International Journal of Computer Applications (0975 – 8887)
Volume 42– No.19, March 2012

Performance improvement in Distributed Systems through Replication and Checkpointing

Sourabh Dave
Department of IT,
Mahakal Institute of Technology,
RGPV University, Ujjain(M.P)

Abhishek Raghuvanshi
Department of IT,
Mahakal Institute of Technology,
RGPV University, Ujjain(M.P)

ABSTRACT

In distributed system fault tolerance is an important issue. Many applications executing in present scenario with several processors have to face with problems related to consistency and availability. Complete process will fail with the failure of a single component. There are many existing approaches which assure reliable execution, are based on fault tolerance mechanisms. We talk about the basic concept of fault tolerance, which is to make a network system tolerant enough to work properly, may be with a little low efficiency, in case of any fault. A good fault tolerant system will avoid further failures.

After transient failures main problem is to bring a distributed system to a consistent state. We worked on two parts of this problem by providing a distributed system to create consistent checkpoints as well as replication is focused. We have given an algorithm for replication and implemented it in Java RMI. We have done two things: First the checkpoints are replicated and Second, Servers are replicated on different system using that algorithm.

General Terms

Fault Tolerance, Distributed System.

Keywords

Checkpointing, Replication, RMI.

1. INTRODUCTION

Systems began being connected to each other through communication system for interchanging data in form of files or any other information. Over time, many computers' capabilities are being shared over the network creating the sphere of distributed computing. Distributed computing is simply applying the two old sayings to the realm of computer resources. The first, "Many hands make light work" refers to the idea that you can take a task and break it down so that many different people or computers can work on it at the same time i.e. in simple sense a task is distributed among several computers. Then obviously it only takes a small amount of work by each computer (or human) to complete the bigger task [1].

The second saying, "The whole is greater than the sum of its parts" applies to the fact that the distributed task results are now recombined to get a whole result easily, that cannot be achieved by the computers working alone[1].

One of the main advantages of distributed system is that it can continue to run even if any fault occurs. This is called reliability of the system. Reliability of distributed system is much better than the computer working alone.

As it is known power of unity always work in any system. When a particular task is being done in a team it completes the task more efficiently and more effectively. For example it is very easy to break a single stick of wood whereas it is quite difficult to break a bunch of same stick. Similarly a distributed system consists of a collection of autonomous computers linked by a computer network and equipped with distributed system software. This software enables computers to coordinate their activities and to share the resources of the system hardware, software, and data.

In network, users experience that there are several machines are present. The working of this machines is not transparent whatever it does either load balancing, replications or any other functionality. But in distributed systems users identify a single and integrated computing facility, despite of having work distributed to different systems. Advantages of distributed systems as applications include: Massively multiplayer online games, virtual reality communities, Aircraft control systems, Distributed rendering in computer graphics and various other field [2].

Research in fault-tolerant [3] distributed computing aims at making distributed systems more reliable and enhancing its performance by handling faults in complex computing environments. Moreover, the society is increasingly dependent on well-designed and well-functioning computer systems, which led to an increasing demand for dependable systems, and the systems with quantifiable reliability properties. As accepted by many, the performance of distributed systems greatly depends on improving fault tolerant schemes [4]. There are many fault tolerant techniques designed on software level, each of which has their own advantages and disadvantages. Some of the schemes in fault tolerant distributed system are heartbeat technique, rollback technique, replication, Byzantine [22] faults etc. Many of the researchers claims that "replication" can sufficiently improve performance in distributed computing. So, here the focus is mainly on 'replication techniques' and trying the best to improve the performance through 'replication' in distributed systems.

2. FAULT TOLERANCE TECHNIQUES

Distributed systems have its application in various fields of information technology. Thus, its performance is a matter of concern in the family of researchers. We have studied a lot of techniques to improve the performance of distributed system. Process Level Redundancy, Fusion Based Technique, Checkpointing, Replication are some fault tolerance techniques used in distributed system. Among these we have focused on two techniques: Replication and Checkpointing.

2.1 Replication

Replication means redundant construct of the data or simply we can say multiple copies of same data. It is used to increase the availability of data. Replication based approach [13] is one of the famous and efficient approach for improving fault tolerance of the distributed system. If at some instance a fault occurs in the system and it gets out of order then at that time this technique would work and retain data from the node, where data is replicated. Replication is a process of maintaining different copies of a data item or object. In these types of techniques, all nodes have the same set of resources and data is replicated to these nodes so client can forward requests to any replica among the set of replicas. Replication adds redundancy in the system [13]. If any one of the node get failed then data of that node is available to its replicas thus failure of some node will not result in the failure of whole system. In this way fault-tolerance is achieved [13] as shown in fig 1.

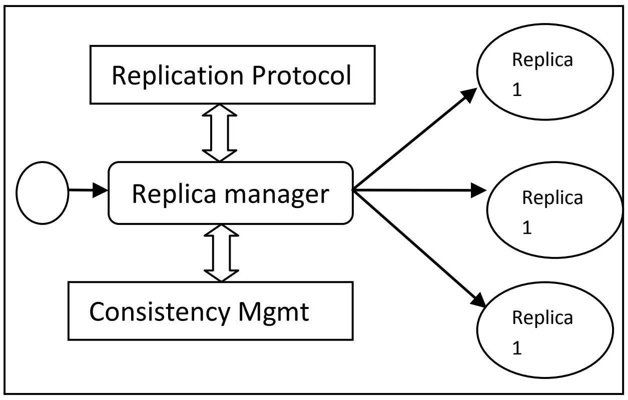

Fig 1: Replication Based Technique

There are various issues in replication base fault tolerance technique. Authors in [13] threw some light on some important issues in replication based techniques these are consistency, degree of replica etc.

The success in replication type of fault tolerance method is depends on the duplication of data. Same data is stored at different location that increases redundancy in the system. Consistency [13] is very important factor in any system. Increased amount of redundancy can compromise with the consistency of the system. For example, if a client is accessing the data from a replicated node on the other hand same copy is being updated by the server then this may generate inconsistent data. It is the responsibility of replication technique to ensure the consistency among all replicas of the same data. There should be some criterion which ensures the consistency. Various types of researches [13] are being done for defining criteria of consistency. Some of the consistency criteria defined in the literature are; linearizability [19], sequential consistency and causal consistency [20] etc. In all above strategies operation is performed on the most recent state or fresh state of the object. The definition of most recent state of the object differs for different consistency criterion. Both linearizability and sequential consistency define strong consistency criterion, whereas causal consistency defines a weak consistency criterion. Sequential consistency informally states that a multiprocessor program executes correctly if its result could have been produced by executing that program on single processor system.

In order to have consistency an efficient strategy is required. There are two main strategies one is active replication strategy and another is passive replication strategy. This classification is based on the way in which modification is done by the server. In passive replication, only one primary server executes the client requests and spreads the changes to all remaining replicas. With this scheme we can get rid of the multiple computations of the same requests. In active replication, client request is spreads to all replicas and computation is carried out at each node. All replicas execute the client request separately. Advantage of active replication is that it takes less network bandwidth and resources than sending update in contrast to the passive replication where primary server sends update to other replicated servers. Also the active replica response to a fault is faster than passive replication. There are various techniques of fault tolerance which used both type of strategies active and passive strategies to implement optimistic replication protocol [11]. Despite of these protocols there is still need of more simple, and effective replication protocol that can ensure consistency considerably.

The second important issue of replication is degree of replication. Number of replica is known as a degree of replication. In order to replicate an object a replication protocol is used [13]. There are various replication protocol given by researchers these are Primary-backup replication [27], voting [23], and primary-per partition protocol [24]. A replication protocol must be practical simple and effective such that it generates the replicas with consistency. The protocol must provide consistency guarantee with an effective performance. One such protocol is Niobe [2]. Degree of replica is an important factor the number of replicas must be sufficient. If a large numbers of replicas are taken, then it will increase the cost of maintaining the consistency. On the contrary if a less number of replicas are taken then, it will affect the performance, scalability and multiple fault tolerance capability. Therefore, there should be a mechanism in the protocol to decide the degree of replication. Some work is still going on in this direction to estimate the number of replicas. Various adaptive replicas creation algorithm are proposed. One such technique is proposed in [26]. There are further research possibilities to build up enhanced algorithm to maintain a balanced replica number. Replica on demand is a feature that can be implemented to make more adaptive, flexible and dynamic. There is research scope to further improve protocols to achieve replication efficiently. There are some critical needs with replication protocol. These critical requirements are support for a balanced number of replicas, strict consistency in the presence of any types of failure viz. network, disk, and machine failures.

2.2 Checkpointing

Checkpoint and rollback-recovery is a very popular technique which is widely used in the distributed systems [2]. Basically Checkpoint is nothing else but it is a snapshot of the state of the system at a specific time. When a system checkpoints, it saves the state of the system in a secondary storage, from where it can be referred later at the time of node failure. The state of the system includes different types of information about the system like process state, environment variables, registers, resources, processors etc. Checkpoints are saved regularly during the normal execution of program. When an error is detected, the process is roll backed to the last saved state [8].

Basically the main aim of a recovery operation is to fetch the system back into a consistent state as in normal condition. Two most important types of rollback recovery are checkpoint based rollback recovery and log based rollback recovery [13]. From these two techniques Checkpoint-based rollback recovery depends only on checkpoints [13] but Log-based rollback-recovery unites checkpointing with logging of those events which are not having determinism [20]. Two types of Checkpointing techniques are coordinated checkpoint and uncoordinated checkpoint associated with message logging used for saving the checkpoints and recovering when failure occurs [21]. In coordinate checkpoint, the name coordinate shows coordination among processes. All processes share their checkpoints and save a complete snapshot of the whole system [2]. In simple we can say process coordinate check points and these are consistent set of checkpoints. Due to consistent set of checkpoints, consistency is more in case of coordinate check points [2]. There is a drawback in coordinated checkpoint because it does the rollback check point of all processes from the last saved state when any failure occurs, even only one process having problem. So it takes long time to recover from the failure state which makes it inappropriate for various applications. This technique cannot be used in the system which suffers from the frequent failures whereas; the performance can be improved by decreasing the recovery time. It takes long time to recover because all the processes restart from the initial state. This time can be lessened if recovery operation brings the whole system to the latest correct state instead of initial state. These issues can be handled by using second type of checkpointing technique i.e. uncoordinated checkpointing. As the name suggests, there is no coordination among the processes like previous technique; whereas all the processes save the checkpoints independently. But this strategy would suffer from lack of coordination. So message logging is mingled with uncoordinated checkpoint to overcome from this problem. This will help the system to reach to the latest correct state after the failure. With message logging, system stores all the messages and resends them in the same order, after the system recovers from the failure.

2.3 Combined Approach (Replicated Checkpointing)

There are several issues with above mentioned techniques. Major issue is cost, checkpointing based fault tolerance mechanism is quite expensive. So researcher proposed new technique in which replication and check-pointing is used jointly to improve the performance [21]. It is known as replicated checkpointing. Some issues are still there which is to be handled while using it. These issues are primarily occurrence of checkpointing, level of replication, storage of check pointing, location and size of checkpoints. Researchers also gave the idea of deciding degree of replication and frequency of checkpointing at run time. This technique is known as adaptive task checkpointing and replication [23]. Two main important features related to these techniques are efficient recovery and recovery in short time. The distributed system is the collection of different types of processors or systems. These are called heterogeneous system. So it is highly required for all the recovery techniques to efficient recover the system from failure in this type of environment. On the other hand apart from efficiency time limit is also a considerable factor. Various researches are going on to improve the recovery time. Old strategy of recovery is to just redo the computation of the failed processes since the last checkpoint [2].

While applying various fault tolerance techniques like replication and checkpointing the major problem which would generally creates disturbance is its overhead. An asynchronous replication strategy is developed [22] that distributes checkpoint or replication overhead over all nodes participating in the computation. Two important factors are addressed in this strategy; first the resiliency of checkpoints to node failure, second; the bottleneck sustained by transferring data to dedicated servers. Walter and Choudhary proposed the use of uniform distribution in [22] for storing checkpoints because success of any strategy involving this relies on the availability of data. Each checkpoint has an equal query probability so uniform distribution is justified for this specific case.

An algorithm is given by Walter and Choudhary in [22] known as random node selection algorithm. The main aim of this algorithm is to provide location, as the output, where checkpoints to be replicated. For Old approach of replication there is a tendency to store replicas at farthest node but this may also cause low performance in the presence of inadequate network bisection bandwidth. Thus this algorithm will be useful in these types of cases [22].

It is implemented by authors in LAM/MPI environment. The aim of this algorithm is to generate same number of replicas to each node. They have managed the algorithm in such a way that no node has its own replica on itself. The number of replicas of a node n is same as the number of replicas of n spread onto the network and all replicas are unique.

The algorithm provides an initial state to the system which satisfies the above mentioned constraint. After that circular shift operation is used to provide the first stage of randomness. Swapping operation is used between nodes various times for maintaining randomness. During each swap operation it is managed that all the constraints should satisfy the constraints imposed. Finally, the actual replica swap is performed after many confirmations and actual result is generated.

3. RMI

RMI i.e. Remote method invocation is a construct of JAVA [6] which is used to help applications in communication with other application that is running on a remote system. There are several other methods exist which can be used for remote communication but they have many limitations like only some defined structure or simple data types can be passed to and from methods. An important feature of RMI is that it permits to load new object types dynamically even if client and server don't know about it.

3.1 Overview:

RMI allows you to call procedures [27] on objects that reside on other virtual machine and treat them as if they were on the local machine. Concept of RMI is very simple there is a client and a server: the server has the method which is to be called remotely by a client application. There is also a record known as RMI registry which is used to handle all remote methods. RMI service provides a new name to remote method and does its entry in RMI registry by using binding method. Once the client has this reference, it can make remote method calls with parameters and return values as if the object (service) were to be on the local host. Java RMI is comprised of three layers [5] that support the interface.

There are three layers in the architecture of RMI. The first layer is the Stub/Skeleton Layer. This layer provides interface to both client and server. Both client and server communicate with each other by using remote object. This layer manages the remote object interface. The name of second layer is Remote Reference Layer (RRL). This layer acts as an interface between first and third layer. The role of this layer is to manage communication between the client/server and virtual machines, (e.g., threading, garbage collection, etc.) for remote objects [5]. The name of third layer is the transport layer. This layer is like data link layer /physical layer of TCP/IP suite. It is also the lowest layer of the protocol suite and it is used to send the information between the client and server over the wire.

4. PROPOSED WORK

The random node selection algorithm [22] is very efficient for replication and authors worked very well and implemented it in LAM/MPI environment. We have taken the same algorithm and tried to develop it in Java RMI technology. There are many technologies exists to create distributed environment but we employed RMI because of its familiar environment and it is presented as efficient technology for distributed system by Jerzy Brzezinski and Cezary Sobaniec in [6].

The random node selection algorithm talked only about replication by providing the addresses of the nodes where replicated data to be stored. But it didn't tell about what to do if there is a requirement of increasing fault tolerance capability by increasing number of nodes or by increasing number of replicas. Thus, we modified and implemented this algorithm to handle these important issues. The issues are as follows:

Case 1: When New Node needed to be inserted in an already developed distributed system.

Case 2: When number of replicas required to be increased in an already developed distributed system to improve its fault tolerance capability.

We developed the new algorithm to handle these issues and got implemented it in RMI. Our implementation contains the following modules:

1) A Computational Coordinator that have algorithm implementation.
2) The Client Systems that needed to communicate with coordinator to know about the replica placement.
3) All the systems in the network needed to communicate each other to find or place their replicas over the network.

This new program has the capability to quench the thirst of increasing fault tolerance. It provides users to increase the no of nodes or replicas after initiating the program. After final execution of the algorithm at the Coordinator side, all clients receive a file containing IP addresses of the nodes selected for the replication of their data. One program has also been developed for communication of the clients. In it, clients finally replicate their data to the addresses received in the file by the coordinator. With the use of this algorithm checkpoints are replicated and fault tolerance is achieved.

4.1 Comparative study

The basis of this work is the random node selection algorithm given by Walter and Choudhary. Their work is quite advantageous to the field of distributed system. We tried to work on the platform created by them. There are some points regarding comparative study of our work and their work.

1) The algorithm given in [22] is implemented in LAM/MPI environment whereas we implemented it in JAVA RMI which is quite easy to understand.
2) We also gave the flexibility of increasing fault tolerance at run time. If there is a need of increasing number of nodes or number of replica at run time then it can be fulfilled by our proposed algorithm.
3) Consistency is a major issue while doing replication. We have used a coordinator algorithm to ensure consistency.
4) We have also implemented idea of saving checkpoints on the local hard disk instead on dedicated server.

5. CONCLUSION

We deeply analyzed the Random Node Selection Algorithm, and find out some shortcomings in that, and then we redesigned the algorithm including various points in such a way that the overall complexity of the algorithm is less as that of earlier. We have proposed an improved method to ensure the consistency by simulating the distributed environment using java RMI. This algorithm is very simple and ensures the consistency in a very simple manner. Checkpointing overhead is reduced by saving the checkpoints on local hard disk instead of SNA (Storage Network Area) or DFS (Distributed File System) following the assumptions given by John Paul Walters. This work will definitely work as a reference for researcher and practitioner to design and develop high performance multiple fault tolerance.

6. ACKNOWLEDGEMENT

We would like to thank the all faculty members of the institute who helped us lot in calculating the facts and figures related to our paper. I would also like to thank the anonymous reviewers who provided helpful feedback on my manuscript.

7. REFEERENCES

[1] Daniel Oelke, “Overview of Distributed computing”, Mithral Communications & Design Inc. 1995-2012 , [Online]Available:http://www.mithral.com/projects/cosm/ch-02.html

[2] Sanjay Bansal, and Sanjeev Sharma, “Identification of Critical Factors in Checkpointing Based Multiple Fault Tolerance for Distributed System”, Journal of Emerging Trends in Computing and Information Sciences, Volume 2 No. 1, 2010.

[3] Halpern, J. and Y. Moses, "Knowledge and Common Knowledge in a Distributed Environment," Proc. of the 3rd ACM Symposium on Principles of Distributed Systems, 1984, pp. 50-61 and Lamport, L., R. Shostak, and M. Pease, "The Byzantine Generals Problem," ACM Transactions on Programming Languages and Systems, Vol. 4 No. 3, July 1982, pp. 382-401.

[4] Jalote, P. Fault Tolerance in Distributed Systems, (Prentice Hall, 1994).

[5] Chris Matthews, “Introduction to Java Remote Method Invocation (RMI)”, The Electronic Developer Magzine, [Online]Available: http://www.edm2.com/0601/rmi1.html

[6] A Concept of Replicated Remote Method Invocation Jerzy Brzezinski and Cezary Sobaniec, Institute of Computing Science, Poznan University of Technology,

Poland{Jerzy.Brzezinski,Cezary.Sobaniec}@cs.put.poznan.pl.

[7] 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).

[8] 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

[9] 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 Institute of Technology, July 94.

[10] 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.

[11] 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

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

[13] Sanjay Bansal, Sanjeev Sharma, Ishita Trivedi, "A Detailed Review of Fault-Tolerance Techniques in Distributed System", International Journal on Internet and Distributed Computing Systems. Vol: 1 No: 1 : 2011

[14] 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.

[15] 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

[16] 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.

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

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

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

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.