

Automated tools for manipulating files in a distributed environment with RSYNC

Alireza Ghobadi*, C. Eswaran*, Chin-Kuan Ho*, Payam Vahdani Amoli**, Roozbeh Karimi**

**Faculty of Information Technology, Multimedia University, Selangor, Malaysia*

***SOHA Sdn. Bhd. ABI, MSC Malaysia Technology Commercialization Centre, Selangor, Malaysia*

alireza.ghobadi@mmu.edu.my, eswaran@mmu.edu.my, ckho@mmu.edu.my, vahdani@soha.com.my, roozbeh@soha.com.my

Abstract— It is well known that the primary objective of a world wide business organization is to find a way to improve the delivery of services and facilities for business interactions. Advancements in information and communication technologies especially in the field of networking can be utilized to enhance the co-ordination and enforcement policies of such organizations. The information needed by an organization agency usually involves collecting information that has been captured or owned by other cooperating agencies which are usually situated in geographically separate locations. Sharing of such information typically involves manual requests which could at times be slow and may be subjected to other administrative procedures. With the use of distributed file technologies supported by sound information sharing policies, disparate information from a myriad of geographically separate organization resources could be collected and distributed as if the information is residing on the requestor's own repository.

This paper suggests a framework for file distribution in an environment where files from different geographically separate resources can be shared by the cooperating agencies. The proposed algorithm performs replication on several servers, and updates the servers when data manipulation occurs. In addition, the algorithm can reduce the transfer time for huge files significantly by using Wrapper RSYNC between two replica servers. The experimental results reveal that the proposed algorithm reduces the overall file transfer time up to 55% and improves the fault tolerance by 20%.

Keywords- Distributed File System, RSYNC, Wrapper RSYNC and File Sharing

I. INTRODUCTION

The graph of internet usage shows that the amount of data shared by business organizations grows rapidly. Many large organizations still experience high costs and long delays in transferring huge amounts of information across the network.

The two main reasons in distributed systems that are responsible for high costs are (i) redundant data or weakness to replicate data (ii) large delay and faults that occur during file transfer [1][2][3]. Replication techniques are used to solve the first problems (which are called calls fault tolerance

techniques) [4]. It often happens that a large amount of the information that is transmitted may be redundant as the recipient may already have this data. According to some researchers, the overall time required for massive data transfers are plagued by various factors such as network condition, file dispatching system at the sender entity, file retrieval system at the receiver entity, file transfer error detection and the processing capabilities of sender and receiver entities [5]. For instance, consider the case of a group of people collaborating over email to produce a large file, sending it back and receiving new updated version of this file. An analysis shows that often only small changes are effected to this file. Therefore, a dramatic reduction in bandwidth can be achieved if just the differences are communicated across the network [6].

For the first problem, depending on the replication policy adopted, several Distribute File Systems (DFS) such as NFS, AFS etc are introduced during the last fourteen years [7][8].

For the second problem, a typical solution to speed-up the overall file transmission process is to upgrade or improve the network condition through the introduction of higher bandwidth or quality of service (QoS) [9]. However, this will incur expenses that do not have a proportionate return of investment. Furthermore, frequently upgrading the processing speed and performance of the peer machines or interconnecting routers in order to speed-up the whole file transmission process will place a tremendous strain on the organization budget [10]. An option is to enhance the file dispatching and retrieval systems (file transfer applications) that they are currently utilizing for file transmission process. Examples of file transfer applications that are currently used are File Transfer Protocol (FTP), Secure Copy (SCP) and RSYNC[11][12][13].

In this paper, we propose a model for providing file distribution in a geographically separate environment which can be used to build an organization file exchange infrastructure with increasing file transmission using RSYNC with Wrapper Algorithm. An overview of the different types of distributed file systems that are currently available is presented in Sections III, IV, and V. RSYNC Wrapper Algorithm is described in section VI. A model called DAPA for supporting file exchange between organization agencies is described in

This work has been supported by SOHA Sdn. Bhd. Malaysia.

sections VII and VIII. Section IX presents the significant, experimental results obtained using DAPA algorithm. Finally, section X concludes this paper.

II. RELATED WORK

A distributed system is a system in which hardware and software components located at networked computers communicate and coordinate their actions only by passing message [14]. A distributed system can also be seen to be consisting of a set of heterogeneous processors connected by an arbitrary interconnection network [15] [16].

In a distributed system, one of the objectives of replication is to achieve availability. Our objective is to build a replication model that would facilitate file transfer between cooperating agencies using existing network infrastructure and distributed file systems.

According to [17] and [18], a distributed file system is a network file system which involves clients, servers and storage devices which are dispersed among the machines of a distributed system or intranet [19]. With this in mind, we examine three major distributed file systems for their suitability to be incorporated as part of an organization file exchange model.

III. NETWORK FILE SYSTEM (NFS)

The Network File System (NFS) is originally developed by Sun Microsystems in 1984, allowing user on a client computer to access files over a network in a simple manner as if the network devices are attached to its local disks. It is well known that NFS is an exceptionally efficient protocol for read-only small files, and login directories are excellent examples of rarely changed, concise configuration files.

However when it involves sizeable files or those that are frequently updated, NFS tries to ensure consistency by forcing all replicas to wait [7]. If NFS were to be implemented in a distributed environment, it means that files replicated across a distributed network will be updated, if any copy of the file has been modified.

The technique of forcing all replicas to wait until the update has completed can be a rather expensive operation since it will slow down the application that is accessing the file. Hence we do not find the NFS a suitable candidate to be adopted for a distributed organization environment. Some examples of network file systems or file server architectures that provide replicated copies are Locus [20], Coda [6], Ficus [21] and Harp [20].

IV. ANDREW FILE SYSTEM (AFS)

The Andrew File System (AFS), is a distributed file system which allows a potentially large set of machines across several networks to share access to a set of files in a uniform manner [8]. AFS uses the client/server model, where all the data are stored on file server machines. Files are transferred to client machines as and when necessary and cached locally. AFS files are stored in structures called volumes which reside on the file servers. Volumes that contain frequently accessed data can be

read-only replicated on several servers. The strength of the AFS lies with its Cache Managers which usually resides on users' client Work stations makes use of the replicated volumes to balance the load [8]. If a replicated copy becomes unavailable due to network problems, it will automatically access another replica copy that is closest to it. Thus physically placing the replica near to a client will improve access and performance. While the AFS seems very appropriate to be used to facilitate distributed Organization file sharing, the granularity level of replication is at volume level. This would mean that large volumes couple with many replica sites will cause very slow updates if most files on most sites needs updating. Also consistency would be an issue for non read-only replicas in such a situation. What we would like to adopt is a non-bulky, file level granularity replication that could be quickly replicated across the distributed environment.

V. GOOGLE FILE SYSTEM (GFS)

The Google File System (GFS) is a proprietary file system that is designed as a result of Google's demand for high data storage to support its core business of web searching. As such, GFS was designed to be a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients. GFS's design has been driven by observation and anticipation of Google's application requirements [22].

While sharing similar goals earlier distributed file systems, GFS has moved away from traditional assumptions in earlier distributed file systems [22].

Although GFS provides a radically new approach at handling distributed files, its proprietary nature leaves it somewhat undesirable to be included in the Organization file sharing.

By analyzing the relative strengths of each of the techniques in existing distributed file systems, we were able to put forth file system model that is simple and yet suitable for use in the Organization environment. The model which we called DAPA moves away from the idea of replicating files by the bulk or chunks or volumes but rather replicate at the granular level of files. This would remove away the excess baggage of having to replicate large volumes across network but rather only the affected file. In our opinion, replicating at file levels could improve reliability and ease the implementation and maintenance of the system supporting this model.

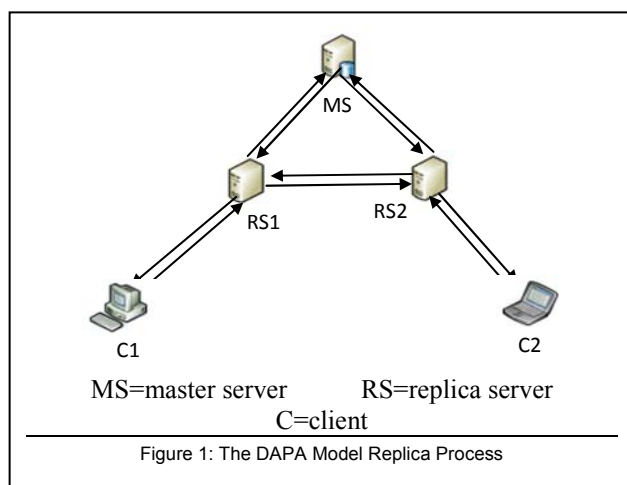
VI. RSYNC WRAPPER ALGORITHM

A Wrapper Algorithm that improves the overall transfer time of large files by compressing the files and splitting the files into smaller chunks of appropriate sizes before they are processed by the RSYNC file transfer application[5][11]. This algorithm tries to reducing file transferring time between two servers.

VII. ALGORITHM AND IMPLEMENTATION

The DAPA model is a model for file sharing across a distributed environment. DAPA typically embraces the principles of distributed file system. A major difference between DAPA and other distributed file system is in the way of its implementation. DAPA supports file sharing at the granular level of files rather than volumes and uses replication to distribute files across geographically separated networks. It makes use of a master server and replica servers to ensure the effective and reliable distribution of files across networks. The following Figure 1, conceptually describes the DAPA model.

In Figure 1, MS refers to the Master Server while RS1 and RS2 respectively refer to Replica Servers. In an organization environment, the Master Server would serve as the coordinating server for all cooperating Organization agencies while RS1 and RS2 would represent cooperating organization agencies.



Although this is a simplistic model, the number of Master Servers and Replica Servers can be increased to support more cooperating agencies.

To illustrate the workings of the model, we assume a simplistic environment of two cooperating Organization agencies which are located in two geographically separated areas. RS1 and RS2 represent the replica servers which are serving agency A and B respectively. C1 refers to a client nearest to RS1 at Organization agency A and C2 refers to a client of Organization agency B nearest to RS2. Client 1 initiates a file creation meant for distributing and sharing to other Organization agencies. The file is then saved in the Replica Server for storage. Upon receiving a new file, the Replica Server updates its local registry with the details of the new files as shown in Figure 2.

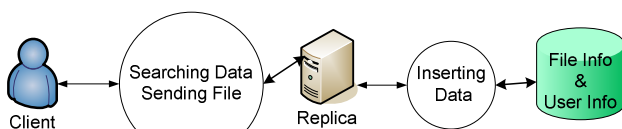


Fig 2. Client-Replica Process

A message is also sent to the Master Server, informing that the Replica Server now has a new file in its possession. The Master Server upon receiving the message will update its registry of files indicating the filename and its location as shown in Figure 3.

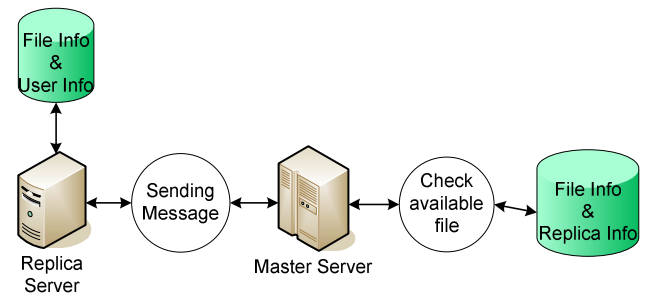


Fig 3. Master Server updated with new details

The Master Server will then send a message to all other Replica Servers as shown in Figure 4 that there is a new file in total holding. The message communicated to other Replica Servers contains a token for verification as well as the location of the file. Communications between two Replica Servers will comprise of verification as well as request for the file to be retrieved. The file delivered to other replicas will now be fortified with access control rights inherited from the originating replica to ensure that appropriate access rights is preserved during file sharing. Only the owner of the file is able to set the access permissions. Physically, the file now exists in both Replica Servers. If there are more than two Replica Servers, the same process is repeated throughout, until all the replica servers are populated with the. There is Neural Network process if there are more than two replicas which will test on future.

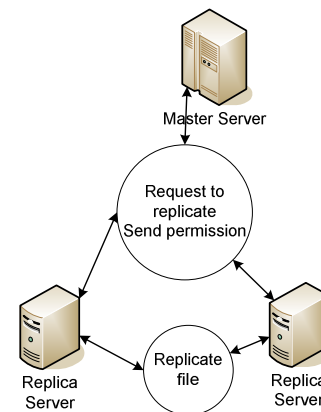


Figure 4 . File replication among replica servers

Subsequently, when there is a request from any client to access a file from any replica server, the replica server will first check if the file exists in its repository. If it does, the replica server will determine the user's access rights to the file. If the user has a read-only access to the file, the user will only be allowed to view the file. However if the user would like to modify the file, he or she is allowed to do so by renaming the file with a new name thus initiating the process of new file creation. This will result in the replica communicating with the

Master Server and eventual distribution to other replica servers. If the user has write permissions to the file, the Master Server will be notified as soon as the user opens the file. In this case, the replica server keeps track of the file and if the file is saved, the replica server will notify the master server that a new version of the file exist and the new version will be replicated again to all replica servers. If the replication happens when another user is currently accessing the same file from a different site, the user will be notified that a new version of the file exists.

The main phases of the proposed algorithm are as follows:

Initial Setup

1. Back-propagation based ANN training to find the nearest server to manipulate.

At the Replica Server Entity

Phase 1: Initialization at the Server Entity

1. Get Client ID and Password and check permission.
2. Send Message permission to Client.
3. Create temporary folder.

Phase 2: Encrypt file data

1. Get files from client and pass to Temporary folder.
2. Encrypt Data according to Data, Client ID, the file name, and the number of the same file.
3. Check Encryption Data with database.
4. Find a position of the file and Transfer file to position.
5. Update Replica database
6. Send a message to Master Server.
7. Get permission from Master server to replicate file.
8. Collect all the details or conditions of the environment: file size, available bandwidth, achievable bandwidth, distance between peers, file transmission time and frequency of the error occurrences and average time for error handling process.
9. Pass the selected conditions to the ANN and determine the file-split constant, k.
10. Compare the file size with the file-split constant and split the file into file chunks.

Phase 3: Transfer File from Client to Replica Server.

1. Find a position of the file and Transfer file to position.
2. Update Replica database.
3. Use First Come First Serve or Matrix based scheduling algorithm for scheduling the file chunks to RSYNC.
4. Send the relevant information (initialization message) to the receiver entity.

Phase 4: Master Server updating

1. Send a message to Master Server.
2. Get permission from Master server to replicate file.

Phase 5: Replicate File to another replica

1. Pass data to ANN process to find the nearest Replica Server.
2. Send a message to update.
3. Get a permission from other replica

4. Send Update data message.
5. Receive acknowledgement from the receiver entity.
6. Transfer file chinks using RSYNC according to the selected scheduling algorithm.
7. Send a file to Replica server.
8. Send a compilation message to Master Server.

At the Master Server

Phase 6: Initialization at the Receiver Entity

1. Receive the initialization message from Replica Server.
2. Update Data on Database.
- 2.1 Update file receive completion and insert to database
- 2.2 Received replicate messages go to Phase 6.

Phase 7: ANN

1. Receive The Replica Location.
2. Send the location as input to ANN process.
3. Get nearest Replica server as output.
4. Send a message to replica which has a file and replica which need a file to transfer file.

VIII. STRENGTHS OF THE DAPA MODEL

The DAPA model is specifically adapted from other distributed system models to support the file sharing requirements in an Organization environment. It is lightweight and does not carry the bulky volume based replications of other major distributed file systems. Because it is designed for file sharing in an Organization environment, replication is downscaled at the granularity of the file level.

The design of the DAPA model makes it scalable depending on the implementation requirements of specific applications. This model supports authentication and verification of replication sites such that only valid and participating replica servers are allowed to receive files based on an authentication and permission token provided by the master server.

The DAPA model is also added with a failsafe feature such that the Master Server can also be replicated and in the event of a failure of any master server, the replicated master server will take over control.

For understanding of performance and differencing between of this method and another three most important models you can see table 1.

Models	AFS	NFS	GFS	DAPA (New Model)
Replication Method	Replica's by copy of the whole data from one replica server to another.		Use chunk server and master sever for control replication and data flow to get better performance.	Use Replica server and master sever for control replication and data flow and Use Database to replica just new or edit file white some clients from one replica server to another.

Table 1: Comparison between NFS, AFS GFS, and DAPA Base on Replication

IX. EXPERIMENTAL RESULTS AND ANALYSES

The DAPA model was implemented in a file sharing application as part of the research project within Multimedia University. The master server module and replica server modules were developed using JAVA and implemented on the LINUX environment supported by MySQL database. The client modules were implemented on both LINUX and MICROSOFT WINDOWS environment. At the moment, the interface is very much command based. It was tested under the same setup and configuration shown in Fig. 1. The experiments were conducted using four files of various sizes (2GB, 4GB, 8GB and 10GB). These four files are existing in each replica servers.

This experiment is replicating of updating file between NFS, AFS, GFS, and DAPA model. This experimental using time of replicating completion calculate. Fig 5 shows the resulting file replicate throughput in seconds (S) and file transfer rate in Megabytes per second (MB/s) for the above mentioned seven files.

File replicate time is computed when all replica server update and ends when Master Server receives the acknowledgement from the last replica, indicating the completion of replication process. Fig. 5 shows that the DAPA Model increases the efficiency of Replication process. For a 2GB file, replicating completion required 167 seconds, whereas replicating completion with AFS required 2112 seconds, NFS required 1893 seconds and GFS required 543.

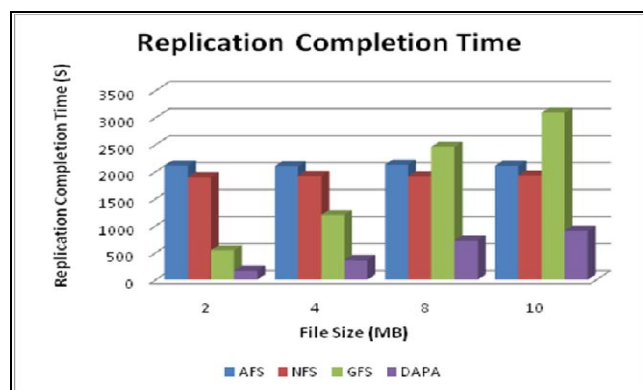


Figure 5: Replication Completion time

The DAPA model highly reduces the impact of replicating completion time when the small files updates. Let's analyze the performance result using two cases:

Case 1. First assume that small update accrues. We update files with 2GB size. According to Table 1. AFS and NFS transfer all of the files (24 GB) to other replica. The DAPA model just transfer files which updated by user. This model is useful when the file which updated is small. Fig.5 shows that DAPA model decreasing replicating completion time up to 52.6% when file size is 2GB. Also this model is better than GFS model because using RSYNC with Wrapper algorithm to reduce time transferring.

Case 2. Second assume that the most of the files is update. GFS does not have problem look like AFS and NFS but still replicating completion time is higher than DAPA model. This

DAPA system using new algorithm base on RSYNC called Wrapper algorithm and reducing time of file transferring.

X. CONCLUSION AND FUTURE WORK

In this paper, first we analyzed the performance of AFS, NFS, and GFS. We noticed that AFS and NFS replicate all of files from one replica server to another. Though GFS is able to solve this problem, it is good only for small files. It has been shown that the proposed DAPA model and RSYNC (with Wrapper Algorithm) can considerably improve the Replicating completion time. The adaptation of this model onto GFS results in up to 50% of improvement compared to GFS. Other measurement parameters are overhead time of the file transfer application. In the proposed model, disk utilization is measured in Replica and Master Server.

The DAPA Algorithm can be extended to other distributed environments to improve accuracy and replication methodology and to decrease the transfer. The algorithm can be improved to handle files to several replications at the same time.

The DAPA model can be extended to four Replicas with a deferent environment to examine and improve efficiency of the model and neural networks can be employed to find the nearest replication. Also it is necessary to test by IOZone and Postmark distributed file system evaluator. In addition,

REFERENCES

- [1] A. Carzaniga, "Design and implementation of a distributed versioning system", Politecnico di Milano, Milan, Italy, Tech. Rep. 98-88, Oct. 1998.
- [2] W. Varavithya, V. Esichaikul, "The Design of Citizen Advisory Service to Support Citizens in Accessing e-Government Services", Conference on Information and Communication Technologies: From Theory to Applications, Page(s): 11 – 12, 2004.
- [3] A. V. Date, "Implementation of Distributed Database and Reliable Multicast for Distributed Conferencing System Version 2," Master's Thesis (M.S.), University of Florida, 2001.
- [4] R. Friedman and E. Hadad, "Analyzing Distributed-System Performance: Latency vs. Throughput", IEEE Distributed Systems Online, vol. 7, no. 1, pp. 1, Jan. 2006.
- [5] A. Ghobadi, C. Eswaran, Nithiapidary, Y. L. Kee, "An Adaptive Wrapper Algorithm for File Transfer Applications to Support Optimal Large File Transfers", The 11th International Conference On Advance Communication Technology, South Korea, February 2009, Volume I, P.P. 315-320.
- [6] M. Satyanarayanan, "A Survey of Distributed File Systems," Annual Review of Computer Science, Vol. 4, 1990, pp. 73-104.
- [7] B. Pawlowski, S. Shepler, C. Beame, B. Callaghan, M. Eisler, D. Noveck, D. Robinson, and R. Thurlow, "The NFS Version 4 Protocol," Proceedings of the 2nd International System Administration and Networking Conference (SANE2000), Maastricht, The Netherlands, May 2000.
- [8] M. Satyanarayanan and M. Spasojevic, "AFS and the Web: Competitors or Collaborators," Proceedings of the Seventh ACM SIGOPS European Workshop, Connemara, Ireland, 1996.
- [9] S. Ganguly, A. Sen, G. Xue, B. Hao, and B.H. Shen, "Optimal Routing for Fast Transfer of Bulk Data Files in Time-Varying Networks", IEEE International Conference on communications, 2004.
- [10] R. Klischewski, "Information Quality as a Common Ground for Key Players in e-Government Integration and Interoperability", pp 1-3, 2006.
- [11] "RSYNC", [online], Available: <http://en.wikipedia.org/wiki/Rsync>

- [12] L. A. Grieco, and S.Mascolo, "Performance Evaluation and Comparison of Westwood+, New Reno and Vegas TCP Congestion Control", ACM CCR, 34(2), April 2004.
- [13] C. Grant, 1998, "FTP Reviewed", RFC 959.
- [14] M. Satyanarayanan, "A Survey of Distributed File Systems," Annual Review of Computer Science, Vol. 4, 1990, pp. 73-104.
- [15] A. Silberschatz, P. Galvin, and G. Gagne, Operating System Concepts, Seventh Edition, John Wiley & Sons, Inc., NY, 2006.
- [16] P. S. Yeager, A distributed file system for distributed conferencing system, University of Florida, 2003 Paul J.Fortier, "Web Based e-government Data Distribution", 2001.
- [17] R. G. Guy, J. S. Heidemann, W. Mak, T. W. Page, Jr., G. J. Popek, and D. Rothmeier, "Implementation of the Ficus Replicated File System", Proc. 1990 Summer USENIX Conf., pp. 63-71 ,1990.
- [18] P. Reiher, J. S. Heidemann, D. Ratner, G. Skinner, and G. J. Popek, Resolving File Conflicts in the Ficus File System, Proc. 1994 Summer USENIX Conf., pp. 183-195 ,1994.
- [19] J. H. Howard, M. L. Kazar, S. G. Menees, D. A. Nichols, M. Satyanarayanan, R. N. Sidebotham, and M. West, "Scale and Performance in a Distributed File System", ACM TOCS, Vol. 6(1), pp. 51-81 ,1988.
- [20] B. J. Walker and S. H. Kiser, "The LOCUS Distributed Filesystem", in The LOCUS Distributed System Architecture, ed. Gerald Popek and Bruce J. Walker, The MIT Press Series in Computer Systems, Herb Schwetman, gen. ed., The MIT Press ,1985.
- [21] B. Liskov, S. Ghemawat, R. Gruber, P. Johnson, L. Shriram and M. Williams Replication in the Harp file system. In 13th Symposium on Operating System Principles, pages 226–238, Pacific Grove, CA, October, 1991.
- [22] S. Ghemawat, H. Gobioff and S. T. Leung, "The Google File System", 19th ACM Symposium on Operating Systems Principles, Lake George, NY, October, 2003.