High Availability for Ultra-scale Scientific High-End Computing

Christian Engelmann\(^1,2\)

\(^1\) Department of Computer Science, The University of Reading, Reading, RG6 6AH, UK

\(^2\) Computer Science and Mathematics Division Oak Ridge National Laboratory, Oak Ridge, TN, USA
Talk Outline

- Computer science research at Oak Ridge National Laboratory: Who we are and what we do…
- Availability deficiencies of today’s scientific high-end computing systems.
- Existing high availability solutions for scientific high-end computing systems.
- Proposed Thesis: High availability framework for scientific high-end computing systems.
- Internship opportunities for current MSc students.
Computer Science Research at Oak Ridge National Laboratory

Christian Engelmann\textsuperscript{1,2}

\textsuperscript{1}Department of Computer Science, The University of Reading, Reading, RG6 6AH, UK

\textsuperscript{2}Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN, USA
Largest Multipurpose Science Laboratory within the U.S. Department of Energy

- 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
- $300 million modernization in progress

• Privately managed for US DOE
• $1.06 billion budget
• 3,900 employees total
  • 1500 scientists and engineers
• 3,000 research guests annually
• 30,000 visitors each year
• Total land area 58mi² (150km²)
ORNL East Campus: Site of World Leading Computing and Computational Sciences

- Computational Sciences Building
- Research Office Building
- Engineering Technology Facility
- Old Computational Sciences Building (until June 2003)
- Joint Institute for Computational Sciences
- Research Support Center (Cafeteria, Conference, Visitor)
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.

- 3 systems in the Top 500 List of Supercomputer Sites:
  - Phoenix: 17. Cray X1E, Vector with 1024 Procs./4 TByte ⇒ 18 TFlop/s.
  - Cheetah: 283. IBM Power 4, Cluster with 864 Procs./1 TByte ⇒ 4.5 TFlop/s.
  - Ram: SGI Altix, SSI with 256 Procs./2 TByte ⇒ 1.4 TFlop/s.
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 (commitment made)
  - 250 TFlop/s in 2007 (commitment made)
  - 1 PFlop/s in 2008 (proposed)

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

- Providing access to computational resources through high-speed networking (10Gbps)
Computer Science Research Groups

- Computer Science and Mathematics (CSM) Division.
  - Applied research focused on computational sciences, intelligent systems, and information technologies.
- CSM Research Groups:
  - Climate Dynamics
  - Complex Systems
  - Computational Chemical Sciences
  - Computational Materials Science
  - Future Technologies
  - Statistics and Data Science
  - Computational Mathematics
  - Network and Cluster Computing
    (~20 researchers, 2 postdocs, 5 postmasters, 4 students, ++)
Network & Cluster Computing Projects

- Parallel Virtual Machine (PVM).
- MPI Specification, FT-MPI and Open MPI.
- Common Component Architecture (CCA).
- Open Source Cluster Application Resources (OSCAR).
- Scalable cluster tools (C3).
- Scalable Systems Software (SSS).
- Fault-tolerant metacomputing (HARNESS).
- High availability for high-end computing (RAS/MOLAR).
- Super-scalable algorithms research.
- Parallel storage systems (Freeloader).
Network & Cluster Computing Projects

- Parallel Virtual Machine (PVM).
- MPI Specification, FT-MPI and Open MPI.
- Common Component Architecture (CCA).
- Open Source Cluster Application Resources (OSCAR).
- Scalable cluster tools (C3).
- Scalable Systems Software (SSS).
- Fault-tolerant metacomputing (HARNESS).
- High availability for high-end computing (RAS/MOLAR).
- Super-scalable algorithms research.
- Parallel storage systems (Freeloader).
Availability Deficiencies of Today’s Scientific HEC Systems

Christian Engelmann$^{1,2}$

$^1$Department of Computer Science, The University of Reading, Reading, RG6 6AH, UK

$^2$Computer Science and Mathematics Division Oak Ridge National Laboratory, Oak Ridge, TN, USA
Scientific High-End Computing (HEC)

- Large-scale HPC systems.
  - Tens-to-hundreds of thousands of processors.
  - Current systems: IBM Blue Gene/L and Cray XT3
  - Next-generation systems: IBM Blue Gene/P and Cray XT4

- 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

http://www.top500.org/
### 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
IBM Blue Gene/L at LLNL

- #1 in Top 500.
- 367 TFLOPS.
- 131072 (700MHz) Power PC processors.
- 32 TB RAM.
- Partition (512 nodes) outage on single failure.
- MTBF = 40-50 hours.
- Weak I/O system prohibits checkpointing.
Vector Machines: Cray X1 (Phoenix)

- Number 17 in Top 500
- 1024 Vector Processors
- 4TB RAM, 18 TFLOPS
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.
- $MTTR = 0$ $\Rightarrow$ $A = 1.00$ (100%) continuous availability.
High Availability Solutions for Scientific HEC Systems

Christian Engelmann$^{1,2}$

$^1$Department of Computer Science, The University of Reading, Reading, RG6 6AH, UK

$^2$Computer Science and Mathematics Division Oak Ridge National Laboratory, Oak Ridge, TN, USA
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 Head/Service Nodes with Heartbeat Package and Shared Storage

- Single active head node.
- Backup to shared storage.
- Simple checkpoint/restart.
- Fail-over to standby node.
- Corruption of backup state when failing during backup.
- Introduction of a new single point of failure.

- Correctness and availability are NOT guaranteed.
- Folks, don’t do this!!!
- Bad examples: SLURM, PVFS2, and Luste.
Active/Standby Head/Service Nodes

- 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.
- Examples: HA-OSCAR, Torque on Cray XT3
Active/Standby PBS with HA-OSCAR
A-Active/Active Head/Service Nodes

- 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.
- Only solution: A-Active/Active HA-OSCAR.
S-Active/Active Head/Service Nodes

- 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.
- Only solution: JOSHUA.
S-Active/Active Torque with JOSHUA
S-Active/Active Torque with JOSHUA

Head Node Fails

No Single Point of Failure

No Single Point of Control

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

To Outside World
To Compute Nodes
Active/Active Redundancy for Nines

\[ A_{\text{component}} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \]

\[ A_{\text{system}} = 1 - (1 - A_{\text{component}})^n \]

\[ T_{\text{down}} = 8760 \text{ hours} \times (1 - A) \]

Single node MTTF of 5000-hours and MTTR 72 of 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>
</tbody>
</table>
Active/Active Redundancy for Nines

\[
A_{\text{component}} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}}
\]

\[
A_{\text{system}} = 1 - (1 - A_{\text{component}})^n
\]

\[
T_{\text{down}} = 8760 \text{ hours} \times (1 - A)
\]

Single node MTTF of 5000-hours and MTTR 72 of 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>
</tbody>
</table>
Active/Active Redundancy for Nines

\[ A_{\text{component}} = \frac{\text{MTTF}}{\text{MTTF} + \text{MTTR}} \]

\[ A_{\text{system}} = 1 - (1 - A_{\text{component}})^n \]

\[ T_{\text{down}} = 8760 \text{ hours} \times (1 - A) \]

Single node MTTF of 5000-hours and MTTR 72 of 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>
</tbody>
</table>
Active/Active Redundancy for Nines

\[ A_{\text{component}} = \frac{MTTF}{MTTF + MTTR} \]
\[ A_{\text{system}} = 1 - (1 - A_{\text{component}})^n \]
\[ T_{\text{down}} = 8760 \text{ hours} \times (1 - A) \]

Single node MTTF of 5000-hours and MTTR 72 of 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 make sense as it does not mask catastrophic events, such as flood, hurricane, tornado, earthquake, and terrorist attack.
High Availability Framework for Scientific HEC Systems

Christian Engelmann\textsuperscript{1,2}

\textsuperscript{1}Department of Computer Science, The University of Reading, Reading, RG6 6AH, UK

\textsuperscript{2}Computer Science and Mathematics Division Oak Ridge National Laboratory, Oak Ridge, TN, USA
Generic High Availability Framework

- **HA-OSCAR:**
  - Heartbeat for monitoring and IP-failover.
  - PBS specific scripts for replication to standby.

- **JOSHUA:**
  - Transis for group communication.
  - TORQUE specific commands for input replication.
  - TORQUE specific scripts for output unification.

- How can we provide active/stand-by and active/active high availability solutions for services in a generic, modular and configurable fashion?
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.
Initial Prototype

- Flexible, modular, pluggable component framework to provide RAS capabilities for services.
- C++ prototype developed as part of the RAS LDRD:
  - Object-oriented communication stack.
  - Dynamic loading of protocol components (Harness-based).
  - TCP and UDP communication drivers.
- Problems with the use of C++ and dynamic loading.
- Performance overhead due to C++ runtime.
  - Ongoing work focuses on pure C implementation.
Follow-on 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.
Other Major Accomplishments

- Development of a high availability taxonomy for HEC system architectures.
  - Definition of high availability terms and metrics for HEC.
  - Identification of single points of failure and control.
  - Evaluation and classification of existing solutions.

- Development of a high availability programming model for symmetric active/active replication.
  - Virtually synchronous environment model for easily making existing single services highly available.
  - JOSHUA prototype as proof-of-concept developed by Kai Uhlemann (2005/6 Reading MSc student internship).
Future Work

- Implementation of individual framework components.
  - Communication drivers and group communication.
- Design of high availability programming models.
  - Implementation of respective components.
- Integration with the JOSHUA solution.
  - Replacing Transis with the framework.
- Development of highly available system services.
  - Metadata server of a parallel file system, etc.
- Investigation and design of further use cases.
  - MPI, software management, etc.
Publications


Internship Opportunities for Current MSc students

Christian Engelmann$^1,2$

$^1$Department of Computer Science, The University of Reading, Reading, RG6 6AH, UK

$^2$Computer Science and Mathematics Division Oak Ridge National Laboratory, Oak Ridge, TN, USA
MSc Internship Basics

- 1-2 students for 6 months at Oak Ridge National Laboratory in Oak Ridge, Tennessee, USA.
- Full-time (40 hours per week) internship supervised by a research staff member.
- Individual leading-edge projects that include background investigation, design, and development.
- Includes MSc thesis and draft research paper writeup as part of the final MSc project.
- $1300-1500 per month stipend plus travel costs depending on student qualifications.
MSc Internship Timeline

- Early June: Application process (now)
  - Specify area of interest/project
  - Submit resume/CV to Vassil

- Late June: Acceptance notification
  Background Check/Subcontracts
  J-1 (Student) Visa application

- August: Visa issued through U.S. Embassy

- September 1: Start of internship

- February 28: End of internship

- March: Defense at the University of Reading
Further Practical Information

- Driver license is a must: No public transport to work.
- $3500 (2700€) in initial minimum funds needed for:
  - First rent and various deposits.
  - One-week car rental (reimbursed afterwards).
    - Is anyone under 25? Car rental/insurance is more expensive.
  - Used car, car sales tax, registration, and insurance.
- Break-even point:
  - 1 student after 4-5 months, 2 students after 2-3 months.
  - Most students leave with a net plus despite extra expenses for: high-speed Internet, cable TV, and weekend trips.
Possible Projects (see Handout)

- **Harness**
  - Design/Prototyping of Harness workbench architecture
  - Analysis of HPC development and deployment tools
  - Experiments with generalizing selected tools and subsystems
  - Development of prototype plug-in components

- **FreeLoader**
  - Diskless (in-memory) FreeLoader prototype
  - Data replication techniques
  - Integration of FreeLoader into Harness.
HARNESS: Pluggable Heterogeneous Distributed Virtual Machine

Exploring New Capabilities in Heterogeneous Distributed Computing

A Collaborative Research Effort Between Oak Ridge National Laboratory, University of Tennessee and Emory University

Fault Tolerance
- Petascale Approaches Beyond Standard Checkpoint/Restart
  - Checksum Based (a la RAID)
  - Localized State Neighborhoods
  - Incremental Checkpointing

Adaptability
- New Dynamic Environments
- Collaborating and Personal VMs
- Pervasive Computing

Multiple Plug-Ins and Parallel Paradigms
- PVM Plug-In
- Application Monitoring
- Fault-Tolerant MPI Plug-In

GRID Lite
- Personally Controlled (VM)
- Resource Sharing
- Minimum Modular Infrastructure
- Complements Existing DOE Data and Science Grids

Near Stateless Computing
- Task Communication
- Minimized Global State

Self-Assembling Virtual Machine
- Parallel Plug-Ins Provide Capabilities
- Parallel Software Modules (Plug-Ins) for Flexibility and Dynamic Customization

Harness Architecture
- Split and Merge with Other DVMs
- DVM Maintains Global State via Distributed Control
- Dynamically Customize and Extend via Plug-ins

H2O Kernel
- Implementations in C and Java
- Portable Multi-Threaded C Implementation

Office of Science

http://www.csm.ornl.gov/harness
The Harness Workbench

Unified and adaptive access to diverse HPC platforms

- Increasing the overall productivity of developing and executing computational codes.
- Optimizing the development and deployment processes of scientific applications.
- Simplifying application scientist activities using uniform and adaptive solutions.
- "Automagically" supporting the diversity of existing and emerging HPC architectures.

Virtualized command toolkit (VCT)
- Unified development, deployment and execution
- Common view across diverse HPC platforms
- User-space installation and virtual environments

Automatic adaptation using pluggable modules
- Virtualized command toolkit plug-ins
- Runtime environment plug-ins

Next generation runtime environment (RTE)
- Flexible, adaptive, lightweight framework
- Management of runtime tasks
- Support for diverse HPC platforms

Development environment and toolkit interfaces
- Easy-to-use interfaces for scientific application development, deployment and execution

Contact: Christian Engelmann · engelmann@ornl.gov
(865) 574-3132

http://icl.cs.utk.edu/harness
http://www.csm.ornl.gov/harness
http://www.mathcs.emory.edu/dc/harness

ORNL
Office of Science
U.S. Department of Energy
FreeLoader Distributed Storage Infrastructure Using Scavenging

Today's Hierarchical Storage Map

Pros:
- Excellent price/performance ratio
- Optimized for wide-area, bulk transfers and reliability

Cons:
- High deployment/maintenance/administrative costs
- Specialized software and central points of failure
- Low availability

Today's Hierarchical Storage Map

Motivation

Idea: Aggregate idle desktop storage to use for caching remote datasets

Benefits:
- Low cost (~$1 / GB)
- Low utilization means high availability for aggregation
- Creates GBs of nearby storage
- Decreases latency & increases bandwidth to remote datasets
- Low impact on individual desktops (load is shared by many)

Concerns:
- Volatility, trust, performance, user impact (disk, CPU, network)

Scalable, Decentralized Architecture

Storage Layer:
- Benefactor Nodes: Unit of contribution (Morsels)
- Basic morsel operations
- Space reclaim
- Data integrity through checksums
- Pools:
  - Benefactor registrations (soft state)
  - Dataset distributions, striping
  - Metadata
  - Selection heuristics

Management Layer:
- Pool registrations
- Replication and selection
- Grid awareness

Design Objectives and Assumptions

Design Goals:
- Scalable: O(100) or O(1000)
- Utilizing commodity components
- Preserving user autonomy
- Heterogeneity tolerant

Assumptions:
- Well-connected & secure corporate setting
- Large, immutable datasets (VORM)
- Use by wide-area and Grid clients

Use Cases

Supercomputing Facility
- Job queue
- Computation
- Data staging as a low-cost intermediate "Hep" for data staging or results transfer based on end-resilience availability

Data Grid
- Computation
- Scavenged Storage as a low-cost intermediate "Hep" for data staging or results transfer based on end-resilience availability

Client Site
- Scavenged Storage as a low-cost intermediate "Hep" for data staging or results transfer based on end-resilience availability

Status and Preliminary Results

Experiment Setup. FreeLoader results with an 8-node stripe width and 1 MiB stripe size. GridFTP transfers with 4 parallel streams and 1 MiB TCP buffers

Conclusions

- What the scavenged storage "is not":
  - Is not a replacement to high-end storage
  - Is not a file system
  - Is not intended to integrate storage resources at a wide-area scale

- What it "is":
  - Is a Low-cost, best-effort alternative
  - Is intended to facilitate:
    - Transient access to large, read-only datasets
    - Data sharing within an administrative domain
  - Is to be used with high-end and archival storage

Project Members:
- Suchi Shrestha, Vazhkuda, Xuefeng Liu, Vincent Frette, Jonathan Strickland, Nandan Tummala, Stephen Scott, and Al Gess

- Computer Science and Mathematics Division, Oak Ridge National Laboratory
- Computer Science Department, North Carolina State University
Questions and Comments
More information: www.csm.ornl.gov/~engelman

FIFA World Cup Opening Match at 5PM: Germany - Costa Rica