A Hybrid Fault Tolerance System for Distributed Environment using Check Point Mechanism and Replication

[1]S. Veerapandi, [2]Dr. K. Alagarsamy
[1]*Research Scholar,* [2]*Associate Professor Rtd*
[1, 2]*School of information Technology,*
[1, 2]*Madurai Kamaraj University*

Abstract

  Managing the distributed environment against the failures plays an important role nowadays. There are so many techniques evolved so far and each have their own merit and demerit. The efficiency of the algorithm depends on how much replication is done and upto what extent the fault tolerance has been achieved. We have here proposed a new method which uses both check point as well as the replication to ensure consistency in the distributed environment. Our method is also easy to implement.

Keywords :*FTPA, PLR, GiFT*

I. INTRODUCTION

  A distributed environment is a collection of multiple independent computers that seems to its users as a single unified system. Distributed environment uses multiple geographically distant computers and solves big and complex task very efficiently. Computing power of idle system is utilized by distributed environment, which offer a better price and performance than mainframes and Computing power can be added in small increments in distributed systems, which leads the way to incremental growth.

  Distributed systems allow many users access to a common computing resource thus provides resource sharing. Thus it allows many users to share expensive peripherals. It makes human-to-human communication easier. Examples of such distributed computing are online railway reservation system, air traffic control, internet banking etc. As the size of distributed system is increasing day by day chances of faults are increasing.

  Mean time to failure is decreasing with increase in size and complexity of distributed system. In large and dynamic distributed system millions of computing devices are working altogether and these millions of computing device are prone to failures. Failures of processors, disks, memory, power, and link failure are some examples of failures. Faults are inevitable in larger and dynamic distributed system. Faults may stop or halt execution of distributed system. It disturbs normal execution and may turn system execution in wrong direction.

  In air traffic control, distributed disaster system, railways reservation system, internet banking a single fault may lead to huge loss of money and even human lives. In such a situation, inclusion of fault tolerance technique is essential. Fault Tolerance Techniques enable systems to perform tasks in the presence of faults [1]. There are high chances that more than one fault may occur in distributed system. For example more than one process may fail one by one or at a one time. Likewise more than one process may also fails in same manner. In such a situation simple fault a tolerance technique having capability to handle one fault are not suitable and does not solve the purpose. Such single fault tolerance algorithm fails to recover and restore the normal execution of dynamic distributed system in case of multiple faults.

  Handling more than one fault is a distinctive feature which is achieved using multiple fault tolerance technique. A multiple fault technique capable of tolerating n number of concurrent faults is known as kfaults tolerance technique. In some situation chain of faults occurs in such a way that the faults occurs when recovery of first is on progress and incomplete. Handling such types of multiple faults situation required a systematic approach and improved algorithms of multiple failure detection and recovery from multiple faults. Performance, scalability, robust, transparency, efficiency and consistency etc are some important issue with multiple fault tolerance implementation of distributed system.

  In case of real time system multiple fault tolerance mechanism must provide performance in both the situation; fault free and faulty situation. Multiple node failures , process failure and failure of another node when recovery of failure of earlier node are some considered as a multiple faults occurrence in distributed system. To enhance the performance of multiple fault tolerance various overheads associate with every technique are required to minimize with improved algorithms. At the same time critical factor responsible for low performance need to be identified and ways need to explore to address these critical factors so that

multiple fault capability can be improved with performance.

II. VARIOUS FAULT TOLERANCE TECHNIQUES

A. Replication Based Fault Tolerance Technique:

Replication based technique is one of the popular fault tolerance techniques [1]-[3]. A replica means multiple copies. Replication is a process of maintaining different copies of a data item or object. In replication techniques, request from client is forwarded to one of replica among a set of replicas. This technique is used for request that do not modify state of service. Replication adds redundancy in system. In this way failure of some nodes will not result in failure in system and thus faulttolerance is achieved as shown in fig 1.
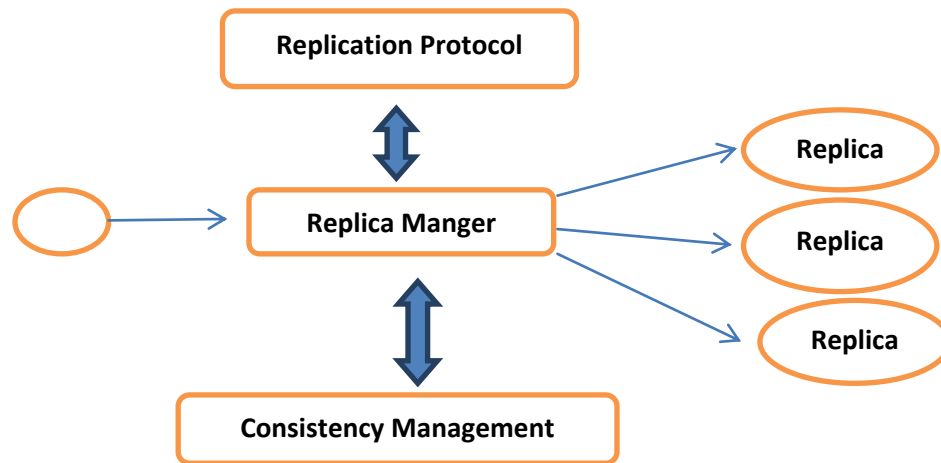

Fig1: Replication System

Replication protocol can be described using five genericphases. These phases are client agreement, contact, execution, server coordination,coordination and client response.Consistencies among replica, replica management, replica ondemand, degree of replica etc. are some important issues inreplication based fault tolerance technique. Major issuesrelated to replication based techniques are consistency, degreeof replica, replica on demand etc.

B. Consistency

Consistency among replicas is a major issue. Multiple copiesof same entity causes problem of consistency due to update ofany copy by one of the user. A replication protocol mustensure the consistency among all replicas of the same object.

Consistency is ensured by some criterion. Many consistencycriteria have been defined in the literature; linearizability [2],sequential consistency and casual consistency [3] etc. In allabove cases, an operation is performed on the most recentstate of the object. However consistency criteria differ in thedefinition of the most recent state. Primary-backup replicationtechnique and active replication technique ensure consistencyby linearizability. Both linearizability and sequentialconsistency define strong consistency criterion, whereascausal consistency defines a weak consistency criterion.

Sequential consistency informally states that a multiprocessorprogram executes correctly if its result could have beenproduced by executing that program on single processorsystem. In order to have consistency an efficient strategy isrequired. Passive strategy and active strategy are mainstrategies. In a passive replication, only one primary executerequests and multicasts state changes to all replicas.

This scheme avoids redundant computation of requests. It copeswith non-deterministic service behavior. In active replica,client request is multicasts to all replicas. This means allreplicas execute the request individually. In this way activereplica takes less network resources than sending update.

Active replica response to a fault is faster than passive.However, replica consistency usually requires deterministicreplica behavior [4].Researcher proposed an algorithm thatuses both active and passive strategies to implementoptimistic replication protocol [5]. Researcher also proposed asimple protocol by combining the token with cache. Thisgives benefits of token as well as cache [6].There is still needof more simple, adaptive and practical replication protocolwith adequate and sufficient ensured consistency.

C. Degree of Replica

Number of replica is known as a degree of replication. Inorder to replicate an object a replication protocol is used.Primary-backup replication [27], voting [23], and primary-perpartition protocol [24] are some of the replication protocol. Areplication protocol must be practical and simple. Theprotocol must provide rigorously-proven yet simply-statedconsistency guarantee with a reasonable performance. Niobeis such protocol purposed by researcher [25].Number ofreplicas must be sufficient. Large numbers of replicas willincrease the cost of maintaining the consistency. Less numberof replicas will affect the performance, scalability andmultiple fault tolerance capability. Therefore, reasonablenumber replicas must be estimate as per system configurationand load. Researcher proposed adaptive replicas creationalgorithm [26].There is further research scope to developimproved algorithm to maintain a rational replica number.

Replica on demand is a feature that can be implemented tomake more adaptive, flexible and dynamic. There is researchscope to further improve protocols to achieve replicationefficiently. There are some crucial requirements withreplication protocol. These crucial requirements are supportfor a flexible number of replicas, strict consistency in thepresence of network, disk, and machine failures and efficientcommon case read and write operations without requiringpotentially expensive two or three-phase commit protocols.

D. Process Level Redundancy

This technique is mainly used as a fault tolerance for transientfaults. A transient fault will eventually disappear without anyapparent intervention. Transient faults are less severe but hardto diagnose and handle. It is caused by temporary malfunctionof some system component. Some environmental interferencealso causes transient fault or faults. Transient faults areemerging as a critical concern in the reliability of distributedsystem. Hardware based fault tolerance is very costly hencesoftware based fault tolerance is used to handle transientfaults.

Process-level redundancy (PLR) is a software basedtechnique for transient fault tolerance, which leveragesmultiple cores for low overhead. PLR creates a set ofredundant processes per application process as shown in fig 2.It systematically compares the processes to guarantee correctexecution. Redundancy at the process level allows theoperating system to schedule freely the processes across allavailable hardware resources. PLR uses a software-centricapproach to transient fault tolerance, which shifts the focusfrom ensuring correct hardware execution to ensuring correctsoftware execution.
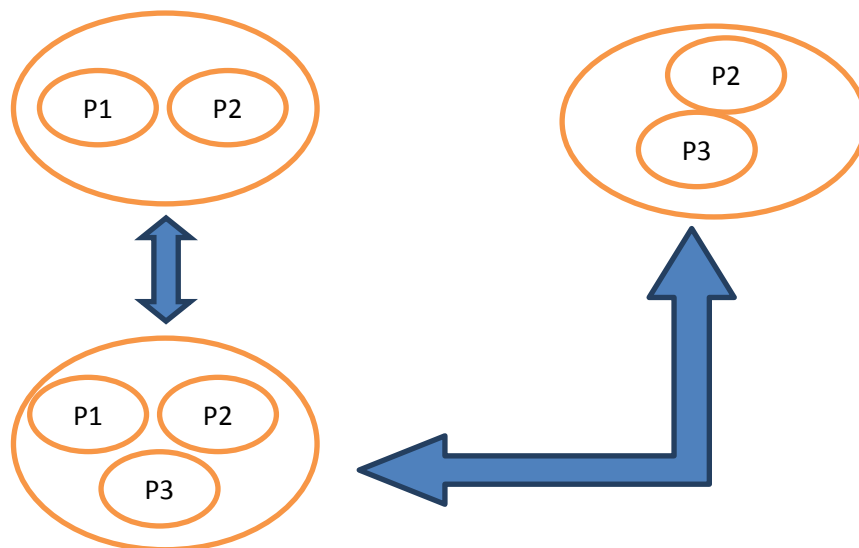

Fig2: Process Level Redundancy

As a result, many benign faults that do not propagate to affect program correctness can be safely ignored. PLR provides improved performance over existing software transient fault tolerance techniques with a 16.9 percent overhead for fault detection [7].However; PLR does not provide an adaptive and configurable fault tolerance on distributed systems. Further there is research scope to make PLR to support simultaneous faults by simply scaling the number of redundant processes and the majority vote logic. Future work remains incharacterizing fault propagation, exploring methods for bounding the time in which

faults remain undetected and performance improvement by minimizing the various overheads.

E. Check pointing and Roll Back

Checkpoint with rollback-recovery is a well-known technique. Checkpoint is an operation which stores the current state of computation in stable storage. Checkpoints are established during the normal execution of a program periodically. This information is saved on a stable storage so that it can be used in case of node failures. The information includes the process state, its environment, the value of registers, etc. When an error is detected, the process is roll backed to the last saved state [8]. Fig 3shown below gives an idea about this technique.
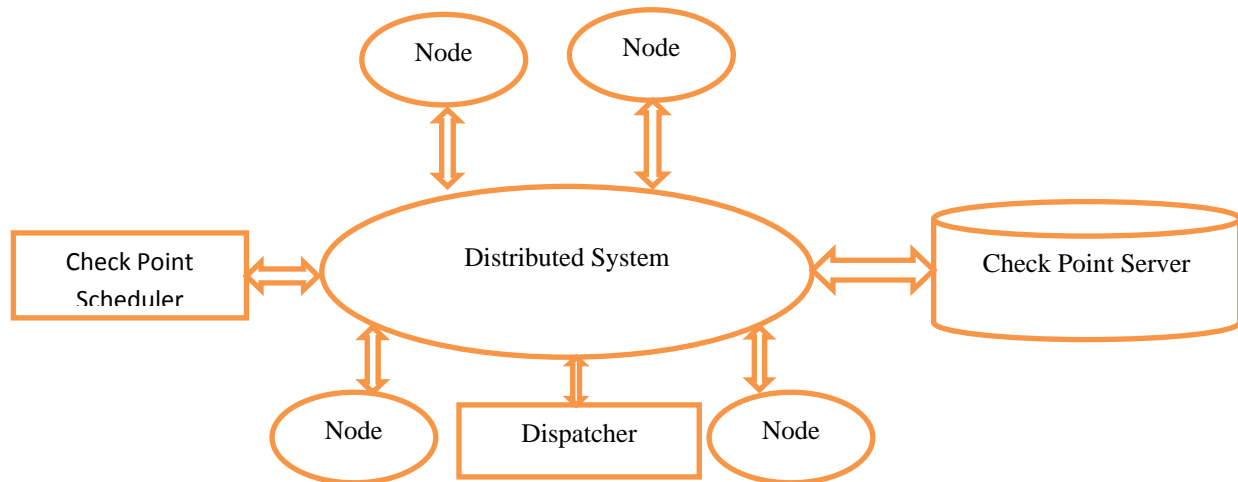

Fig: 3 Check Point & Rollback Process

The main function of a recovery is to recover the system again in consistent and operation state as it continues to work in normal condition. Two most important types of rollback recovery are checkpoint based rollback recovery and log based rollback recovery. Checkpoint-based rollback recovery relies only on checkpoints .Log-based rollback-recovery combines' checkpointing with logging of non-deterministic events [9]. Coordinated checkpoint and uncoordinated checkpoint associated with message logging are the two main techniques used for saving the distributed execution state and recovering from system failures [10].

In coordinate check point processes coordinate their checkpoints in order to save a system wide consistent state. Coordinate check points are consistent set of checkpoints. These consistent check points are used to bound rollback propagation. Consistency is more in case of coordinate check points due to consistent set of checkpoints [9]. Coordinated checkpoint involves the rollback check point of all processes from the last snapshot when a faulty situation is detected, even when a single process crashes. Therefore recovery time is very large and it makes unsuitable for real time applications. In case of frequent failures and multiple faults coordinate check point technique cannot be used. Performance can be improved by decreasing the recovery time .Main reason for large recovery time is restarting all the initial state. Recovery time can be reduced by enabling the restart from last correct state instead of from very first state. There must be some mechanism to ensure restarting from last correct state will reach a state matching the the system, as before the crash.

Uncoordinated checkpoint protocols are designed to handle such critical issues to some extend. Message logging is combined with uncoordinated checkpoint to restart the system from last correct state.In Uncoordinated checkpoint protocols, all processes execute a checkpoint independently of the others so that recovery can be done independently with one another. It is combined with message logging to ensure the complete description of a process execution state in case of its failure.

Besides logging of all received messages, re-sending the same relevant messages in the same order to the crashed processes during their reexecution is also main function of message logging. There are three kinds of message logging protocols: optimistic, pessimistic and causal. Pessimistic protocols ensure that

all messages received by a process are logged on reliable media before it sends information in the system. Log information on reliable media can be re-sent later and only if necessary during rollback.

Message logging optimistic protocols just ensure that all messages will eventually be logged. So, one usual way to implement optimistic logging is to log the messages on non-reliable media. Causal protocols log message information of a process in all causally dependent processes [11]. Check pointing based fault tolerance is very costly. Researcher proposed replication based check-pointing to improve the performance [12].There are many issues related to replication based check pointing fault-tolerance technique.

These issues are mainly degree of replication, check pointing storage type and location, check pointing frequency, check point size and check point run time. At the same time researcher suggested an adaptive check pointing and replication to adapt dynamically the check pointing frequency and the number of replicas as a reaction on changing system properties (number of active resources, resource failure frequency and system load) [13]. In case of fault, the most important issue is efficient recovery in dynamic heterogeneous systems. Recovery under different numbers of processors is highly desirable. The fault tolerant and recover approaches must be suitable for applications with a need for adaptive or reactionary configuration control.

Researcher proposed flexible rollback recovery in dynamic heterogeneous computing for such crucial requirements [14]. Still overhead of this technique is significant and need to be address further. Performance of any fault tolerant technique depends on recovery time. Researchers and practitioners are trying to improve the recovery time by improving the recovery time. Conventional rollbackrecovery protocols redo the computation of the crashed process since the last checkpoint on a single processor. As a result, the recovery time of all protocols is no less than the time between the last checkpoint and the crash. Researcher proposed a new application-level faulttolerant approach for parallel applications called the Fault-Tolerant Parallel Algorithm (FTPA), which provides fast self-recovery. When fail-stop failures occur and are detected, all surviving processes recomputed the workload of failed processes in parallel. FTPA, however, requires the user to be involved in fault tolerance. In order to ease the FTPA implementation, Researcher developed Get it Fault-Tolerant (GiFT),a source-tosource precompiled tool to automate the FTPA implementation.

Researcher evaluates the performance of FTPA with parallel matrix multiplication and five kernels of NAS Parallel Benchmarks on a cluster system with 1,024 CPUs. The experimental results show that the performance of FTPA is better than the performance of the traditional check pointing approach due to fast recovery [15].However this is only suitable for large problem. If the problem size is not large enough, not all processes will contribute to parallel recomputing In order to tolerate multiple faults using checkpoint and recovery, three critical functionalities that are necessary for fault tolerance: a lightweight failure detection mechanism, dynamic process management that includes process migration, and a consistent checkpoint and recoverymechanism. Hugo Jung et al. proposed a technique to address this critical functionality [9].
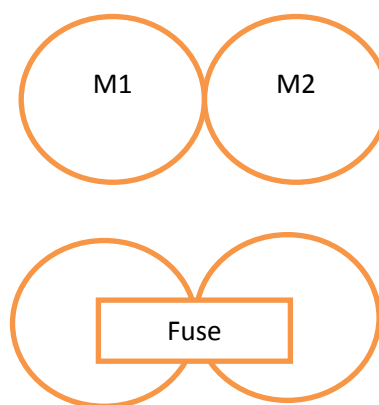
F. Fusion Based Technique

Although replication method is widely used as a fault tolerance technique but number of backups is a main drawback. Number of backups increases drastically as coverage against number of faults increases. As the number of backup increases management of these backups is very costly. Fusion based techniques overcome this problem. It is emerging as a popular technique to handle multiple faults. Basically it is an alternate idea for fault tolerance that requires fewer backup machines than replication based approaches. In fusion based fault tolerance a technique, back up machines is used which is cross product of original computing machines. These backup machines are called as fusions corresponding to the given set of machines [28]. Overhead in fusion based techniques is very high during recovery from faults. Hence this technique is acceptable if probability of fault is low.

III. PROPOSED TECHNIQUE

In computing, systems checkpoint is an essential for ensuring system availability. Checkpoint enables the system to continuously take snapshots or current status of running applications; in the presence of

a fault, the application can be rolled back to the most recent snapshot and continue execution with minimal downtime. Under error free execution, checkpoint incurs performance overhead. To make checkpointing attractive, the performance overhead must be minimized. All checkpoint mechanisms work on the basis of taking a snapshot of the running application. The snapshot required to recover an application consists of all application memory, opened files, sockets and IO devices.

This research focuses on check pointing applications with the concentrateon require memory state recovery. The application memory state has the largest footprint and therefore will be the most time consuming.In this research work the I/O traffic is buffered during each checkpoint and if a recovery is required the I/O is played back so that the application receives the same input during the second execution. The biggest factor to affect the performance overhead is the method adopted for the memory duplication process. The most rudimentary approach is to perform a full memory copy of the application at each checkpoint interval. This strategy, however, causes a great performance overhead since it requires a large amount of memory bandwidth.

For this reason, in our approach we will duplicate only a select region of application memory during each checkpoint which looks like incremental checkpoint and memory process. Since every check point we have to backup the add on memory space only. Fixed & standard memory spaces will not need to back up every time in the new check point. New status will be added into the existing memory back up. Replication also will be done in the newly updated staus only to avoid the memory overhead. The method of selecting which data to save and at what time interval varies between the different checkpoint types. In this we are implementing a memory space reduction scheme by deleting the unwanted checkpoints which are not at all further required.

This system has been implemented in java platform because garbage collection can be effectively implement. By this way our scheme proves to be space saving as well as time complexity is also reduced. Even if we need the deleted checkpoints details that can be also recovered by a replication scheme. The full memory copy of the application at each checkpoint interval is copied into a secondary storage such as the hard disk of the system.

	Existing System	Proposed Method
Memeory Usage	576 MB	324 MB
Swap Time	13 ns	5 ns
Response Time	6 ns	3.5ns

Table 1: Performance Analysis

IV. CONCLUSION

In this paper we have proposed an improved and efficient technique which ensures the consistency in the distributed environment using java RMI. The proposed technique involves lock leased protocol for performing the various write-write or read-write operations. Simultaneous concurrent read operations are possible in this environment. Our method is very simple and easy to be implemented. It efficiently reduces the check pointing overhead by saving the checkpoints on local hard disk as well as only selected data in memory.

REFERENCES

[1] M. Wiesmann, F. Pedone, A. Schiper, B. Kemme, G. Alonso," Understanding Replication in Databases and Distributed Systems," Research supported by EPFLETHZ DRAGON project and OFES).

[2] M. Herlihy and J. Wing."Linearizability: a correctness condition for concurrent objects," ACM Trans. on Progr. Languages and Syst., 12(3):463-492, 1990. (IJIDCS) International Journal on Internet and Distributed Computing Systems.Vol: 1 No: 1, 39

[3] M. Ahamad, P.W. Hutto, G. Neiger, J.E. Burns, and P. Kohli., "Causal Memory:Definitions, implementations and Programming," TR GIT-CC-93/55, Georgia In-stitute of Technology, July 94.

[4] H.P. Reiser, M.J. Danel, and F.J. Hauck., " A flexible replication framework for scalable andreliable .net services.,"

In Proc. of the IADIS Int. Conf. on Applied Computing, volume1, pages 161–169, 2005.

[5] A. Kale, U. Bharambe, "Highly available fault tolerant distributed computing using reflection and replication," Proceedings of the International Conference on Advances in Computing, Communication and Control ,Mumbai, India Pages: 251-256 ,: 2009

[6] X. China, "Token-Based Sequential Consistency in Asynchronous Distributed System ," 17 thInternaional Conference on Advanced Information Networking and Applications (AINA'03),March 27-29, ISBN: 0-7695- 1906-7

[7] A. Shye, , J. Blomstedt, , T. Moseley,V. Reddi, , and Daniel A. Connors, "PLR: A Software Approach to Transient Fault Tolerance for Multicore Architectures" Pp135-148.

[8] V. Agarwal, Fault Tolerance in Distributed Systems, I. Institute of Technology Kanpur, www.cse.iitk.ac.in/report-repository, 2004. ,

[9] H. Jung, D. Shin, H. Kim, and Heon Y. Lee, "Design and Implementation of Multiple FaultTolerant MPI over Myrinet (M3) ," SC|05 Nov 1218,2005, Seattle, Washington, USA Copyright 2005 ACM.

[10] M. Elnozahy, L. Alvisi, Y. M. Wang, and D. B. Johnson. A survey of rollback-recovery protocols in message passing systems. Technical Report CMU-CS-96-81, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, October 1996.

[11] L. Alvisi and K. Marzullo. Message logging : Pessimistic, optimistic, and causal. In Proceedings of the 15th International Conference on Distributed Computing,Systems (ICDCS 1995), pages ,229–236. IEEE CS Press, May-June 1995.

[12] J. Walters and V. Chaudhary," Replication-Based Fault Tolerance for MPI Applications," Ieee Transactions On Parallel And Distributed Systems, Vol. 20, No. 7, July 2009

[13] M Chtepen, F..Claeys, B. Dhoedt, , and P. Vanrolleghem," Adaptive Task Checkpointing and Replication:Toward Efficient Fault-Tolerant Grids", IEE Transactions on Parallel and Distributed Systems, Vol. 20, No. 2, Feb 2009

[14] S. Jafar, A. Krings, and T. Gautier," Flexible Rollback Recovery in Dynamic Heterogeneous Grid Computing", IEEE Transactions On Dependable and Secure Computing, Vol. 6, No. 1, Jan-Mar 2009

[15] X. Yang, Y. Du, Panfeng W. Fu, and Jia "FTPA: Supporting Fault-Tolerant Parallel Computing through Parallel Recomputing," Ieee Transactions On Parallel And Distributed Systems, Vol. 20, No. 10, October 2009

[16] S. Gorender, and M Raynal, "An Adaptive Programming Model for Fault-Tolerant Distributed Computing" Ieee Transactions On Dependable And Secure Computing, Vol. 4, No. 1, January-March 2007.

[17] A. Luckow B. Schnor, ,,"Adaptive Checkpoint Replication for Supporting the Fault Tolerance of Applications in the Grid," Seventh IEEE International Symposium on Network Computing and Applications, 2008 IEEE.

[18] A. Bouteiller, F. Cappello, T. H Krawezik, Pi Lemarinier, F Magniette, "MPICH-V2: a Fault Tolerant MPI for Volatile Nodes based on Pessimistic Sender Based Message Logging, " SC'03, NoV 15-21, 2003, Phoenix, Arizona, USA Copyright 2003 ACM 1-58113-695- 1/03/001

[19] I. Saha, D. Mukhopadhyay and S. Banerjee, "Designing Reliable Architecture For Stateful Fault Tolerance," Proceedings of the Seventh International Conference on Parallel and Distributed Computing,Applications and Technologies (PDCAT'06) 2006 .

[20] N. Gorde, S. Aggarwal, "A Fault Tolerance Scheme for Hierarchical Dynamic Schedulers in Grid" InternationalConference on Parallel Processing Workshops, 2008 IEEE

[21] Y. Li, , Z. Lan, , P. Gujrati and , X. Sun, , "Fault-AwareRuntime Strategies for High-Performance Computing," IEEE Transactions on Parallel And Distributed Systems, Vol. 20, No. 4, April 2009

[22] G. Jakadeesan, D. Goswami, "A Classification-Based Approach to Fault-Tolerance Support in Parallel Programs", International Conference on Parallel and Distributed Computing, Applications and Technologies, 2009 IEEE.

[23] D.K. Gifford, "Weighted voting for replicated data," In SOSP '79: Proc. of the seventh ACM symposium on Operating systems principles, pages 150–162, 1979.

[24] J. Osrael, L. Froihofer, K.M. Goeschka, S. Beyer,P. Gald´amez, , and F. Mu˜noz. "A system architecture for enhanced availability of tightly coupled distributed systems," In Proc. of 1st Int. Conf. on Availability, Reliability, and Security.IEEE, 2006

[25] J Maccormick1, C Thekkath, M.Jager,K. Roomp, and L. Peterson , "Niobe: A Practical Replication Protocol." ACM Journal Name, Vol. V, No. N, Month 20YY.

[26] Cao Huaihu, Zhu Jianming, "An Adaptive Replicas Creation Algorithm with Fault Tolerance in the Distributed Storage Network" 2008 IEEE..

[27] N. Budhiraja, K. Marzullo, F.B. Schneider, and S. Toueg.The Primary-Backup Approach. In SapeMullender, editor, Distributed Systems, pages 199-216. ACM Press, 1993.

[28] V.K Garg,. "Implementing fault-tolerant services using fused state machines," Tech-nical Report ECE-PDS-2010- 001, Parallel and Distributed Systems Laboratory,ECE Dept. University of Texas at Austin (2010).

[29] N. Xiong, M. Cao, J. He and L. Shu, "A Survey on Faulttolerance in Distributed Network Systems," 2009 International Conference on Computational Science, 978- 0-7695-3823-5/09

[30] D. Tian , K. Wu X. Li, "A Novel Adaptive Failure Detector for Distributed Systems," Proceedings of the 2008 International Conference on Networking, Architecture, and Storage ©2008 , ISBN: 978-0-7695- 3187-8