

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

C. Engelmann<sup>1,2</sup>, S. L. Scott<sup>1</sup>, C. Leangsuksun<sup>3</sup>, X. He<sup>4</sup>

<sup>1</sup>*Computer Science and Mathematics Division  
Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA*

<sup>2</sup>*Department of Computer Science  
The University of Reading, Reading, RG6 6AH, UK*

<sup>3</sup>*Computer Science Department  
Louisiana Tech University, Ruston, LA 71272, USA*

<sup>4</sup>*Department of Electrical and Computer Engineering  
Tennessee Technological University, Cookeville, TN 38505, USA  
engelmannc@ornl.gov, scottsl@ornl.gov, box@latech.edu, hexb@ntech.edu*

## Abstract

*During the last several years, our teams at Oak Ridge National Laboratory, Louisiana Tech University, and Tennessee Technological University focused on efficient redundancy strategies for head and service nodes of high-performance computing (HPC) systems in order to pave the way for high availability (HA) in HPC. These nodes typically run critical HPC system services, like job and resource management, and represent single points of failure and control for an entire HPC system. The overarching goal of our research is to provide high-level reliability, availability, and serviceability (RAS) for HPC systems by combining HA and HPC technology. This paper summarizes our accomplishments, such as developed concepts and implemented proof-of-concept prototypes, and describes existing limitations, such as performance issues, which need to be dealt with for production-type deployment.*

---

\*This research was partially sponsored by the Mathematical, Information, and Computational Sciences Division; Office of Advanced Scientific Computing Research; U.S. Department of Energy. The work was performed in part at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725. It was also performed in part at Louisiana Tech University under U.S. Department of Energy Grant No. DE-FG02-05ER25659. The research performed at Tennessee Tech University was partially supported by the U.S. National Science Foundation under Grant No. CNS-0617528 and the Research Office of the Tennessee Tech University under a Faculty Research Grant. The work at Oak Ridge National Laboratory and Tennessee Tech University was also partially sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory.

## 1. Introduction

High-performance computing (HPC) plays a significant role for the scientific research community as an enabling technology. Scientific HPC applications, like the Terascale Supernova Initiative [29] or the Community Climate System Model (CCSM) [4], help to understand the complex nature of open research questions and drive the race for scientific discovery through advanced computing in various scientific disciplines, such as in nuclear astrophysics, climate dynamics, fusion energy, nanotechnology, and human genomics.

The demand for continuous availability of HPC systems has risen dramatically with the recent trend towards capability computing, where scientific applications desire significant amounts of time (weeks and months) without interruption on the fastest HPC machines available. These high-end computing (HEC) systems must be able to run in the event of failures in such a manner that the capability is not severely degraded.

High availability (HA) computing has for a long time played an important role in mission critical applications, such as in the military, banking, and telecommunication sectors. Reliability, availability, and serviceability (RAS) solutions have dealt with HA issues, such as transparent masking of failures using redundancy strategies, for some time now.

The overarching goal of our research is to provide high-level RAS for HPC systems by combining HA and HPC technology, thus allowing scientific applications to better take advantage of the provided capability of HPC systems. This will invariably improve scientific

application efficiency and productivity.

In order to pave the way for HA in HPC, our research initially focuses on efficient redundancy strategies for head and service nodes of HPC systems. These nodes typically run critical HPC system services, like job and resource management. Head and service nodes represent single points of failure and control for an entire HPC system as they render it inaccessible and unmanageable in case of a failure until repair.

With this paper, we summarize our accomplishments in the area of high availability for HPC system services, such as developed concepts and implemented proof-of-concept prototypes. Furthermore, we describe existing limitations, such as performance issues, which need to be dealt with for production-type deployment. We also briefly discuss future research plans that try to address the most pressing issues.

## 2. Existing Solutions

Only a few solutions existed when we started our research. Even today, high availability solutions for HPC system services are rare.

PBS Pro for the Cray XT3 system [21] supports high availability using a hot standby redundancy strategy involving Crays proprietary interconnect for replication and transparent fail-over. Service state is replicated to the the standby node, which takes over based on the current state without loosing control of the system. This solution has a mean time to recover (MTTR) of practically 0. However, it is only available for the Cray XT3 and its availability is limited by the deployment of two redundant nodes.

The Simple Linux Utility for Resource Management (SLURM) [25, 37] as well as the metadata servers of the Parallel Virtual File System (PVFS) 2 [22, 23] and of the Lustre [16, 17] cluster file system employ an active/standby solution using a shared storage device, which is a common technique for providing service-level high availability using a heartbeat mechanism [12]. However, this technique has its pitfalls as service state is saved on a shared storage device upon modification and may be corrupted during fail-over. An extension of this technique uses a crosswise hot standby redundancy strategy in an asymmetric active/active fashion. In this case, both are active services and additional standby services for each other. In both cases, the MTTR depends on the heartbeat interval. As with most shared storage solutions, correctness and quality of service are not guaranteed due to the lack of commit protocols.

## 3. Accomplishments

The work we performed within the scope of high availability for head and service nodes of HPC systems consists of three parts:

- We investigated the overall background of HA technologies in the context of HPC, including a more detailed problem description, conceptual models, and a review of existing solutions.
- We developed different replication strategies for providing active/standby, asymmetric active/active, and symmetric active/active high availability for HPC system services.
- We implemented proof-of-concept prototypes based on the developed replication strategies, which offer high availability for critical HPC system services, like job and resource management.

In the following, we give a more detailed description of our major accomplishments.

### 3.1. High Availability in the HPC Context

Traditional HA computing deals with providing redundancy strategies for single services, such as a Web server. While our research initially focuses on efficient redundancy strategies for single services that run on head and service nodes, the entire HPC system architecture needs to be considered as components (nodes) are distributed and interdependent. Our work investigated the overall need and applicability of HA technologies in the context of HPC [6].

A HPC system typically employs a significant number of compute nodes that perform the actual parallel scientific computation, while a single head node and optional service nodes handle system management tasks, such as user login, resource management, job scheduling, data storage, and I/O. Large-scale HPC systems may be partitioned to improve scalability and to enable isolation in case of failures. In this case, several sets of partition compute nodes are supported by their respective partition service nodes, while a single head node is still controlling the entire system.

This Beowulf cluster system architecture [26, 27] has been proven to be very efficient as it permits customization of nodes and interconnects to their purpose. Many vendors adopted the Beowulf architecture either completely in the form of HPC clusters or in part by developing hybrid HPC solutions.

HPC systems run critical and non-critical system services on head, service, and compute nodes. A service is critical to its system if it cannot operate without it.

Any such service is a single point of failure and control for the entire system. A service is non-critical if the system can continue to operate in a degraded mode. Typical critical HPC system services are: user login, network file system, job and resource management, communication services, and in some cases the OS, *e.g.*, for single system image (SSI) systems. User management, software management, and programming environment are usually non-critical system services.

If a system has a head node running critical system services, this head node is a single point of failure and control for the entire system. A typical head node of a HPC system may run the following critical system services: user login, job and resource management, and network file system. It may also run the following non-critical services: user management, software management, and programming environment. Most HPC systems employ a head node, such as clusters, vector machines, massively parallel processing (MPP) systems, and SSI solutions. Examples are: Cray X1 [35], Cray XT3 [36], IBM Blue Gene/L [3], IBM MareNostrum [18], and SGI Altix [1].

If a system employs service nodes running critical system services, any such service node is a single point of failure and control for the entire system. If a system has service nodes running non-critical system services, any such service node is still a single point of failure for the entire system. Service nodes typically offload head node system services, *i.e.*, they may run the same critical and non-critical system services. Most of the advanced HPC systems currently in use employ service nodes, *e.g.*, Cray X1, Cray XT3, IBM Blue Gene/L, and IBM MareNostrum.

### 3.2. Replication Strategies for Services

We also focused on describing the principles, assumptions and techniques employed in providing high availability for services. Based on earlier work in refining a modern high availability taxonomy for generic computing systems [24, 34], we developed a high availability taxonomy for computing services and adopted it to the complexity of HPC system architectures [6].

High availability for computing services is invariably achieved through a replication mechanism appropriate to the service, a redundancy strategy. When a service fails, the redundant one replaces it. The degree of transparency in which this replacement occurs can lead to a wide variation of configurations. Warm and hot standby are active/standby configurations commonly used in high availability computing. Asymmetric and symmetric active/active configurations are commonly used in mission critical applications.

- Warm Standby requires some service state replication and an automatic fail-over. The service is interrupted and some state is lost. Service state is regularly replicated to the redundant service. In case of a failure, it replaces the failed one and continues to operate based on the previous replication. Only those state changes are lost that occurred between the last replication and the failure.
- Hot Standby requires full service state replication and an automatic fail-over. The service is interrupted, but no state is lost. Service state is replicated to the redundant service on any change, *i.e.*, it is always up-to-date. In case of a failure, it replaces the failed one and continues to operate based on the current state.
- Asymmetric active/active requires two or more active services that offer the same capabilities at tandem without coordination, while optional standby services may replace failing active services ( $n + 1$  and  $n + m$ ). Asymmetric active/active provides improved throughput performance, but it has limited use cases due to the missing coordination between active services.
- Symmetric active/active requires two or more active services that offer the same capabilities and maintain a common global service state using virtual synchrony. There is no interruption of service and no loss of state, since active services run in virtual synchrony without the need to fail-over.

These redundancy strategies are entirely based on the fail-stop model, which assumes that system components, such as individual services, nodes, and communication links, fail by simply stopping. They do not guarantee correctness if a failing system component violates this assumption by producing false output.

### 3.3. Implemented Proof-of-Concept Prototypes

We implemented several proof-of-concept prototypes based on the developed replication strategies. Taking into account the need and applicability of these HA technologies in the context of HPC, we initially focused on the batch job management system. Ongoing implementation efforts not described in this paper deal with parallel file system metadata.

**3.3.1. Active/Standby HA-OSCAR.** High Availability Open Source Cluster Application Resources (HA-OSCAR) [10, 11, 15] is a high availability framework for OpenPBS [20] and the PBS compliant TORQUE [30]. OpenPBS is the original version of the

Portable Batch System (PBS), a flexible batch queuing system developed for NASA in the early to mid-1990s. Its service interface has become a standard in HPC job and resource management.

HA-OSCAR supports high availability for Open PBS/TORQUE using a warm standby redundancy strategy involving a standby head node. Service state is replicated to the the standby upon modification, while the standby service takes over based on the current state. The standby node monitors the health of the active node using a heartbeat mechanism and initiates the fail-over. However, OpenPBS/TORQUE does temporarily loose control of the system in this case. All previously running jobs are automatically restarted.

The MTTR of HA-OSCAR depends on the heartbeat interval, the fail-over time, and the time currently running jobs need to recover to their previous state. HA-OSCAR integrates with the compute node checkpoint/restart layer for LAM/MPI [13], BLCR [2], improving its MTTR to 3-5 seconds for detection and fail-over plus the time to catch up based on the last application checkpoint.

**3.3.2. Asymmetric Active/Active HA-OSCAR.** As part of the HA-OSCAR research, an asymmetric active/active prototype implementation [14] has been developed that offers HPC job and resource management in a high-throughput computing scenario.

Two different job and resource management services, OpenPBS and the Sun Grid Engine (SGE) [28], run independently on different head nodes at the same time, while an additional head node is configured as a standby. Fail-over is performed using a heartbeat mechanism and is guaranteed for only one service at a time using a priority-based fail-over policy. Similar to the active/standby HA-OSCAR variant, OpenPBS and SGE do loose control of the system during fail-over, requiring a restart of currently running jobs controlled by the failed head node. Only one failure is completely masked at a time due to the  $2 + 1$  configuration. A second failure results in a degraded operating mode with one head node serving the entire system.

The MTTR of the asymmetric active/active HA-OSCAR solution depends on the heartbeat interval, the fail-over time, and the time currently running jobs need to recover to their previous state. However, the system is still operable during a fail-over due to the second active head node. Furthermore, only those jobs need to be restarted that were controlled by the failed head node.

**3.3.3. Symmetric Active/Active JOSHUA.** JOSHUA [32, 33] offers symmetric active/active high availability for HPC job and resource management services with

a PBS compliant service interface. It represents a virtually synchronous environment using external replication [9] based on the PBS service interface providing high availability without any interruption of service and without any loss of state.

Conceptually, the JOSHUA software architecture consists of three major parts: a server process (joshua) running on each head node, a set of control commands (jsub, jdel, and jstat) reflecting PBS compliant behavior to the user, and a set of scripts (jmutex and jdone) to perform a distributed mutual exclusion during job launch. Furthermore, JOSHUA relies on the Transis [5, 31] group communication system for reliable, totally ordered message delivery. The JOSHUA prototype is based on the PBS compliant TORQUE HPC job and resource management system that employs the TORQUE PBS server together with the Maui [19] scheduler on each active head node and a set of PBS mom servers on compute nodes.

The communication between the JOSHUA commands, the Transis group communication system, and the JOSHUA server introduces a latency and throughput overhead, which increases with the number of active head nodes. However, both are in an acceptable range as a 4-way active/active head node system still provides a job submission latency of 349 milliseconds and a throughput of 3 job submissions per second.

## 4. Limitations

Based on theoretical analysis and practical experiments, we identified several existing limitations, which need to be dealt with for a production-type deployment of our developed technologies:

- The developed active/hot-standby technology interrupts the service during a fail-over, resulting in a non-transparent masking of failures.
- The developed asymmetric active/active technology has shown a similar behavior in the  $2 + 1$  configuration. Generic  $n + 1$  or  $n + m$  configurations have not been developed yet, and the  $2 + 1$  configuration uses two different service implementations.
- The developed symmetric active/active technology has certain unnecessary stability issues due to the reliance on the Transis group communication system. Also, the used communication protocol has an inherent latency problem.
- All developed prototypes need a customized active/hot-standby, asymmetric active/active, or symmetric active/active environment, resulting in insufficient reuse of code.

- The interaction with compute node fault tolerance mechanisms is limited to checkpoint/restart in case of a non-transparent head node fail-over. There is no solution that deals with the failure of a compute node that communicates directly with the head node, *e.g.*, a PBS mom failure.

The most pressing issues for production-type deployment of our developed technologies are stability, performance, and interaction with compute node fault tolerance mechanisms. The most pressing issue for extending the developed technologies to other critical system services is portability and ease-of-use.

## 5. Conclusions

Considering our accomplishments and their limitations in providing efficient redundancy strategies for head and service nodes of HPC systems and our path towards high-level RAS for HPC systems, we conclude that the developed technologies have shown promising results. However, further improvements are needed for production-type deployment of our prototypes.

The need for performance improvement for the active/active solution has become apparent during the ongoing prototype implementation efforts not described in this paper that deal with parallel file system metadata, where low latency and high bandwidth is essential. The protocol used in the Transis group communication system has an inherent latency problem. Recent work by our group in eliminating this issue shows promising results. Future work needs to target the optimization of group communication protocols for the actual active/active replication use case, which involves only a small (2-4) number of nodes and requires only a specific set of group communication properties.

To increase the reuse of code and to simplify the adaptation of the developed high availability technologies to other existing services, a component-based high availability framework is needed that provides the necessary high availability mechanisms and easy-to-use interfaces based on the developed replication strategies. We already performed initial work in this area [7, 8] by developing a flexible, pluggable component framework that allows adaptation to system properties, like network technology and system scale; and application needs, such as programming model and consistency requirements. Future research needs to focus on further development of high availability programming models and on the implementation of respective interfaces.

Stability and performance improvement can certainly be achieved by rigorous debugging and testing of our solutions in testbed environments as well as in real

production HPC systems. Our strategy needs to initially target small-scale HPC systems with less critical impact, *i.e.*, we do not need to guarantee a specific quality of service. Once our solutions are mature enough and can guarantee a specific quality of service, large-scale production deployment may be performed.

The most intriguing and challenging issue we have to deal with is the combination of redundancy strategies for head, service, and compute nodes in form of a comprehensive RAS solution for HPC systems. All system components within a HPC system interact with and depend on each other. Interfaces are not always clear cut. For example, the batch job system runs on the head node and uses a separate process on compute nodes to actually start a parallel job. If this process fails, the batch job system loses control over the parallel job, which itself loses the communication path to report back to. Keeping an HPC system alive in the event of failures in such a manner that the capability is not severely degraded requires a resilient redundancy strategy for such a job start process on compute nodes.

To summarize this paper, we presented our accomplishments in the area of high availability for HPC system services. We briefly described a survey of existing solutions, the overall need and applicability of HA technologies in the context of HPC, a high availability taxonomy for computing services adopted to the complexity of HPC system architectures, and our proof-of-concept prototype implementations for the the batch job management system. We also described existing limitations, which need to be dealt with for production-type deployment. The most pressing issues we identified were stability, performance, portability, and interaction with compute node fault tolerance mechanisms.

Our future work will focus on enhancing the developed prototypes, deploying production-type solutions, and combining our HA solutions with compute node fault tolerance mechanisms.

## References

- [1] Altix Computing Platform at SGI, Mountain View, CA, USA. Available at <http://www.sgi.com/products/servers/altix>.
- [2] Berkeley Lab Checkpoint/Restart (BLCR) project at Lawrence Berkeley National Laboratory, Berkeley, CA, USA. Available at <http://ftg.lbl.gov/checkpoint>.
- [3] Blue Gene/L Computing Platform at IBM Research. Available at <http://www.research.ibm.com/bluegene>.
- [4] Community Climate System Model (CCSM) at the National Center for Atmospheric Research, Boulder, CO, USA. Available at <http://www.cesm.ucar.edu>.
- [5] D. Dolev and D. Malki. The Transis approach to high availability cluster communication. *Communications of*

- the ACM*, 39(4):64–70, 1996.
- [6] C. Engelmann and S. L. Scott. Concepts for high availability in scientific high-end computing. In *Proceedings of High Availability and Performance Workshop (HAPCW) 2005*, Santa Fe, NM, USA, Oct. 11, 2005.
  - [7] C. Engelmann and S. L. Scott. High availability for ultra-scale high-end scientific computing. In *Proceedings of 2<sup>nd</sup> International Workshop on Operating Systems, Programming Environments and Management Tools for High-Performance Computing on Clusters (COSET-2) 2005, in conjunction with 19<sup>th</sup> ACM International Conference on Supercomputing (ICS) 2005*, Cambridge, MA, USA, June 19, 2005.
  - [8] C. Engelmann, S. L. Scott, D. E. Bernholdt, N. R. Gottumukkala, C. Leangsuksun, J. Varma, C. Wang, F. Mueller, A. G. Shet, and P. Sadayappan. MOLAR: Adaptive runtime support for high-end computing operating and runtime systems. *ACM SIGOPS Operating Systems Review (OSR)*, 40(2):63–72, 2006.
  - [9] C. Engelmann, S. L. Scott, C. Leangsuksun, and X. He. Active/active replication for highly available HPC system services. In *Proceedings of International Symposium on Frontiers in Availability, Reliability and Security (FARES) 2006, in conjunction with 1<sup>st</sup> International Conference on Availability, Reliability and Security (ARES) 2006*, pages 639–645, Vienna, Austria, Apr. 20–22, 2006.
  - [10] HA-OSCAR at Louisiana Tech University, Ruston, LA, USA. Available at <http://xcr.cenit.latech.edu/ha-oscar>.
  - [11] I. Haddad, C. Leangsuksun, and S. L. Scott. HA-OSCAR: Towards highly available linux clusters. *Linux World Magazine*, Mar. 2004.
  - [12] Heartbeat. Available at <http://www.linux-ha.org/HeartbeatProgram>.
  - [13] LAM-MPI Project at Indiana University, Bloomington, IN, USA. Available at <http://www.lam-mpi.org>.
  - [14] C. Leangsuksun, V. K. Munganuru, T. Liu, S. L. Scott, and C. Engelmann. Asymmetric active-active high availability for high-end computing. In *Proceedings of 2<sup>nd</sup> International Workshop on Operating Systems, Programming Environments and Management Tools for High-Performance Computing on Clusters (COSET-2) 2005*, Cambridge, MA, USA, June 19, 2005.
  - [15] K. Limaye, C. Leangsuksun, Z. Greenwood, S. L. Scott, C. Engelmann, R. Libby, and K. Chanchio. Job-site level fault tolerance for cluster and grid environments. In *Proceedings of IEEE International Conference on Cluster Computing (Cluster) 2005*, Boston, MA, USA, Sept. 26–30, 2005.
  - [16] Lustre at Cluster File Systems, Inc., Boulder, CO, USA. Available at <http://www.lustre.org>.
  - [17] Lustre Architecture Whitepaper at Cluster File Systems, Inc., Boulder, CO, USA. Available at <http://www.lustre.org/docs/whitepaper.pdf>.
  - [18] MareNostrum eServer Computing Platform at IBM. Available at <http://www.ibm.com/servers/eserver/linux/power/marenostrum>.
  - [19] Maui Cluster Scheduler at Cluster Resources, Inc., Spanish Fork, UT, USA. Available at <http://www.clusterresources.com/maui>.
  - [20] OpenPBS at Altair Engineering, Troy, MI, USA. Available at <http://www.openpbs.org>.
  - [21] PBS Pro for the Cray XT3 Computing Platform at Altair Engineering, Troy, MI, USA. Available at [http://www.altair.com/pdf/PBSPro\\_Cray.pdf](http://www.altair.com/pdf/PBSPro_Cray.pdf).
  - [22] Parallel Virtual File System (PVFS). Available at <http://www.pvfs.org/pvfs2>.
  - [23] PVFS2 Development Team. PVFS2 High-Availability Clustering. Available at <http://www.pvfs.org/pvfs2> as part of the PVFS2 source distribution.
  - [24] R. I. Resnick. A modern taxonomy of high availability, 1996. Available at <http://www.generalconcepts.com/resources/reliability/resnick/HA.htm>.
  - [25] SLURM at Lawrence Livermore National Laboratory, Livermore, CA, USA. Available at <http://www.llnl.gov/linux/slurm>.
  - [26] T. Sterling. *Beowulf cluster computing with Linux*. MIT Press, Cambridge, MA, USA, 2002.
  - [27] T. Sterling, J. Salmon, D. J. Becker, and D. F. Savarese. *How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters*. MIT Press, Cambridge, MA, USA, 1999.
  - [28] Sun Grid Engine (SGE) at Sun Microsystems, Inc, Santa Clara, CA, USA. Available <http://gridengine.sunsource.net>.
  - [29] Terascale Supernova Initiative at Oak Ridge National Laboratory, Oak Ridge, TN, USA. Available at <http://www.phy.ornl.gov/tsi>.
  - [30] TORQUE Resource Manager at Cluster Resources, Inc., Spanish Fork, UT, USA. Available at <http://www.clusterresources.com/torque>.
  - [31] Transis Project at Hebrew University of Jerusalem, Israel. Available at <http://www.cs.huji.ac.il/labs/transis>.
  - [32] K. Uhlemann. High availability for high-end scientific computing. Master’s thesis, Department of Computer Science, University of Reading, UK, Mar. 2006.
  - [33] K. Uhlemann, C. Engelmann, and S. L. Scott. JOSHUA: Symmetric active/active replication for highly available HPC job and resource management. In *Proceedings of IEEE International Conference on Cluster Computing (Cluster) 2006*, Barcelona, Spain, Sept. 25–28, 2006.
  - [34] E. Vargas. High availability fundamentals. *Sun Blueprints series*, Nov. 2000.
  - [35] X1 Computing Platform at Cray, Seattle, WA, USA. Available at <http://www.cray.com/products/x1>.
  - [36] XT3 Computing Platform at Cray, Seattle, WA, USA. Available at <http://www.cray.com/products/xt3>.
  - [37] A. Yoo, M. Jette, and M. Grondona. SLURM: Simple linux utility for resource management. In *Lecture Notes in Computer Science: Proceedings of Job Scheduling Strategies for Parallel Processing (JSSPP) 2003*, volume 2862, pages 44–60, Seattle, WA, USA, June 24, 2003.