SELF-HEALING NETWORK FOR SCALABLE FAULT TOLERANT RUNTIME ENVIRONMENTS *

Thara Angskun[1], Graham Fagg[1], George Bosilca[1],
Jelena Pješivac–Grbović[1], and Jack Dongarra[1,2,3]
[1] *University of Tennessee,* [2] *Oak Ridge National Laboratory,* [3] *University of Manchester*
{angskun, fagg, bosilca, pjesa, dongarra}@cs.utk.edu

Abstract Scalable and fault tolerant runtime environments are needed to support and adapt to the underlying libraries and hardware which require a high degree of scalability in dynamic large-scale environments.

This paper presents a self-healing network (SHN) for supporting scalable and fault-tolerant runtime environments. The SHN is designed to support transmission of messages across multiple nodes while also protecting against recursive node and process failures. It will automatically recover itself after a failure occurs. SHN is implemented on top of a scalable fault-tolerant protocol (SFTP). The experimental results show that both the latest multicast and broadcast routing algorithms used in SHN are faster than the original SFTP routing algorithms.

Keywords: Fault tolerance, Routing, Runtime Environment, Scalability, Self-healing.

1. Introduction

Recently, several of high performance computing platforms have been installed with more than 10,000 CPUs, such as Blue-Gene/L at LLNL, BGW at IBM and Columbia at NASA [6]. However, as the number of components increases, so does the probability of failure. To satisfy the requirements of such a dynamic environment (where the available number of resources is fluctuating), a scalable and fault-tolerance framework is needed. Many large-scale applications are implemented on top of

*This material is based upon work supported by "Los Alamos Computer Science Institute (LACSI)", funded by Rice University Subcontract No. R7B127 under Regents of the University Subcontract No. 12783-001-05 49 and "Open MPI Derived Data Type Engine Enhance and Optimization", funded by the Regents of the University of California (LANL) Subcontract No. 13877-001-05 under DoE/NNSA Prime Contract No. W-7405-ENG-36

message passing systems for which the de-facto standard is the Message Passing Interface (MPI) [11]. MPI implementations require support of parallel runtime environments, which are extensions of the operating system services, and provide necessary functionalities (such as naming resolution services) for both the message passing libraries and applications. However, currently available parallel runtime environments are either not scalable or inefficient in dynamic environments. The lack of scalable fault-tolerance parallel runtime environments motivates us to design and implement such a system. A self-healing network (SHN) that can be used as a basis for constructing a higher level fault-tolerant parallel runtime environment is described in this paper. SHN was designed to support transferring messages across multiple nodes efficiently, while protecting against recursive node or process failures. It was built on top of a scalable and fault-tolerant protocol (SFTP) [1] and automatically recovered itself after a failure occurs.

The structure of this paper is as follows. The next section discusses previous and related work. Section 3 introduces the self-healing network and its recovery algorithm, while the section 4 presents the routing algorithm along with some experimental results, followed by conclusions and future work in the section 5.

2. Previous and Related Work

Although there are several existing parallel runtime environments for different types of systems, they do not meet some of the major requirements for MPI implementations: scalability, portability and performance. Typically, distributed OS and single system image systems are not portable while the nature of Grid middle-wares has performance problems.

The MPICH implementation [9] uses a parallel runtime environment called Multi-Purposed Daemon (MPD) [4] to providing scalability and fault-tolerance through a ring topology for some operations and a tree topology for others. Runtime environments of other MPI implementations, such as Harness [2] of FT-MPI [7], Open RTE [5] of Open MPI [8] and LAM of LAM/MPI [3], do not currently provide both scalable and fault tolerance solutions for their internal communications.

The scalability and fault-tolerance issues have been addressed in several networking areas. However, those approaches could not be used or they are not efficient in the parallel runtime environments. Structured peer-to-peer networking based on distributed hash tables such as CAN [12], Chord [15], Pastry [14] and Tapestry [16] was designed for resource discovery. They are only optimized for unicast messages. Tech-

niques used in sensor or large scale ad-hoc networking based on gossiping (or the epidemic algorithm) [10] [13] mainly focus on information aggregation.

The scalable and fault-tolerant protocol (SFTP) [1] was introduced to support parallel runtime environments. The protocol is based on a *k-ary* sibling tree. The *k-ary* sibling tree topology is a *k-ary* tree, where k is number of fan-out ($k \geq 2$), and the nodes on the same level (same depth on the tree) are linked together using a ring topology. The tree is primarily designed to allow scalability for broadcast and multicast operations, while the ring is used to provide a well understood secondary path for transmission when the tree is damaged during failure conditions. The protocol could be used to build a self-healing network which automatically recovers itself to overcome the orphan situation, the situation where nodes are unreachable because the tree becomes a bisection.

3. Self-Healing Network (SHN)

3.1 Overview

Although the self-healing network (SHN) is designed to support generic runtime environments of MPI implementations, the current work is in a progress to integrate it in a fault-tolerance implementation of message passing interface called FT-MPI as well as in the modular MPI implementation called Open MPI. The network is designed to support various operations needed by scalable and fault-tolerant MPI runtime environments. The example of those operations and the details of how SHN could be used for the operations are as follows.

Distributed Directory Service Directory service is a storage which maintains information used during running an MPI job such as contact information of each process, coordinator of recovery algorithm in FT-MPI etc. The SHN provides a possibility to use the network as a distributed directory service by mapping those information into the logical node ID. Scalable and fault-tolerant information management (update, query) could be done with unicast messages of SFTP routing (similar to resource discovery in the structured peer-to-peer networking).

Standard I/O Redirection Although MPI standard did not define how an input and an output could be treated, most of the MPI implementation redirect the standard output and the standard error to the user terminal (if not run under the batch scheduling). This operation could be done using the k-ary tree as a main route to forward the standard output/error and using the ring in case of failures.

Monitoring Framework A monitoring framework provides information such as processes, nodes, messages for tool and application de-

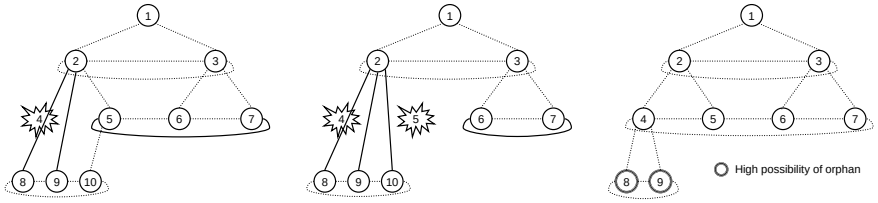


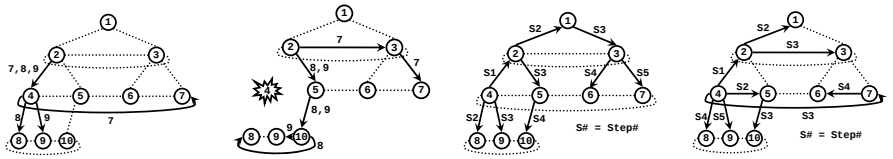
Figure 1. (a) SHN after recovery [4 dies] (B) SHN after recovery [4,5 die] (c) high possibility of orphan

velopment. Examples of those tools are parallel debuggers, runtime fault detectors, runtime verification and load balancers etc. To build a scalable and fault-tolerant monitoring framework, all of the communication underneath the framework can use multiple types of message transmissions (unicast, multicast and broadcast) provided by the SHN.

In general, the SHN provides a capability to send unicast, multicast and broadcast messages from any nodes while additionally protecting against node and process failures, from effecting message delivery.

3.2 SHN Recovery

There are some situations where nodes do not die but become unreachable due to network bisectioning. This situation can be prevented by self-recovery, when a node detects that its neighbor dies, it will send a unicast message to establish the connection with the next neighbor in the same direction of the dead node. If the next neighbor also dies, it will continue trying to establish the connection with the next node and so on until success or the next node in that direction is the node itself. If two nodes try to establish a connection at the same time, the connection which is initiated by higher ID will be dropped. Figure 1(a) illustrates an example where logical node 4 dies. All neighbors of node 4 will begin to recover the logical topology by reestablishing their connections in the appropriate direction. If node 5 also dies, the same recovery procedure will occur as shown in figure 1(b). However, there is an exception where the number of node in the last level (highest depth) of the tree is less than or equal to k, where k is fanout as shown in figure 1(c). In this case, the grandparent of the last level needs to know the contact information of the last level, because if the parent of the last level dies, those nodes in the last level will become orphans before the self-recover procedure can occur.



(a) (b) (c) (d)

Figure 2. (a) mcast (b) mcast-failure (c) bcast-updown (d) bcast-spanning tree

4. Routing Algorithm in SHN

The SHN routing algorithm is based on the SFTP routing algorithm [1]. The initialize system protocol, unicast message protocol and broadcast from a specific root protocol are the same as the SFTP protocol. The new multicast and broadcast routing algorithms from any nodes in the network, which are an extension of the SFTP routing algorithm have been added. They can be used both before (including some node failures) and after recovery of the logical topology. The SHN routing algorithms can be described as follows.

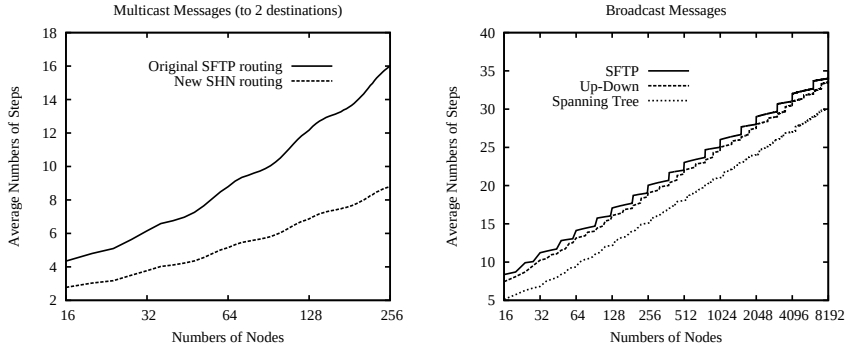
Multicast messages in SHN

The multicast from any nodes in the SHN is a capability to send messages to several destinations (1 to m, where m < n). Unlike the IP multicast, multicast group management (group creation and termination) is not required. The multicast group members are embedded in the message header. Multicast messages in SFTP are delivered by a sender to the first destination in the destination lists. Then, the first destination will forward the message to the next destination and so on. If an intermediate node is one of the nodes in the destination list, it will remove itself from the list. The order of nodes in the destination list is a descending order sorted by number of hop from a sender to those destinations (i.e. the largest number of hop first). This routing algorithm works fine if the destination nodes are consecutive or they are located in the same area of the tree. The new multicast routing algorithm in SHN is an enhancement of the SFTP multicast routing algorithm. The multicast message can be splitted at an intermediate node, if the shortest paths to those destination nodes are not in the same direction from the intermediate node point of view. However, if there are more than one shortest path to a destination, the intermediate node will choose the next hop which can go along with other destinations. When a node receives a multicast message, it will first determine the header and choose the next hop for each multicast destination according to the shortest path to them. The node will recreate the header corresponding to the

direction of each next hop. Messages that contain the largest number of hops will be forwarded first to increase network throughput by utilize multiple links simultaneously. Figure 2(a) shows an example of node 2 sending a multicast message to nodes 7, 8 and 9 with the new routing algorithm. In case of failure, if a node detects that the next hop for the multicast messages has died, it automatically reroutes the multicast messages using an alternate next hop as shown in Figure 2(b). Figure 3(a) depicts that the new multicast routing algorithm is faster than the original algorithm used in the SFTP. The experiment results were obtained from an average number of steps for sending multicast messages to 2 destinations with a dead node (fanout=2). The 2 destination nodes (D) were obtained from combinations of all possible nodes (N) i.e. $\binom{N}{D}$, where a source node $\notin$ D and the dead node was randomly selected.

Broadcast messages in SHN

Broadcast from any node routing protocol is an enhancement of broadcast routing in SFTP. In SFTP, the broadcast is done by sending messages to a root of the tree and it will forward the messages to the rest of the tree. Only the tree portion of SFTP is used to prevent a broadcast storm and duplicate messages. The ring is used only in the case of failure. The first obvious improvement of this routing protocol is to allow a node between source and a root of the tree to send messages to their children after they send the messages to their parent (called up-down) as shown in 2(c) with node 4 as the root. The second improvement is using a logical spanning tree from the source as shown in 2(d). When each node receives broadcast messages, it will calculate the next hops using spanning trees from the source node. There are two steps involving the next hop calculation. The first step is to create a spanning tree using a source node as the root node of the tree. The spanning tree creation algorithm is based on a modified version of the breath first search with a graph coloring algorithm. The second step is to calculate the next hop. The next hop is chosen from children of each node according to the spanning tree which has the highest cost among its children. The cost is computed from the number of steps used to send a message to all nodes in the children's subtrees. In case of failure, a broadcast message is encapsulated into a multicast message, and then the message is sent from parent of the failure node to its children in the spanning tree. Figure 3(b) indicates that the up-down algorithm is a marginally faster than the original SFTP, while the new spanning tree broadcast routing algorithm is significant faster than the SFTP broadcast routing algorithm due to increased parallelism. The experimental results were obtained from an average number of steps for sending a broadcast message from every node (fanout=2).



(a) (b)

Figure 3. A comparison of routing protocols (a) multicast (b) broadcast

5. Conclusions and Future Work

The self-healing network (SHN) for parallel runtime environments was designed and developed to support runtime environments of MPI implementations. The SHN is implemented on top of a scalable fault-tolerant protocol (SFTP). Simulated performance results indicate that the new routing algorithms of SHN are faster than the original SFTP routing algorithms.

There are several improvements that we plan for the near future. Making the protocol aware of the underlying network topology (in both the LAN and WAN environments) will greatly improve the overall performance for both the broadcast and multicast message distribution. This is equivalent to adding a function cost on each possible path and integrating this function cost to the computation of the shortest path. In the longer term, we hope that the SHN will become the basic message distribution of the runtime environment within the FT-MPI and Open MPI runtime systems.

References

[1] T. Angskun, G. E. Fagg, G. Bosilca, J. Pjesivac-Grbovic, and J. Dongarra. Scalable fault tolerant protocol for parallel runtime environments. In *Proceedings of the 13th European PVM/MPI User's Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Bonn, Germany, September 2006. Springer-Verlag.

[2] M. Beck, J. J. Dongarra, G. E. Fagg, G. A. Geist, P. Gray, J. Kohl, M. Migliardi, K. Moore, T. Moore, P. Papadopoulous, S. L. Scott, and V. Sunderam. HARNESS: A next generation distributed virtual machine. *Future Generation Computer Systems*, 15(5–6):571–582, 1999.

[3] G. Burns, R. Daoud, and J. Vaigl. LAM: An Open Cluster Environment for MPI. In *Proceedings Supercomputing Symposium*, pages 379–386, 1994.

[4] R. Butler, W. Gropp, and E. L. Lusk. A scalable process-management environment for parallel program. In *Proceedings of the 7th European PVM/MPI User's Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 168–175, London, UK, 2000. Springer-Verlag.

[5] R. H. Castain, T. S. Woodall, D. J. Daniel, J. M. Squyres, B. Barrett, and G. E. Fagg. The open run-time environment (openrte): A transparent multi-cluster environment for high-performance computing. In *Proceedings 12th European PVM/MPI User's Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Sorrento(Naples), Italy, September 2005. Springer-Verlag.

[6] J. J. Dongarra, H. Meuer, and E. Strohmaier. TOP500 supercomputer sites. *Supercomputer*, 13(1):89–120, 1997.

[7] G. E. Fagg, E. Gabriel, G. Bosilca, T. Angskun, Z. Chen, J. Pjesivac-Grbovic, K. London, and J. Dongarra. Extending the mpi specification for process fault tolerance on high performance computing systems. In *Proceedings of the International Supercomputer Conference (ICS) 2004*, Heidelberg, Germany, June 2006. Primeur.

[8] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings 11th European PVM/MPI User's Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface*, pages 97–104, Budapest, Hungary, September 2004. Springer-Verlag.

[9] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high - performance, portable implementation of MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, 1996.

[10] I. Gupta, R. van Renesse, and K. Birman. Scalable fault-tolerant aggregation in large process groups. In *Proceedings of The International Conference on Dependable Systems and Networks (DSN)*, pages 433–442, 2001.

[11] MPI Forum. MPI: A message-passing interface standard. Technical report, 1994.

[12] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content addressable network. Technical Report TR-00-010, Berkeley, CA, 2000.

[13] R. V. Renesse, Y. Minsky, and M. Hayden. A gossip-style failure detection service. Technical Report TR98-1687, 28, 1998.

[14] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. *Lecture Notes in Computer Science*, 2218:329–350, 2001.

[15] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[16] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.