Transparent Symmetric Active/Active Replication for Service-Level High Availability*

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

As service-oriented architectures become more important in parallel and distributed computing systems, individual service instance reliability as well as appropriate service redundancy becomes an essential necessity in order to increase overall system availability. This paper focuses on providing redundancy strategies using service-level replication techniques. Based on previous research using symmetric active/active replication, this paper proposes a transparent symmetric active/active replication approach that allows for more reuse of code between individual service-level replication implementations by using a virtual communication layer. Service- and client-side interceptors are utilized in order to provide total transparency. Clients and servers are unaware of the replication infrastructure as it provides all necessary mechanisms internally.

1. Introduction

Services are an integral part of today‘s parallel and distributed computing systems. While service-oriented architectures (SOA) [12] play a significant role in distributed Grid computing systems [14], the trend toward services also emerges in large-scale closely coupled massively parallel high performance computing (HPC) systems for scientific computing [6].

*This research was sponsored by the Mathematical, Information, and Computational Sciences Division; Office of Advanced Scientific Computing Research; U.S. Department of Energy. The work 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.

In both cases, individual service instances are susceptible to hardware and software failures, such as caused by a hard disk crash or a programming error. Additional causes of service outages are disaster scenarios, such as flood, tornado, and terrorist attack. Individual service instance reliability as well as appropriate service redundancy are essential to increase overall distributed computing system availability.

There are various techniques for providing service redundancy, such as active/standby and active/active. Implementations are based on replication mechanisms on multiple redundant service nodes.

Our previous research in service-level symmetric active/active replication resulted in two different replication methods [8, 9], internal and external, and in various proof-of-concept prototype implementations for HPC system services, such as the batch job scheduler [22] and the parallel file system metadata server. A recent evaluation of our accomplishments and their limitations [10] revealed several issues. One reported problem was the insufficient reuse of code between individual prototype implementations. Each service required a customized symmetric active/active environment.

This paper proposes a transparent symmetric active/active replication software architecture for service-level high availability that accommodates both replication methods, internal and external, by using a virtual communication layer (VCL). The original internal and external symmetric active/active replication methods are refined to utilize service- and client-side interceptors in order to provide total transparency. Adaptation of these interceptors to clients and services is only needed with regards to the used communication protocols and its semantics. Clients and servers are unaware of the symmetric active/active replication infrastructure as it provides all necessary mechanisms internally.

This paper is structured as follows. First, we provide an overview of the service-level high availability concept and the symmetric active/active replication model. Second, we introduce a refined software architecture for transparent symmetric active/active replication, discuss its internal fault tolerant communication mechanisms, and present preliminary performance test results. Third, we briefly describe related past and ongoing work in this area. We conclude with a short summary of the presented research.

2. Background Overview

2.1. Service-Level High Availability

A service is a communicating process that interacts with other local or remote services and/or with users via an input/output interface, such as network connection(s) and command line interface(es) [11]. Services are stateful or stateless. They also show either deterministic or non-deterministic behavior. The research presented in this paper focuses on stateful deterministic services only, as most services in parallel and distributed computing systems display these properties.

A service may fail by simply stopping its operation, *i.e.*, by stopping to respond (timeout).

Service high availability mechanisms require to consistently replicate service process state to multiple redundant services on different service nodes. Implementations are based on either system-level or service-level replication mechanisms.

System-level replication is based on the concept of replicating service process state from the operating system (OS) perspective. The Berkeley Lab Checkpoint/Restart (BLCR) [1] layer is an example for a system-level replication mechanism.

On the other hand, service-level replication inserts a middleware between OS and service to replicate service process state from the service perspective. The earlier mentioned symmetric active/active batch job scheduler solution [22] is an implementation example for a service-level replication mechanism.

Both mechanisms have their advantages and disadvantages. System-level replication mechanisms are inherently transparent to the service, but less efficient due to the limited knowledge about the service. In contrast, service-level replication mechanisms are able to adapt to specific service properties, such as interaction with other services/users and quality of service requirements, but they often require to modify the original service.

The research presented in this paper targets transparent service-level replication mechanisms that provide the advantages of both approaches.

2.2. Symmetric Active/Active Replication

In the symmetric active/active (A/A) replication model for service-level high availability, two or more active services offer the same capabilities and maintain a common global service state [8, 9, 11].

Service-level symmetric active/active replication is based on guaranteeing the same initial states and a linear history of state transitions for all active services, *i.e.*, virtual synchrony [18]. Service state replication is performed by totally ordering all incoming request messages and reliably delivering them to all active services. A process group communication system is used to assure total message order, reliable message delivery, and service group membership management. Consistent output messages produced by all correct active services is unified either by simply ignoring duplicated messages or by using the group communication system for a distributed mutual exclusion. The latter is required if duplicated messages cannot be simply ignored by dependent services and/or users.

The virtual synchrony model requires that each replicated service performs the same order of state transitions based on the same order of incoming request messages, which are delivered to each service by a group communication system. Adaptation of the service to this event-based or request/response programming model can be performed either internally by modifying the service itself or externally by wrapping it into a virtually synchronous environment [8, 9].

Internal replication allows each active service to accept request messages individually, while using the group communication system for total message order and reliable message delivery to all members of the service group. This method requires modification of existing code, which may be unsuitable for complex and/or large services. However, it may lead to performance enhancements as internal replication implies fine-grain synchronization of state transitions.

External replication avoids modification of existing code by wrapping a service into a virtually synchronous environment. Interaction with dependent services and users is intercepted, totally ordered, and reliably delivered to each service using the group communication system to mimic the service interface. This method not only does not modify existing service code, it also allows reusing the same solution for different services with the same interface. However, it may lead to performance degradation as external replication implies coarse-grain synchronization of state transitions.

In both cases, an internal or external service-side interceptor component deals with receiving incoming request messages and routing them through the group

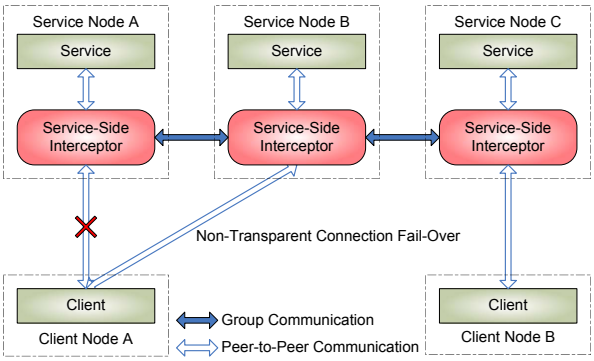

Figure 1. Original Symmetric Active/Active Replication Software Architecture with Non-Transparent Client Connection Fail-Over

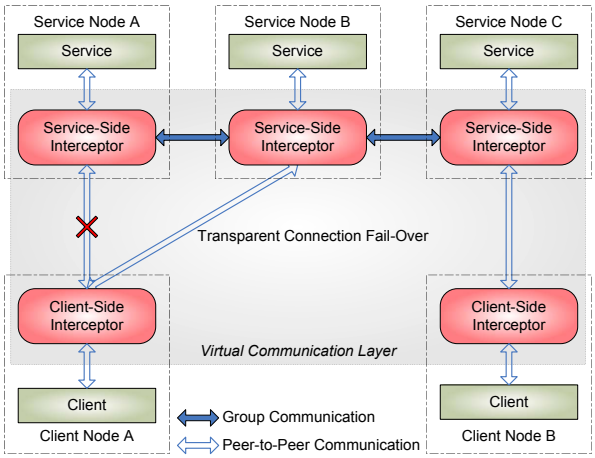

Figure 2. Refined Symmetric Active/Active Replication Software Architecture with Transparent Client Connection Fail-Over

communication system for total message order and reliable message delivery (Figure 1). It also routes output back to the client, which interacts with the service-side interceptor component instead with the original service. In case of a failure, clients need to be reconfigured in order to interact with the service-side interceptor component of another member of the active service group. This requires clients to be informed about service group membership and to perform a consistent connection fail-over in case of a failure. Clients need to be made aware of the service-level symmetric active/active replication technique and need to be modified for internal and external service-side replication. More details about the implemented high availability programming model can be found in an earlier paper [11].

Our previously developed prototype implementations for the HPC batch job scheduler service Torque [22] using external symmetric active/active replication and for the Parallel Virtual File System 2 (PFVS2) [19] metadata server (MDS) using internal symmetric active/active replication utilized a Transis [4, 21] group communication system feature at the client side by interacting with the active service group without actually being a full member of the group.

The lack of transparency in both symmetric active/active replication methods resulted in an insufficient reuse of code between individual prototype implementations. Each service required a customized symmetric active/active environment at the service-side as well as at the client-side. Moreover, the client needed to be significantly modified even in the external symmetric active/active replication method.

In the following, we propose to refine the internal and external symmetric active/active replication software architecture using a virtual communication layer that accommodates both replication methods and allows for transparent client connection fail-over as well as for more reuse of code between individual service-level replication implementations.

3. Transparent Symmetric A/A Replication

3.1. Architecture

The main idea behind the service-side interceptor concept of our original internal and external symmetric active/active replication software architecture was to hide the interaction of the service with the group communication system from the service. While the internal replication method tightly integrates the service with the service-side interceptor, the external replication method utilizes the service interface. In both cases, the client interacts with the service-side interceptor.

In the proposed refined symmetric active/active replication software architecture with transparent client connection fail-over (Figure 2), an additional client-side interceptor hides the interaction of the client with the service-side interceptor from the client in the same fashion the service-side interceptor hides the interaction of the service with the service-side interceptor.

Similar to the service-side interceptor, the client-side interceptor may be implemented internally by tightly integrating the client with the client-side interceptor or externally by utilizing the service interface at the client-side interceptor. In both cases, the client recognizes the client-side interceptor as the service.

The client- and service-side interceptors maintain a virtual communication layer (VCL), which client and service are unaware of. In fact, the client is only aware

of a connection to a local service represented by the client-side interceptor, while the service is only aware of a connection to a local client represented by the service-side interceptor.

The VCL enforces certain group communication semantics at the service-side interceptor as well as at the client-side interceptor based on the existing symmetric active/active high availability model [11]. In addition to maintaining the previous role of service-side interceptors, the VCL assures that client-side interceptors are informed about service group membership and perform a consistent connection fail-over in case of a failure.

The proposed VCL not only provides transparency to services, but to clients as well. Additionally, the client- and service-side interceptors communicate with client and service via standard input/output channels, such as sockets, network connections, and command line, in order to further increase reuse of code between individual service-level replication implementations. An adaptation to a specific client/server protocol may be performed by using a modular pluggable communication protocol substrate, as exemplified by the RMIX communication framework [5].

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, authorization, and auditing (AAA) using the client- and server-side interceptors as virtual protocol routers. Second, the VCL itself may need to perform similar AAA mechanisms to assure its own integrity across administrative domains.

At this moment, we do not consider complex AAA mechanisms in the proposed symmetric active/active replication software architecture as they require mapping of peer-to-peer AAA protocols to group communication semantics. This is an open research issue we currently do not address.

3.2. Transparent Connection Fail-Over

In addition to fault-tolerant group communication mechanisms for the group of service-side interceptors described in earlier papers [8, 9], the VCL provides fault-tolerant communication fail-over for client connections [11] in a transparent fashion, *i.e.*, clients are unaware of the failure of a service-side interceptor.

Upon initial connection to a service-side interceptor, the client-side interceptor receives the current list of service-side interceptor group members. All client-side interceptors are notified about membership changes by the service-side interceptor they are connected to after all service-side interceptor group members agree.

A client-side interceptor that detects a failure of its service-side interceptor performs a connection fail-over to another service-side interceptor based on its current list of service-side interceptor group members. After reconnection to the service-side interceptor group, a recovery protocol retrieves any undelivered messages.

The connection between client- and service-side interceptors uses a basic message numbering and acknowledgment scheme in both directions to assure correct message delivery even in case of failures. While the message numbering assures that already received messages can be ignored, acknowledgments are used in certain intervals to clear cached messages on the sending side. However, acknowledgments from the client-side interceptor are interleaved with request messages, reliably multicast, and totally ordered, so that each service-side interceptor is able to maintain a consistent message cache for service-side output messages in order to perform the connection fail-over transparently.

In case of a connection fail-over, all cached messages are resent in the same order and doubled messages are ignored accordingly to the number of the last received message before the failure due to the enforced total message order in both directions.

3.3. Client- and Service-Side VCL Interfaces

The provided interfaces of the VCL at the client-side and service-side inteceptors depend on actual client and service interfaces as well as on the replication method, *i.e.*, internal or external. The standard input/output channels provided by the VCL to clients and services virtualize client/service communication based on the utilized replication method.

In case of internal replication, inter-process communication is intercepted inside the client and the service by libraries that provide the same internal communication protocols and semantics to client and service while communicating over the VCL. For example, a simple TCP/IP-based client/service system is modified by replacing the system calls for TCP/IP communication (open/read/write/close) in the client and service codes with calls to client- and service-side interceptor library equivalents.

In case of external replication, inter-process communication is intercepted outside the client and the service by processes that provide the same external communication protocols and semantics to client and service while communicating over the VCL. For example, a simple TCP/IP-based client/service system is modified by configuring the client-side interceptor process as a service for the client, and the service-side interceptor process as a client for the service.

In both cases, the main advantage provided by the transparent symmetric active/active replication approach is that client- and service-side interceptor libraries and processes can be reused for different services that have the same communication protocols and semantics. For example, for any simple TCP/IP-based client/service system. The VCL provides transparency and high availability.

Furthermore, the client- and service-side interceptor libraries and processes of the VCL may be configurable and extensible in order to adapt to different communication protocols and semantics, including supporting multiple different communication methods, *e.g.*, network and command line, at the same time. This will not only help to further improve reuse of code, it will also enable efficient symmetric active/active replication in complex distributed computing scenarios, where multiple services are clients of each other.

3.4. Performance Impact

Introducing external interceptors into the communication path of a client/service system inherently results in a certain performance degradation. Also, totally ordering messages using a group communication system at the service-side interceptor is further impacting client/service communication performance.

Preliminary results (Tables 1 and 2) in a 100Mbps LAN environment using external replication show that latency increases and throughput decreases with a service-side interceptor, and both further degrade with an additional client-side interceptor. The performance penalty for small payloads ($<= 1KB$) for using client and service-side interceptors can be as high as 22% for latency and 17% for bandwitdh in comparison to an unmodified client/service system, and 19% and 15%, respectively, in comparison to using server-side interceptors only. However, the overall performance impact dramatically decreases with increasing payload.

The tests emulate a remote procedure call (RPC) pattern by sending a payload to the service and waiting for its return. The latency is measured at the client for the entire round trip of the payload. The tests do not include any group communication system as its performance impact has been studied earlier [9, 3].

Payload	Without Interceptors	With Service Interceptor	With Both Interceptors
$100B$	$149.9\mu s$	$150.6\mu s$/ +0.5%	$178.4\mu s$/+19.0%
$1KB$	$284.3\mu s$	$314.6\mu s$/+10.7%	$346.7\mu s$/+21.9%
$10KB$	$1.9ms$	$1.9ms$/ ±0.0%	$2.0ms$/ +5.3%
$100KB$	$22.3ms$	$22.5ms$/ +0.8%	$22.7ms$/ +1.8%

Table 1. Ping-Pong Latency Comparison

Payload	Without Interceptors	With Service Interceptor	With Both Interceptors
$100B$	$667KBps$	$664KBps$/−0.4%	$561KBps$/−15.9%
$1KB$	$3.5MBps$	$3.2MBps$/−8.6%	$2.9MBps$/−17.1%
$10KB$	$5.3MBps$	$5.2MBps$/−1.9%	$5.0MBps$/ −5.7%
$100KB$	$4.5MBps$	$4.4MBps$/−2.2%	$4.4MBps$/ −2.2%

Table 2. Ping-Pong Bandwidth Comparison

4. Related Work

Apart from our already mentioned related efforts, there has been a substantial amount of work on process group communication algorithms that focuses on providing quality of service guarantees for networked communication in distributed systems with failures. An overview of process group communication system implementations and semantics, as well as references to further related work can be found in [2] and [3].

Further related research focuses on programming models for replicated objects using virtual synchrony. The Object Group Pattern [16] designs objects as state machines, which may be replicated using totally ordered and reliably multicast state transitions. The follow-on research projects Orbix+Isis and Electra [15] focus on extending this object-oriented high availability support to CORBA using object request brokers (ORBs) on top of virtual synchrony toolkits.

Ongoing research and development efforts in adding security features to group communication systems, such as mapping of peer-to-peer AAA protocols to group communication semantics, focuses on group keys for securing multicast channels [20].

Lastly, the presented research is part of the MOLAR [7, 17] project, which concentrates on adaptive, reliable, and efficient operating and runtime system solutions for ultra-scale scientific high-end computing as part of the Forum to Address Scalable Technology for Runtime and Operating Systems (FAST-OS) [13].

5. Conclusions and Future Work

With this paper, we propose a refined symmetric active/active replication software architecture using a virtual communication layer (VCL) that accommodates both replication methods, internal and external, and allows for transparent client connection fail-over as well as for more reuse of code between individual service-level replication implementations.

With the introduction of client-side interceptors, the proposed solution additionally hides the interaction of the client with the service-side interceptor from the

client in the same fashion the service-side interceptor hides the interaction of the service with the service-side interceptor. Adaptation of these interceptors to clients and services is only needed with regards to the used communication protocol and its semantics. Clients and services are unaware of the symmetric active/active replication infrastructure as it provides all necessary mechanisms internally via the VCL.

Preliminary test results using external replication in a 100Mbps LAN environment show that latency increases and throughput decreases with increasing number of interceptors, while the greatest impact, 22% for latency and 17% for bandwitdh, occurs with small payloads ($<= 1KB$). The performance overhead for internal replication is neglegible, since there are no intermmediate communicating processes.

The proposed solution is able to provide transparent service-level high availability using the symmetric active/active replication approach for services in parallel and distributed computing systems. It is applicable to any service-oriented architecture, such as in distributed Grid computing, or service-dependent architecture, like large-scale closely coupled massively parallel high performance computing systems.

Future work will focus on fully implementing respective proof-of-concept prototypes with added basic communication security mechanisms.

References

[1] Berkeley Lab Checkpoint/Restart (BLCR) project at Lawrence Berkeley National Laboratory, Berkeley, CA, USA. Available at http://ftg.lbl.gov/checkpoint.

[2] G. V. Chockler, I. Keidar, and R. Vitenberg. Group communication specifications: A comprehensive study. *ACM Computing Surveys*, 33(4):1–43, 2001.

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

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

[5] C. Engelmann and G. A. Geist. RMIX: A dynamic, heterogeneous, reconfigurable communication framework. In *Lecture Notes in Computer Science: Proceedings of International Conference on Computational Science, Part II*, volume 3992, pages 573–580, Reading, UK, May 28-31, 2006.

[6] C. Engelmann, H. Ong, and S. L. Scott. Middleware in modern high performance computing system architectures. In *Lecture Notes in Computer Science: Proceedings of International Conference on Computational Science*, Beijing, China, May 27-30, 2007.

[7] 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.

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

[9] 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.

[10] 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 High Availability and Performance Workshop*, Santa Fe, NM, USA, Oct. 17, 2006.

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

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

[13] Forum to Address Scalable Technology for Runtime and Operating Systems (FAST-OS). Available at http://www.fastos.org.

[14] C. Kesselman and I. Foster. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, San Francisco, CA, USA, 1998.

[15] S. Landis and S. Maffeis. Building reliable distributed systems with CORBA. *Theory and Practice of Object Systems*, 3(1):31–43, 1997.

[16] S. Maffeis. The object group design pattern. *Proceedings of* 2^{nd} *USENIX Conference on Object-Oriented Technologies (COOTS)*, June 17-21, 1996.

[17] Modular Linux and Adaptive Runtime Support for High-end Computing Operating and Runtime Systems (MOLAR). Available at http://www.fastos.org/molar.

[18] L. Moser, Y. Amir, P. Melliar-Smith, and D. Agarwal. Extended virtual synchrony. *Proceedings of IEEE* 14^{th} *International Conference on Distributed Computing Systems*, pages 56–65, June 21-24, 1994.

[19] Parallel Virtual File System (PVFS). Available at http://www.pvfs.org/pvfs2.

[20] M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11:769–780, Nov. 2000.

[21] Transis Project at Hebrew University of Jerusalem, Israel. Available at http://www.cs.huji.ac.il/labs/transis.

[22] 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*, Barcelona, Spain, Sept. 25-28, 2006.