Symmetric Active/Active Replication for Dependent Services[*]

C. Engelmann[1,2], S. L. Scott[1], C. Leangsuksun[3], and 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

During the last several years, we have established the symmetric active/active replication model for service-level high availability and implemented several proof-of-concept prototypes. One major deficiency of our model is its inability to deal with dependent services, since its original architecture is based on the client-service model. This paper extends our model to dependent services using its already existing mechanisms and features. The presented concept is based on the idea that a service may also be a client of another service, and multiple services may be clients of each other. A high-level abstraction is used to illustrate dependencies between clients and services, and to decompose dependencies between services into respective client-service dependencies. This abstraction may be used for providing high availability in distributed computing systems with complex service-oriented architectures.

1. Introduction

During the last several years, our teams at Oak Ridge National Laboratory, Louisiana Tech University, and Tennessee Technological University have established the symmetric active/active replication model for service-level high availability and implemented several proof-of-concept prototypes [9, 10, 11, 12, 13, 14, 15, 16, 28, 29, 30, 33]. The overall goal was to provide high availability for critical system services in closely-coupled extreme-scale high-performance computing (HPC) systems [5, 19, 21, 20, 26], such as for the metadata service of a parallel file system [29] or for the parallel job management service [33]. Such services are typically the "Achilles heel" of a HPC system as they represent single points of failure and control, rendering the entire system useless for the time of repair.

The symmetric active/active replication model [13] allows to provide high availability for any type of client-service scenario using the well known state-machine replication concept [22, 32] using a group communication system [2, 6] for totally ordered and reliably delivered messages in a virtual synchronous service group [27]. Our past work has shown that implementations are correct, can be quite efficient, and have a practical value in the field [29, 33].

With this paper, we address one important limitation of our symmetric active/active replication model. While client-service scenarios are quite common, the parallel and distributed computing world is moving toward more complex service-oriented architectures (SOAs) [17]. Today, dependent services cannot only be found in distributed computing scenarios, but also within closely-coupled parallel HPC systems [8].

The Lustre cluster file system [3, 4], for example, employs a metadata service (MDS) as well as object storage services (OSSs). While file system drivers on compute nodes communicate in a client-service fashion with these services, MDS and OSSs communicate with each other in a service-to-service fashion incompatible with the current client-service architecture of our symmetric active/active replication model.

In fact, our previous proof-of-concept prototype for the symmetric active/active parallel job management solution, JOSHUA [33], showed exactly this deficiency, as the job management service on the head node of a HPC system and the parallel job start and process mon-

[*]This research was sponsored by the Office of Advanced Scientific Computing Research; U.S. Department of Energy. The work was performed in part at Oak Ridge National Laboratory (ORNL), which is managed by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725. The research performed at Tennessee Tech University was partially supported by the U.S. National Science Foundation under Grant Nos. CNS-0617528 and CNS-0720617.

itoring service on the compute nodes depend on each other to function correctly.

Up until now, our symmetric active/active replication model did not clearly address dependent services. This paper extends the symmetric active/active replication model to dependent services using its already existing mechanisms and features. The presented concept is based on the idea that a service may also be a client of another service, and multiple services may be clients of each other.

This paper is organized as follows. First, the existing transparent symmetric active/active replication model for client-service scenarios is briefly explained including its two different implementation methods, internal and external replication. Second, the transparent symmetric active/active replication model for client-service scenarios is generalized and applied to scenarios with dependent services. Third, implementation details are given for scenarios with dependent services as well as some initial test results. Fourth, related work is examined. Lastly, the presented work is briefly summarized and future work is outlined.

2. Client-Service Scenarios

Our existing transparent symmetric active/active replication model for client-service scenarios [16], is based on the concept of state-machine replication [22, 32] by guaranteeing the same initial states and a linear history of state transitions for all service replicas, *i.e.*, virtual synchrony [27]. This requires total order and reliable delivery of all incoming request messages, typically provided by a process group communication system [2, 6] that also assures correct group membership. Output messages produced by service replicas is unified either by simply ignoring duplicated messages or by using the group communication system for a distributed mutual exclusion. Based on our observations in the application area, we assume that services are deterministic and fail by simply stopping.

In order to hide the complexity of the underlying communication layer and consistency mechanisms, and to improve portability, our transparent symmetric active/active replication model separates the replication infrastructure from client and service implementations. This separation can be performed either internally by modifying client and service implementation itself or externally by wrapping them into a transparent virtually synchronous environment [16].

Internal replication (Figure 1) utilizes adaptor modules within clients and services in order to allow for better adaptation resulting in improved performance. This model allows each active service to accept request messages individually, while using the group communication system internally for all state changes. Client- and service-side adapters form a virtual communication layer (VCL), such that client-service connection fail-overs and group communication services are transparently performed by the adapters.

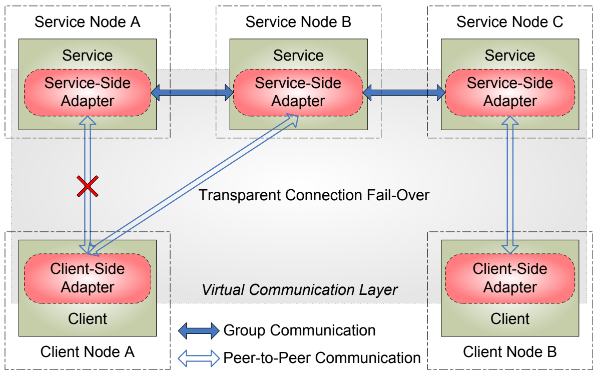

Figure 1. Semi-Transparent Internal Symmetric Active/Active Replication Implementation for Client/Service Scenarios

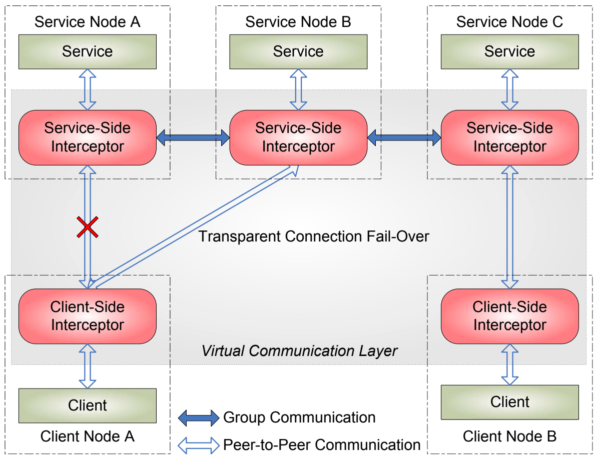

Figure 2. Transparent External Symmetric Active/Active Replication Implementation for Client/Service Scenarios

In contrast, external replication (Figure 2) utilizes separate interceptor processes in order to allow for better portability resulting in improved ease-of-use. In this model, interceptor processes imitate client and service behavior, such that clients operate as they communicate to a local service and services act as they communicate to a local client. The entire replication infrastructure is hidden in the VCL by the interceptor processes, where client-service connection fail-overs and group communication services are transparently performed.

The trade-off between both models is performance vs. portability and ease of use. The utilization of internal adapters requires modification of existing code, which may result in a rather extensive implementation effort if source code is available at all. However, internal adapters allow direct adaptation for improved performance, *e.g.*, to the locking and communication characteristics of the Parallel Virtual File System (PVFS) metadata service (MDS) [29].

The introduction of external interceptor processes in the communication path clearly degrades performance, while it allows to provide replication for legacy or complex services, such as the parallel job and resource management service TORQUE [33], without the need to touch client or service implementations.

As previously mentioned, the existing transparent symmetric active/active replication model addresses client-service scenarios only. In the following, it is extended to scenarios with dependent services.

3. Dependent Services

Two networked services depend on each other, when one service is a client of the other service or when both services are clients of each other. More than two services may depend on each other through a composition of such service-to-service dependencies.

We previously introduced the example of the Lustre cluster file system architecture [3, 4] with its metadata service (MDS) and object storage services (OSSs). In this scenario, MDS and OSSs are not only services for the file system driver client on the compute nodes of a HPC system, but also clients for each other. Assuming n compute, 1 MDS and m OSS nodes, this architecture consists of a n to 1 dependency for file system driver and MDS, a n to m dependency for file system driver and OSSs, a m to 1 dependency for OSSs and MDS, and a 1 to m dependency for MDS and OSSs.

In order to deal with such dependencies between services, we propose to extend the existing transparent symmetric active/active replication model by using its already existing mechanisms and features for client-service systems. While each interdependency between two services is decomposed into two respective orthogonal client-service dependencies, services utilize client-side adapters/interceptors for communication to services they depend on.

A high-level abstraction of the existing transparent symmetric active/active replication model is needed to illustrate dependencies between clients and services, and to decompose dependencies between services into respective client-service dependencies. With the help of the recently introduced virtual communication layer

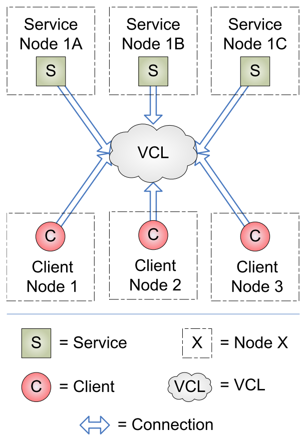

Figure 3. Transparent Symmetric Active/Active Replication for Client/Service Scenarios

(VCL) that hides the replication infrastructure as much as possible [16], the model can be simplified into five components: nodes, clients, services, VCLs, and connections (Figure 3). This high-level abstraction is independent of the implementation method (internal or external), while it clearly identifies clients, services, replicas, client-service dependencies, and decomposed service-to-service dependencies.

In this abstraction, a node represents some specific state. Each node may host one service and multiple clients, where clients on a node that hosts a service belong to that service. Each VCL belongs to one group of replicated services and their clients. Any client or service may belong to only one VCL.

The notion of node in this abstraction may not directly fit to real-world applications as a single physical service node may host multiple services, like the head node in a HPC system. This may be noted in the abstraction by using a node for each service that runs independently on a physical node. Services that do depend on each other and run on the same physical node are considered as a one service as they are replicated as a single state machine.

Service-independent clients may be noted by using a node for each client. Independent clients may not be grouped together on a node in the abstraction, even if they reside on the same physical node.

For the respective VCL they are connected to, clients utilize their client-side adaptor/interceptor and services utilize their service-side adaptor/interceptor. However, client-side adapters/interceptors that belong to a replicated service need to deal with the replicated output produced by this service either directly using the earlier mentioned distributed mutual exclusion, or indirectly by ignoring duplicated messages in the service-side adapters/interceptors. In both cases, the client-side adapters/interceptors of a replicated service form a single virtual client in the VCL layer, thus allowing client-side adapters/interceptors of clients or of non-replicated services to connect to the same VCL.

Based on this high-level abstraction, a variety of distributed computing scenarios may be expressed to demonstrate the application of the transparent symmetric active/active replication model in more complex scenarios with dependent services.

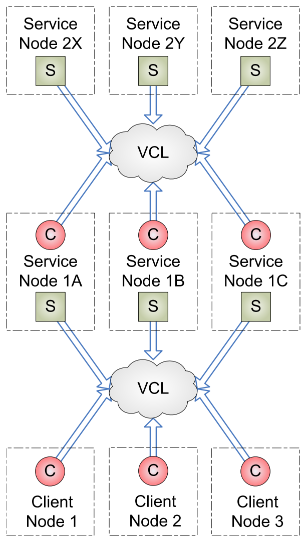

Figure 4. Transparent Symmetric Active/Active Replication for Client/Client+Service/Service Scenarios

Figure 4 depicts a scenario with a group of clients (client nodes 1-3) accessing a replicated service group (service nodes 1A-C), which itself relies on another replicated service group (service nodes 2X-Z). The service nodes 1A-C provide a replicated service to client nodes 1-3, while they are clients for service nodes 2X-Z. Each of the two VCLs performs the necessary replication mechanisms for its replicated service.

We previously mentioned our earlier work on providing symmetric active/active replication support for the parallel job and resource management service TORQUE [33]. While we provided symmetric active/active replication support the TORQUE service process running on HPC system head nodes that is used for parallel job scheduling and system resource management, one of the major deficiencies of this solution was the inability to survive a failure of the TORQUE service process running on HPC system compute nodes that is used for parallel job startup, control and monitoring. This issue was due to the fact that both services depend on each other.

The scenario depicted in Figure 4 solves this issue as services 1A-C represent the replicated TORQUE service process running on HPC system head nodes and services 2X-Z represent the replicated TORQUE service process running on HPC system compute nodes. Both TORQUE services can be fully and transparently replicated by using serial VCLs to hide the replication infrastructure from both services.

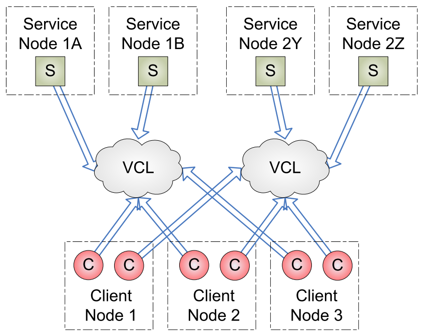

Figure 5. Transparent Symmetric Active/Active Replication for Client/2 Services Scenarios

Figure 5 shows a group of clients (client nodes 1-3) communicating with two different replicated service groups (service nodes 1A-B and 2Y-Z). The client nodes 1-3 host two clients, each for a different VCL belonging to a different replicated service group.

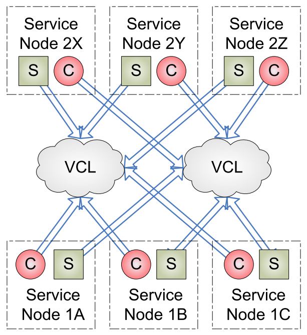

Figure 6. Transparent Symmetric Active/Active Replication for Service/Service Scenarios

Figure 6 illustrates a scenario with two interdependent replicated service groups. The service nodes 1A-B provide a replicated service and are clients of 2Y-Z, while the service nodes 2Y-Z provide a replicated service and are clients of 1A-B.

The introduced high-level abstraction may be used to guide the implementation of transparent symmetric active/active replication solutions in real-world distributed computing systems with complex service-oriented architectures (SOAs).

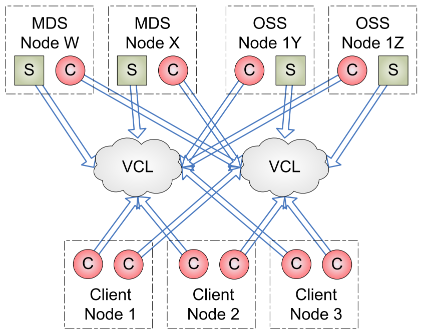

Figure 7. Transparent Symmetric Active/Active Replication for the Lustre cluster file system

Going back to the Lustre cluster file system example, Figure 7 illustrates the Lustre architecture using this high-level abstraction. This example uses 3 file system driver clients (client nodes 1-3), a replicated MDS service group (MDS nodes W-X), and one replicated OSS service group (OSS nodes Y-Z). This architecture is a combination of a group of clients communicating with two different replicated service groups (Figure 5) and two interdependent replicated service groups (Figure 6). Since Lustre supports many clients and several OSSs, the depicted example may be extended with respective components if needed.

4. Intitial Results

While this paper is mainly intended to present the concept of transparent symmetric active/active replication in distributed computing scenarios with dependent services, we also want to address several questions regarding performance of the presented concept.

As mentioned before, our past work has shown that implementations are correct, can be quite efficient, and have a practical value in the field [29, 33]. The always questioned overhead introduced by the process group communication system has been previously tested and discussed [30], and can be classified between low and negligible in small replicated service groups with 2-4 services. The overhead introduced by the external replication method using interceptor processes has been previously evaluated as well [16], and can be classified as acceptable in application scenarios that are generally not communication latency sensitive.

This paper introduces two new basic implementation configurations: (1) multiple serial VCLs, *i.e.*, nodes with a service and one or more clients (Figure 4), and (2) multiple parallel VCLs, *i.e.*, nodes with multiple clients (Figure 5). Service-to-service scenarios (Figure 6) are a combination of both.

Multiple parallel VCLs may interfere with each other if the process group communication traffic is routed through the same network, *i.e.*, via the same network switch. Since this interference is highly application dependent (bandwidth usage, collisions, network quality of service), a generic performance evaluation does not make much sense. Furthermore, a separation of process group communication traffic for additional performance may be implemented using a separate network between replicated service nodes.

Multiple serial VCLs interfere with each other by adding latency in a request/response scenario commonly used in distributed computing systems in the form of remote procedure calls (RPCs).

Reiterating the Lustre cluster file system example, the file system driver client on the compute nodes communicates with the MDS, which in turn communicates with OSSs. This scenario can be observed when deleting a file, where the MDS deletes the file record and notifies the OSSs to delete the file object before returning a response back to the file system driver client that initiated the file deletion request.

Since Lustre is a high-performance cluster file system and its MDS request/response latency a major performance measure, we performed initial tests with a generic client/service/service architecture to measure the performance impact of using interceptor processes in a serial VCL configuration.

We performed two test series using (1) a single service and (2) two serial services, each with and without interceptor processes, *i.e.*, VCLs, between client and service 1, and service 1 and service 2 (Figure 4). The tests were performed in a 1Gbps TCP/IP LAN environment using the 64-bit Fedora Core 5 Linux distribution on Intel Pentium D nodes with 2GB RAM. We conducted round trip request/response latency and bandwidth measurements with various payloads for both request and response messages.

Services	1		2		
Interceptors	0	2	0	2	4
100B (*ms*/%)	0.10	0.19/189	0.20	0.27/133	0.37/180
1KB (*ms*/%)	0.16	0.24/152	0.32	0.38/121	0.47/148
10KB (*ms*/%)	0.35	0.45/128	0.70	0.78/110	0.88/125
100KB (*ms*/%)	2.21	2.40/107	3.86	4.25/110	4.72/122
1MB (*ms*/%)	17.2	24.0/140	34.4	40.8/119	47.7/139

Table 1. Request/Response Latency Comparison with Various Payloads and Scenarios

The request/response latency comparison (Table 1) clearly shows the increasing performance impact of adding interceptor processes into the communication path. The penalty can be as high as 89% in the single client-service scenario, and 80% in a serial VCL configuration of 2 services. The highest impact can be observed with small messages.

Services	1		2		
Interceptors	0	2	0	2	4
100B (MBps/%)	0.99	0.52/53	0.49	0.37/75	0.27/56
1KB (MBps/%)	6.28	4.15/66	3.17	2.62/83	2.15/68
10KB (MBps/%)	28.3	22.0/78	14.2	12.8/90	11.4/80
100KB (MBps/%)	45.2	41.7/92	25.9	23.6/91	21.2/82
1MB (MBps/%)	58.0	41.7/72	29.0	24.5/85	21.0/72

Table 2. Request/Response Bandwidth Comparison with Various Payloads and Scenarios

The bandwidth comparison (Table 2) also clearly shows the increasing impact of adding interceptor processes into the communication path. The penalty can be as high as 47% in the single client-service scenario, and 46% in a serial VCL configuration of 2 services. Similar to the latency impact, the highest bandwidth impact is with small messages.

There are two factors that influence this performance impact: (1) latency added by the interceptor processes, and (2) local traffic (and respective congestion) added by the interceptor processes. Both are due to the fact that the interceptor processes cause network traffic to go twice through the TCP/IP stack of the operating system, once to send/receive to/from the network and once to send/receive to/from its client or service.

Similar to our previous performance evaluation of the overhead introduced by the external replication method using interceptor processes [16], latency sensitive scenarios should rather rely on internal replication using adapters. In scenarios where portability or ease-of-use are more important, external replication using interceptor processes should be employed.

5. Related Work

A substantial amount of evaluation of related research and development efforts can be found in our earlier papers on symmetric active/active replication [9, 10, 11, 12, 13, 14, 15, 16, 28, 29, 30, 33].

Most notably, there is a plethora of past work on process group communication algorithms and systems focusing on semantics, correctness, efficiency, adaptability, and programming models. A taxonomy and survey can be found in the following two papers [2, 6].

Our work on symmetric active/active replication was primarily motivated by previous research and development efforts in process group communication systems and its missing application to critical system services in HPC environments. The virtual communication layer utilizes an improved variant [30] of the Transis group communication system [7].

Other related work is in process group communication toolkits, like Spread [1], which aim to provide highly tuned application-level multicast, group communication, and point to point support for distributed applications. Some solutions, like Cactus [18], offer a framework for supporting customizable dynamic fine-grain Quality of Service (QoS) attributes related to dependability, real time, and security in distributed systems.

Depending on the actual use-case requirements, all of these solutions may be used internally by the virtual communication layer (VCL) of our transparent symmetric active/active replication model. The VCL provides an abstraction for transparently replicating a deterministic service with fail-stop behavior by utilizing a process group communication toolkit to perform state-machine replication with virtual synchrony.

Our approach also relates to the Object Group Pattern [24], which offers programming model support for replicated objects using a group communication system with virtual synchrony. In this design pattern, objects are constructed as state machines and replicated using totally ordered and reliably multicast state changes. The Object Group Pattern also provides the necessary hooks for copying object state, which is needed for joining group members. In our work on transparent symmetric active/active replication for already existing critical services, the interface to copy service state is still a service-proprietary implementation.

Orbix+Isis and Electra are follow-on research projects [23] that focus on extending high availability support to CORBA using object request brokers (ORBs) on top of virtual synchrony toolkits.

Related work also includes recent research in low-overhead solutions for practical Byzantine fault tolerance for networked services [25, 31], where the fail-stop requirement is not assumed and incorrect, arbitrary service behavior is detected.

6. Conclusion

With this paper, we have addressed one important limitation of our existing transparent symmetric active/active replication model for providing service-level high availability. Its deficiency, the inability to deal with dependent services, has been resolved by extending the model 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 recently introduced virtual communication layer (VCL) to hide the replication infrastructure for a specific service group as much as possible.

This presented concept is able to transparently provide high availability in distributed computing systems with complex service-oriented architectures, such as for the critical middleware and system service infrastructure within modern HPC systems [9, 8].

Planned work focuses on implementing the concept with specific services in the field.

This paper already contains initial results from the ongoing application of the presented concept to the Lustre cluster file system in terms of design of the replication infrastructure and of its individual subsystems. It has to be noted that Lustre is a very special case, as its clients and services reside in the Linux kernel space, while complex internal distributed locking as well as partial recovery mechanisms are employed.

We also hope that the presented concept is picked up by the service-oriented architecture (SOA) community for providing service-level high availability with strong consistency semantics in critical SOA infrastructures, such as for Web services.

References

[1] Y. Amir, C. Danilov, M. Miskin-Amir, J. Schultz, and J. Stanton. The Spread toolkit: Architecture and performance. Technical Report CNDS-2004-1, Johns Hopkins University, Center for Networking and Distributed Systems, Baltimore, MD, USA, 2004.

[2] G. V. Chockler, I. Keidar, and R. Vitenberg. Group communication specifications: A comprehensive study. *ACM Computing Surveys (CSUR)*, 33(4):427–469, 2001. ACM Press, New York, NY, USA.

[3] Cluster File Systems, Inc., Boulder, CO, USA. Lustre Cluster File System, 2007. http://www.lustre.org.

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

[5] Cray Inc., Seattle, WA, USA. Cray XT4 Computing Platform Documentation, 2007. http://www.cray.com/products/xt4.

[6] X. Défago, A. Schiper, and P. Urbán. Total order broadcast and multicast algorithms: Taxonomy and survey. *ACM Computing Surveys (CSUR)*, 36(4):372–421, 2004. ACM Press, New York, NY, USA.

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

[8] 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 (ICCS) 2007, Part II*, volume 4488, pages 784–791, Beijing, China, May 27-30, 2007. Springer Verlag, Berlin, Germany.

[9] C. Engelmann and S. L. Scott. Concepts for high availability in scientific high-end computing. In *Proceedings of the High Availability and Performance Workshop (HAPCW) 2005, in conjunction with the Los Alamos Computer Science Institute (LACSI) Symposium 2005*, Santa Fe, NM, USA, Oct. 11, 2005.

[10] 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 (COSET-2) 2005, in conjunction with the 19th ACM International Conference on Supercomputing (ICS) 2005*, Cambridge, MA, USA, June 19, 2005.

[11] C. Engelmann, S. L. Scott, and G. A. Geist. High availability through distributed control. In *Proceedings of the High Availability and Performance Workshop (HAPCW) 2004, in conjunction with the Los Alamos Computer Science Institute (LACSI) Symposium 2004*, Santa Fe, NM, USA, Oct. 12, 2004.

[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 (ARES) 2006*, pages 639–645, Vienna, Austria, Apr. 20-22, 2006. IEEE Computer Society.

[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 (JCP)*, 1(8):43–54, 2006. Academy Publisher, Oulu, Finland.

[14] C. Engelmann, S. L. Scott, C. Leangsuksun, and X. He. Towards high availability for high-performance computing system services: Accomplishments and limitations. In *Proceedings of the High Availability and Performance Workshop (HAPCW) 2006, in conjunction with the Los Alamos Computer Science Institute (LACSI) Symposium 2006*, Santa Fe, NM, USA, Oct. 17, 2006.

[15] 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 (ARES) 2007*, pages 999–1006, Vienna, Austria, Apr. 10-13, 2007. IEEE Computer Society.

[16] 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 (CCGrid) 2007*, pages 755–760, Rio de Janeiro, Brazil, May 14-17, 2007. IEEE Computer Society.

[17] T. Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall PTR, Upper Saddle River, NJ, USA, Aug. 2005.

[18] M. A. Hiltunen and R. D. Schlichting. The Cactus approach to building configurable middleware services. In *Proceedings of International SRDS Workshop on Dependable System Middleware and Group Communication (DSMGC) 2000*, Nuernberg, Germany, Oct. 16-18, 2000.

[19] IBM Corporation, Armonk, NY, USA. IBM Blue Gene Computing Platform Documentation, 2007. http://www-03.ibm.com/servers/deepcomputing/bluegene.html.

[20] IBM Corporation, Armonk, NY, USA. MareNostrum eServer Computing Platform Documentation, 2007. http://www.ibm.com/servers/eserver/linux/power/marenostrum.

[21] S. M. Kelly and R. Brightwell. Software architecture of the light weight kernel, Catamount. In *Proceedings of 47th Cray User Group (CUG) Conference 2005*, Albuquerque, NM, USA, May 16-19, 2005.

[22] L. Lamport. Using time instead of timeout for fault-tolerant distributed systems. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 6(2):254–280, 1984. ACM Press, New York, NY, USA.

[23] S. Landis and S. Maffeis. Building reliable distributed systems with CORBA. *Theory and Practice of Object Systems*, 3(1):31–43, 1997. Wiley InterScience, John Wiley & Sons, Inc., Hoboken, NJ, USA.

[24] S. Maffeis. The object group design pattern. In *Proceedings of 2nd USENIX Conference on Object-Oriented Technologies (COOTS) 1996*, page 12, Toronto, ON, Canada, June 17-21, 1996. USENIX Association, Berkeley, CA, USA.

[25] M. G. Merideth, A. Iyengar, T. Mikalsen, S. Tai, I. Rouvellou, and P. Narasimhan. Thema: Byzantine-fault-tolerant middleware for web-service applications. In *Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems (SRDS) 2005*, pages 131–142, Orlando, FL, USA, Oct. 26-28, 2005. IEEE Computer Society.

[26] J. Moreira, M. Brutman, n. José Casta T. Engelsiepen, M. Giampapa, T. Gooding, R. Haskin, T. Inglett, D. Lieber, P. McCarthy, M. Mundy, J. Parker, and B. Wallenfelt. Designing a highly-scalable operating system: The Blue Gene/L story. In *Proceedings of International Conference on High Performance Computing, Networking, Storage and Analysis (SC) 2006*, page 118, Tampa, FL, USA, Nov. 11-17, 2006. ACM Press, New York, NY, USA.

[27] 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 (ICDCS) 1994*, pages 56–65, Poznan, Poland, June 21-24, 1994. IEEE Computer Society.

[28] D. I. Okunbor, C. Engelmann, and S. L. Scott. Exploring process groups for reliability, availability and serviceability of terascale computing systems. In *Proceedings of the 2nd International Conference on Computer Science and Information Systems 2006*, Athens, Greece, June 19-21, 2006.

[29] 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 (PDCS) 2007*, Cambridge, MA, USA, Nov. 19-21, 2007. ACTA Press, Calgary, AB, Canada.

[30] 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 (ICCCN) 2007*, Honolulu, HI, USA, Aug. 13-16, 2007. IEEE Computer Society.

[31] R. Rodrigues, M. Castro, and B. Liskov. BASE: Using abstraction to improve fault tolerance. volume 35, pages 15–28, 2001. ACM Press, New York, NY, USA.

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

[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 the 8th IEEE International Conference on Cluster Computing (Cluster) 2006*, Barcelona, Spain, Sept. 25-28, 2006. IEEE Computer Society.