ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Distributed System with Improved Replication Technique

Lalit Gehlod*

* Department of Computer Engineering, Institute of Engineering and Technology, Indore

Abstract

Distributed system is one of the important backbones of today's computer systems. One of the major problematic issues in distributed system is Fault tolerance. Most of the systems which are based on distributed system face faults in various forms like unavailability, consistency, isolation etc. There are many fault tolerance techniques are available like check pointing, Replication etc.Both the above mentioned techniques have their own pros and cons. Replication plays a vital role in fault tolerance. In this work we are maintain replicas of data over different-different systems so that we can recover our data in case of fault. During Replica placement we assure that some replicas should be on nearer nodes and some on farther nodes. In this work we have chosen Java RMI to implement the system. In this work we have added dynamism in the system by adding number of nodes who participate in distributed system at run time. We can also increase the number of replicas at run time in the distributed system.

Keywords: Fault Tolerance, DS, Replication

Introduction

Systems were connected with each other through several channels for interchanging data like files or other data. Replication plays a vital role in fault tolerance. In this work we are maintain replicas of data over different-different systems so that we can recover our data in case of fault. During Replica placement we assure that some replicas should be on nearer nodes and some on farther nodes. In this work we have chosen Java RMI to implement the system. In this work we have added dynamism in the system by adding number of nodes who participate in distributed system at run time. We can also increase the number of replicas at runtime in the distributed system. In Distributed computing there is a saying. First, "Many hands make light work" and "whole are greater than the sum of its parts" means you can take a task and break it so that many persons can work on it simultaneously and the distributed task results are now recombined to get a whole result easily, that cannot be achieved by the computers working alone[1].

The another 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].

Main advantages with distributed system are that it can continue even if any fault occurs in the system called reliability of the system. Reliability of distributed system is better than stand alone system.

As we known power of unity always work in system. When a particular task is done in a group it completes the task more accurately 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. In a similar way a distributed system consists of collection of autonomous computers connected by a computer network and equipped with distributed machines. This program enables computers to coordinate their activities and share resources of the system hardware, software, and data.

In networking systems, users experience that there are many machines are present. The working of these machines is not clear whatever it is doing load balancing, replications or any other. 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 controlling machines, distributed rendering in computer graphics and many other field [2].

Research activity in fault-tolerant [3] distributed computing aims to make distributed systems more trustworthy and enhancing its performance by handling faults in complex computing environments. Furthermore, the society is increasingly reliant on

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

well-designed and well-functioning computer systems, which led to an increasing demand for dependable systems, and the systems with quantifiable dependability properties. As accepted by a lot of, the performance of distributed systems significantly depends on improving fault tolerant methods [4]. There are many fault tolerant techniques designed on software level, each of which has their own advantages and disadvantages. Some methodologies used in fault tolerant distributed system are heartbeat method, rollback technique, replication, Byzntine [22] faults etc. Many researchers tell that "replication" can sufficiently improve performance of distributed computing system. our focus is mainly on 'replication techniques' and are trying to improve the system efficiency with the help of 'replication techniques' in distributed systems.

Fault Tolerance Techniques

Performance of distributed system is most important in many applications and for researchers too.We have studied so many techniques like Process Level Redundancy, Fusion Based Technique, Checkpointing, Replication etc.Out of all we have focused on replication.

2.1 Replication

Replication is the key to providing high availability, fault tolerance, and enhanced performance in a distributed computing system. As companies move toward systems that are more open and distributed, replication is becoming increasingly important in the ability to provide data and services that are current, correct and available, which is a key factor in maintaining a competitive advantage over rivals.It add redundancy in system and we can recover from redundant data in case of failure as shown in fig. 1

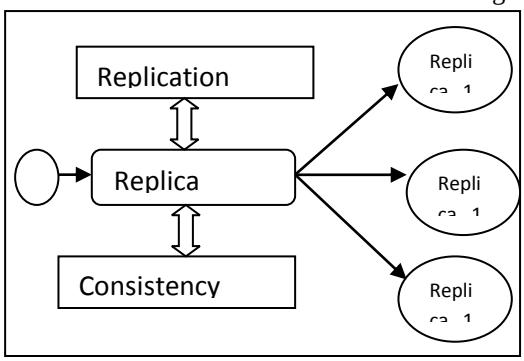

Fig 1: Replication Based Technique

There are various issues in replication base fault tolerance technique. Important issues [13] in replication based techniques are consistency, degree of replica etc.

2.1.1 Consistency

Consistency is important construct. Because success of replication is totally based on redundant data so it can provide contradictory data at the time of action. For instance, a client is accessing the facts from a replicated node which is being updated by the server then; it can access not consistent data. A replication procedure must ensure the consistency among all replicas of the same thing. Consistency is ensured by some criterion. Many consistency criteria have been clear in the literature; linearizability [19], sequential consistency and causal consistency [20] etc. In all above cases, an operation is performed on the most recent state of the object. However consistency criteria differ in the definition of the most recent state. Primary-backup replication technique and active replication technique ensure consistency by linearizability. 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. Passive strategy and active strategy are main strategies. In a passive replication, only one principal execute requests and multicasts state changes to all replicas. This scheme avoids duplicate computation of requests. It copes with non-deterministic service behavior. In active replica, client request is multicasts to all replicas. This means all replicas execute the request individually. In this way

Active replica takes less network resources than sending update. Active replica response to a fault is faster than passive. However, replica consistency usually requires deterministic replica behavior [10].Researcher proposed an algorithm that uses both active and passive strategies to implement optimistic replication protocol [11]. Researcher also proposed a simple protocol by combining the token with cache. This gives benefits of token as well as cache [12].There is still need of more simple, adaptive and practical replication protocol with adequate and sufficient ensured consistency.

2.1.2 Degree of Replica

Number of replica is called as a degree of replication. In order to replicate an item a replication protocol is used [13]. Primary-backup replication [27], voting [23], and primary-per partition protocol [24] are some of the replication protocol. A replication protocol must be realistic and simple. The protocol must offer rigorously-proven yet simply-stated steadiness guarantee with a reasonable performance.

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Niobe is such protocol purposed by researcher [25].Number of replicas must be sufficient. Large numbers of replicas will increase the cost of maintaining the consistency. Less number of replicas will concern the performance, scalability and multiple fault tolerance capability. Therefore, rational number replicas must be guess as per system arrangement and load. Researcher projected adaptive replicas creation algorithm [26].There is further research scope to develop improved algorithm to maintain a rational replica number. Replica on demand is a feature that can be implemented to make more adaptive, flexible and vibrant. There is research scope to further improve protocols to achieve replication proficiently. There are some crucial requirements with replication protocol. These crucial necessities are sustain for a flexible number of replicas, firm consistency in the occurrence of network, disk, and machine failures and efficient common cases read and write operations deficient requiring potentially costly two or three-phase commit protocols.

2.2 Checkpointing

Checkpointing and rollback-recovery are well-known techniques that allow processes to make progress in spite of failuresi2. The failures under consideration are tran- sient problems such as hardware errors and transaction aborts, i.e., those that are unlikely to recur when a process restarts. With this scheme, a process takes a checkpoint from time to time by saving its state on stable storage9 When a failure occurs, the process rolls back to its most recent checkpoint, assumes[25]

Checkpointing is an important feature in distributed computing systems. It gives fault tolerance without requiring additional efforts from the programmer. A checkpoint is a snapshot of the current state of a process. It saves enough information in non-volatile stable storage such that, if the contents of the volatile storage are lost due to process failure, one can reconstruct the process state from the information saved in the non-volatile stable storage. This strategy usually works well in uniprocessor systems. The reconstruction of a distributed system with multiple processes is, however, not easy. The action of the receiver of a message depends on the content of the message. If the processes communicate with each other through messages, rolling back a process may cause some inconsistency. In the time since its last checkpoint, a process may have sent some messages. If it is rolled back and restarted from the point of its last checkpoint, it may create orphan messages, i.e., messages whose receive events are recorded in the states of the destination processes but the send events are lost. Similarly, messages received during the rolled back period, may also cause problem. Their sending processes will have no idea that these messages are to be sent again. Such messages, whose send events are recorded in the state of the sender process but the receive events are lost, are called missing messages. A set of checkpoints, with one checkpoint for every process, is said to be Consistent Global checkpointing State (CGS), if it does not contain any orphan message or missing message. However, generation of missing messages may be acceptable, if messages are logged by sender. In a distributed system, each process has to take checkpoints periodically on non-volatile stable storage. In case of a failure, the system rolls back to a consistent set of checkpoints. If all the processes take checkpoints at the same time instant, the set of checkpoints would be consistent. But since globally synchronized clocks are very difficult to implement, processes may take checkpoints within an interval. In order to achieve synchronization, sometimes processes take temporary checkpoints. When all processes agree, these checkpoints are made permanent. There are several schemes for checkpointing and rollback recovery.

2.3.1 Random Node Selection

The Random node selection algorithm [22] is given by John Paul Walters and Vipin Choudhary. It is implemented by authors in LAM/MPI environment. The goal is to randomly generate r replicas per node, subject to the following constraints:

1. Self replication of node is not allowed
2. A node should replicate to exactly r nodes, each of which may only store r replicas.
3. Replica of each of a node should be unique.

The key to the algorithm is to begin with an initial state in the form of a table which contains the information about the nodes on which data of a particular node is replicated. The initial state satisfies the above replica constraints, and incrementally refines the replica choices through swapping. After this a circular shift operation is used to ensure the final state of replicas should be valid according to the constraints. This means that each column should be circular-shifted such that no two rows contain duplicate replicas and no row should be assigned itself.

Once the early state is attained, the algorithm proceeds through each node and each replica and swaps replicas with randomly chosen nodes. Because algorithm started the swapping with a convincing initial state, it is only needed to keep the constraints while introducing an element of randomness into the

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

replica production. Two nodes may swap replicas as long as neither node already contains a copy of their potential replicas, and provided that the swap would not result in either node replicating to itself. Next, the candidate node is generated that may later be used for swapping. The only constraint that is enforced is that the node should not attempt to swap with itself. Once a valid replica candidate has been generated, it is simply checked to ensure that swapping would maintain a valid state. Finally, the actual replica swap is performed after many confirmations.

RMI

The RMI (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.The RMI provides remote communication between the applications using two objects stub and skeleton. stub and skeleton-

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

It initiates a connection with remote Virtual Machine (JVM),

It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),

It waits for the result

It reads (unmarshals) the return value or exception, and

It finally, returns the value to the caller.

The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

It reads the parameter for the remote method

It invokes the method on the actual remote object, and

It writes and transmits (marshals) the result to the caller. As shown in figure 2

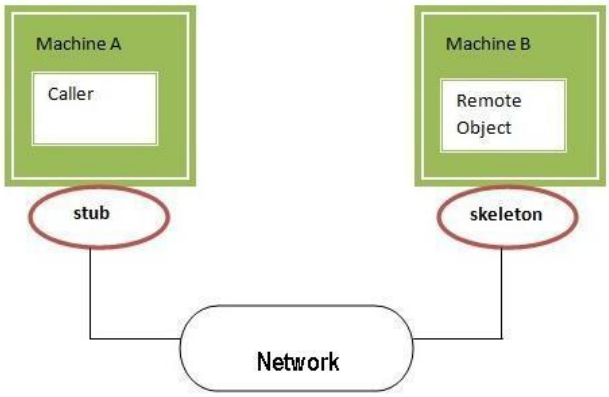

Fig. 2

3.1 Overview:

RMI allows you to identify procedures [27] on objects that reside on other effective machine and treat them as if they were on the local machine. Perception of RMI is very easy 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.

The first layer is the Stub/Skeleton Layer. This layer is responsible for managing the remote object interface between the client and server. The second layer is the Remote Reference Layer (RRL). This layer is responsible for managing the "liveliness" of the remote objects. It also manages the communication between the client/server and virtual machine s, (e.g., threading, garbage collection, etc.) for remote objects. The third layer is the transport layer. This is the actual network/communication layer that is used to send the information between the client and server over the wire. It is currently TCP/IP based. If talked upon RPC, it is a UDP-based protocol which is fast but is stateless and can lose packets. TCP is a higher-level protocol that manages state and error correction automatically, but it is correspondingly slower than UDP.

Proposed Work

The random node selection algorithm [5] is very well-organized for replication and authors worked very well and implemented it in LAM/MPI settings. We have taken the same algorithm and tried to develop it in Java RMI technology. There are many

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

tools exists to build distributed environment but we employed RMI because of its recognizable environment and it is presented as competent tools for distributed system by Jerzy Brzezinski and Cezary Sobaniec in [6].

The random node selection algorithm conversed only about replication by providing the addresses of the nodes where replicated data to be saved. But it didn't says regarding what to do if there is a condition of escalating fault tolerance competence by escalating number of nodes or by increasing number of replicas. Thus, we modified and executed this algorithm to handle these vital issues. The issues are as follows:

There are two possible cases in case one is when new node desired to be inserted in an already developed distributed structure and case two when number of replicas required to be increased in an previously developed distributed system to improve its fault tolerance capability.

We designed the new algorithm to touch these issues and got implemented it in RMI. Our execution contains the following modules:

1. A dedicated Coordinator that have algorithm implementation.
2. The Client environment that required to converse with coordinator to concern about the replica placement.
3. All the systems in the network required to converse each other to discover or place their replicas over the network.

This new program has the facility to satisfy the desire of escalating fault tolerance. It provides users to boost the no of nodes or replicas after initiating the program. After final implementation of the algorithm at the Coordinator side, all clients receive a file including 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 lastly 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 attained

Conclusion

We intensely examine the Algorithm, and find out some limitation in that, and then we redesigned the algorithm including various points in such a way that the on the whole complexity of the algorithm is less as that of earlier.We have proposed an better method to ensure the consistency by simulating the distributed environment using java RMI. This algorithm is very simple and guarantees the consistency in a very simple manner. Check pointing operating cost is reduced by storing the checkpoints on local hard disk instead of SNA (Storage Network Area) or DFS (Distributed File System) following the postulations specified by John Paul Walters. This work will absolutely work as a reference for researcher and practitioner to design and extend high performance multiple fault tolerance.

REFERENCES

1. http://www.mithral.com/projects/cosm/ch-02.html
2. http://encyclopedia2.thefreedictionary.com/Distributed+system
3. http://www.javatpoint.com/RMI
4. Jalote, P. Fault Tolerance in Distributed Systems, (Prentice Hall, 1994).
5. 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

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

Applications (AINA'03),March 27-29, ISBN: 0-7695- 1906-7
13. Checkpointing and Rollback-Recovery for Distributed Systems by Richard KooD.K.
14. Gifford, "Weighted voting for replicated data," InSOSP '79: Proc. of the seventh ACM symposium on Operating systems principles, pages 150–162, 1979.
15. http://www.isical.ac.in
16. J Maccormick1, C Thekkath, M.Jager,K. Roomp, and L. Peterson , "Niobe: A Practical Replication Protocol." ACM Journal Name,
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. http://www.scpe.org

Author Bibliography

Lalit Gehlod
Author has been working as a lecturer in computer engineering department at IET,DAVV