Towards High Availability for High-Performance Computing System Services: Accomplishments and Limitations

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
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.
- 10 TFLOP – 1PFLOP with 10 TB – 1 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

Scientific High-End Computing

IBM Blue Gene/L

1 PFlop/s ~2008
### Availability Measured by the Nines

<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
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.

\[ MTTR = 0 \Rightarrow A = 1.00 \text{ (100\%) continuous availability.} \]
Projects Overview

- Initial **HA-OSCAR** research in active/standby technology for the batch job management system
- Ongoing **MOLAR** research in active/standby, asymmetric and symmetric active/active technology
- Recent **RAS LDRD** research in symmetric active/active technology
- 3-4 years of research and development in high availability for high-performance computing system services
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
High Availability Models

- **Active/Standby (Warm or Hot)**
  - For one active component at least one redundant inactive (standby) component
  - Fail-over model with idle standby component(s)
  - Level of high-availability depends on replication strategy

- **Active/Active (Asymmetric or Symmetric)**
  - Multiple redundant active components
  - No wasted system resources
  - State change requests can be accepted and may be executed by every member of the component group
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
- JOSHUA prototype for Torque
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
  - Fully transparent replication
Towards High Availability for High-Performance Computing System Services: Accomplishments and Limitations
Asymmetric Active-Active HA-OSCAR

Internet

PBS Server
Submitting PBS jobs to selected nodes

Heart Beat

On fly selection of computing nodes for PBS jobs

Standby Server

2GE Server
Submitting SGE jobs to selected nodes

Heart Beat

On fly selection of computing nodes for SGE jobs

Computing nodes can be overlapped performing PBS & SGE jobs simultaneously

March 14, 2007
Towards High Availability for High-Performance Computing System Services: Accomplishments and Limitations 16/42
Normal Active-Active Operation
Failover Active-Active Operation
Asymmetric Active/Active Availability

HA-OSCAR solution vs traditional Beowulf
Total Availability impacted by service nodes

Model assumption:
- scheduled downtime=200 hrs
- nodal MTTR = 24 hrs
- failover time=10s
- During maintenance on the head, standby node acts as primary

<table>
<thead>
<tr>
<th>Noda-wise mean time to failure (hr)</th>
<th>1000</th>
<th>2000</th>
<th>4000</th>
<th>6000</th>
<th>8000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beowulf</td>
<td>0.905797</td>
<td>0.915751</td>
<td>0.920810</td>
<td>0.922509</td>
<td>0.923361</td>
<td>0.923873</td>
</tr>
<tr>
<td>HA-oscar</td>
<td>0.999684</td>
<td>0.999896</td>
<td>0.999951</td>
<td>0.999962</td>
<td>0.999966</td>
<td>0.999968</td>
</tr>
</tbody>
</table>
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
**Introduced Overhead**

- Group communication system adds overhead for reliable and atomic multicast
- Latency increases with number of active nodes
- Throughput decreases with number of active nodes
- Overhead in acceptable range for this scenario

- **Nodes**: Pentium III 450MHz on 100MBit/s Ethernet

<table>
<thead>
<tr>
<th>System</th>
<th>#</th>
<th>Latency</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>TORQUE</td>
<td>1</td>
<td>98 ms</td>
<td></td>
</tr>
<tr>
<td>JOSHUA/TORQUE</td>
<td>1</td>
<td>134 ms</td>
<td>36 ms / 37%</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>265 ms</td>
<td>158 ms / 161%</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>304 ms</td>
<td>206 ms / 210%</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>349 ms</td>
<td>251 ms / 256%</td>
</tr>
</tbody>
</table>

**Job Submission Latency Overhead**

<table>
<thead>
<tr>
<th>System</th>
<th>#</th>
<th>10 Jobs</th>
<th>50 Jobs</th>
<th>100 Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>TORQUE</td>
<td>1</td>
<td>0.93 s</td>
<td>4.95 s</td>
<td>10.18 s</td>
</tr>
<tr>
<td>JOSHUA/TORQUE</td>
<td>1</td>
<td>1.32 s</td>
<td>6.48 s</td>
<td>14.08 s</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2.68 s</td>
<td>13.09 s</td>
<td>26.37 s</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>2.93 s</td>
<td>15.91 s</td>
<td>30.03 s</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3.62 s</td>
<td>17.65 s</td>
<td>33.32 s</td>
</tr>
</tbody>
</table>

**Job Submission Throughput Overhead**
Symmetric Active/Active Availability

- $A_{\text{component}} = \frac{MTTF}{MTTF + MTTR}$
- $A_{\text{system}} = 1 - (1 - A_{\text{component}})^n$
- $T_{\text{down}} = 8760 \text{ hours} \times (1 - A)$
- 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.
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
Communicating Process Generalization

Request Messages:
\[ r_p^1, r_p^2, r_p^3, \ldots \]

Output Messages:
\[ o_p^{1,1}, o_p^{1,2}, o_p^{3,1}, o_p^{3,2}, \ldots \]

Query Messages:
\[ q_p^{1,1}, q_p^{1,2}, q_p^{2,1}, q_p^{3,1}, \ldots \]

Output Messages:
\[ o_p^{1,1}, o_p^{1,2}, o_p^{2,1}, \ldots \]

Dependent Services/Users

Deterministic State Machine

Service p

User...

Service...

User...

Dependent Services/Users
Active/Standby Generalization
Asymmetric Active/Active Generalization
Symmetric Active/Active Generalization

Service A

Request Messages:
$r_A^{1}, r_B^{2}, r_C^{3}, \ldots$

Query Messages:
$q_C^{1,1}, q_A^{1,2}, q_B^{2,1}, q_A^{3,1}, \ldots$

Virtual Synchrony

Output Messages:
$o_A^{1,1}, o_A^{1,2}, o_C^{3,1}, o_C^{3,2}, \ldots$

Service B

Connection Fail-Over/Load Balancing

Service A

Active Service(s)

Service B

Connection Fail-Over/Load Balancing

Service A

Output Messages:
$o_C^{1,1}, o_A^{1,2}, o_B^{2,1}, o_A^{3,1}, o_A^{3,2}, \ldots$

Service A

User...

Dependent Services/Users

User...

Dependent Services/Users
Symmetric Active/Active Replication

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

- Service Node B
  - Service
  - Service-Side Interceptor
  - Group Communication

- 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

Non-Transparent Connection Fail-Over

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

Service Node A

Service

Service-Side Interceptor

Client-Side Interceptor

Client

Client Node A

Service Node B

Service

Service-Side Interceptor

Service-Side Interceptor

Client-Side Interceptor

Client

Client Node B

Service Node C

Service

Service-Side Interceptor

Client-Side Interceptor

Client

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
Modular HA Framework

- Pluggable component framework.
  - Communication drivers.
  - Group communication.
  - Virtual synchrony.
  - Applications.

- Interchangeable components.

- Adaptation to application needs, such as level of consistency.

- Adaptation to system properties, such as network and system scale.
Current Prototype

- Unique, flexible, dynamic, C-based component framework: Adaptive Runtime Environment (ARTE)
- Dynamic component loading/unloading on demand
- XML as interface description language (IDL)
- “Everything” is a component:
  - Communication driver modules
  - Group communication layer modules
  - Virtual synchrony layer modules
Future Work

- Continued implementation of framework components
  - Implementation of HA programming model components
- Integration with existing prototypes
  - For example, replacing Transis with the framework
- Availability and reliability modeling
- Testing and benchmarking
- Journal paper
- What about communication security/integrity?
  - For client-server connections across administrative domains
  - For distributed computing scenarios
- Start writing my PhD thesis!!!
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):
Directly Related Publications


Indirectly Related Publications


Towards High Availability for High-Performance Computing System Services: Accomplishments and Limitations

Christian Engelmann\textsuperscript{1,2}

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