Service-Level High Availability in Parallel and Distributed Systems

Christian Engelmann\textsuperscript{1,2}

\textsuperscript{1} Oak Ridge National Laboratory, Oak Ridge, USA
\textsuperscript{2} The University of Reading, Reading, UK
Talk Outline

- Scientific high-end computing (HEC)
- Availability deficiencies of today’s HEC systems
- Projects and accomplishments overviews
- High availability (HA) models for services
- Developed prototypes overview
- Existing limitations and most pressing issues
- Generalization of HA programming models
- Enhancing the transparency of the HA infrastructure
- Generic HA framework infrastructure
Scientific High-End Computing (HEC)

- **Large-scale HPC systems.**
  - Tens-to-hundreds of thousands of processors.
  - Current systems: IBM Blue Gene/L and Cray XT4
  - Next-generation: petascale IBM Blue Gene and Cray XT

- **Computationally and data intensive applications.**
  - 100 TFLOP – 1PFLOP with 100 TB – 10 PB of data.
  - Climate change, nuclear astrophysics, fusion energy, materials sciences, biology, nanotechnology, …

- **Capability vs. capacity computing**
  - Single jobs occupy large-scale high-performance computing systems for weeks and months at a time.
Projected Performance Development

1 PFlop/s ~2008/9

Scientific High-End Computing

IBM Blue Gene/L
Earth Simulator
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: 58. 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
  - 119 TFlop/s in 2007 (recently installed)
  - 250 TFlop/s in 2007/8 (commitment made, after SC07)
  - 500 TFlop/s in 2008 (commitment made, UT/ORNL)
  - 1 PFlop/s in 2008/9 (commitment made)

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

- Providing access to computational resources through high-speed networking (10Gbps)
## Availability Measured by the Nines

See [http://info.nccs.gov/resources](http://info.nccs.gov/resources) for current status of HPC systems 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
 Typical failure causes:
• Overheating !!!
• Memory errors
• Network errors
• Other hardware issues
• Software bugs

Different scale requires different solutions:
• Compute nodes (10,000+)
• Front-end, service, and I/O nodes (50+)

Image source: Moreira et al., “Designing a Highly-Scalable Operating System: The Blue Gene/L Story”
Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, Nov. 11-17, Tampa, FL, USA.
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.
Projects Overview

- Initial **HA-OSCAR** research in active/standby technology for the batch job management system
- Recent **MOLAR/FAST-OS** research in compute/service node fault tolerance & high availability
- Recent **RAS LDRD** research in service node fault tolerance & high availability
- Ongoing **RAS/FAST-OS** research in compute node fault tolerance & high availability

⇒ 4-5 years of research and development in fault tolerance & high availability for HPC
Accomplishments Overview

- Investigated the overall background of HA technologies in the context of HPC
  - Detailed problem description
  - Conceptual models
  - Review of existing solutions
- Developed different replication strategies for providing high availability for HPC system services
  - Active/standby
  - Asymmetric active/active
  - Symmetric active/active
- Implemented several proof-of-concept prototypes
Active/Standby 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, meta data 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 necessary
- No restore-over necessary
- Virtual synchrony model
- Complex algorithms
- Prototype for Torque & PVFS
Developed Prototypes Overview (1/2)

- **Active/Standby HA-OSCAR**
  - High availability for Open PBS/TORQUE
  - Integration with compute node checkpoint/restart

- **Asymmetric active/active HA-OSCAR**
  - High availability for Open PBS & SGE
  - High throughput computing solution

- **Symmetric active/active JOSHUA**
  - High availability for PBS TORQUE and for PVFS metadata server
Existing Limitations

- The active/standby and asymmetric active/active technology interrupts the service during fail-over.
- Generic $n+1$ or $n+m$ asymmetric active/active configurations have not been developed yet.
- The 2+1 asymmetric active/active configuration uses two different service implementations.
- The developed symmetric active/active technology has certain stability and performance issues.
- All developed prototypes use a customized high availability environment.
- Missing interaction with compute node fault tolerance mechanisms (except for HA-OSCAR for head node fail-over).
Most Pressing Issues

- For production-type deployment
  - Stability – guaranteed quality of service
  - Performance – low replication overhead
  - Interaction with compute node fault tolerance mechanisms
    - e.g. procedure for failing PBS mom
  - Testing, enhancements, and staged deployment

- For extending the developed technologies
  - Portability – ability to apply technology to different services
  - Ease-of-use – simplified service HA management (RAS)
  - Generic HA framework needed
Next Step: Generic HA Framework

- Generalization of HA programming models
  - Active/Standby
  - Asymmetric active/active
  - Symmetric active/active

- Enhancing the transparency of the HA infrastructure
  - Minimum adaptation to the actual service protocol
  - Virtualized communication layer for abstraction

- Portability

- Ease-of-use
Next Step: Generic HA Framework

- Generalization of HA programming models
  - Active/Standby
  - Asymmetric active/active
  - Symmetric active/active

- Enhancing the transparency of the HA infrastructure
  - Minimum adaptation to the actual service protocol
  - Virtualized communication layer for abstraction

- Portability
- Ease-of-use
Failure Model

- **Fail-stop**
  - The service, its node, or its communication links, fail by simply stopping.
  - Failure detection mechanisms may be deployed to assure fail-stop behavior in certain cases, such as for incomplete or garbled messages.

- **Permanent failures**
  - Non-transient behavior assured by detection mechanisms via node fencing.
  - Recovery requires external intervention, such as repair or replacement of the failed component.

- Both assumptions match real-world properties.
Most, if not all, HPC system services are deterministic.

- Non-determinism introduced by random number generators or unsynchronized timers:
  - Removal of the use of random number generators in HPC system services.
  - Synchronization of timers (clocks) between replicas is trivial:
    - Closely coupled local area networks with low and constant latency.
    - Clock skew tolerable within certain boundaries (not real-time, not fully synchronous).
Active/Standby Generalization

- **Warm-Standby:**
  - Regular state updates from Active Service to Standby Service (push or pull)
- **Hot-Standby**
  - On-change state updates from Active Service to Standby Service (push)
  - Group communication style consistency required for state updates to multiple Standby Services
Asymmetric Active/Active Generalization

- Replication of service capability via multiple Active Services
- No replication of state among Active Services
- Mechanisms and semantics for optional Standby Services are the same as for Active/Standby
Symmetric Active/Active Generalization

- Replication of service capability via multiple Active Services
- Replication of state among Active Services
- Virtual synchrony (active replication) model
## Comparison of Replication Methods

<table>
<thead>
<tr>
<th>Model</th>
<th>MTTR</th>
<th>Latency Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Warm-Standby</td>
<td>$T_d + T_f + T_c + T_r$</td>
<td>0</td>
</tr>
<tr>
<td>Hot-Standby</td>
<td>$T_d + T_f + T_r$</td>
<td>$2l_{A,B}$</td>
</tr>
<tr>
<td>Asymmetric</td>
<td>$T_d + T_f + T_c + T_r$ or</td>
<td>0 or $2l_{A,A}$</td>
</tr>
<tr>
<td></td>
<td>$T_d + T_f + T_r$</td>
<td></td>
</tr>
<tr>
<td>Symmetric</td>
<td>$T_r$</td>
<td>$O(n)$</td>
</tr>
</tbody>
</table>

$T_d = \text{time between failure occurrence and detection (Heartbeat)}$

$T_f = \text{time between failure detection and fail-over (STONITH)}$

$T_c = \text{time to recover from checkpoint to previous state}$

$T_r = \text{time to reconfigure client/user connection}$

$l_{A,B} = \text{communication latency between A and B}$
Next Step: Generic HA Framework

- Generalization of HA programming models
  - Active/Standby
  - Asymmetric active/active
  - Symmetric active/active
- Enhancing the transparency of the HA infrastructure
  - Minimum adaptation to the actual service protocol
  - Virtualized communication layer for abstraction

➡ Portability
➡ Ease-of-use
Symmetric Active/Active Replication

Service Node A  
Service  
Service-Side Interceptor  
Client  
Client Node A

Service Node B  
Service  
Service-Side Interceptor  

Service Node C  
Service  
Service-Side Interceptor  
Client  
Client Node B

Group Communication
Peer-to-Peer Communication
Non-Transparent Connection Fail-Over

Service Node A
Service
Service-Side Interceptor
Client
Client Node A

Service Node B
Service
Service-Side Interceptor

Service Node C
Service
Service-Side Interceptor

Client
Client Node B

Group Communication
Peer-to-Peer Communication

Non-Transparent Connection Fail-Over
Transparent Connection Fail-Over

Virtual Communication Layer

Group Communication
Peer-to-Peer Communication
Interceptors in the Communication Path: What about Performance?

<table>
<thead>
<tr>
<th>Payload</th>
<th>Without Interceptors</th>
<th>With Service Interceptor</th>
<th>With Both Interceptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>100B</td>
<td>149.9μs</td>
<td>150.6μs/ +0.5%</td>
<td>178.4μs/ +19.0%</td>
</tr>
<tr>
<td>1KB</td>
<td>284.3μs</td>
<td>314.6μs/ +10.7%</td>
<td>346.7μs/ +21.9%</td>
</tr>
<tr>
<td>10KB</td>
<td>1.9ms</td>
<td>1.9ms/ ±0.0%</td>
<td>2.0ms/ +5.3%</td>
</tr>
<tr>
<td>100KB</td>
<td>22.3ms</td>
<td>22.5ms/ +0.8%</td>
<td>22.7ms/ +1.8%</td>
</tr>
</tbody>
</table>

Table 1. Ping-Pong Latency Comparison

<table>
<thead>
<tr>
<th>Payload</th>
<th>Without Interceptors</th>
<th>With Service Interceptor</th>
<th>With Both Interceptors</th>
</tr>
</thead>
<tbody>
<tr>
<td>100B</td>
<td>667KBps</td>
<td>664KBps/−0.4%</td>
<td>561KBps/−15.9%</td>
</tr>
<tr>
<td>1KB</td>
<td>3.5MBps</td>
<td>3.2MBps/−8.6%</td>
<td>2.9MBps/−17.1%</td>
</tr>
<tr>
<td>10KB</td>
<td>5.3MBps</td>
<td>5.2MBps/−1.9%</td>
<td>5.0MBps/−5.7%</td>
</tr>
<tr>
<td>100KB</td>
<td>4.5MBps</td>
<td>4.4MBps/−2.2%</td>
<td>4.4MBps/−2.2%</td>
</tr>
</tbody>
</table>

Table 2. Ping-Pong Bandwidth Comparison

Test Results from a 100 Mbit/s LAN Environment
Future Work

- Availability and reliability modeling
- Testing and benchmarking
- What about communication security/integrity?
  - For client-server connections across administrative domains
  - For distributed computing scenarios
- What about interdependent services?
How does this relate to P2P and the Grid

- All presented concepts and prototypes are applicable to any service-oriented architectures (SOAs).
- All prototypes are designed for local area replication, but can be easily used for wide area replication.
- Introduced latency overhead for local area replication protocols are negligible in the wide area context.
- Stateful Grid service replication:
  - Replication of Grid services to meet QoS/SLA guarantees.
- Stateful P2P service replication:
  - Replication of P2P support services for high availability, e.g., for directory servers and brokers.
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):

[Logos of participating institutions]

Office of Science
U.S. Department of Energy

Oak Ridge National Laboratory
Managed by UT-Battelle for the Department of Energy

NC State University
Ohio State University
Louisiana Tech University
The University of Reading
Cray, The Supercomputer Company
Service-Level High Availability in Parallel and Distributed Systems

Christian Engelmann$^{1,2}$

$^1$ Oak Ridge National Laboratory, Oak Ridge, USA
$^2$ The University of Reading, Reading, UK