Volume 2, Issue 5, May 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Analysis of Replication and Replication Algorithms in Distributed System

Nikhil Chaturvedi
Research Scholar in Dept. of CSE
S.V.I.T.S
Indore (M.P)
chaturvedinikhil87@gmail.com

Prof. Dinesh Chandra Jain
Reader , Dept. of CSE
S.V.I.T.S
Indore (M.P)

***Abstract*- This paper focuses on replication in distributed file systems. Replication is a key strategy for improving reliability, availability and performance in distributed systems. This paper gives a brief introduction to replication and various algorithms have been discussed and a detailed study has been performed. The comparison table shows it clearly that an algorithm satisfies the replication requirement.**

Keyword : DFS, Replication, replication algorithms, comparison of algorithm

I. INTRODUTION

The basic function of file system is to provide a long time reliable storage. Similar to a local based file system, distributed file system (DFS)[2] stores files on one or more computers called servers DFS often provide file replication as a service to their clients. In other words multiple copies of selected files are maintained, with each copy on a separate file server. Major reasons[3] for offering such service are

- **Reliability** – DFS results in the elimination of the single point of failure that has dogged all timesharing systems. It also supports replication for all of its network services. If one of the servers becomes unavailable, a client automatically switches over to one of the replicated servers.

- **Availability** – The wining feature of DFS is that its components are available to users at all times. Use of replication enables an administrator to do file system backups while the system is up and running. The replicated copy remains stable even while the user is changing the original file.

- **Fault Transparency or Tolerance** - Components in a distributed system can fail independently. A file may be made more available in the face of failures of a file server if it appears on more than one file server. Availability is one of the main measures of the advantage of a replication algorithm.

- **Load Balancing** - Replication of files in a DFS allows better load balancing.

- **Performance** - Another advantage of a distributed file system is the ability to share information with many diverse users. DFS is an efficient, extensible system. In addition, the server tracks which clients have cached copies of files, reducing the need for the client to constantly query the server, as well as reducing the network and server load. the server tracks which clients have cached copies of files, reducing the need for the client to constantly query the server, as well as reducing the network and server load.

II. REPLICATION AND REPLICATION TECHNIQUES

The main idea in replication[5] is to keep several copies or replicas of the same resources at various different servers. A client can contact any of these available servers, preferably the closest, for the same document. This helps in reducing server load, access latency and network congestion. Some of the important factors for replication are to maintain consistency among the replicas, to find the best replica server for each client and also keep it transparent to the users.

There are number of technique for file replication that are used to maintain data consistency. Replication services maintain all the copies or replicas having the same versions of updates. This is known as maintaining consistency or synchronization.

Replication techniques to provide consistency can be divided into two main classes:

• **Optimistic**- These schemes assume faults are rare and implement recovery
schemes to deal with inconsistency.

• **Pessimistic-** These schemes assume faults are more common, and attempt
to ensure consistency of every access.
Schemes that allow access when all copies are not available use voting protocols to decide if enough copies are available to proceed.

Pessimistic Replication

This is a more conservative type scheme using prime site techniques, locking or voting for consistent data update. As this approach assumes that failure is more common it guards against all concurrent updates. An update cannot be written if a lock cannot be obtained or if majority of other sites cannot be queried. In doing so you sacrifice data availability. The pessimistic model is a bad choice where frequent disconnections network and network partitions are common occurrence. It is used for more traditional DFS[7].

Optimistic Replication

This approach assumes that concurrent updates or conflicts are rare. This scheme allows concurrent updates can be done at any replica or copy. This increases the data availability. However, when conflicts do occur, special action must be taken to resolve the conflict and merge the concurrent updates into a single data object. The merging is referred to as conflict resolution. When conflicts do occur, many can be resolved transparently and automatically without user involvement. This approach is used for mobile computing.

III. REPLICATION MODEL

There are three basic replication models the master-slave, client-server and peer-to-peer models.

- ***Master-slave model***

In this model one of the copy is the master replica and all the other copies are slaves. The slaves should always be identical to the master. In this model the functionality of the slaves are very limited, thus the configuration is very simple. The slaves essentially are read-only. Most of the master-slaves services ignore all the updates or modifications performed at the slave, and "undo" the update during synchronization, making the slave identical to the master. The modifications or the updates can be reliably performed at the master and the slaves must synchronize directly with the master.
The client-server model like the master-slave designates one server, which serves multiple clients. The functionality of the clients in this model is more complex than that of the slave in the master-slave model. It allows multiple inter-communicating servers, all types of data modifications and updates can be generated at the client. One of the replication systems in which this model is successfully implemented is Coda. Coda is a distributed file system with its origin in AFS2. It has many features that are very desirable for network file systems. Optimistic replication can use a client-server model. In Client- server replication all the updates must be propagated first to the server,which then updates all the other clients. In the client-server model, one replica of the data is designated as the special server replica. All updates created at other replicas must be registered with the server before they can be propagated further.

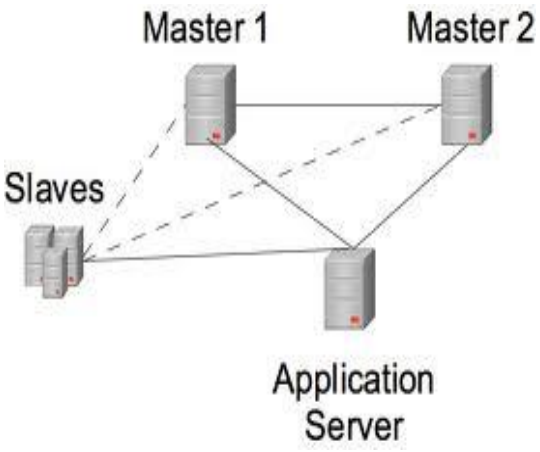

Fig 1: master slave model

- ***Client-server model***

This approach simplifies replication system and limits cost, but partially imposes a bottleneck at the server. Since all updates must go through the server, the server acts as a physical synchronization point. In this model the conflicts which occur are always be detected only at the server and only the server needs to handle them. However, if the single server machine fails or is unavailable, no updates can be propagated to other replicas. This leads to inconsistency as individual machines can accept their local updates, but they cannot learn of the updates applied at other machines. In a mobile environment where connectivity is limited and changing, the server may be difficult or impossible to contact, while other client replicas are simple and cheap to contact. The peer model of optimistic replication can work better in these conditions.

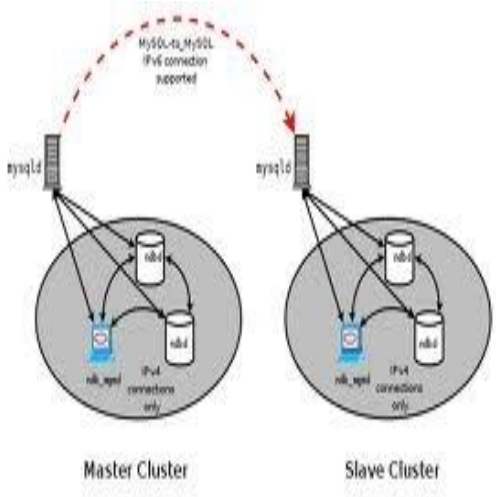

Fig 2:client server model

- ***Peer-to-peer model***

The Peer-to-peer model[8] is very different from both the master-slave and the client-server models. Here all the replicas or the copies are of equal importance or

they are all peers. In this model any replica can synchronize with any other replica, and any file system modification or update can be applied at any replica. Optimistic replication can use a peer-to-peer model. Peer-to-peer systems allow any replica to propagate updates to any other replicas. The peer-to-peer model has been implemented in Locus, Rumor and in other distributed environments such as xFS in the NOW project. Peer-to-peer systems[11] can propagate updates faster by making use of any available connectivity. They provide a very rich and robust communication framework. But they are more complex in implementation and in the states they can achieve. One more problem with this model is scalability. Peer models are implemented by storing all necessary replication knowledge at every site thus each replica has full knowledge about everyone else. As synchronization and communication is allowed between any replicas, this results in exceedingly large replicated data structures and clearly does not scale well. Additionally, distributed algorithms that determine global state must, by definition, communicate with or hear about (via gossiping) each replica at least once and often twice. Since all replicas are peers, any single machine could potentially affect the outcome of such distributed algorithms; therefore each must participate before the algorithm can complete, again leading to potential scaling problems . Simulation studies in the file system arena have demonstrated that the peer model increases the speed of update propagation among a set of replicas, decreasing the frequency of using an outdated version of the data.

IV. Replication Algorithms

- ***Less Log: A Legless File replication Algorithm for peer – to Peer Distributed Systems***

In this paper, the author present Less Log, a legless file replication algorithm, developed for a peer – peer distributed system[12]. A look – up tree is constructed for each node. Less Log uses bitwise operations to determine the location of the replicated node without any client – access history. In addition, each replication is guaranteed to reduce the workload of the replicating node by half. A fault – tolerant Less Log model is also presented. The experimental results show that Less Log successfully and efficiently reduces the load of overloaded nodes.

Work to be done:

The future work is to implement LessLog in a large – scaled P2P system and obtain performance data in a real – world scenario where nodes dynamically join and leave the system.

- ***Research of replication in unstructured P2P network***

In this paper research is based on the sub dividable area. The author proposes a new mechanism calls junction replication Method (JRM)[13] in unstructured decentralized P2P network which reflects the most usual P2P network in the real environment. Set a new limit, if a normal node's request over that limit, the author also sends the replica to it directly. In this way, the author can avoid nodes become overloaded with low node in nodes.

Work to be done:

a. First, Security of JRM needs to be considered as the node's location is easy to be exposed.

b. Second, for the development of computer, the gap between super node and normal ones is smaller; the author needs to find methods to spread replicas averagely and decrease the pressure of super node. And a better load –Balancing in the whole network is the final destination.

- ***File clustering based replication algorithm in a grid environment***

This algorithm groups files according to a relationship of simultaneous accesses between files and stores replicas of the clustered files into storage nodes , to satisfy expected most of future read access times to the clustered files andreplication times for individual files being minimized under the given storage capacity limitation. The experiments on a given grid environments, 20 nodes of 5 sites, suggest that the proposed algorithm achieves accurate file clustering and efficient replica management.

Work to be done:

In future, the author will continue to evaluate our algorithm in complex workloads on live environments.

- ***DORA: A dynamic file assignment strategy with replication***

In this paper, the author proposes a new dynamic file assignment strategy called DORA (dynamic round robin with replication). The advantages of DORA can be attributed to its two main characteristics. First, it takes the dynamic nature of file access patterns into account to adapt to a changing workload condition. Second, it utilizes file replication techniques to complement file assignment schemes so that system performance can be further improved. Experimental results demonstrate that DORA performs consistently better than existing algorithms.

b. Three stages in DORA algorithm:

i. Initial file assignment

ii. Dynamic replication and replica allocation

iii. Replica garbage collection

Work to be done:

In this research DORA only considers read dominant workload like web search where read bandwidth is prevailing while write bandwidth is minimal.

- ***Impact of peer – to – peer communication on real – time performance of file replication algorithms***

In this study, performance of four replication algorithms, two from the literature and two new algorithms, are evaluated. For this evaluation, a process

oriented and discrete – event driven simulator is developed. A detailed set of simulation studies are conducted using the simulator and the results obtained are presented to elaborate on the real time performance of these replication algorithms.

Work to be done:
These initial yet detailed results on the impact of the peer – to – peer communication on the replication algorithms in terms of the real time grid performance motivate the development of more sophisticated replication algorithms to better use of the grid resources, which will be topic of the future research.

- ***Dynamic replica Placement and Location strategies for data grid***

In data grid, replication data on multiple nodes can improve availability and response time. Yet determining when and where to replicate data in order to meet performance goals with many users and files, dynamic network characteristics, and changing user behavior is difficult. Also it is a challenging problem to find the best fit replica efficiency according to location and performance of physical storage systems at which the replicas dwell.

Work to be done:
As part of future work, the author plans to validate his model on real data grids and evaluate it with other replica algorithm.

- ***Access – Pattern and bandwidth aware file replication algorithm in grid environment[13]***

The author proposes an automated replication algorithm that allows most of I/O accesses to be performed within a given time threshold, while simultaneously minimizing the space overhead by replication. This algorithm models the replication problem as a combinatorial optimization problem, where the constraints are derived from the given access time threshold and various system parameters, while the objective function being to minimize file replication costs.
Work to be done:
As a future work, the author will continue to evaluate algorithm for larger task sets, combining both simulation and live environments, including those on the In Trigger environment using other data intensive applications.

- ***Plover: A Proactive Low – overhead file replication scheme for structured P2P Systems***

This paper presents a proactive low – overhead file replication scheme, namely Plover. By making file replication among physically close nodes based on node available capacities, plover not only achieves high efficiency in file replication but also supports low – cost and timely consistency maintenance.

Work to be done:
The author plans to explore the methods that help to fully exploit file replication for efficient lookups

- ***Study and Optimize the process of Batch small files replication[14]***

This paper analyzes and optimizes replication process for batch small files in Linux file system. In local case, six algorithms are achieved by using parallel, consecutive and aggregating policies in different stages of the whole process. In network case, achieve and compress strategies are also introduced and compared with aggregating algorithm.

Work to be done:
As the creation of the files will be involved in the implementation of allocation algorithm, therefore it is much slow and average time for each item is about 0.3 – 0.4 millisecond. These experiment results will be helpful for the further optimizing the performance of the file system.

- ***A dynamic data grid replication strategy to minimize the data missed***

The author has introduced two metrics of data availability to evaluate the reliability of the system data in the data grid system. Then the author discussed how to model the system availability problem and how to transfer this to a classic optimal problem.
Work to be done:
In future work, the author plans to study how to enhance algorithms so one can differentiate the system file missing rate and system bytes missing rate in the grid when the file size is not unique.

Table 1: Comparison of various Replication Algorithms

Parameters/ Algorithms	Availability	Reliability	Scalability	Throughput	Network Traffic	Response Time	Autonomous operation	Limitation
Less Log	High	High	No	Low	Fast	Yes	Yes	Large – scaled P2P system
APCP (Asynchronous Primary Copy Protocol)	High	High	High	High	average	Fast	Yes	-
JRM (Junction Replication	high	High	High	High	low	fast	Yes	Security

Method								
File clustering based replication algorithms	high	High	high	high	low	average	Yes	Complex workloads on live environments
DORA	high	High	high	high	average	fast	Yes	Read dominant
OPR (on line pointer replication algorithm)	high	High	high	high	average	fast	Yes	-
Access – pattern and bandwidth aware replication algorithm	high	High	high	high	high	Fast	Yes	Locality of access – patterns
Plover (Proactive low – overhead replication method)	high	High	high	High	High	fast	Yes	Efficient lookups
EDFRS (effective distributed file replication system)	high	High	Incremental l scalability	high	High	fast	Yes	-

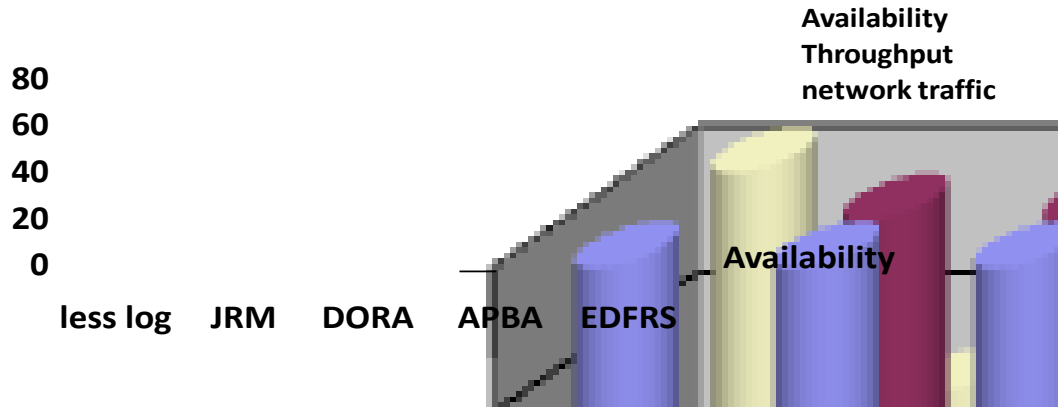

Fig 3 : Comparison bar chat of algorithm with parameter

V. CONCLUSION

As we have seen the various models of file replication. Optimistic replication approach is generally followed in DFS. This is a trade off with consistency against availability, as it allows updates to take place even in the presence of communication failures at a cost of sometimes violating single-copy serializability. Although optimistic replication systems have been in use for some time, it is seen that there is still a lack of adequate metrics. Client-Server model is generally used but when dealing with mobile users Peer-to-peer model is used. This allows direct communication and synchronization between all the peers but has scalability problems. Thus depending upon the system the replication approach and model is selected. As both have some trade off.

Acknowledgement

I am pursuing Master of Engineering in Computer Science branch. I had done my research work in the area replication algorithms in distributed computing environment. I would thank to Dinesh Jain for supporting me in this work.

References

[1] Giacomo Cabri, Antonio Corradi, Giacomo Cabri, Antonio Corradi, Franco Zambonelli "Experience of Adaptive Replication in Distributed File Systems"- Copyright IEEE. Published in the Proceedings of EUROMICRO '96, September 1996 at Praha, Chzech Republic.

[2] Gerald Popek, Richard Guy, Thomas Page, John Heidemann "Replication in Ficus Distributed File Systems"- Proceedings of the Workshop on Management of Replicated Data, November 1990.

[3] David Ratner "Roam: A Scalable Replication System for Mobile and Distributed

Computing"- UCLA Computer Science Department Technical Report UCLA-CSD-97044, January 1998.

[4] James Jay Kistler "Disconnected Operation in a Distributed File System"- Carnegie Mellon University Pittsburgh, PA 15213, May 1993

[5] Carl Downing Tait "A File System for Mobile Computing "-Columbia University 1993.

[6] Qi Lu "Improving Data Consistency for Mobile File Access Using Isolation-Only
Transactions "- School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213, May 1996

[7] A. Hisgen, A. Birrel, T. Mann, M. Schroeder, and G. Swart, "Granularity and Semantic Level of Replication in the Echo Distributed File System", *Proc. Of Workshop on Management of Replicated Data*, Houston (Nov. 1990).

[8] Muthitacharoen, R. Morris, T.M. Gil, and B. Chen, "Ivy: A Read/Write Peer-to-peer File System", *Proc. Of 5th OSDI*, Boston (Dec. 2002)

[9] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peerto- Peer Lookup Service for Internet Applications. In *Proceedings of ACM SIGCOMM*, 2001.

[10] F. Liu, X. Lu, Y. Peng, and J. Huang. An efficient distributed algorithm for constructing delay and degree-bounded application-level multicast tree. In ISPAN '05: Proceedings of the 8th International Symposium on Parallel Architectures,
Algorithms and Networks, pages 72–77, Washington, DC, USA, 2005. IEEE Computer Society.

[11] S. Tewari and L. Kleinrock, "Proportional replication in peer-to-peer
networks," in *Proc. of IEEE Infocom*, 2006.

[12] E. Cohen and S. Shenker, "Replication strategies in unstructured peerto-
peer networks," in *Proc. of ACM Sigcomm*, 2002

[13] E. Cohen and S. Shenker, "Replication strategies in unstructured peerto-
peer networks," in *Proc. of ACM Sigcomm*, 2002

[14] Giacomo Cabri, Antonio Corradi, Giacomo Cabri, Antonio Corradi, Franco Zambonelli"Experience of Adaptive Replication in Distributed File Systems"- Copyright IEEE. Published in
the Proceedings of EUROMICRO '96, September 1996

[15] Gerald Popek, Richard Guy, Thomas Page, John Heidemann "Replication in Ficus Distributed
File Systems"- Proceedings of the Workshop on Management of Replicated Data, November
1990.

