on current system health status and historic information
  - Application parts (processes or virtual machines) are migrated away from compute nodes that are about to fail
  - Migration may be performed by stopping the application or live, while keeping the application running
Preemptive Migration with Xen

- Stand-by Xen host, no guest (spare node)
- Deteriorating health → migrate guest (w/ MPI app) to spare node
Preemptive Migration with Xen

- Stand-by Xen host, no guest (spare node)
- Deteriorating health → migrate guest (w/ MPI app) to spare node
- Destination host generates unsolicited ARP reply
  - indicates Guest VM IP has moved to new location
  - ARP tells peers to resend packets to new host
Algorithmic Fault Tolerance Approaches

- Naturally fault tolerant algorithms
  - Processes have only limited knowledge mostly about other processes in their neighborhood
  - Application is composed of local algorithms, where a failure has only a minor local impact
  - Examples: Chaotic relaxation, peer-to-peer communication

- Recovery & erasure codes
  - Reconstruction of lost information through algorithmic redundancy within the application
  - Rollback to consistent state through reverse computation
MIT Research: Paintable Computing

- In the future, embedded computers with a radio device will get as small as a paint pigment
- Supercomputers can be easily assembled by just painting a wall of embedded computers
- Applications are driven by cellular algorithms
Cellular Architecture (Smart Dust) Simulator

- Developed at ORNL in Java with native C and Fortran application support using JNI
- Runs as standalone or distributed application
- Lightweight framework simulates up to 1,000,000 lightweight processes on 9 real processors
- Standard and experimental networks:
  - Multi-dimensional mesh/torus
  - Nearest/Random neighbors
- Message driven simulation is not in real-time
- Primitive fault-tolerant MPI support
Each dot is a full processor/OS
864 IBM Power 4
2.3 Tera FLOPS

Earth Simulator

Cheetah at ORNL
Summary and Conclusion

- Presented several traditional and advanced fault tolerance technologies for HPC
- Different scale requires different solutions:
  - Compute nodes
  - Front-end, service, and I/O nodes
- Scalable fault tolerance technologies are paramount to the success of large-scale HPC systems
MOLAR: Adaptive Runtime Support for High-end Computing Operating and Runtime Systems

- Addresses the challenges for operating and runtime systems to run large applications efficiently on future ultra-scale high-end computers.
- Part of the Forum to Address Scalable Technology for Runtime and Operating Systems (FAST-OS).
- MOLAR is a collaborative research effort (www.fastos.org/molar):

![Oak Ridge National Laboratory]
![NC State University]
![Ohio State University]
![Louisiana Tech University]
![The University of Reading]
![CRAY - The Supercomputer Company]
Advanced Fault Tolerance Solutions for High Performance Computing

Christian Engelmann

Oak Ridge National Laboratory, Oak Ridge, USA
The University of Reading, Reading, UK