ISSN 1598-0170 (Print)
ISSN 2287-1136 (Online)
http://www.jksii.or.kr

Design and Cost Analysis for a Fault-Tolerant Distributed Shared Memory System☆

AL-Harbi Fahad Jazi[1] Kangseok kim[2] Jai-Hoon Kim[3*]

ABSTRACT

Algorithms implementing distributed shared memory (DSM) were developed for ensuring consistency. The performance of DSM algorithms is dependent on system and usage parameters. However, ensuring these algorithms to tolerate faults is a problem that needs to be researched. In this study, we proposed fault-tolerant scheme for DSM system and analyzed reliability and fault-tolerant overhead. Using our analysis, we can choose a proper algorithm for DSM on error prone environment.

☞ keyword : Distributed Shared Memory, Fault-Tolerant, Consistency, Reliability, Data Replication

1. INTRODUCTION

Distributed shared memory (DSM) system is a memory architecture that logically implements the shared memory model on a physically distributed memory system. It provides a virtual address space shared among processes on loosely coupled processors. DSM allows high performance multiprocessor systems that have no physically shared memory to be programmed through the shared address space [1, 2, 3, 4].

The advantages offered by DSM include minimizing average access time and maintaining data consistency. In earlier research works, algorithms implementing DSM were developed for ensuring consistency. In [1], the four basic algorithms (central-server, the migration, the read-replication and the full-replication algorithms) implementing DSM were briefly described and compared for implementing DSM by analyzing their performance. Detailed review of each of these algorithms are found in [1,5].

As the number of nodes increases in DSM, the possibility of any node failure also increases. If the failure node has a unique copy of page (or block) in DSM, DSM will lost memory contents and cannot provide services correctly for applications. Thus, ensuring these algorithms to tolerate faults needs to be researched. In this paper we proposed fault-tolerant scheme on each of the aforementioned basic algorithms and analyzed improvement reliability and fault-tolerant overhead. Parameters are described in Figure 6.

Related works are described in chapter 2, our fault-tolerant algorithms are proposed in chapter 3, we analyze fault-tolerant overhead and reliability in chapter 4, and conclusion is in chapter 5.

2. RELATED WORK

Reference [1] described and analyzed all four algorithms ensuring consistency in distributed shared memory. The four basic algorithms are central-server algorithm, migration algorithm, read-replication algorithm and full-replication algorithm. Some of these four basic distributed memory algorithms can be replicated and non-replicated, on the other hand able to be non-migration and migration as shown in Figure 1. Reference [1] identified the parameters in data access costs and investigated the application behaviors which affect on the performance of the data consistency algorithm.

[1] Computer Engineering, Ajou Univ., Suwon, Korea (443-749)
[2,3] Cyber Security, Ajou Univ., Suwon, Korea.
* Corresponding author (jaikim@ajou.ac.kr)
[Received 3 December 2016, Reviewed 13 January 2016(R2 15 April 2016, R3 2 June 2016), Accepted 20 June 2016]
☆ Early short version of this paper[11] was presented in Fall Conference of KSII, 2015.10.
☆ This paper is based on the MS thesis (AL-Harbi Fahad Jazi, Cost Analysis for Fault-Tolerant Distributed Shared Memory System, Feb. 2016, Ajou University).
☆ This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education (2015R1D1A1A01060034)

Also reference [1] demonstrates that DSM algorithms show different consistency overhead for different parameters. They explained the main role of each basic algorithm and compared performance between algorithms. Central-server algorithm maintains the only copy of the shared data and provides all accesses from other nodes to shared data. Second algorithm is migration algorithm using single-reader/single-writer protocol [1] and the data is available by migrating to the site where it is accessed. Third algorithm is read replication algorithm. The algorithm maintains consistency because a read access always returns the value of the most recent write to the same location. Read algorithm repeats consistent because the read access at all times returns the worth of the fresh writing to the same location.

Fourth basic is full replication algorithm that uses a multiple readers / multiple writers (MRMW) protocol. Even when data blocks are written, full replication allows it to be replicated. One disadvantage of the algorithms is that, in the same block and at any given time, only the threads on one host able to access data contained [1].

Many fault tolerance algorithms of DSM were proposed to improve system availability. Reference [18] proposes efficient replication for distributed fault tolerant memory. It is for in-memory applications and replicates updates to a peer node for a fault tolerance. Experimental results are described without a cost analysis model. Replication scheme for fault tolerant transactional systems is proposed [19]. Shared objects are replicated across all nodes (full-replication) and replica consistency is ensured by an optimistic atomic broadcast. Authors implemented an optimized ordering layer and a concurrency control mechanism which exploit multiple nodes for replicating the transactional state. Reference [20] presents software diversity to support security and reliability. Two patterns describing commonly used practices of realizing automated software diversity are described.

2.1 ALGORITHMS OF DSM

All four algorithms they described in their paper [1] ensure consistency in distributed shared memory. However their performance is sensitive to the data access costs and application conducts that have considerable bearings on the performance of the algorithms. Also they compared the relative merits of the algorithms based on some simple analyses, and performed to identify the relationship between access patterns of applications and the shared memory algorithms. Using the analysis, we can expect the performance of their algorithms.

	Non-replicated	Non-replicated	Replicated	Migrating
Central	O	O		
Migration	O			O
Read-replication			O	O
Full-replication		O	O	

Figure 1. Four basic distributed memory algorithms [1]

2.2 THE MAIN CONTRIBUTION OF THE PAPER AND DETAIL EXPLANTION

In this paper our goal is to provide basic performance analysis for DSM including fault-tolerance. Fault-tolerance describes a computer system or component designed so that, in the event that a component fails, a backup component or procedure can immediately take its place with no loss of service [1].

Fault tolerance can be provided with software, or embedded in hardware, or provided by some combination. In the software implementation, the operating system provides an interface that allows a programmer to "checkpoint" critical data at pre-determined points within a transaction. We measure the extent of the possibility of the fault-tolerance for the basic four algorithms for distributed shared memory, and compare the performance of fault tolerance schemes for each basic algorithms.

Although references [1,10] is old classic paper, it categorized consistency algorithms of the distributed shared memory (DSM) systems. Most recent researches related to DSM systems [12-17] belong to one of four algorithm as shown in Table 1. Thus our analysis for fault tolerant distributed shared memory systems also can be applied for recent DSM researches.

Table 1. Distributed shared memory algorithms

	Recent related works	remarks
Central	Software DSM with Transactional Coherence [12]	Lease based protocol
	Region-Based Prefetch Techniques for Software DSM Systems[13]	prefetch
Read-replication	[13]	prefetch
Read-replication Full-replication	A Distributed Real-Time Operating System with DSM for Embedded Control Systems [16]	MRSW MRMW
	Software based DSM model using Shared variables between Multiprocessors[17]	memory controller
	Samhita, software DSM on multicore architectures [14]	multiple-writer protocol
Full-replication	Delay Tolerant Lazy Release Consistency for Distributed Shared Memory in Opportunistic Networks [15]	Delay Tolerant
	[16]	MRSW MRMW

3. FAULT-TOLERANT ALGORITHMS OF DSM

In this chapter, we analyze fault-tolerant overhead and reliability of fault-tolerance DSM schemes. Fault-tolerance describes a computer system or component designed so that, in the event of a component fails, a backup component or procedure can immediately take its place with no loss of service. Fault tolerance can be provided with software, embedded in hardware, or provided by some combination. In the software implementation, the operating system provides an interface that allows a programmer to "checkpoint" critical data at pre-determined points within a transaction.

We propose and analyze the fault-tolerance scheme for the basic algorithms of distributed shared memory by maintaining at least two copies for each data block, and compare the reliability and fault tolerance overhead. We assume the parameters shown in Figure 6 based on reference [1].

Reference [10] also presented fault-tolerant distributed shared memory algorithms. However, our algorithms and analysis are different from [10] as shown in Table 2. Reference [10] requires two additional parameters while our analysis does not require additional parameters. We also analyze reliability besides cost of fault-tolerant schemes.

Table 2. Algorithms and analysis comparisons.

	Reference [10]	Our analysis
Parameters	Additional two parameters are necessary. m: The possibility of a page transfer request being made for a dirty block. d: The average number of dirty blocks a host has when servicing a page transfer request of a dirty page owner by this host.	No additional parameters is necessary.
Reliability	No analysis for reliability	Reliability analysis (probability of system failure)
Migration algorithms	Passive backup: Data update is lost at primary active node failure.	Active back up: Data update is always propagated to an active backup node. Active backup node can be primary node or forward a block at the next migration.

3.1 THE CENTRAL-SERVER ALGORITHM

In this type of algorithm, the central-server maintains all the shared data and services the read requests from clients by returning the data items to them. The central server updates the data on write requests by clients and returns acknowledgements. A request is retransmitted after each timeout period if acknowledgment fails, as depicted in Figure 2. The probability of accessing a remote data item is 1-1/S, in which case 4 packet events (send data request, receive data request, send response, and receive response) are necessary for the access. Replicating the data contained by the main server to a backup server helps this algorithm to tolerate faults. The main server does not reply to the client until it receives the acknowledgement from the backup. At write operation, a cost of (1/(1+r))*2p is required in addition to the cost of the original algorithm (send backup and receive backup on a write operation). For an error analysis, the probability of system error on an original algorithm is e as a copy only exists on the

central server, while a fault tolerant algorithm is as two copies exist on a central server and a backup node.

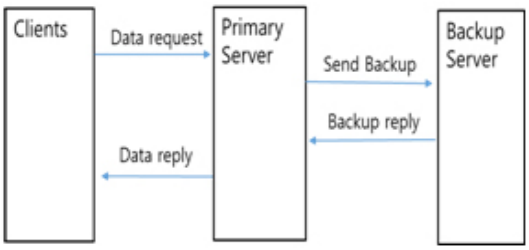

Figure 2. The central-server algorithm ([1,10])

3.2 THE MIGRATION ALGORITHM

In this algorithm every data access request is forwarded to the site where it is accessed, and data in the migration is shipped to the location of the data access request, allowing subsequent accesses to the data to be performed locally. Data is migrated between hosts in blocks in order to facilitate the management of the data. If a block is migrated from the first host to another host, the copy at the first host is maintained, but marked invalid the previous backup copy, as depicted in Figure 3.

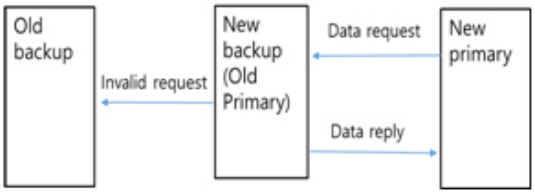

Figure 3. The migration algorithm ([1,10])

The cost of accessing a data item in that case is equal to the cost of bringing the data block containing the data item to the local site, which includes a total of one block transfer (2P for send and receive the a block) and four packet events (4p) distributed across the local, manager and remote host (send and receive the two data request packets). For a fault tolerant algorithm, additional 2p is required on a block access fault to delete the previous back copy. At write operation, (1/(1+r))*2p is also required for backup (send and receive the backup packet). For an error analysis, a migration algorithm is the same as a central-server algorithm.

3.3 THE READ-REPLICATION ALGORITHM

It extends the migration algorithm by replicating data blocks and allowing multiple nodes to have read access or one node to have read-write access. The remote access cost of read replication is similar to the migration algorithm (2P+4p). In the case of a write fault that occurs with a probability of 1/(r+1), a multicast invalidation packet must be handled by all S sites except a backup (Sp/(1+r)). (See Figure 4). Fault tolerant algorithm requires additional cost on a write even without a write fault for backup of cost 2p. For an error analysis, upper bound of system error of original algorithm is $f'(2p + 4p + \frac{Sp}{r+1})$ one as one copy exists on a write while multiple copies exist on a read. However, system error of a fault tolerant algorithm is e^2 as it always keeps a backup.

Figure 4. The write operation case in read-replication ([1,10])

3.4 THE FULL-REPLICATION

It is an extension of read-replication algorithm in that it allows multiple nodes to have both read and write access to shared data blocks. It uses a gap-free sequencer to maintain consistency of data for multiple writers. In this kind of algorithms, the probability of a remote access is equal to the probability of a write access 1/(1+r) in which (S+2)*p cost is required to send a update packet to sequencer and forward the packet to all the other nodes.

The cost of this write is the message from the local site to the sequencer (2 packet events), followed by a multicast update message to all other sites (S packet events). As this algorithm replicates on each host, the fault tolerant scheme does not need further replication.

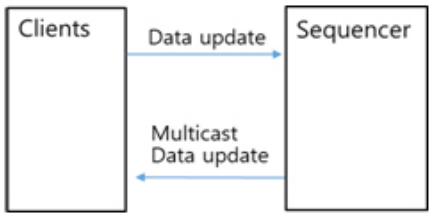

Figure 5. The full replication algorithm [1,10]

4. PERFORMANCE COMPARISON OF THE ORIGINAL AND FAULT-TOLERANT DSM ALGORITHM:

4.1 THE COST ANALYSIS

In this section, we analyze the average costs of accessing shared data in the original and the fault tolerant DSM algorithms by parametric analysis of cost models for maintaining data consistency. The parameters as shown in Figure 6 are considered for characterizing the basic costs of accessing shared data, application behavior.

Using the basic parameters in Figure 6 and some assumptions, the average access costs of the original algorithms are stated in Table 3 of column two [1]. In our study, we have stated the error formula of the original algorithms, the cost formula and error formula of the proposed fault tolerant algorisms in columns 4, 5 and 6 respectively.

e: Probability of node failure.
p: The cost of a packet event, i.e., the processing cost of sending or receiving a short packet.
P: The cost of sending or receiving a data block. This is similar to *p*, except that *P* is typically significantly higher.
S: The number of sites involving in DSM.
r: Read/write ratio, i.e., there is one write operation for every r reads on average.
f: Probability of an access fault on a non-replicated data block used in the migration algorithm.
f': Probability of an access fault on replicated data blocks used in the read-replication algorithm.

Figure 6. Parameters that characterize the basic costs of accessing shared data and fault tolerance overhead [1,10]

Table 3. The cost formula and error formula of the original and fault tolerant algorithms

DSM Algorithm	Original Algorithm		Fault Tolerant Algorithm		
	Cost [1]	Error	Cost		Our Error
			Reference [10]	Our Analysis	
Central-Server	$\left(1-\frac{1}{S}\right)4p$	e	$\frac{r}{r+1}\left(1-\frac{1}{S}\right)4p + \frac{1}{r+1}\left(1-\frac{1}{S}\right)8p$	$\left(1-\frac{1}{S}\right)4p + \frac{1}{r+1}2p$	e^2
Migration	$f(2p+4p)$	e	$f[2(md+(1-m))P + 8p]$	$f(2P+6p) + \frac{1}{r+1}2p$	e^2
Read-Rep	$f'(2p+4p + \frac{Sp}{r+1})$	$\le \frac{r}{1+r}e^2 + \frac{1}{1+r}e$	$f'[2(md+(1-m))p + 8p + \frac{Sp}{r+1}]$	$f'(2P+4p+\frac{Sp}{r+1}) + (1-f')\frac{1}{r+1}2p$	$\le e^2$
Full-Rep	$\frac{1}{r+1}(S+2)p$	e^S	$\frac{1}{r+1}(S+2)p$ without backup $\frac{1}{r+1}(S+2)p$ with backup	$\frac{1}{r+1}(S+2)p$	e^S

Comparing the cost formulas of the original and fault tolerant DSM algorithms, we observed that the average accessing costs of the fault tolerant versions, central-server, migration, and read-replication DSM algorithms are higher than that of their corresponding original versions.

4.2 THE COST COMPARISONS

For the central algorithm, the cost of both versions of the original algorithms and fault tolerant cost by varying S (number of sites) are shown in Figure 7. The cost of fault tolerant scheme is higher than that of original scheme to maintain a backup copy.

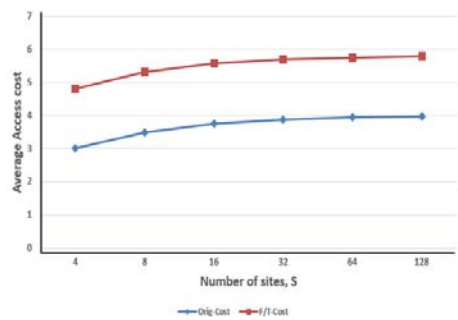

Figure 7. Cost as a function of s in central algorithm (r=0.1 and p=1)

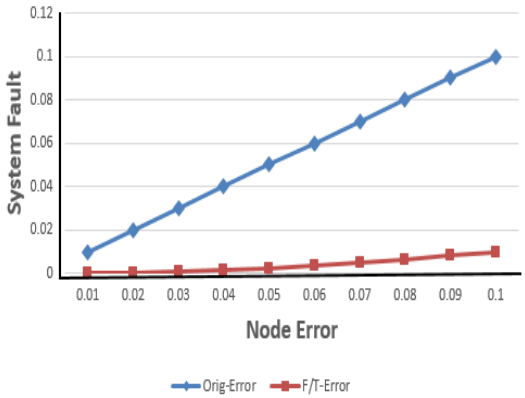

Figure 8. Probability of error in the original and F/T central algorithm

Central-algorithms operation case is shown in Figure 8. When fault tolerant scheme is compared with original version, the probability of system failures decreases. Therefore system reliability is improved, because the probability of system failures decreases due to backup server. For an example, system failure rate of original scheme is 0.1 when node error rate is 0.1, while fault-tolerant scheme is 0.01 as a backup node as well as the central node keeps the copy.

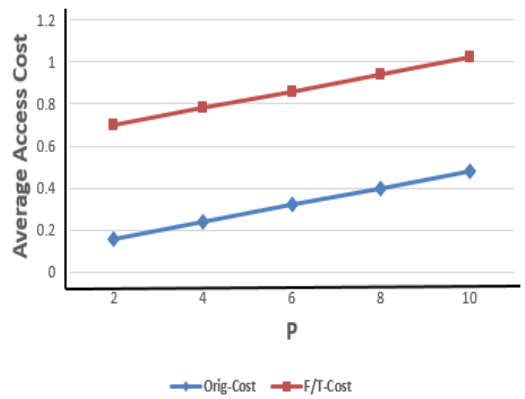

Figure 9. Cost as a function of P in migration algorithm (r=3, p=1 and f=0.02)

Average accessing cost of the fault tolerant migration algorithm (Figure 9) is somewhat higher than original version to maintain a backup.

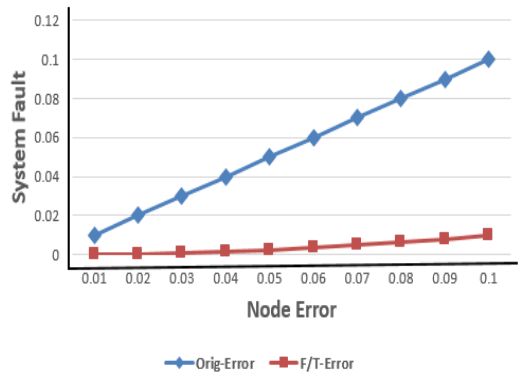

Figure 10. comparison of error in the original and F/T migration algorithm

The system failures in Figure 10 decreases almost similarly with fault tolerant central algorithm as shown in Figure 8. Both algorithms maintain a backup node to increase reliability.

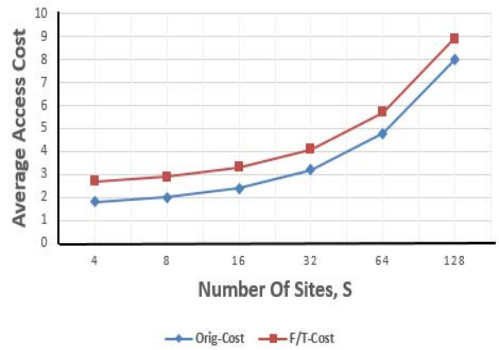

Figure 11. Average access cost per number of hosts in read-replication algorithm (r=3, P=4, f'=0.1 and p=2)

Figure 11 demonstrates our observation graphically that the average accessing cost of the fault tolerant read-replication DSM algorithm is a little bit higher than that of its original version. Fault tolerant algorithm requires an additional cost when one node has a unique copy of data.

Also the results shown in Figure 12 illustrates the impact of the probability of node failure on error on both versions of the read-replication algorithm. As the probability of node failures increases, compared to the original version, the amount of system failure occurred in the fault tolerant read-replication DSM algorithm is lower, thereby fault tolerant scheme

improves system reliability. The difference of reliability of two algorithms will increases as read ratio (r) decreases, because read-replication algorithm may have multiple copies on high read ratio while the fault tolerant read-replication algorithm always has at least two copies.

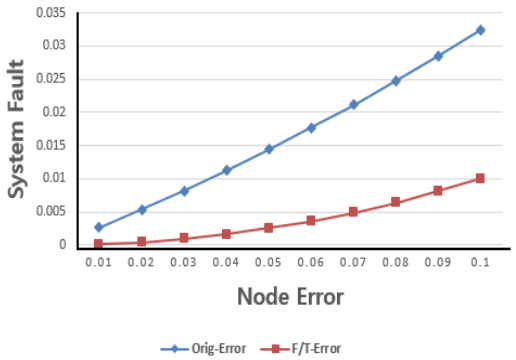

Figure 12. Probability of node failure on error (upper limit) in the read-replication algorithm (r =3)

Fault tolerant probability of fault tolerant algorithm is less than 0.01 when node error is under 0.1 by maintaining at least two copies as seen in Figure 12.

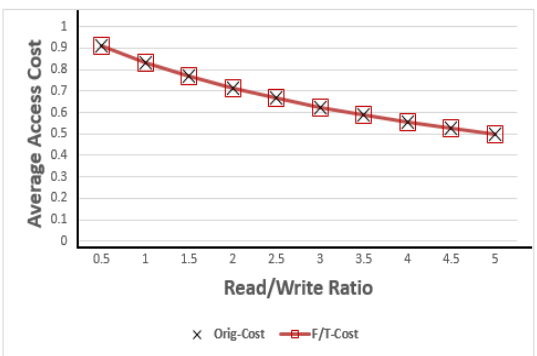

Figure 13. Average access cost in original and F/T Full replication algorithm (S=8, p=0.1)

In Figure 13, the costs of full replication of original version and fault tolerant scheme are the same as both algorithms always keep the copy on all nodes. The average access cost decreases as the read/write ratio increases.

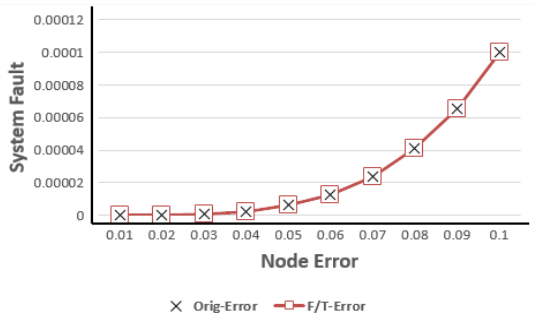

Figure 14. Comparison of error in the original and F/T Full replication algorithm (S=4)

The probability of system failures in full replication increases as the node error increasing as shown in Figure 14.

5. CONCLUSION

In this paper, we amended each of the four basic DSM algorithms [1] to tolerate failures and analyzed their performance. Based on the parameters that characterize the costs of accessing shared data, results showed that we can reduce error rate by minimal overhead of fault-tolerant scheme.

References

[1] Michael Stumn and Songnian Zhou, “Algorithms Implementing Distributed Shared Memory”, IEEE Computer, pp. 54-64, May 1990. http://dx.doi.org/10.1109/2.53355

[2] John L. Hennessy and David A Patterson, Computer Architecture: a quantitative approach, Fourth Edition, Morgan Kaufmann Publishers, 2007. https://app.knovel.com/web/toc.v/cid:kpCAAQAE02/viewerType:toc/root_slug:computer-architecture-a

[3] Larry Brown and Jie Wu, “Dynamic Snooping in a Fault_Tolerant Distributed Shared Memory”, Proc. of International Conference on Distributed Computing Systems, pp. 218-226, Jun 1994. http://dx.doi.org/ 10.1109/ICDCS.1994.302415

[4] Bill Nitzberg and Virginia Lo, “Distributed Shared Memory: A Survey of Issues and Algorithms,” IEEE Computer, pp. 54-64, May 1991.

http://dx.doi.org/ 10.1109/2.84877

[5] Jelica Protic, Milo Tomasevic, and Veljko Milutinovic, "Distributed Shared Memory: Concepts and Systems," IEEE Computer pp. 63-79, June 1996. http://dx.doi.org/10.1109/88.494605

[6] Krishna Kavi and Hyong-Shik Kim, "Shared Memory and Distributed Shared Memory Systems: A Survey," IEEE System Sciences, pp. 74 – 84, 3-6 Jan 1995.

[7] Arun K. Somani and Nitin H. Vaidya, "Understanding Fault Tolerance and Reliability," IEEE Computer, pp. 45-50, April 1997. http://dx.doi.org/ 10.1109/MC.1997.585153

[8] Bill Nitzberg and Virginia Lo, "Distributed Shared Memory: A Survey of Issues and Algorithms," IEEE Computer, pp. 52-60, August 1991. http://dx.doi.org/ 10.1109//2.84877

[9] Kjetil Nrvag, "An Introduction to Fault-Tolerant Systems," IDI Technical Report 6/99, ISSN 0802-6394, pp.3-19, July 2000.

[10] Michael Stumn and Songnian Zhou, "Fault Tolerant Distributed Shared Memory Algorithms," Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing, pp. 719-724, December 1990. http://dx.doi.org/ 1109/SPDP.1990.143633

[11] AL-Harbi Fahad Jazi A. and Jai-Hoon Kim, "Cost Analysis for Fault-Tolerant Distributed Shared Memory System," Proc. of Fall Conference of KSII, Oct. 2015. http://www.dbpia.co.kr/Journal/ArticleDetail/NODE06554560

[12] Michele Di Santo et al., "Software Distributed Shared Memory with Transactional Coherence," Proc. of 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, pp.175-179, 2010. http://dx.doi.org/10.1109/PDP.2010.28

[13] Jie Cai et al., "Region-Based Prefetch Techniques for Software Distributed Shared Memory Systems," Proc. of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, pp. 113-122, 2010. http://dx.doi.org/ 10.1109/CCGRID.2010.16

[14] Bharath Ramesh et al., "Is It Time To Rethink Distributed Shared Memory Systems?" IEEE 17th International Conference on Parallel and Distributed Systems, 212-219, 2011. http://dx.doi.org/10.1109/ICPADS.2011.75

[15] Chance Eary and Mohan Kumar, "Delay Tolerant Lazy Release Consistency for Distributed Shared Memory in Opportunistic Networks," Proc. of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp.1-6, 2012. http://dx.doi.org 10.1109/WoWMoM.2012.6263745

[16] Takahiro Chiba et al., "A Distributed Real-Time Operating System with Distributed Shared Memory for Embedded Control Systems," Proc. of IEEE 11th International Conference on Dependable, Autonomic and Secure Computing, pp. 248-255, 2013. http://dx.doi.org/10.1109/DASC.2013.71

[17] Hemant D. Vasava and Jagdish M. Rathod, "Software based Distributed Shared Memory (DSM) model using Shared variables between Multiprocessors," Proc. of IEEE ICCSP, pp. 1431-1435, 2015. http://dx.doi.org/10.1109/ICCSP.2015.7322749

[18] Deepavali Bhagwat et al, "Efficient Replication for Distributed Fault Tolerant Memory," Proc. of ACM SYSTOR, 2015. http://dx.doi.org/ 10.1145/2757667.2757686

[19] Sachin Hirve et al., "SMASH: Speculative State Machine Replication in Transactional Systems," Proc. of ACM Middleware, 2013. http://dx.doi.org/ 10.1145/2541614.2541630

[20] ANDREA HÖLLER et al., "Patterns for Automated Software Diversity to Support Security and Reliability," Proc. of EuroPloP, 2015. http://dx.doi.org/ 10.1109/IEEESTD.1990.101064

◐ Authors ◑

AL-Harbi Fahad Jazi A.

2008 Technical and Vocational Training Corporation (TVTC), The Associate Degree of College of Technology from the department of Computer Technology in the field of Technical Support, 2013 Bachelor Degree in Computer Engineering from Korea Polytechnic University, and 2016 Master Degree in Computer Engineering from Korea Ajou University (South Korea).

Kangseok Kim

received Ph.D. in Computer Science from Indiana University at Bloomington, IN, USA. He is currently a research professor of the Cyber Security department at Ajou University, Suwon, Korea. His main research interests include ubiquitous computing, cloud computing, pervasive collaborative applications with smart phones, IoT/Smartphone grid, bioinformatics and applied security in big data

Jai-Hoon Kim

received the B.S. degree in Control and Instrumentation Engineering, Seoul National University, Seoul, South Korea, in 1984, M.S. degree in Computer Science, Indiana University, USA, in 1993, and his Ph.D. degree in Computer Science, Texas A&M University, USA, in 1997. He is currently a professor of the Cyber Security department at Ajou University, South Korea. His research interests include distributed systems, cyber-physical systems, and mobile computing.