Provided for non-commercial research and education use.
Not for reproduction, distribution or commercial use.

This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

http://www.elsevier.com/copyright

Author's personal copy

Future Generation Computer Systems 26 (2010) 479–485

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Self-healing network for scalable fault-tolerant runtime environments

Thara Angskun *, Graham Fagg, George Bosilca, Jelena Pješivac-Grbović, Jack Dongarra

Department of Computer Science, The University of Tennessee, 1122 Volunteer Blvd. Knoxville, TN 37996, USA

ARTICLE INFO

Article history:
Received 7 December 2006
Received in revised form
17 September 2007
Accepted 29 April 2009
Available online 9 May 2009

Keywords:
Fault tolerance
Routing protocols
Runtime environments
Scalability
Self-healing

ABSTRACT

The number of processors embedded on high performance computing platforms is growing daily to satisfy the user desire for solving larger and more complex problems. 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 and more reliable than the original SFTP routing algorithms.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

Recently, several 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 [1]. 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-tolerant framework is needed. Many large-scale applications are implemented on top of message passing systems for which the de facto standard is the Message Passing Interface (MPI) [2]. MPI implementations require support from 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. 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-ware has performance problems. The MPICH implementation [3] uses a parallel runtime environment called Multi-Purposed Daemon (MPD) [4] to provide 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 [5] of FT-MPI [6], Open RTE [7] of Open MPI [8] and LAM of LAM/MPI [9], do not currently provide scalable or fault-tolerant solutions for their internal communications.

The scalability and fault-tolerance issues have been addressed in several networking areas. However, these approaches cannot 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 [10], Chord [11], Pastry [12] and Tapestry [13] were designed for resource discovery. They are only optimized for unicast messages. Techniques used in sensor or large-scale ad hoc networking based on gossiping (or the epidemic algorithm) [14,15] mainly focus on information aggregation.

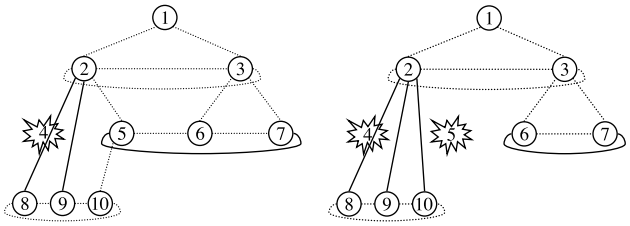
A self-healing network (SHN) that can be used as a basis for constructing a higher level, fault-tolerant parallel runtime environment is described in this article. SHN was designed to support transferring messages across multiple nodes efficiently, while protecting against recursive node or process failures. SHN automatically recovers itself to overcome the orphan situation (the situation where nodes are unreachable because the network is bisectioned).

SHN was built on top of a scalable and fault-tolerant protocol (SFTP) [16] and automatically recovers itself after a failure occurs. The SFTP is based on a *k-ary* sibling tree. The *k-ary* sibling tree topology is a *k-ary* tree, where k is the number of fan-outs ($k \geq 2$), 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 has been formally proven by SPIN [17] to work under both normal and failure modes.

* Corresponding author. Tel.: +1 865 974 6321.
E-mail address: angskun@cs.utk.edu (T. Angskun).

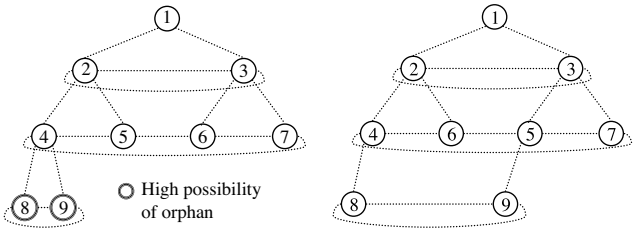
0167-739X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2009.04.001

Author's personal copy



(a) Node 4 dies. (b) Nodes 4 and 5 die.

Fig. 1. SHN after recovery.



(a) Exceptional case. (b) Rearranged nodes.

Fig. 2. Orphan prevention.

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

2. Self-healing network (SHN)

2.1. Overview

The self-healing network (SHN) is designed to support generic runtime environments of MPI implementations. Currently, the integration of SHN in a fault-tolerant implementation of message passing interface called FT-MPI, as well as in the modular MPI implementation called Open MPI, is in progress. The network is designed to support various operations needed by scalable and fault-tolerant MPI runtime environments. The examples of those operations and the details on their usage are described below.

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

Standard I/O redirection Although the MPI standard does not define how an input and an output can be treated, most of the MPI implementation redirects the standard output and the standard error to the user terminal (if not run under the batch scheduling). This operation can 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 development. 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 SHN.

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

2.2. 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 a neighbor disappears, it will send a unicast message to establish the connection with the next neighbor in the ring in the direction of the dead node. This procedure will be continued until the connection with one of the nodes in the ring can be established or until the node identifies itself as the last remaining node in the ring. If two nodes try to establish a connection at the same time, the connection which is initiated by the higher ID will be dropped.

Fig. 1(a) illustrates an example where logical node 4 dies. All neighbors of node 4 will begin to recover the logical topology by re-establishing their connections in the appropriate direction. If node 5 also dies, the same recovery procedure will occur as shown in Fig. 1(b). There is an exception when the number of nodes on the last level (highest depth) of the tree is at most equal to k (k is the fan-out as shown in Fig. 2(a)). In this case, the contact information of the nodes on the last level should be propagated to the grandparent in order to avoid the network bisection if the parent disappears. Alternatively, if there are at least two nodes in the last level, these nodes can be rearranged to reduce the possibility of orphan nodes as shown in the Fig. 2(b). However, the initialization phase of this topology is more complex. The simplest solution to prevent this problem is to change the fan-out for a particular number of nodes such that the number of nodes in the last level of the tree is always more than that of fan-out as shown in Fig. 3. The fan-outs between the lower bound and the upper bound, except those exceptional cases in the Fig. 3, are safe from the high possibility of an orphan problem. The experiments have been conducted on an AMD Athlon™64 Processor 3500 + 2.2 GHz machine with 1 GB of main memory, running on Linux kernel 2.6.15.

3. Routing algorithm in SHN

The SHN routing algorithm is based on the SFTP routing algorithm [16]. The initial system protocol, unicast message protocol,

Author's personal copy

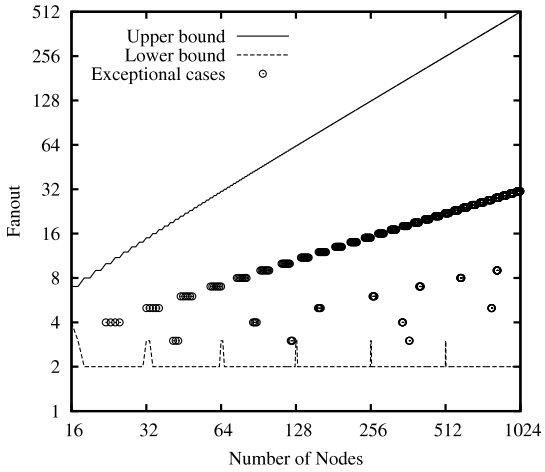


Fig. 3. Safe fan-out.

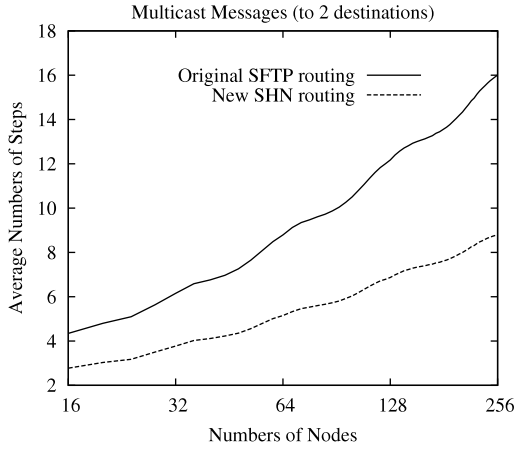


Fig. 5. Multicast result (with 1 failure node).

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 extensions of the SFTP routing algorithm, have been added. Both can be used before (including some node failures) and after recovery of the logical topology. The SHN routing algorithms can be described as follows.

3.1. Multicast messages in SHN

The multicast from any nodes in SHN is the 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 the number of hops from a sender to those destinations (i.e., the largest number of hops first). This routing algorithm works fine if the destination nodes are consecutive or if they are located in the same area of the tree.

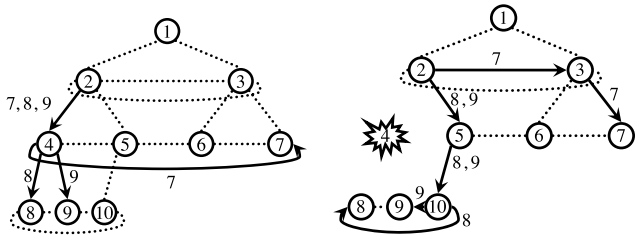
The new multicast routing algorithm in SHN is an enhancement of the SFTP multicast routing algorithm. The multicast message can be split at an intermediate node, if the shortest paths to those destination nodes are not in the same direction from the intermediate node point of view. However, if there are more than one shortest path to a destination, the intermediate node will choose the next hop that can go along with other destinations.

Fig. 4(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 Fig. 4(b). 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, in order to increase the network throughput (by allowing a large number of messages simultaneously into the network).

Fig. 5 confirms that the new multicast routing algorithm is faster than the original algorithm used in the SFTP. The experimental results were obtained from an average number of steps for sending multicast messages to two destinations with a dead node (fan-out = 2). The two 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 were randomly selected.

3.2. Broadcast messages in SHN

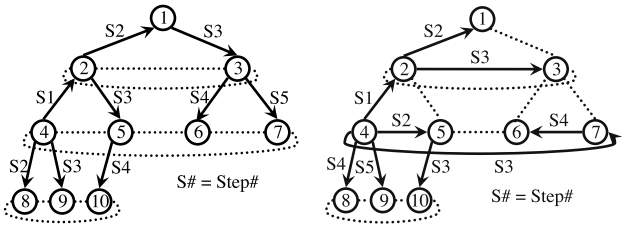
Broadcast from any node routing protocol is an enhancement of broadcast routing in SFTP. In SFTP, the broadcast is done by sending messages to a root of the tree, which will then forward the messages to the rest of the tree. Only the tree topology of SFTP is used to prevent a broadcast storm and duplicate 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 a 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 Fig. 6(a) with node 4 as the root. The second improvement is using a logical spanning tree from the source as shown in Fig. 6(b). When a



(a) Normal circumstances. (b) Failure circumstances.

Fig. 4. Multicast message transmission.

Author's personal copy



(a) Up–down. (b) Spanning tree from source.

Fig. 6. Broadcast message transmission.

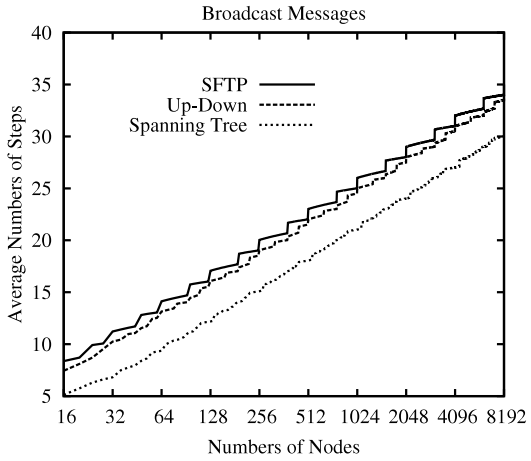


Fig. 7. Broadcast result.

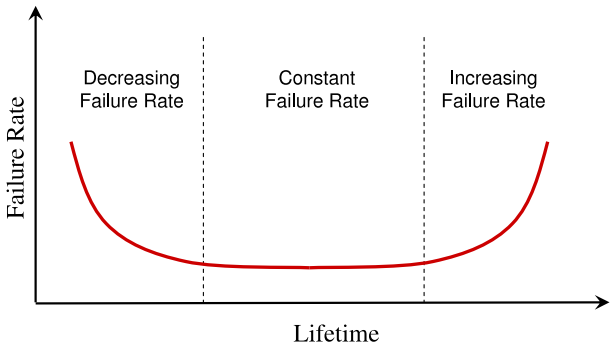


Fig. 8. Bathtub curve.

node receives a broadcast message, it will calculate the next hops using spanning trees from the source node. There are two steps involved in the computation of the next hop. 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 that 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 the parent of the failure node to its children in the spanning tree.

Fig. 7 indicates that the up–down algorithm is marginally faster than the original SFTP, while the new spanning-tree broadcast routing algorithm is significantly 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 (fan-out = 2).

4. Reliability analysis of SHN

The reliability of SHN has been analyzed using the discrete-event simulation technique [18]. The reliability is defined as the ability to maintain an operation over a period of time t, i.e., the reliability $R(t) = Pr$ (the network is operational in $[0, t]$). SHN is "*operational*" if it can successfully deliver messages from any source to the alive destination(s), even when some nodes in the routing path die.

The cumulative distribution function (cdf), $F(t)$ can be defined as

$$F(t) = \int_0^t f(t)\mathrm{d}t$$

where $f(t)$ is the probability density function (pdf) that is associated with the lifetime of the network.

There are several characteristics that are commonly used in reliability analysis. These characteristics can be determined from the pdf and cdf, e.g., reliability function, hazard function and mean time between failures.

The reliability function (or survival function), $R(t)$, is the probability that SHN survives up to time t. It can be defined as

$$R(t) = 1 - F(t).$$

The simulation assumes that there is no failure at the initial time, i.e., $t = 0, R(0) = 1$.

The Hazard function, $h(t)$ is the failure rate of the network. The $h(t)$ is defined by

$$h(t) = \frac{f(t)}{R(t)}.$$

The failure rate in practice has a bathtub shape [19]. The hazard function of SHN is also assumed to change as the bathtub curve, which consists of three phases: decreasing failure rate (burn in), constant failure rate and increasing failure rate (wearing out) as shown in Fig. 8.

The mean time between failure (MTBF) is defined to be the average (or expected) lifetime of the network. The MTBF is defined by

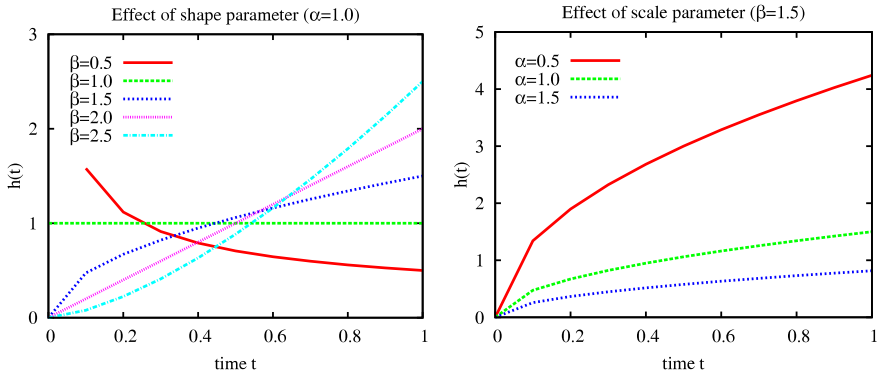
$$MTBF = \int_0^\infty R(t)\mathrm{d}t.$$

The probability density function of SHN is assumed to follow the Weibull distribution [20]. This distribution has the capability to model the bathtub curve. The pdf of the Weibull distribution is given by

$$f(t) = \beta\alpha^{-\beta}t^{\beta-1}\exp\left[-\left(\frac{t}{\alpha}\right)^{\beta}\right]$$

where α is the scale parameter and β is the shape parameter. The associate functions of the Weibull distribution can be summarized

Author's personal copy



(a) β effect. (b) α effect.

Fig. 9. Effects of shape (β) and scale (α).

Table 1
Associate characteristic functions of distributions.

Characteristics	General	Weibull
CDF, $F(t)$	$\int_0^t f(t)\mathrm{d}t$	$1 - \mathrm{e}^{-(\frac{t}{\alpha})^{\beta}}$
Reliability function, $R(t)$	$1 - F(t)$	$\mathrm{e}^{-(\frac{t}{\alpha})^{\beta}}$
Hazard function, $h(t)$	$\frac{f(t)}{R(t)}$	$\beta\alpha^{-\beta}t^{\beta-1}$
MTBF	$\int_0^{\infty} R(t)\mathrm{d}t$	$\alpha\Gamma\left(1+\frac{1}{\beta}\right)$

in Table 1. The Γ denotes a gamma function where $\Gamma(n)$ is defined as

$$\Gamma(n) = \int_0^{\infty} \mathrm{e}^{-x}x^{n-1}\mathrm{d}x.$$

If n is an integer then $\Gamma(n) = (n-1)!$. In the most general form, the 3-parameter form [21] of the Weibull includes an additional waiting time parameter μ (sometimes called a shift or location parameter). The formulas for the 3-parameter Weibull can be easily obtained from these formulas by substituting occurrences of t by $(t-\mu)$.

The Weibull lifetime distribution assumes that the hazard function is time dependent. The hazard function is dependent on the value of β as shown in Fig. 9(a).

- If $\beta < 1$, the hazard function is the decreasing function (infant mortality or burn in), i.e., the older it is, the less likely it fails (the first phase of the bathtub curve).
- If $\beta = 1$, the age has no effect. It is the second phase of the bathtub curve.
- If $\beta > 1$, the hazard function is the increasing function (wearing out), i.e., the older it is, the more likely it is to fail. It is the third phase of the bathtub curve. If $1 < \beta < 2$, the hazard function is concave (increasing at a decreasing rate). On the other hand, the hazard function is convex (increasing at an increasing rate), if $\beta > 2$.

Fig. 9(b) shows the effects of the characteristic life (α) on the failure rate, which affects the spread (scale) of the distribution.

The simulation assumes that MTBF of the network is three years (26,280 h). Several β and its corresponding α parameters have been tested as shown in Table 2. If β equals to 1, the hazard function is time independent, i.e., the network is equally likely to fail at any moment during its lifetime, regardless of how old it is. The failure rate is known to be a constant ($\frac{1}{\alpha}$). This is a special case where Weibull becomes the exponential distribution [22]. Fig. 10 illustrates the effect of β and its corresponding α parameters (as shown in Table 2) in the Weibull lifetime distribution to the percent average of success of multicast operations. It shows that the new multicast routing used in SHN is more reliable than the original SFTP routing for every value of the β parameter. Fig. 11

Table 2
Weibull parameters (MTBF=26,280).

β	α	h(t)	R(t)
0.5	13,140.00	$0.5 \times 13,140^{-0.5}t^{-0.5}$	$\mathrm{e}^{-\left(\frac{t}{13,140}\right)^{0.5}}$
1.0	26,280.00	3.8×10^{-5}	$\mathrm{e}^{-\left(\frac{t}{26,280}\right)}$
1.5	29,111.21	$1.5 \times 29,111,21^{-1.5}t^{0.5}$	$\mathrm{e}^{-\left(\frac{t}{29,111.21}\right)^{1.5}}$
2.0	29,653.80	$2.0 \times 29,653.80^{-2.0}t$	$\mathrm{e}^{-\left(\frac{t}{29,653.80}\right)^{2.0}}$
2.5	29,619.14	$2.5 \times 29,619.14^{-2.5}t^{1.5}$	$\mathrm{e}^{-\left(\frac{t}{29,619.14}\right)^{2.5}}$

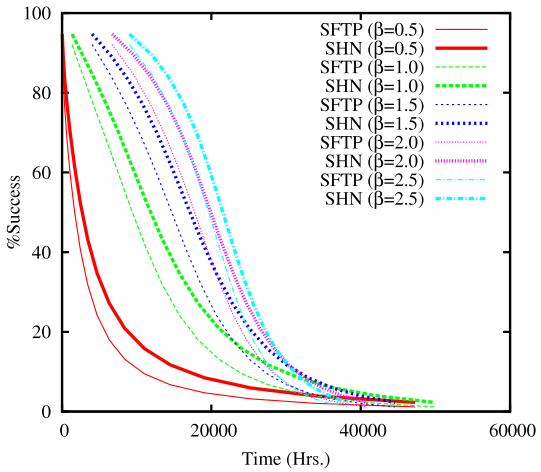


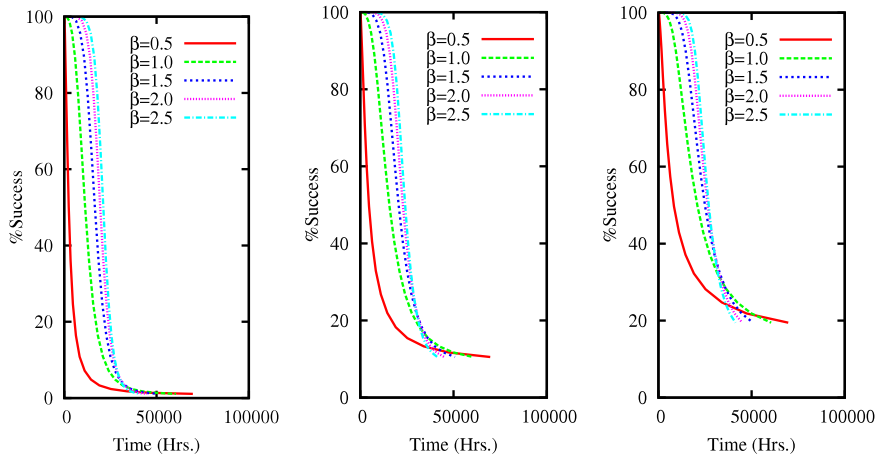
Fig. 10. Weibull on multicast (MTBF=26,280).

illustrates the effect of β and its corresponding α parameters (as shown in Table 2) in the Weibull lifetime distribution to the percent average of success of broadcast operations. It shows that the spanning-tree (from source) broadcast routing algorithm used in SHN is the most reliable routing algorithm when compared with the up–down and the original SFTP routing for all values of the β parameter.

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. SHN is implemented on top of a scalable fault-tolerant protocol (SFTP). Simulated performance results indicate that the new broadcast and multicast routing algorithms of SHN are faster and more reliable than the original SFTP routing algorithms.

Author's personal copy



(a) SFTP. (b) Up–down. (c) Spanning tree.

Fig. 11. Weibull on broadcast (MTBF=26,280).

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 of both the broadcast and multicast message distribution. This is equivalent to adding a function cost on each possible path and integrating this function cost with the computation of the shortest path. In the longer term, we hope that SHN will become the basic message distribution of the runtime environment within the FT-MPI and Open MPI runtime systems.

References

[1] J.J. Dongarra, H. Meuer, E. Strohmaier, TOP500 supercomputer sites, Supercomputer 13 (1) (1997) 89–120.
[2] MPI Forum, MPI: A message-passing interface standard, Tech. Rep., 1994.
[3] W. Gropp, E. Lusk, N. Doss, A. Skjellum, A high-performance, portable implementation of MPI message passing interface standard, Parallel Computing 22 (6) (1996) 789–828.
[4] R. Butler, W. Gropp, E.L. Lusk, A scalable process-management environment for parallel program, in: Recent Advances in PVM and MPI, in: LNCS, vol. 1908, Springer, 2000, pp. 168–175.
[5] G.E. Fagg, J.J. Dongarra, HARNESS fault tolerant MPI design, usage and performance issues, Future Generation Computer Systems 18 (8) (2002) 1127–1142.
[6] 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, V. Sunderam, HARNESS: A next generation distributed virtual machine, Future Generation Computer Systems 15 (5–6) (1999) 571–582.
[7] R.H. Castain, T.S. Woodall, D.J. Daniel, J.M. Squyres, B. Barrett, G.E. Fagg, The open run-time environment (openrte): A transparent multi-cluster environment for high-performance computing, in: Recent Advances in PVM and MPI, in: LNCS, vol. 3666, Springer, 2005, pp. 225–232.
[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, T.S. Woodall, Open MPI: Goals, concept, and design of a next generation MPI implementation, in: Recent Advances in PVM and MPI, in: LNCS, vol. 3241, Springer, 2004, pp. 97–104.
[9] G. Burns, R. Daoud, J. Vaigl, LAM: An open cluster environment for MPI, in: Proceedings Supercomputing Symposium, 1994, pp. 379–386.
[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, A scalable content addressable network, Tech. Rep. TR-00-010, Berkeley, CA, 2000.
[11] I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, in: Proceedings of the 2001 ACM SIGCOMM Conference, 2001, pp. 149–160.
[12] A. Rowstron, P. Druschel, Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems, Lecture Notes in Computer Science 2218 (2001) 329–350.
[13] B.Y. Zhao, J.D. Kubiatowicz, A.D. Joseph, Tapestry: An infrastructure for fault-tolerant wide-area location and routing, Tech. Rep. UCB/CSD-01-1141, UC Berkeley, April 2001.
[14] I. Gupta, R. van Renesse, K. Birman, Scalable fault-tolerant aggregation in large process groups, in: Proceedings of The International Conference on Dependable Systems and Networks, DSN, 2001, pp. 433–442.
[15] R.V. Renesse, Y. Minsky, M. Hayden, A gossip-style failure detection service, Tech. Rep. TR98-1687, 28, 1998.
[16] T. Angskun, G.E. Fagg, G. Bosilca, J.P. vac Grbovic, J. Dongarra, Scalable fault tolerant protocol for parallel runtime environments, in: Recent Advances in PVM and MPI, in: LNCS, vol. 4192, Springer, 2006, pp. 141–149.
[17] G.J. Holzmann, The model checker SPIN, IEEE Transactions on Software Engineering 23 (5) (1997) 279–295.
[18] L.M. Leemis, S.K. Park, Discrete-Event Simulation: A First Course, Prentice Hall, 2005.
[19] M. Xie, C.D. Lai, Reliability analysis using an additive Weibull model with bathtub-shaped failure rate function, Reliability Engineering and System Safety 52 (1) (1995) 87–93.
[20] A. Plait, The Weibull distribution with tables, Industrial Quality Control 19 (5) (1962) 17–26.
[21] G.W. Cran, Moment estimators for the 3-parameter Weibull distribution, IEEE Transactions on Reliability 37 (4) (1988) 360–363.
[22] N. Balakrishnan, A.P. Basu, The Exponential Distribution: Theory, Methods and Applications, Gordon and Breach Publishers, 1995.

Thara Angskun received his Bachelor and Master degree in Computer Engineering from Kasetsart University, Thailand. Currently, he is a Ph.D. student and Graduate Research Assistant at the Innovative Computing Laboratory, Department of Computer Science, University of Tennessee, Knoxville. His major research interests are in parallel and distributed environments, message passing, high performance computing, computer networking, cluster and grid computing. He is a developer of several projects including OpenSCE, KSIX, ACI, CAMETA, Harness/FT-MPI and Open MPI.

Graham Fagg received his B.Sc. in Computer Science and Cybernetics from the University of Reading (UK) in 1991 and a Ph.D. in Computer Science in 1998. Currently he is a Research Associate Professor at the University of Tennessee. His current research interests include distributed scheduling, resource management, performance prediction, profiling, benchmarking, cluster management tools, parallel and distributed IO and high speed networking.

He is currently involved in the development of a number of distributed, metacomputing and GRID middleware systems including SNIPE/2, HARNESS, fault-tolerant MPI (FT-MPI) and Open MPI. He is a member of the IEEE.

George Bosilca is a Senior Research Associate at the Innovative Computing Laboratory (ICL). He is also an Adjunct Professor in Computer Science Department at the University of Tennessee, Knoxville. He received a Ph.D. degree in parallel architectures from the University de Paris XI. He was the main developer of the channel memory subsystem for MPICH-V. Dr. Bosilca is currently working on the Open MPI project.

Author's personal copy

Jelena Pjesivac-Grbovic is a Graduate Research Assistant at the Innovative Computing Laboratory at the University of Tennessee at Knoxville, working toward Ph.D. degree in Computer Science. She received M.S. in Computer Science from the University of Tennessee at Knoxville, and B.S. degrees in Computer Science and Physics from Ramapo College of New Jersey. Her research interests are parallel communication libraries and computer architectures, scientific and grid computing, and modeling of biophysical systems. She is a developer on Harness/FT-MPI project.

Jack Dongarra holds an appointment as University Distinguished Professor of Computer Science in the Computer Science Department at the University of Tennessee and holds the title of Distinguished Research Staff in the Computer Science and Mathematics Division at Oak Ridge National Laboratory (ORNL), and is an Adjunct Professor in the Computer Science Department at Rice University. He specializes in numerical algorithms in linear algebra, parallel computing, use of advanced-computer architectures, programming methodology, and tools for parallel computers. He is a Fellow of the AAAS, ACM, and the IEEE and a member of the National Academy of Engineering.