De-replication: A Dynamic Memory-aware Mechanism

Manu Vardhan, Paras Gupta, Dharmender Singh Kushwaha
Department of Computer Science and Engineering
Motilal Nehru National Institute of Technology Allahabad
Allahabad, INDIA
e-mail: {rcs1002, cs1006, dsk}@mnnit.ac.in

***Abstract*—Resource replication in distributed environment produces issues of secondary storage. De-replication of resources is required when replication mechanism is hindered due to lack of secondary storage. This paper introduces de-replication approaches that depend upon last modification time, number of replica available and resource size. Comparative study shows that de-replication can be used to overcome the space overhead issue and reduces the de-replication time. Result shows that in case the space required is same but number of files to be de-replicated varies, de-replication time also varies depending on number of files to be de-replicated. De-replication time will be more for case having large number of files. With the proposed approach, if file size increases by the multiple of 7, de-replication time will get increase just by the multiple of 1.5. This shows that de-replication time is decoupled from size of files that are de-replicated on the fly dynamically and does not increase proportionally with respect to file size.**

Keywords-De-replication; Distributed Systems; Replication

I. INTRODUCTION

Use of computer systems and Internet is becoming the part of day to day life, with the increasing demand for the services provided by them. To fulfill the requirement of services requested by an individual, service availability is an important issue. Distributed systems provide the environment to various experts, where services, resources and information are distributed and can be accessed by the members of that environment, as compared to the centralized systems.

A basic definition of a distributed system in [1] is that a distributed system is a collection of independent entities that cooperate to solve a problem that cannot be individually solved. This is a term that describes a wide range of computers, from weakly coupled systems, such as wide area networks, to strongly coupled systems, such as local area networks, to very strongly coupled systems such as multiprocessor systems [2].

Replication is a mechanism of service or resource placement to provide their availability in case of unavailability of resources and services. Replication is how to replicate data and request actors using adaptive and predictive techniques for selecting where, when and how fast replication should proceed [3].

De-replication is a mechanism to de-replicate / garbage-collect data or request actors and optimizes utilization of distributed storage based on current system load and expected future demands for the object [3].

De-replication is done to optimize the utilization of storage space when the demand for a resource arises. The file to be de-replicated must be carefully taken into consideration of the future demands of a file. File currently being serviced cannot be de-replicated. The number of previously replicated files selected for de-replication can fulfill the requirement for storage space need of the upcoming file to be replicated. De-replication is considered as a part of resource management process where as replication is considered as a part of resource placement process.

The rest of the paper is organized as follows. The next section discusses a brief literature survey of existing theories and work done so far. Section III discusses the problem definition. Section IV describes the proposed solution, followed by the results and discussion section. Finally, Section V concludes the work followed by references.

II. RELATED WORK

Various resource management policies and mechanisms are globally available that represent a step towards the adaptive resource management techniques, thus improving the utilization of resources, which results in improving the overall performance of the system by reducing several overheads. Venkatasubramanian [3] discusses about the security and timeliness application requirements using a customizable and safe middleware framework called CompOSE|Q. He describes the design and implementation of CompOSE|Q, which is a QoS-enabled reflective middleware framework. Also, to improve the performance of the system in the field of continuous media application, resource management technique is helpful in improving the utilization of resources. Chou [4] describes various resource management policies on threshold basis in context of continuous media (CM) servers in the area of multimedia application. Venkatasubramanian[5] discusses the two replication policies, these are static and dynamic. The division is based upon the number of copies of a file which is termed as degree of replication. In static replication policies, the degree of replication is constant, while dynamic replication policies allow it to vary with time. Santry [6] identified four file retention policies for Elephant and have implemented these policies in their prototype. The policies are viz. Keep One, Keep All, Keep Safe and Keep Landmarks. Keep One provides the non versioned semantics of a standard file system. Keep All retains every version of the file. Keep Safe provides versioning for undo but does not retain any long term history. Keep Landmarks enhances Keep Safe to also retain a long-term history of landmark versions. Hurley and Yeap [7] propose a file de-replication method based on beta time interval that decides the frequency of invoking the de-replication operation. Over time, all files will eventually be candidates for migration/replication. Although many exist, the one we choose is as follows: every beta time units (where beta is a uniform time interval which defines the time between de-replication events), storage sites will decide which file qualifies for de-replication. The de-replication policy chosen applies the least recently used concept (i.e., the file selected for de-replication is the file which was not requested for the longest period of time at the storage site). Once the file has been selected, it will be removed from this storage site. Using beta, it is possible to create a variety of de-replication policies: the smaller the value of beta, the greater the frequency of de-replication, and the larger the value of beta, the longer a file copy remains in the system. Resource replication is basically of two types, active and passive. In passive replication, all the resources are fixed in advance depending upon the application requirement. In active replication, mutual information about the peer nodes is maintained and the replicated resources can be accessed at any site. The traditional resource replication is passive and does not participate in the decision on when to replicate, where to replicate and the number of copies to replicate. In a blind-replica service model proposed by Tang [10], request routing is independent of where the replicas are located. Each replica simply serves the requests flowing through it under a given routing strategy. Various replication strategies have been proposed on the basis of the relative popularity of individual files, based on their query rate. Helen [8] proposed a query-based file popularity approach for replication. Common techniques include the square-root, proportional, and uniform distributions. File clustering-based replication algorithm in a grid environment is proposed by Hitoshi [9], which presents the location based replication mechanism. The files stored in a grid environment are grouped together based on the relationship of simultaneous file accesses and on the file access behavior.

III. Problem Definition

During replication, when a File Replicating Server (FRS) creates a replica of file on the peer nodes, space management issue arises, i.e., whether space is available or not in the secondary storage of the peer nodes on which the file needs to be replicated. If space is available, the file will get copied, but if space is not available de-replication of previously replicated files needs to be done in the secondary storage of that peer node.

De-replication of files will take place in a manner such that it will fulfill the size requirement of upcoming files. While maintaining the space management overhead, decision for deleting a file, depends on three criteria that are discussed in Section III-A.

A. Parameters to be Used

Solution to this problem will be represented on the basis of three parameters of a file which are last modification time of the file, number of replica available of a file and file size.

- *Last Modification Time of a File:* Last modification time is the time at which the file was last modified or last used.
- *Number of Replicas Available of a File:* Number of replicas available of a file is a count on number of copies available for a particular file. Whenever a copy of file is created, it will increase the number of replicas available of a file.
- *Size of a File:* File size is the size of a file required on a disk.

IV. Proposed Solution

With everything being lodged on Internet, computing paradigm is changing fast to harness this capability. Many information servers and files are resident on various machines and this can be effectively utilized by the users. We present a scenario discussed in Section IV-A, although on a smaller scale where geographically disparate clusters interact with each other for information sharing through replication. Each of these clusters are owned by respective organizations.

In our proposed model, we talk about space overhead in replicating file on the storage site. If space is available, the file will get replicated; otherwise, de-replication of previously replicated files needs to be done in that directory.

A. Architecture Used

One node in each cluster is designated as FRS. FRS can also be replicated on some other node in the cluster for backup and recovery. The scenario presented in the paper is illustrated in Figure 1 and is elaborated subsequently.

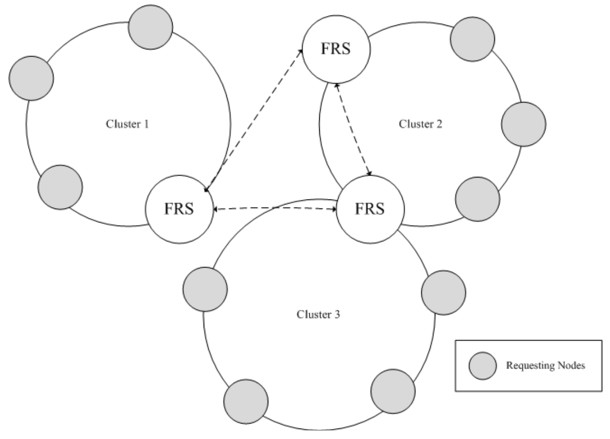

Figure 1. Architecture

The proposed architecture consists of loosely coupled systems, capable of providing various kinds of services like replication, storage, I/O specific, computation specific and discovery of resources. Based on the application requirement, the resources are made available to other nodes. Figure 1 shows a network of three clusters that are connected to each other via intercommunication network. Each cluster consists of a group of trusted nodes and a File Replicating Server (FRS) assigned to these nodes. A FRS can be 'local' or 'remote'. A FRS is assigned to a subset of nodes known as local FRS and FRS positioned outside that cluster, will be called as remote FRS. Each subset of nodes (denoted as requesting nodes) receives the list having IP-address of remote FRS, to increase fault tolerance capability. But, the nodes of a cluster will send the file request only to the local FRS. In case of the failure of the local FRS, a node can automatically select a remote FRS from the list and file request will be routed to the selected remote FRS. This makes the model robust and capable of handling crashes in case of local or even remote FRS fails. The system will keep functioning under all circumstances and will never come to halt. Each FRS maintains two tables:

- File request count table with the following attributes: <file_id, file_name, request_count, meta data>.
- Peer FRS table with the following attributes: <FRS_IP, FRS_PORT>.

Each FRS is informed whenever a new FRS is added to the network, to updates its peer FRS table. FRS does not monitor and maintains the status of remote FRS, instead FRS request for the current status of remote FRS on-demand. FRS status can either be 'busy' or 'ready'.

Threshold based file replication works as follows:

Each local FRS is responsible for accepting the file request and based on its current status (checks if the number of requests currently serving for a particular file is below the threshold or not), in the following manner:

- If the status of local FRS is 'ready', the local FRS will fulfill the request.
- If the status of local FRS is busy, it looks for a remote FRS that can handle the request, by one of the following manner, described as under:

The local FRS contacts the remote FRS that can handle the request by the available copy of the requested file i.e. the status of remote FRS is ready. If not so, the local FRS contacts those remote FRS on which the requested file is not available. In that case file replication will be initiated, by the local FRS of the cluster and the file replica will be created on remote FRS on which the file is not available. For both the cases mentioned above, IP address of the remote FRS that can handle the request will be send to the requesting node. On receiving the IP address, the requesting node will connect to the remote FRS and receives the file, without any user intervention. Thus the overhead of polling and broadcasting is reduced.

B. *Approaches Proposed for De-replication*

De-replication of files will take place in a manner such that it will fulfill the size requirement of upcoming files. While maintaining the space management overhead, three approaches for file de-replication are discussed below.

1) Last Modification Time Based Approach: In this approach, files are sorted on the usage basis file that was not requested for longest period of time will be selected for de-replication. A drawback of this approach is that if only one requested file is there before deletion, it causes loosing of information. So, a check is performed before de-replicationwhich will be done on number of replica available basis approach.

2) Number of Replicas Available of a File Based Approach: In this approach, files having many copies or the files with more than one replica are de-replicated only when there is not sufficient space available for new replicated files. Files with one replica are not de-replicated to avoid losing information of the file. In this case, before the de-replication of file, a check is performed, whether or not there are other copies of file available or not. If only single copy of file exists in the system, in that case next probable file for de-replication will be selected from the sorted file list on the basis of last modification time.

3) File Size Based Approach: File size based de-replication approach is used when time required for de-replication considered as important factor. When there is a very little difference in the last modification time of the two files and number of replicas available of both files is more than one, de-replication of file with minimum file size among them will take place to avoid the delay in the process and complete it in the less time.

The proposed approach for de-replication will be described in Figure 2. The detailed description of the number labeled arcs will be described in sequential manner as follows:

1. *Node A of cluster$_1$* sends connection request to *FRS$_1$*.
2. *FRS$_1$* sends ip addresses of peer FRS and resource list to n*ode A of cluster$_1$*.
3. *Node A of cluster$_1$* sends request for file *f$_1$* to *FRS$_1$* at time *t$_0$*.
4. *Node A of cluster$_1$* starts receiving requested file *f$_1$* from *FRS$_1$*.
5. *Node D of cluster$_1$* sends connection request to *FRS$_1$*.
6. *FRS$_1$* sends IP addresses of peer FRS and resource list to n*ode D of cluster$_1$*.
7. *Node D of cluster$_1$* sends request for same file *f$_1$* to *FRS$_1$* at time *t$_1$*.

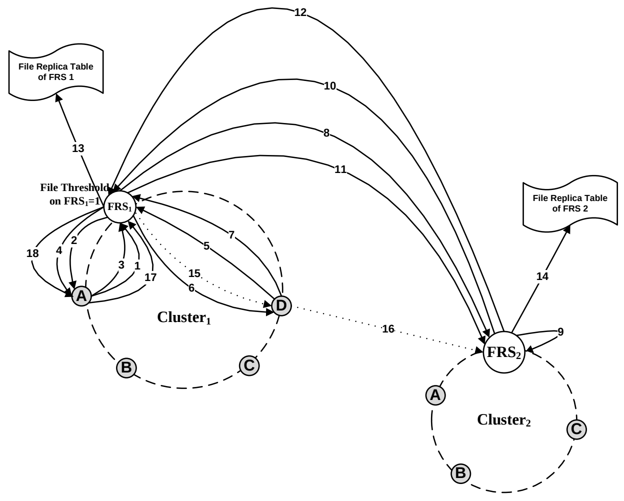

Figure 2. Proposed Model

8. As *FRS₁* can fulfill only one request at a time because the threshold value for a particular file on *FRS₁* is *1*, so *node D of cluster₁* will receive the requested file from another *FRS* in the system, here *FRS₂*, to fulfill its request. To fulfill the request of *node D of cluster₁*, replication of requested files is initiated by *FRS₁* as the requested file is not present on *FRS₂*. This is because the FRS does not maintain any information about the "requesting node (e.g. *node D*)" at any point of time. So *FRS₁*will replicate the requested file to other FRS as its shared resource information is being maintained, as discussed in section IV-A. Now, *FRS₁*sends the size of the file to be replicated to *FRS₂*.
9. *FRS₂* does not accept the file replication request because of space/storage scarcity. *FRS₂* initiates de-replication operation on set of previously replicated files. The required amount of space is made available on *FRS₂*. If the secondary storage on *FRS₂*did not contains any replicated files then user interruption will come, as de-replication of non-replicated file is not allowed.
10. *FRS₂* sends message 'ready to receive file f_1' to *FRS₁*.
11. *FRS₁* starts replicating the file f_1 to *FRS₂*.
12. *FRS₂* sends message 'replication of file $f_1$to be done successfully' to *FRS₁*.
13. *FRS₁* updates its file replica table.
14. *FRS₂* updates its file replica table.
15. *FRS₁* sends IP address and port of *FRS₂* to n*ode D of cluster₁*informing that the file f_1 is now available on *FRS₂*.
16. Request of *node D of Cluster₁* for file f_1 will now be fulfilled by peer FRS, *FRS₂*.
17. After some time *node A of cluster₁* request same file $f_1$from *FRS₁*.
18. In case file with the same name already exists on the n*ode A of Cluster₁*,file de-replication will be done on that node then the file transfer from *FRS₁* to *node A of cluster₁* will be initiated.

C. Stability Analysis

According to Figure 3, the communication between a requesting node and a FRS (*Source* A and FRS_1) is described as follows: *Source A* sends a file request to FRS_1through$\overline{M_1}$.FRS_1 will receive the request of *Source A* represented as M_1. In return, FRS_1 sends file to *Source A* shown by M_3 received on *Source A* using $\overline{M_3}$.

The total communication between requesting node *Source* A and FRS_1 with internal actions (τ) will be given by equation 1 as follows:

$$SourceA \overset{\text{def}}{=} \overline{M_1}.M_3.\tau.SourceA \qquad (1)$$

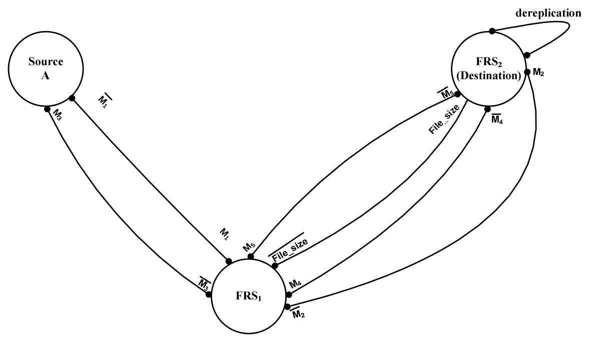

Figure 3. File De-replication Model Flow Graph in Process Algebraic Approach

Also as shown in Figure 3, communication between the two existing FRS in the architecture (FRS_1 and FRS_2) is described as follows: FRS_1 will send file size of the file to be replicated using$\overline{File_size}$ which will be received at FRS_2 end by$File_size$. When file size is received by FRS_2, it initiates de-replication operation on set of previously replicated files which will be represented by $\tau.derplicate\_file$, which is file de-replication with internal actions (τ). After the successful completion of de-replication operation, the required size for replication will be available on FRS_2. Now, FRS_2 will send 'ready to receive replicated file' message to FRS_1 represented through$\overline{M_4}$. FRS_1 received this message using M_4. After receiving the message, FRS_1 will send the file to be replicated to FRS_2 represented by message $\overline{M_2}$. FRS_2 will receive the file send by FRS_1 represented as M_2. When the file will be replicated successfully on FRS_2, it will send a message 'successful replication done' to FRS_1 by$\overline{M_5}$which was received by FRS_1 using M_5.

1) Illustration of State Transition of Source Node: As shown in Figure 3, FRS_1 will act as a source node. Status change illustration of source node (FRS_1), as shown in Figure 4, will be described as follows:

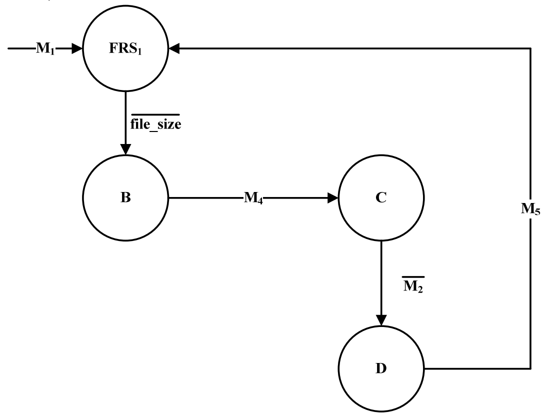

Figure 4. State Transition Diagram of Source Node (FRS_1)

After the action of sending file size of the file to be replicated through message $\overline{File\_size}$, source node (FRS_1) transit to state B of FRS_1 shown in equation 2,

$$FRS_1 \overset{\text{def}}{=} \overline{File\_size}.B \quad (2)$$

State B of FRS$_1$ switch to state C of FRS$_1$ through message M$_4$, which represents the action of receiving 'ready to receive replicated file' message by FRS$_1$shown in equation 3,

$$B \overset{\text{def}}{=} M_4.C \quad (3)$$

State C of FRS$_1$ switch to state D of FRS$_1$ through message $\overline{M_2}$ which represents the action of sending the file to be replicated by FRS$_1$ shown in equation 4,

$$C \overset{\text{def}}{=} \overline{M_2}.D \quad (4)$$

State D of FRS$_1$ upon the action of receiving message 'successful replication done' by FRS$_1$through message M$_5$ switch to starting state FRS$_1$ shown in equation 5,

$$D \overset{\text{def}}{=} M_5.FRS_1 \quad (5)$$

From the equations 2, 3, 4 and 5 of various states of FRS$_1$, we can build the definition of FRS$_1$, which is defined as by the equation 6:

$$FRS_1 \overset{\text{def}}{=} M_1.\overline{File\_size}.M_4.\overline{M_2}.M_5.\overline{M_3}.FRS_1 \quad (6)$$

*2) Illustration of State Transition of Destination Node:*As shown in Figure 3, FRS$_2$ will act as a destination node. Status change illustration of destination node (FRS$_2$) as shown in Figure 5 will be described as follows:

After the action of receiving file size of the file to be replicated through message$File_siz$, destination node (FRS$_2$) transit to state E of FRS$_2$ shown in equation 7,

$$FRS_2 \overset{\text{def}}{=} File\_size.E \quad (7)$$

State E of FRS$_2$ switch to state F of FRS$_2$ through message $\overline{dereplicate\_file}$ which represents the action of de-replication on previously replicated files by FRS$_2$ shown in equation 8,

$$E \overset{\text{def}}{=} \overline{dereplicate\_file}.F \quad (8)$$

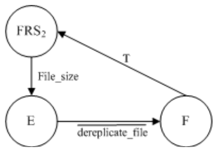

Figure 5. State Transition Diagram of Destination Node (FRS$_2$)

State F of FRS$_2$ upon some internal actions (τ) by FRS$_2$ switch to starting state FRS$_2$ shown in equation 9,

$$F \overset{\text{def}}{=} \tau.FRS_2 \quad (9)$$

From the equations 7, 8 and 9 of various states of FRS$_2$, we can build the definition of FRS$_2$ which is defined as by the equation 10:

$$FRS_2 \overset{\text{def}}{=} File\_size.\tau.\overline{dereplicate\_file}.\overline{M_4}.M_2.\overline{M_5}.FRS_2 \quad (10)$$

From the equations 1, 6 and 10, we can build the complete system as defined by the equation 11:

$$FDM \overset{\text{def}}{=} Source \| FRS \| Destination \quad (11)$$

V. RESULTS AND DISCUSSION

To overcome from the overhead of space management issue, a data structure consisting of a table considered which is described in Table 1. The proposed model is simulated on linux platform and LAN having speed of 100.0 Mbps.

TABLE I. ATTRIBUTES

Attribute Name	Type
Last Modification Date	yyyy-mm-dd
Last Modification Time	hh:mm
File Name	String
File Size	Long
File replica	Integer

Replicated files on the storage site will be sorted based on least recently used parameter which will be obtained using the combination of both last modification date and last modification time. The list of replicated files will be sorted in descending order. Example of a data structure of available files maintained at the storage site is described in Table 2.

TABLE II. DATA STRUCTURE EXAMPLE FOR COMPARISON BETWEEN APPROACHES

Last Modification Date	Last Modification Time	File Name	File Size (in MB)	File replica
2011-12-21	20:08	a.mp3	3	4
2011-12-08	22:48	b.mp3	500	1
2011-11-23	16:36	c.mp3	100	2
2011-11-23	16:03	d.mp3	250	1
2011-11-09	20:11	e.mp3	50	1
2011-11-09	18:47	f.mp3	5	4
2011-11-09	18:43	g.mp3	10	2

The Figure 6 plots efficiency of all the three approaches versus load based on the data shown in Table 2 and the three approaches based on least recently used parameter, replica counts and file size parameters. Efficiency calculated is proportional to the reciprocal of extra memory size vacated during de-replication based on low, low-medium, medium-high and high load for which range of file size (R_f) is R_f<20MB, 20MB<R_f<60MB, 60MB<R_f<100MB, and R_f>100MB, respectively.

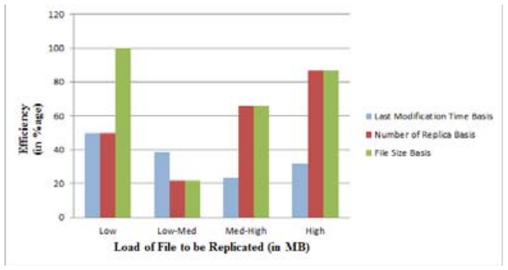

Figure 6. Comparison of the Three Approaches

Unlike 2nd and 3rd approaches (i.e., number of replica available of a file basis and file size basis respectively), 1stapproach(i.e. last modification time basis) is based only on least recently used parameter and disregards the replica counts and file size parameters. Thus it may even delete the last replica of file present in system. While 2ndapproach is based on both least recently used and replica counts parameters and disregards the file size parameter. The 3rd approach is based on all the three parameters, least recently used, replica counts parameters and file size parameter. Most of the time, the percentage efficiency of the 2nd and 3rd approach is equal and better than of the 1st approach, except in case low-medium load. Only in case of low load percentage efficiency of 3rdapproach is better than 2ndapproach. All the three approaches said to be 100% efficient only when space required before and after de-replication is exactly the same.

De-replication time increases, as the number of files not accessed for the longest period and smaller in size, are more as compared to the files that are larger in size. Table 3 shows when the space required is same but the number of files to be de-replicated varies, de-replication time also varies depending on the number of files to be de-replicated. De-replication time will be more for the case having large number of files. Table 3 shows that if file size increases by the multiple of 7, i.e., from 6 MB to 43.9405 MB, de-replication time will get increase by the multiple of 1.5, i.e., from 60millisecond to 98 millisecond. This shows that the de-replication time is decoupled from the size of files that are de-replicated dynamically and does not increase proportionally with respect to the file size.

TABLE III. DE-REPLICATION TIME IN REQUIRED SPACE

Number of Files de-replicated	Space Required (in MB)	Space Freed (in MB)	De-replication Time (in msec)
1	6	6.0523	60
2	7.8607	13.1792	75
3	20.0399	21.0399	77
3	36.2634	39.7985	79
5	36.2634	59.7151	96
5	43.9405	51.0140	98

Finally, equation 11 establishes a relationship between the formal aspect of file de-replication server and its architectural model through process algebra approach. The stability analysis ensures that the system will run in the finite sequences of interaction and transitions. On the basis of these equations, a transparent, reliable and safe file de-replication model is build.

VI. CONCLUSION

This paper proposed an approach that tackles the issue of space overhead in a distributed system environment. The proposed solution resolves this issue of space overhead. De-replication time increases, as the number of files increases that are not accessed for the longest time period and smaller in size as compared to the files that are larger in size. Result shows that, in case when the space required is same but the number of files to be de-replicated varies, de-replication time also varies depending on the number of files to be de-replicated. De-replication time will be more for the case having large number of files. With the proposed approach, if file size increases by the multiple of 7, de-replication time will get increase just by the multiple of 1.5. This shows that the de-replication time is decoupled from the size of files that are de-replicated on the fly dynamically and does not increase proportionally with respect to the file size.

REFERENCES

[1] A. D. Kshemkalyani and M. Singhal, "Distributed Computing: Principles, Algorithms, and Systems", ISBN: 9780521189842, paperback edition, Cambridge University Press, March 2011 (corrects the errata in the 2008 edition). 756 pages.

[2] M. Gupta, M. H. Ammar, and M. Ahamad, "Trade-offs between reliability and overheads in peer-to-peer reputation tracking," Computer Networks, pp. 501-522, 2006.

[3] N. Venkatasubramanian, "CompOSE|Q-a QoS-enabled customizable middleware framework for distributed computing," Electronic Commerce and Web-based Applications/Middleware. in 19th IEEE International Conference on Distributed Computing Systems, pp. 134-139, 1999.

[4] Cheng-Fu Chou, L. Golubchik, and J. C. S. Lui, "Striping doesn't scale: how to achieve scalability for continuous media servers with replication," in 20th International Conference on Distributed Computing Systems, pp. 64-71, 2000.

[5] N. Venkatasubramanian, M. Deshpande, S. Mohapatra, S. Gutierrez-Nolasco, and J. Wickramasuriya, "Design and implementation of a composable reflective middleware framework," in 21st International Conference on Distributed Computing Systems, pp. 644-653, Apr 2001.

[6] D. S. Santry, M. J. Feeley, N. C. Hutchinson, A. C. Veitch, R. W. Carton, and J. Ofir, "Deciding when to forget in the Elephant file system," vol. 33, issue 5, pp. 110-123, Dec 1999.

[7] R. T. Hurley, and Soon Aun Yeap, "File migration and file replication: a symbiotic relationship," IEEE Transactions on Parallel and Distributed Systems, vol. 7, no. 6, pp. 578-586, Jun 1996.

[8] S. Helen, "IRM: Integrated file replication and consistency maintenance in P2P Systems", IEEE Trans. on Parallel and Distributed Systems, Vol. 21, No. 1, pp. 100-113, 2010.

[9] S. Hitoshi, S. Matsuoka and T. Endo, "File Clustering Based Replication Algorithm in a Grid Environment", 9th IEEE/ACM Int. Sym. on Cluster Computing and the Grid, pp. 204-211, 2009.

[10] X. Tang, C. Huicheng, and S. T. Chanson, "Optimal Replica Placement under TTL-Based Consistency", IEEE Transactions on Parallel and Distributed Systems, vol.18, no. 3, pp. 351-363, March 2007.