Symmetric Active/Active High Availability for High-Performance Computing System Services: Accomplishments and Limitations*

C. Engelmann[1,2], S. L. Scott[1], C. Leangsuksun[3], X. He[4]
[1]*Computer Science and Mathematics Division, Oak Ridge National Laboratory, USA*
[2]*Department of Computer Science, The University of Reading, UK*
[3]*Computer Science Department, Louisiana Tech University, USA*
[4]*Department of Electrical and Computer Engineering, Tennessee Technological University, USA*
engelmannc@ornl.gov, scottsl@ornl.gov, box@latech.edu, hexb@tntech.edu

Abstract

This paper summarizes our efforts over the last 3-4 years in providing symmetric active/active high availability for high-performance computing (HPC) system services. This work paves the way for high-level reliability, availability and serviceability in extreme-scale HPC systems by focusing on the most critical components, head and service nodes, and by reinforcing them with appropriate high availability solutions. This paper presents our accomplishments in the form of concepts and respective prototypes, discusses existing limitations, outlines possible future work, and describes the relevance of this research to other, planned efforts.

1. Introduction

During the last decade, high-performance computing (HPC) has become an important tool for scientists world-wide to understand problems, such as in climate dynamics, nuclear astrophysics, and nanotechnology. Every year, new larger scale HPC systems emerge with better performance capabilities. The performance increase is not only aided by advances in network and processor design, but also by employing more and more processors within closely-coupled systems.

The growth in system scale poses a substantial challenge for system software and scientific applications with respect to reliability, availability and serviceability (RAS). Although the mean-time to failure (MTTF) for each individual component, *e.g.*, processor and memory module, may be above consumer product standard, the combined probability of failure for a system scales proportionally with the number of its components. The enormous quantity of components results in a much lower MTTF for the overall system, causing more frequent system-wide interruptions than displayed by previous, smaller scale systems.

In contrast to the loss of HPC system availability, the demand for continuous availability has risen dramatically with the recent trend toward capability computing, which drives the race for scientific discovery by running applications on the fastest machines available while desiring significant amounts of time (weeks and months) without interruption.

Both, the telecommunication and the general information technology (IT) communities, have dealt with these issues for their particular solutions using traditional high availability concepts, such as active/standby. It is time for the HPC community to follow the IT and telecommunication industry lead and provide high-level RAS for extreme-scale HPC systems.

This work paves the way by focusing on the most critical components and by reinforcing them with appropriate high availability solutions. Head and service nodes are the "Achilles heel" of a HPC system. A failure typically results in a system-wide outage until repair. This research targets efficient redundancy for such service components in extreme-scale HPC systems.

When this effort started in 2004, there were only a small number of high availability solutions for HPC system head and service nodes. Most of them focused on an active/standby redundancy strategy with some or full service state replication. Other research in provid-

*This work was sponsored by the Office of Advanced Scientific Computing Research; U.S. Department of Energy. It was performed at Oak Ridge National Laboratory (ORNL), which is managed by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725. It was also performed at Louisiana Tech University under U.S. Department of Energy Grant No. DE-FG02-05ER25659. The work at Tennessee Tech University was sponsored by the Laboratory Directed Research and Development Program of ORNL and by the U.S. National Science Foundation under Grant Nos. CNS-0617528 and CNS-0720617.

ing high availability in distributed systems using state-machine replication, virtual synchrony, and group communication has been performed in the 1990s.

This work combines both efforts. In addition to practical proof-of-concept solutions, this work offers a theoretical foundation for head and service node high availability as part of the greater effort in defining a HPC RAS taxonomy. Within this context, we:

- examined relevant past and ongoing efforts in providing high availability for:
  - IT and telecommunication services,
  - HPC head, service, and compute nodes, and
  - distributed systems;
- provided a high availability taxonomy suitable for HPC head and service nodes;
- generalized HPC system architectures and identified availability deficiencies;
- defined various methods for providing high availability for HPC head and service nodes;
- developed proof-of-concept prototypes for HPC head and service node high availability using:
  - internal symmetric active/active replication
  - external symmetric active/active replication
- developed a prototype framework for:
  - transparent symmetric active/active replication in client-service scenarios, and
  - transparent symmetric active/active with dependent services.

As this research project comes to an end, this paper summarizes our efforts over the last 3-4 years, presents our accomplishments in the form of concepts and respective prototypes, discusses existing limitations, outlines possible future work in this area, and describes the relevance of this research to other, planned efforts.

2. Previous Work

The concept of using *shared storage* for saving service state is a common technique for providing high availability, but it has its pitfalls. State is saved on the shared storage upon modification, while the standby takes over in case of a failure of the active service. The standby monitors the health of the active service using a heartbeat mechanism [24]. An extension of this technique uses a crosswise active/standby redundancy strategy, where both are active services and standbys for each other. In both cases, the mean-time to recover (MTTR) depends on the heartbeat interval.

While the shared storage device is typically an expensive redundant array of independent drives (RAID), it remains a single point of failure. Furthermore, file system corruption due to failures occurring during write operations are not masked unless a journaling file system is used and an incomplete backup is discarded. This requirement impacts the fail-over procedure by adding a file system check, which in-turn extends the MTTR.

Active/standby solutions using shared storage exist for the following HPC system services:

- Simple Linux Utility for Resource Management (SLURM) [30] job and resource management service,
- Parallel Virtual File System (PVFS) [22] metadata service (MDS), and
- Lustre cluster file system [4] MDS.

Service-level *active/standby* high availability solutions typically perform state change validation to maintain consistency of backup state. The primary service copies its state to the standby service in regular intervals or on any change. A two-phase commit protocol ensures consistency between active and standby nodes and provides transparent fail-over, *i.e.*, no state is lost and only an interruption of service may be noticed.

Service-level active/standby solutions exist for the following HPC system services:

- High Availability Open Source Cluster Application Resources (HA-OSCAR) [18] for the OpenPBS [1] job and resource management service, and
- Moab Workload Manager [5] job and resource management service.

High availability clustering commonly describes the concept of using a group of nodes to provide the same single service using load balancing for uninterrupted processing of incoming requests. Active/standby replication for all nodes together, for a subset, or for each node individually may be used to provide continued processing of existing requests. This leads to a variety of configurations, such as $n+1$ and $n+m$ with n active nodes and 1 or m standby nodes. High availability clustering targets high-throughput processing of a large number of small service requests with no or minimal service state change, such as a Web service.

High availability clustering solutions exist for the following HPC system services:

- Asymmetric Active/Active HA-OSCAR [17] for the OpenPBS [1] and Sun Grid Engine [27] job and resource management services.

State-machine replication [26] is a common technique for fault tolerance in distributed systems. Assuming that a service is a deterministic finite state machine,

consistent replication may be achieved by guaranteeing the same initial states and a linear history of state changes. All replicas perform the same state changes and produce the same output.

State-machine replication involves *process group communication*, which deals with reliable delivery and consensus issues, such as *liveness* and *consistency*. There is a plethora of previous work [7] focusing on semantics, correctness, efficiency, adaptability, and programming models.

Total order broadcasting [7] not only reliably delivers all messages within a process group, but also in the same order to all its members, which is essential for state-machine replication. Three approaches are widely used to implement total ordering: *sequencer*, *privilege-based*, and *communication history* algorithms.

In sequencer algorithms, one group member is responsible for ordering and reliably broadcasting messages on behalf of all other members. Isis [3] utilizes a sequencer algorithm.

Privilege-based algorithms rely on the idea that group members can broadcast messages only when they are granted the privilege to do so, for example using a rotating token as in the Totem protocol [2].

In communication history algorithms, messages can be broadcast by any group member at any time, without prior enforced order, and total order is ensured by delaying delivery until enough information of communication history has been gathered from other group members. Transis [8] is an example.

Sequencer and privilege-based algorithms provide good performance when a system is relatively idle. However, latency is limited by the time to rotate the token or produce total order via a sequencer. Communication history algorithms have a post-transmission delay that is most apparent when the system is relatively idle, since less communication history is produced and a response from all group members is needed.

The *virtual synchrony* paradigm was first established in the early work on Isis [3]. It defines the relation between regular and control messages in a group. Whenever group membership changes, all remaining members observe a membership change event. Conceptually, virtual synchrony guarantees that membership changes within a group are observed in total order as well. *Extended virtual synchrony* [19] additionally supports crash recoveries and network partitions. It has been implemented in Transis [8].

Recent work focused on practical solutions for *Byzantine fault tolerance*, where arbitrary errors are handled as well. These approaches go beyond the *fail-stop model*, which assumes that components, such as services, nodes, or links, fail by simply stopping. Since more extensive failure detection mechanisms are deployed to verify correct process behavior, Byzantine fault tolerance mechanisms incur a higher performance overhead during normal operation.

Byzantine Fault Tolerance with Abstract Specification Encapsulation (BASE) [25] is a framework for state-machine replication with Byzantine fault tolerance. Using BASE, each replica can be repaired periodically using an abstract view of the state stored by correct replicas. Furthermore, each replica can run distinct or nondeterministic service implementations, which reduces the probability of common mode failures.

3. Taxonomy, Architecture, and Methods

As part of this effort, earlier work in refining a modern high availability taxonomy [23, 29] has been extended [10, 13] with definitions for *asymmetric* and *symmetric active/active* replication to capture high availability clustering and state-machine replication. While the term symmetric active/active captures state-machine replication, asymmetric active/active defines high availability clustering. This distinction was necessary as some used active/active to describe state-machine replication, while the telecommunication industry used it for high availability clustering. The original definition for active/standby has been extended as well to distinguish more clearly between the various active/standby configurations. The terms passive and active replication were omitted for more clarity.

Further work [10, 9], investigated and generalized HPC system architectures, and identified availability deficiencies, such as single points of failure and control. A failure at a single point of failure interrupts the entire system, while a failure at a single point of control additionally renders it useless until repair. HPC head and service nodes are typical single points of failure and control, since they run critical system services, such as the job and resource management service or the MDS of the parallel file system.

Based on the extended high availability taxonomy, we also defined a conceptual service model, described various service-level high availability methods and their properties, compared them with each other with regards to performance overhead and provided availability, and examined similarities and differences in their programming interfaces [14]. While active/warm-standby roughly provides the lowest runtime overhead in a failure free environment and the highest recovery impact in case of a failure, symmetric active/active roughly provides the lowest recovery impact and the highest runtime overhead.

To determine if the runtime overhead is acceptable

in an environment that demands high performance, several symmetric active/active prototypes were developed. Furthermore, a high availability framework prototype was developed to provide replication with a minimum amount of modification of existing code and in complex scenarios with dependent services.

4. External Replication for the HPC Job and Resource Management Service

JOSHUA [13, 28] is a generic approach for symmetric active/active high availability of HPC job and resource management services with a PBS compliant interface. The proof-of-concept prototype is based on PBS TORQUE [6]. Transis [8] is used for total ordering of incoming messages and for assuring that a job gets started only once. Consistently produced output is delivered from every active head node to PBS clients, which filter duplicated messages. JOSHUA performs a distributed mutual exclusion using Transis to assure at-most-once job launch.

Transis automatically excludes failed head nodes from any further communication, while PBS mom services on compute nodes simply ignore them when sending job statistics reports. PBS clients are not affected by a head node failure, since Transis delivers all messages even if a client needs to reconnect to the service group via a different service group member. The distributed mutual exclusion relies on Transis membership change events for releasing any locks.

With JOSHUA, the first fully functional solution for providing symmetric active/active high availability for a HPC system service was developed using the external replication approach [12] that wraps an existing service into a virtually synchronous environment. The prototype performed correctly and offered an acceptable response latency and throughput performance.

The reliance of the PBS service on the PBS mom service on the compute nodes revealed the existence of more complex interdependencies between individual system services on head, service and compute nodes. Although the response latency overhead of 251ms on four symmetric active/active head nodes is in an acceptable range for HPC job and resource management, this may not be true for more latency sensitive services, such as the MDS of a parallel file system [22, 4].

5. Internal Replication for the Parallel File System Metadata Service

The developed follow-on proof-of-concept prototype [20] is a customized implementation for the MDS of PVFS [22]. An improved Transis with a fast delivery protocol [21] is used for total ordering. Consistently produced MDS output is delivered by the service a client is directly connected to. The last produced output is temporarily cached for each client at all services to allow seamless connection fail-over. A non-partitioning network is assumed due to the close coupling of active service nodes.

Transis automatically excludes failed service nodes from any further communication. Clients that are directly connected a failed service node initiate a connection fail-over. After reconnecting to an active service node, the last output message is resent and may be ignored by a client if duplicated.

With the symmetric active/active PVFS MDS, the first fully functional solution was developed that uses the internal replication approach [12], which modifies an existing service to inferface it with a group communication system for virtual synchrony. The prototype performed correctly and offered a remarkable performance with only 26ms latency overhead for MDS writes and 300% of PVFS MDS baseline throughput for MDS reads in a 4 service node system.

The performance results are encouraging for widely deploying symmetric active/active replication infrastructures in HPC systems to re-enforce critical system services with high-performance redundancy. The developed fast delivery protocol enhancement was instrumental to this success. The performance improvement may also help to reduce the response latency overhead of JOSHUA. While the internal replication approach provides very good performance, it requires modification of existing services. The PVFS MDS is small and easy-to-modify. This may not be true with other services, such as the MDS of Lustre [4].

6. Transparent Replication for Services

In the symmetric active/active replication software architecture, a service-side interceptor or adaptor component deals with receiving incoming messages and routing them through the group communication system for total order and reliable delivery, and with sending produced output messages back to the client.

The transparent symmetric active/active replication software framework [15] is a follow-on prototype that accommodates both replication methods, external and internal, by using a virtual communication layer (VCL) abstraction for group communication and client-service connection fail-over semantics. The original replication methods are refined to additionally utilize client-side interceptors or adaptors to provide total transparency, *i.e.*, transparent client-service connection fail-over in

addition to the already transparent virtual synchrony of services. Adaptation to clients and services is only needed with regards to the used communication protocols. Clients and services are unaware of the replication infrastructure as it provides all necessary mechanisms internally via the VCL. While there are some similarities to Virtual Private Networking (VPN), the VCL only hides the replication mechanisms from clients and services using client- and service-side proxies. A VCL instance is inherently tied to a specific service it provides replication mechanisms for.

The developed framework is able to transparently provide service-level high availability using the internal or external symmetric active/active replication approach. The transparency provided by the VCL also hides any communication across administrative domains, *i.e.*, communication appears to be local. This has two consequences. First, client and server still need to perform any necessary authentication and authorization using the interceptors or adaptors as virtual protocol routers. Second, the VCL itself may need to perform similar mechanisms to assure its own integrity across administrative domains.

7. Replication of Dependent Services

While client-service scenarios are quite common, dependencies between critical HPC system services exist. Lustre [4], for example, employs a MDS as well as object storage services that communicate with each other in a service-to-service fashion incompatible with the replication architecture. With a follow-on prototype [16], this limitation, the inability to deal with dependent services, has been resolved by extending the framework using its already existing mechanisms and features to allow services to be clients of other services, and services to be clients of each other.

By using a high-level abstraction, dependencies between clients and services, and decompositions of service-to-service dependencies into respective orthogonal client-service dependencies can be mapped onto an infrastructure consisting of multiple symmetric active/active replication subsystems. Each subsystem utilizes the VCL to hide the replication infrastructure for a specific service group as much as possible.

The developed prototype is able to transparently provide high availability for dependent HPC system services, while avoiding an unnecessary increase in framework size and complexity.

8. Conclusions and Future Work

Our research effort over the last 3-4 years produced a number of concepts and prototypes. The most important accomplishments are the extended high availability taxonomy, a theoretical foundation for service-level high availability, and the symmetric active/active PVFS prototype, a fully functional solution that offers high availability as well as high performance.

While our work focused on defining and exploring symmetric active/active high availability in high-performance environments, it did not address certain production-type deployment issues. For example, all developed prototypes rely on the Internet protocol (IP). However, most HPC systems employ specialized networks. A modular communication framework, as proposed earlier [11], is needed to adapt to system properties, such as network type, and service needs, such as efficient group communication.

Since our work focused on high-performance environments, Byzantine fault tolerance was not targeted. However, certain practical Byzantine fault tolerance mechanisms may be added in the future, such as comparing service output to catch soft errors.

Our work provides a foundation for building highly available high-performance services in HPC systems, distributed systems, peer-to-peer networks, and many other service-oriented or service-based infrastructures. Our research shows that state-machine replication using total order broadcasting provides high availability as well as high performance. Implementations are correct and offer better availability than active/standby or asymmetric active/active solutions.

References

[1] Altair Engineering, Troy, MI, USA. OpenPBS, 2007. http://www.openpbs.org.

[2] Y. Amir, L. E. Moser, P. M. Melliar-Smith, D. A. Agarwal, and P. W. Ciarfella. The Totem single-ring ordering and membership protocol. *ACM Transactions on Computer Systems*, 13(4):311–342, 1995.

[3] K. P. Birman and R. van Renesse. *Reliable Distributed Computing with the Isis Toolkit*. IEEE Computer Society, Apr. 1994.

[4] Cluster File Systems, Inc., Boulder, CO, USA. Lustre Cluster File System Architecture, 2007. http://www.lustre.org/docs/whitepaper.pdf.

[5] Cluster Resources, Inc, Salt Lake City, UT, USA. Moab Workload Manager Administrator‘s Guide, 2007. http://www.clusterresources.com/products/mwm/docs.

[6] Cluster Resources, Inc, Salt Lake City, UT, USA. TORQUE Resource Manager, 2007. http://www.clusterresources.com/torque.

[7] X. Défago, A. Schiper, and P. Urbán. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Computing Surveys*, 36(4):372–421, 2004.

[8] D. Dolev and D. Malki. The Transis approach to high availability cluster communication. *Communications of the ACM*, 39(4):64–70, 1996.

[9] C. Engelmann, H. H. Ong, and S. L. Scott. Middleware in modern high performance computing system architectures. In *Lecture Notes in Computer Science: Proceedings of the 7th International Conference on Computational Science, Part II*, volume 4488, pages 784–791, Beijing, China, May 27-30, 2007.

[10] C. Engelmann and S. L. Scott. Concepts for high availability in scientific high-end computing. In *Proceedings of the High Availability and Performance Workshop*, Santa Fe, NM, USA, Oct. 11, 2005.

[11] C. Engelmann and S. L. Scott. High availability for ultra-scale high-end scientific computing. In *Proceedings of the 2nd International Workshop on Operating Systems, Programming Environments and Management Tools for High-Performance Computing on Clusters*, Cambridge, MA, USA, June 19, 2005.

[12] C. Engelmann, S. L. Scott, C. Leangsuksun, and X. He. Active/active replication for highly available HPC system services. In *Proceedings of the 1st International Conference on Availability, Reliability and Security*, pages 639–645, Vienna, Austria, Apr. 20-22, 2006.

[13] C. Engelmann, S. L. Scott, C. Leangsuksun, and X. He. Symmetric active/active high availability for high-performance computing system services. *Journal of Computers*, 1(8):43–54, 2006.

[14] C. Engelmann, S. L. Scott, C. Leangsuksun, and X. He. On programming models for service-level high availability. In *Proceedings of the 2nd International Conference on Availability, Reliability and Security*, pages 999–1006, Vienna, Austria, Apr. 10-13, 2007.

[15] C. Engelmann, S. L. Scott, C. Leangsuksun, and X. He. Transparent symmetric active/active replication for service-level high availability. In *Proceedings of the 7th IEEE International Symposium on Cluster Computing and the Grid*, pages 755–760, Rio de Janeiro, Brazil, May 14-17, 2007.

[16] C. Engelmann, S. L. Scott, C. Leangsuksun, and X. He. Symmetric active/active replication for dependent services. In *Proceedings of the 3rd International Conference on Availability, Reliability and Security*, Barcelona, Spain, Mar. 4-7, 2008. To appear.

[17] 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 the 2nd International Workshop on Operating Systems, Programming Environments and Management Tools for High-Performance Computing on Clusters*, Cambridge, MA, USA, June 19, 2005.

[18] K. Limaye, C. Leangsuksun, Z. Greenwood, S. L. Scott, C. Engelmann, R. M. Libby, and K. Chanchio. Job-site level fault tolerance for cluster and grid environments. In *Proceedings of the 7th IEEE International Conference on Cluster Computing*, pages 1–9, Boston, MA, USA, Sept. 26-30, 2005.

[19] L. E. Moser, Y. Amir, P. M. Melliar-Smith, and D. A. Agarwal. Extended virtual synchrony. In *Proceedings of the 14th IEEE International Conference on Distributed Computing Systems*, pages 56–65, Poznan, Poland, June 21-24, 1994.

[20] L. Ou, C. Engelmann, X. He, X. Chen, and S. L. Scott. Symmetric active/active metadata service for highly available cluster storage systems. In *Proceedings of the 19th IASTED International Conference on Parallel and Distributed Computing and Systems*, Cambridge, MA, USA, Nov. 19-21, 2007.

[21] L. Ou, X. He, C. Engelmann, and S. L. Scott. A fast delivery protocol for total order broadcasting. In *Proceedings of the 16th IEEE International Conference on Computer Communications and Networks*, Honolulu, HI, USA, Aug. 13-16, 2007.

[22] Philip H. Carns. PVFS2 High-Availability Clustering using Heartbeat 2.0, 2007. http://www.pvfs.org/doc/pvfs2-ha-heartbeat-v2.pdf.

[23] R. I. Resnick. A modern taxonomy of high availability, 1996. http://www.verber.com/mark/cs/systems/A\%20Modern\%20Taxonomy\%20of\%20High\%20Availability.htm.

[24] A. L. Robertson. The evolution of the Linux-HA project. In *Proceedings of the UKUUG LISA/Winter Conference - High-Availability and Reliability*, Bournemouth, UK, June 21, 2004.

[25] R. Rodrigues, M. Castro, and B. Liskov. BASE: Using abstraction to improve fault tolerance. *ACM SIGOPS Operating Systems Review*, 35(5):15–28, 2001.

[26] F. B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, 1990.

[27] Sun Microsystems, Inc, Santa Clara, CA, USA. Sun Grid Engine Documentation, 2007. http://gridengine.sunsource.net.

[28] K. Uhlemann, C. Engelmann, and S. L. Scott. JOSHUA: Symmetric active/active replication for highly available HPC job and resource management. In *Proceedings of the 8th IEEE International Conference on Cluster Computing*, Barcelona, Spain, Sept. 25-28, 2006.

[29] E. Vargas. High availability fundamentals. *Sun Blueprints*, Nov. 2000.

[30] A. B. Yoo, M. A. Jette, and M. Grondona. SLURM: Simple linux utility for resource management. In *Lecture Notes in Computer Science: Proceedings of the 9th International Workshop on Job Scheduling Strategies for Parallel Processing*, volume 2862, pages 44–60, Seattle, WA, USA, June 24, 2003.