Dynamic Replication Algorithm for Data Replication to Improve System Availability: A Performance Engineering Approach

Sreekumar Vobugari, D. V. L. N. Somayajulu & B. M. Subaraya

Dynamic Replication Algorithm for Data Replication to Improve System Availability: A Performance Engineering Approach

Sreekumar Vobugari[1], **D. V. L. N. Somayajulu**[1] **and B. M. Subaraya**[2]

[1]Department of Computer Science and Engineering, National Institute of Technology, Warangal, India, [2]Infosys Limited, Mysore, India

ABSTRACT

Information technology systems deployed by enterprises should not only fulfil their business functionalities but also cater to Quality of Service concerns such as Availability, Scalability, and Performance. To enhance the system performance, the system availability is an important factor and to improve the system availability, one of the strategies is replicating the frequently accessed data to multiple suitable locations which is a practical choice as the users can access the data from a nearby site. This is, however, not the case for replicas which must have a preset number of copies on several locations. How to decide a sensible number and right location for replicas have become an important issue in cloud computing. In this paper, we show a dynamic data replication strategy to enhance the performance of software system. To identify the suitable file to replicate and to decide respective number of replicas, we calculate popularity degree and replica factor. We use the fuzzy logic system to identify the system to place the replicas and we use the round robin method to place the replicas in the identified systems. We compare the performance of our technique with the existing technique.

Keywords:
Cloud Computing, Data Replication, Fuzzy Logic System, Popularity Degree, Round Robin.

1. INTRODUCTION

Information technology (IT) systems are increasingly gaining strategic importance in enterprises and have become an integral part of business operations today. Thus, it has become imperative that the IT systems deployed by enterprises not only fulfil their business functionalities but also cater to Quality of Service (QoS) parameters such as availability, scalability, and performance. From the end-users perspective, some of the QoS parameters such as availability, scalability may not be directly visible but play vital role in successful execution of the transaction initiated by end-users while parameters like performance and usability are directly experienced by the end-users. It is, therefore, important that utmost care should be taken in designing the IT systems to ensure that relevant QoS parameters are considered.

Availability, as discussed above has a high impact on any IT system because the downtime of the IT systems is harmful to the business competitiveness of an organization and it is just stating the obvious. What is not often obvious is the different ways in which downtime can affect the enterprises bottom line. The costs that the enterprise can incur due to downtime (non-availability) of the IT systems can be tangible and intangible. Hence, availability of the IT systems play a vital role in the success of large enterprises through successful uptime of the IT systems. Mathematically, the availability (A) of an IT system is defined as:

$$A = \mathrm{MTTF}/(\mathrm{MTTF} + \mathrm{MTTR})$$

where MTTF is the mean time to failure of the business system and MTTR is mean time to repair. Performance of software system is one of the most essential QoS parameters for user satisfaction. It is clear that any application system that satisfies all business requirements but fails to satisfy the performance quality of service parameter will lead to greater dissatisfaction of software application end-users. Also, degradation in any of the above-stated QoS parameters poses high risk to enterprises in terms of productivity, revenue, and business continuity. With business transactions increasingly going online, competition is only a click away and hence due consideration need to be given during architecting the software systems to address the performance and availability aspects. A report estimates that poorly performing IT applications is costing industrialized nations almost $70 billion annually [1]. Clearly, the impact of poorly performing applications is high on the owners of the software application. Hence, performance aspect of any application system will be dealt in numerous ways. Some of the following areas of performance engineering techniques can be identified as the ones having major impact on the overall performance of a software system: (1) Data Access [2–5], (2) Networking [6–10], (3) Replica Management [10–14], (4) Replica Optimization, and (5) Scheduling.

Managing performance through replication demands for additional infrastructure in terms of storage and database servers. With the advent of cloud, customers are opting for cloud-based infrastructure in order to cater infrastructure needs of high performing software applications and this is true even for applications based on data replication strategies. Cloud computing is rapidly transforming the way organizations view their IT resources. From a scenario of a single system consisting of single operating system and single application, organizations are moving into cloud computing where resources are available in abundance and the user has a wide range to choose from [15]. In cloud computing, the end-users need not know the details of a specific technology while hosting their application, as the service is completely managed by the cloud service provider (CSP).

Users can consume services at a rate that is set by their particular needs. This on-demand service can be provided any time. CSP takes care of all the necessary complex operations on behalf of the user. It provides the complete system which allocates the required resources for execution of user applications and management of the entire system flow. Figure 1. depict the visual representation of the architecture of cloud computing [16]. With reference to Replication strategies, replica optimization is

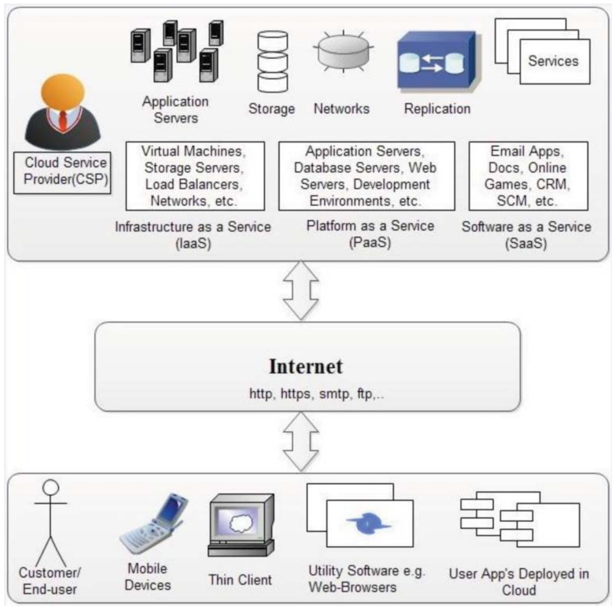

Application Servers
Storage
Networks
Replication
Services
Cloud Service Provider(CSP)
Virtual Machines, Storage Servers, Load Balancers, Networks, etc.
Application Servers, Database Servers, Web Servers, Development Environments, etc.
Email Apps, Docs, Online Games, CRM, SCM, etc.
Infrastructure as a Service (IaaS)
Platform as a Service (PaaS)
Software as a Service (SaaS)
Internet
http, https, smtp, ftp,..
Customer/ End-user
Mobile Devices
Thin Client
Utility Software e.g. Web-Browsers
User App's Deployed in Cloud

Figure 1: Cloud computing architecture.

one of the performance enhancement techniques for software system. One of the objectives of replica optimization is to lessen file access times by pointing access requests to corresponding replicas and pro-actively replicating repeatedly used files based on access statistics gathered. Replica optimization techniques can thus be divided into (1) replica selection, (2) replica initiation.

Replica selection: The replica selection aspect of replica optimization intends to select the best replica corresponding to network and storage access latencies. In other words, if for a given file several replicas exist, the optimization algorithm identifies the replica that should be accessed from a given location. Similarly, the algorithm may also be used to identify the best location for new replicas, i.e. where to store additional replicas of an existing file.

Replica initiation: The objective of replica initiation is to trigger replication and thus the generation of new replicas dynamically. The decision about when to create new replicas can be based on the file access history in order to optimize data locality for repeatedly requested files. This assumes that based on historical events we can partially predict future file access. By increasing the replication factor of a file, one can achieve better load balancing of file requests to less loaded sites and also fault tolerance in case some sites become unavailable [15].

To achieve the dynamic data replication [17–21] we have to solve three essential issues. The first issue is which data should be replicated and when to replicate in the cloud systems to satisfy the user needs which can reduce the reduction of waiting time and speeding up data access. The wrong selection of file to replicate and too early replication will not satisfy the user requirements. The second issue is how many suitable replicas to be created in the cloud to meet the system availability requirement. The increasing number of replicas will increase the system maintenance cost and too many needless replicas will brings superfluous spending instead of increasing system availability. The third issue is where to place the new replicas to meet the system task successful execution rate and bandwidth consumption. By maintaining all replicas active, the replicas may enhance the system task successful execution rate and bandwidth utilization if the replicas and requests are reasonably distributed. However, suitable replica placement in ultra-large-scale, dynamically scalable and entirely virtualized data centers is much more complicated [22].

In this paper, we develop a dynamic data replication strategy to enhance the performance of software system. We considered the mathematical model given in [22] to develop a dynamic data replication algorithm. In [22], the processes involved are which file should be replicated, how many replicas we need to create, and where the new replicas should be placed. We modify the popularity degree and replica factor given in [22]. The popularity degree is modified with respect to three factors. The first factor considers the inverse access frequency and the second factor considers the table weightage function of a file and the third factor is the popularity degree in [22]. Thereafter, we calculate the replica factor using the values of positive factor which is based on popularity degree and negative factor which is based on negative degree. We then use the fuzzy logic system to identify the systems to place the replicas and we use the round robin method to place the replicas in the identified systems.

This paper is organized as follows: the second section shows a brief review of some of the related works and the third section shows the motivation of our work and the fourth section explains our proposed technique and the fifth section shows the performance of our technique and the sixth section concludes our technique.

2. RELATED WORKS: A BRIEF REVIEW

Literature presents several techniques for data replication. Here, we review some of the techniques presented. Sun et al. [22] have developed a dynamic data replication strategy, it includes: (1) analysing and modelling the relationship within system availability and the number of replicas; (2) assessing and identifying the popular data and cause a replication operation when the popularity data passes a dynamic threshold; (3) computing an appropriate number of copies to meet a logical system byte effective rate (SBER) necessity and placing replicas among data nodes in a balanced way; (4) formulating the dynamic data duplication algorithm in a cloud. Experimental outcomes demonstrated the efficiency and effectiveness of the enhanced system brought by the proposed strategy in a cloud.

Sashi and Thanamani [23] have presented a modified bandwidth hierarchy-based replication (BHR) algorithm to overcome the limitations of the standard BHR algorithm. The algorithm was simulated using a Data Grid simulator, OptorSim, developed by European Data Grid projects. The performance of the algorithm was improved by minimizing the data access time and avoiding unnecessary replication. Lee et al. [23] have developed an adaptive data replication algorithm, called the Popular File Replicate First algorithm (PFRF), which was developed on a star-topology data grid with limited storage space based on aggregated information on previous file accesses. The PFRF periodically calculated file access popularity to track the variation of users' access behaviours, and then replicates popular files to appropriate sites to adapt to the variation. They employed several types of file access behaviours, including Zipf-like, geometric, and uniform distributions, to evaluate PFRF. The simulation results showed that PFRF has effectively improved average job turnaround time, bandwidth consumption for data delivery, and data availability as compared with those of the tested algorithms.

Hussein and Mousa [24] have developed an adaptive replication strategy in the cloud environment. The strategy investigated the availability and efficient access of each file in the data center, and studied how to improve the reliability of the data files based on prediction of the user access to the blocks of each file. The developed adaptive replication strategy redeploys dynamically large-scale different files replicas on different data nodes with minimal cost using heuristic search for each replication. The developed adaptive strategy was based on a formal description of the problem. The strategy identified the files which were popular files for replication based on analysing the recent history of the data access to the files using Holt's linear and exponential smoothing (HLES) time series. Once a replication factor based on the popularity of the files was less than a specific threshold, the replication signal was triggered. Hence, the adaptive strategy identifies the best replication location based on a heuristic search for the best replication factor of each file.

Kamali et al. [25] have considered the problem of fragment allocation in lazily replicated systems and address both placement and replication issues in an integrated approach. While replication has improved performance via increased locality, excessive replication can incur extra overhead cost to maintain replicas. A comprehensive model that took into account network topology, fragment correlation, and data access patterns is presented. Based on this model, they developed an algorithm to find near-optimal dynamic allocation solutions. Zhe Wang et al. [26] have presented a dynamic data replication strategy based on two ideas. The first one employed historical access records which were useful for picking up a file to replicate. The second one was a proactive deletion method, which was applied to control the replica number to reach an optimal balance between the read access time and the write update overhead. A unified cost model was used as a means to measure and compare the performance of their data replication algorithm and other existing algorithms.

Ruay-Shiung Chang and Hui-Ping Chang [27] have developed a dynamic data replication mechanism called Latest Access Largest Weight (LALW). LALW selects a popular file for replication and calculated a suitable number of copies and grid sites for replication. By associating a different weight to each historical data access record, the importance of each record was differentiated. A more recent data access record had a larger weight. It indicated that the record was more pertinent to the current situation of data access. A Grid simulator, OptorSim, was used to evaluate the performance of that dynamic replication strategy. The simulation results showed that LALW successfully increases the effective network usage.

Najme Mansouri [28] has presented a dynamic data replication strategy, called Modified Latest Access Largest Weight (MLALW). This strategy was an enhanced version of LALW strategy. MLALW deletes files by considering three important factors: least frequently used replicas, least recently used replicas, and the size of the replica. MLALW stores each replica in an appropriate site, i.e. appropriate site in the region that has the highest number of access in future for that particular replica. The algorithm was simulated using a Data Grid simulator, OptorSim, developed by European Data Grid projects. The experiment results showed that MLALW strategy gave better performance compared to the other algorithms and prevent unnecessary creation of replica which leads to efficient storage usage.

The advantages of our technique compared to the aforementioned techniques are as follows: In [22], they had considered only one factor which is based on access frequency to calculate the popularity degree. In our technique, we consider two more factors which are based on inverse access frequency and table weightage function to improve the performance and also we use the fuzzy logic system to identify the exact systems to replicate the files. In [23], they store the replicated data in a particular location. It would increase the response time to the end-user. Because if a user gives a query, the system would search for the available data which is required to process the query and if the data is not in the nearer system of the end-user, it would increase the response time. But in our technique, we use the fuzzy logic system to find where to place the replicas and the replicas would be placed in the necessary system and it would reduce the response time. In [23], they only considered the access frequency of the files for replication and the replicas are only within the user's local cluster. In our technique, we additionally consider the inverse access frequency and table weightage function to calculate the popularity degree which impacts the replica factor and the replicas is within user's local sites. In [24,27], they did not use any model to place the replicas equally to the systems. In our technique, we use the fuzzy logic system which identifies where to place the replicas and also we use round robin method which equally allocates the replicas in the required systems. In [25,28], they use some mathematical functions to place the replicas. In our technique, we calculate mathematical functions and we give the solution to the fuzzy logic system to decide where to place the replicas. Because fuzzy logic can deal with reasoning that is approximate rather than fixed and exact.

3. MOTIVATION OF OUR WORK

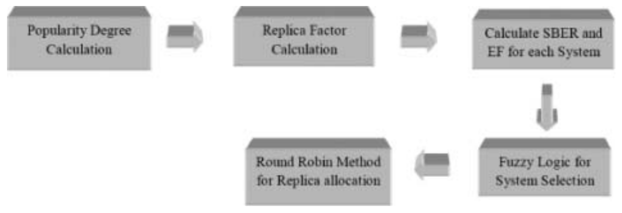
To develop a dynamic data replication algorithm, we have taken the mathematical model given in [22] as motivation for our research. There, they have considered the SBER as the primary issue and they developed a mathematical formula to identify which file needs to replicate, the number of replica needed, and where to place the new replicas by considering the popularity degree. While examining their work [22], they have given that further study is needed to reduce the user waiting time, speeding up data access, and enhancing the data availability. By considering these facts, we modify the popularity degree of [22] with respect to three factors. The first factor is based on inverse access frequency and the second factor is based on table weightage function and the third factor is based on access frequency that is in [22]. We then evaluate the replica factor based on the positive and negative factor. Thereafter, we use the fuzzy logic system to identify the systems to replicate the files, and for allocating the replicas in the systems, we use the round robin technique.

4. PROPOSED TECHNIQUE TO ENHANCE THE PERFORMANCE OF SOFTWARE SYSTEM

This section delineates our proposed dynamic data replication strategy to enhance the performance of software system. To enhance the performance, we modify the calculation of popularity degree in [22] and we use the fuzzy logic to identify in which system we have to modify and we also use round robin technique for replica allocation. The process involved in our proposed technique is shown in Figure 2.

4.1 Popularity Degree Calculation

This section shows the calculation of the popularity degree of our proposed technique. The popularity degree is used to calculate the replica factor. The



Popularity Degree Calculation
Replica Factor Calculation
Calculate SBER and EF for each System
Fuzzy Logic for System Selection
Round Robin Method for Replica allocation

Figure 2: Process of our proposed dynamic data replication.

calculation is based on three factors and the formula to calculate the popularity degree is shown below:

$$\mathrm{PD} = P1 + P2 + P3$$

where, PD denotes the popularity degree of a file; $P1$ denotes the first factor; $P2$ denotes the second factor; and $P3$ denotes the third factor. The equation to calculate $P1$ is given below:

$$P1 = \sum_{t_i=t_s}^{t_p} \mathrm{AF}_{(t_i,t_{i+1})} * log\left(\frac{\mathrm{nu}}{\mathrm{nr}}\right)_{(t_i,t_{i+1})}$$

In the above equation, AF denotes the access frequency between t_i and t_{i+1}; and nu denotes the number of unique users between t_i and t_{i+1}; and nr denotes the number of repeated users between t_i and t_{i+1}; and t_s is the start time; and t_p is the present time. The first factor is calculated by taking the product of access frequency and the inverse access frequency. The inverse access frequency is used to identify the importance to replicate a file by considering the number of unique users who used a file, i.e. the logarithmic ratio of number of unique users nu to the number of repeated users nr used a file in a time interval. If the number of unique users is high, then that file has more importance to replicate. Therefore, the inverse access frequency is the term that gives more importance to the number of unique users from the total number of users who used a file in a time interval. The formula to calculate the second factor is given below:

$$P2 = \sum_{t_i=t_s}^{t_p} \mathrm{AF}_{(t_i,t_{i+1})} * W_{tf}$$

where W is total weight of tables in a file. The second factor is calculated by taking the product value of the access frequency in a time interval and the total weight value of each table in a file. We then sum the values obtained on each interval to calculate the value for the second factor. In each file, there are a number of tables and each table has weight values. The total weight value is adding the weight values of each table in a file. The third factor is the same that used to calculate the popularity degree in [22]. It is shown by an equation below:

$$P3 = \sum_{t_i=t_s}^{t_p} \mathrm{an}_k(t_i, t_{i+1}) \times \omega(t_i, t_p)$$

After calculating the first factor $P1$, second factor $P2$, and the third factor $P3$ of a file, the popularity degree PD of a file can be calculated. Similarly, we have to find the popularity degree for each file.

4.2 Replication Factor Calculation

The replica factor calculation is used to find whether the data file should be replicated or not. We calculate the replica factor by calculating the positive factor (PF) and negative factor NF. The purpose of positive factor is to identify how important is a file for replication. The formula to calculate the positive factor PF is given below:

$$\mathrm{PF} = \frac{\mathrm{PD}_{\mathrm{current}} - \mathrm{PD}_{min}}{\mathrm{PD}_{max} - \mathrm{PD}_{min}}$$

In the above equation, PF represents positive factor; and $\mathrm{PD}_{\mathrm{current}}$ represents the popularity degree of the current file; and $\mathrm{PD}_{\mathrm{min}}$ represents minimum popularity degree; and $\mathrm{PD}_{\mathrm{max}}$ represents maximum popularity degree. The positive factor PF is calculated by identifying the difference between the popularity degree of the current file $\mathrm{PD}_{\mathrm{current}}$ and the minimum popularity degree value $\mathrm{PD}_{\mathrm{min}}$ and dividing the solution by the difference between the maximum popularity degree value $\mathrm{PD}_{\mathrm{max}}$ and the minimum popularity degree value $\mathrm{PD}_{\mathrm{min}}$. The positive factor is calculated to find the importance of the file to replicate. Thereafter, we have to evaluate the negative degree (ND) for each file. The negative degree calculation is used to find the negative factor (NF).

NF of a file specifies if a file should not be replicated. The equation to evaluate negative degree ND for each file is given below:

$$\mathrm{ND} = M \times R \times \mathrm{QRT}$$

where, ND denotes negative degree; and M denotes the memory size of a file; and R denotes the number of replica exists; and QRT denotes query response time. The negative degree ND is then used to calculate the negative factor NF of a file. The formula to calculate the negative factor NF is as follows:

$$\mathrm{NF} = \frac{\mathrm{ND}_{\mathrm{current}} - \mathrm{ND}_{min}}{\mathrm{ND}_{max} - \mathrm{ND}_{min}}$$

In the above equation, NF represents negative factor; $\mathrm{ND}_{\mathrm{current}}$ represents negative degree value of current file; $\mathrm{ND}_{\mathrm{min}}$ represents minimum negative degree value; and $\mathrm{ND}_{\mathrm{max}}$ represents maximum negative degree value. The negative factor NF value is calculated by taking the difference amid the negative degree value of the current file $\mathrm{ND}_{\mathrm{current}}$ and the minimum negative degree value ND_{min} and dividing the solution with the difference amid the maximum negative degree ND_{max} and the minimum negative degree ND_{min}. The negative factor is calculated to find the importance of not to replicate the file. The replica factor is then calculated using the equation given below:

$$\mathrm{RF} = \frac{\alpha \mathrm{PF} + \beta(1 - \mathrm{NF})}{\alpha + \beta}$$

Here, RF denotes replica factor; PF denotes the positive factor of a file; NF denotes negative factor of a file; and α, β are the constant values which we assigned as 1 by checking performance with different values. The number of replica is then generated by the following condition:

$$\text{no. of Replica} = \begin{cases} \mathrm{RF}_t + 2 * \mathrm{RF}_{t-1}; & \text{if } \mathrm{RF}_t > \mathrm{RF}_{t-1} \\ 0 \qquad ; & \textit{if } \mathrm{RF}_t < \mathrm{RF}_{t-1} \end{cases}$$

where, RF_t denotes Replica factor at a time interval and RFt-1 denotes Replica factor at previous time interval.

4.3 Fuzzy Logic for System Selection

This section shows the system selection of our technique based on fuzzy logic. To identify the systems to replicate the files, we have to calculate the SBER for each system and effective factor for each system. Figure 3 shows the model system architecture and explanation.

When the user gives task to access a file, the system manager will check the traffic free system that has the file requested by the user and directs the path to the user. Here, we calculate the SBER [22] and we have to calculate the effective factor for each system. The effective factor for each system is calculated as follows:

$$\mathrm{EF} = \frac{1}{F}\sum_{i=1}^{k} \mathrm{RF}_i$$

Here, EF is the effective factor for a system; F denotes the number of files in a system; and RF_i is the replica factor of each file in the system. In the above equation, we find the number of files exists in a system and we sum the replica factor value for each file in that system. From Figure 3, we can see that each system has number of files. After calculating the SBER and effective factor values for each system, we give the values to the fuzzy logic system as input to identify the systems to replicate files. Figure 4 shows the process of identifying systems to replicate files.

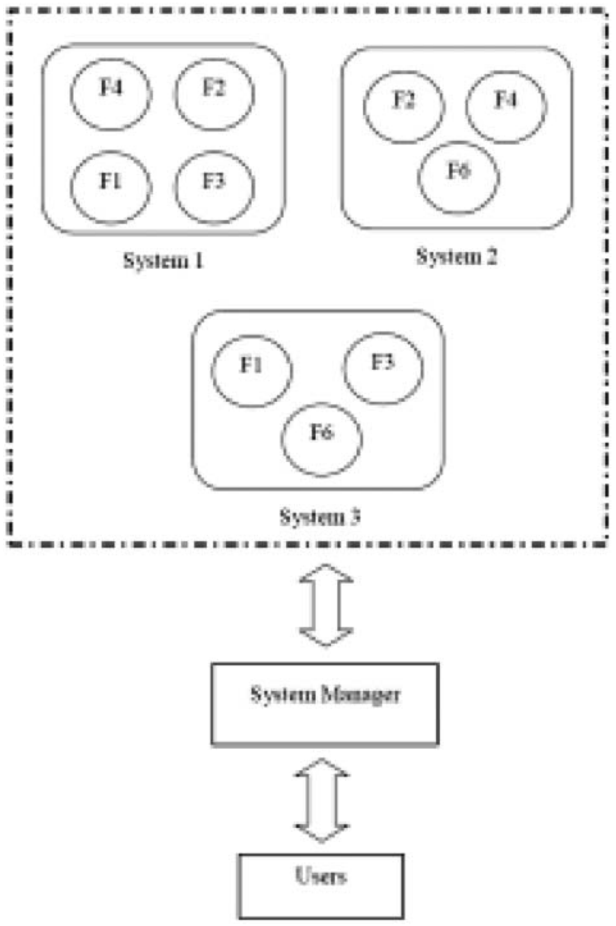


Figure 3: Model system architecture.

The input variables in fuzzy logic system are mapped by set of membership functions. The fuzzification is the process that determines the percentage level of input membership in overlapping sets. The rules we give determine the outputs based on inputs. The defuzzification is the process that combines all fuzzy actions into a single fuzzy action and converts the single fuzzy action into a crisp, executable system output. In Figure 4, we give the SBER and effective factor for each system as input to the fuzzy logic system and it would give the output as score values. We can set any number of memberships for the inputs we give. For instance, if we set the memberships for

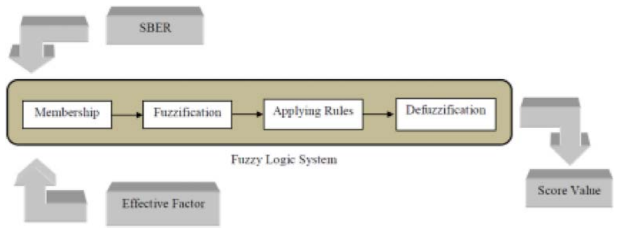


Figure 4: Process of fuzzy logic system.

SBER and effective factor as low and high, we will set the rules based on low and high memberships. The low and high memberships are based on the values of SBER and effective factor. If the values are from 0 to 10, we take the values from 0 to 5 as low and from 6 to 10 as high. An example for rule formation is, if the SBER is high and the effective factor is high, the fuzzy score would be high. Similarly, we would form different rules to get the score value based on the membership functions. Thereafter, we set a threshold for the score value to identify the systems to replicate the files. So, the systems that have the score values above the threshold would get selected to replicate the files.

4.4 Round Robin Method for Replica Allocation

This section explains the allocation of replicas in the identified systems. Consider we have identified the first system and the second system to replicate the files using fuzzy logic system based on Figure 3. The files we need to replicate are third and fourth files and the number of replicas we need to generate are two for the third file and three for the fourth file. While considering the third file, the round robin technique first replicates the first copy in the first system and the second copy in the second system; and while considering the fourth file, the round robin technique replicates the first replica in the first system and the second replica in the second system and the third replica in the first system. Figure 5 shows the algorithm of the entire process of our proposed technique.

5. RESULTS AND DISCUSSION

This section explains the results we obtained for our technique and we compare the performance of our

```
Input: File availability, file unavailability, number of replicas, and number of accesses.
Output: System Byte Effective Rate (SBER)
1.  For each file
2.  Calculate $PD$
3.  Calculate $RF$
4.  If $RF_t > RF_{t-1}$
5.  Calculate number of replicas
6.  Else ignore
7.  End if
8.  End for
9.  For each system
10. Calculate $EF$
11. Give SBER and $EF$ as input to fuzzy logic
12. Score value as output from fuzzy logic
13. End for
14. Set threshold
15. If score value > threshold
16. Select that system to replicate file
17. Else ignore
18. End if
19. Place replicas in the selected system using round robin
20. Calculate new SBER
```

Figure 5: Algorithm of our proposed dynamic replication strategy.

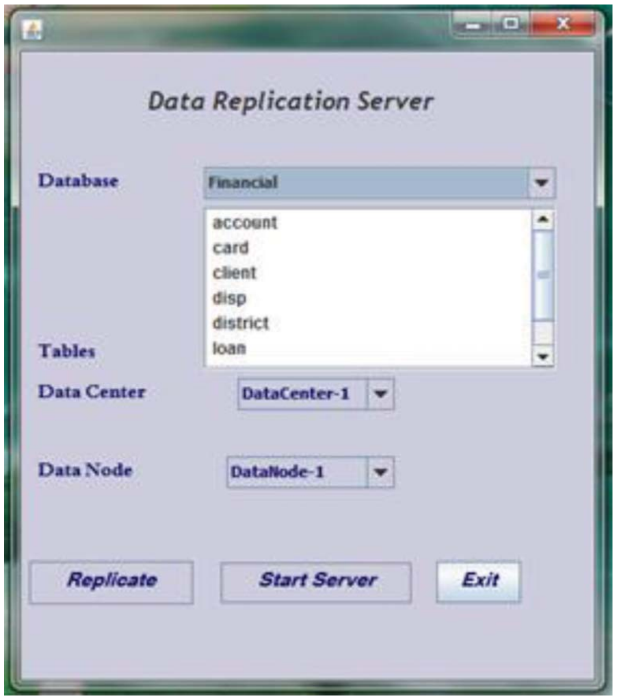


Figure 6: Sample dataset with tables.

technique with the performance of the existing technique [22].

5.1 Experimental Setup

Our proposed technique is implemented using Java (jdk 1.6) which is installed in a system that has following configuration: i5 processor with 3.20GHZ clock speed, 4GB RAM. We used three different datasets which are Financial, Medical, and RDB to process our proposed technique. Figure 6 shows the dataset "Financial" and the tables in it.

In Figure 6, the terms "account", "card", "client", etc. denote the tables in the financial dataset (file). Similarly, we have two other datasets which are "Medical" and "RDB" with its respective table contents.

Algorithm of our technique:

5.2 Experimental Setup

We compared the performance of our proposed technique with the existing technique in terms of SBER and execution time. The SBER is calculated for different time intervals. For each time intervals, first we identify the replica numbers generated and after identifying the replica numbers we calculate the SBER, because if the replica generated is for the right file which the users requested a lot, the system performance would get improve. If the replica

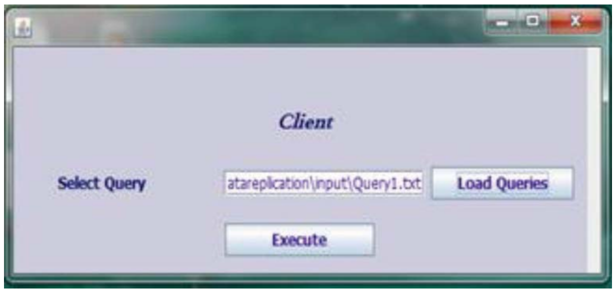


Figure 7: Queries we give to process.

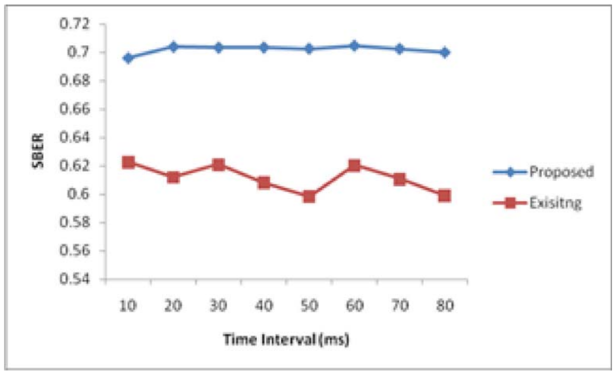
generated is not for the right file, the system performance will not get improved because the wrongly selected file and too early selected file will not reduce the waiting time or speed up data access. To check the execution time, we give different number of tasks and evaluate the time taken to complete it. The tasks are number of queries and the queries are given by different users. Figure 7 shows the queries we give to process.

In Figure 7, the "Query1.txt" in the text box of "Select Query" contains different queries of different users. Based on the queries, the system would decide to generate replica or not to improve the performance of the system.

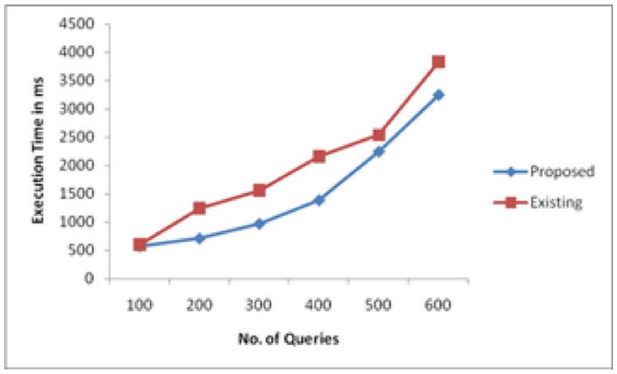
5.2 Performance Analysis

The performance of our proposed technique is analysed with the existing work [22] in terms of SBER and execution time. Graph 1 shows the performance of SBER of our proposed technique and the existing work.

The performance improvement is based on the SBER. If the replica generated is for the right file, the SBER will get improved and it would enhance the system availability. In Graph 1, we evaluated the SBER for our proposed technique and for the



Graph 1: Performance of SBER.



Graph 2: Performance of SBER.

existing technique in different time intervals. While comparing the SBER for our proposed technique and the existing technique, our proposed technique achieved higher SBER compared to the existing work. This implies that our proposed technique has higher system availability and the performance is improved compared to the existing technique.

Graph 2 shows the performance comparison of our technique and the existing technique based on execution time. The execution time is calculated for our technique and the existing technique by varying the number of queries. From this graph, we can understand that the execution time of our technique is less compared to the existing technique for different number of queries we give.

6. Conclusion

In this paper, we proposed a dynamic data replication strategy to improve the performance of software system. The processes involved in our technique are identifying which file to replicate, how many numbers to replicate, and where to place it. The file to replicate and the number of files to replicate were identified by calculating the popularity degree and the replica factor which we designed by modifying the existing technique. Thereafter, we used fuzzy logic system to identify the system to place the replicas and we used the round robin method to place the replicas in the identified system. We also compared the performance of our technique with the existing technique in terms of SBER and execution time. From the performance measure, we showed that our technique improved the performance compared to the existing technique.

Acknowledgements

The work is solely done by us at NITW and hence no need of acknowledging any entity or person.

REFERENCES

1. The Cost Benefit of Monitoring Applications, Butler Group May 2005. Available: http://www.wilytech.com/solutions/resource/index.php.
2. C. Mayr, U. Zdun, and S. Dustdar, "View-based model-driven architecture for enhancing maintainability of data access services", *Data Knowl. Eng.*, Vol. 70, pp. 794–819, May 2011.
3. S. Arslan, A. Yazıcı, A. Saçan, I. H. Toroslu, and E. Acar, "Comparison of feature-based and image registration-based retrieval of image data using multidimensional data access methods," *Data Knowl. Eng.*, Vol. 86, pp. 124–45, Jul. 2013.
4. M. Akon, M. T. Islam, X. (Sherman) Shen, and A. Singh, "A bandwidth and effective hit optimal cache scheme for wireless data access networks with client injected updates," *Comput. Netw.*, Vol. 56, pp. 2080–95, May 2012.
5. H. Chen, Y. Xiao, and S. V. Vrbsky, "An update-based step-wise optimal cache replacement for wireless data access," *Comput. Netw.*, Vol. 57, pp. 197–212, Sept. 2013.
6. B. Tagger, D. Trossen, A. Kostopoulos, S. Porter, and G. Parisis, "Realising an application environment for information-centric networking," *Comput. Netw.*, Vol. 57, pp. 3249–66, Aug. 2013.
7. E. Talipov, J. Yin, Y. Chon, and H. Cha, "A context-rich and extensible framework for spontaneous smartphone networking," *Comput. Commun.*, Vol. 37, pp. 25–39, Jan. 2014.
8. G. Floros, and K. Siomos, "The relationship between optimal parenting, Internet addiction and motives for social networking in adolescence," *Psychiatry Res.*, Vol. 209, pp. 529–34, Feb. 2013.
9. Y. Xu, Y. Li, T. Lin, Z. Wang, W. Niu, H. Tang, and S. Ci, "A novel cache size optimization scheme based on manifold learning in Content Centric Networking," *J. Netw. Comput. Appl.*, Vol. 37, pp. 273–81, Mar. 2014.
10. R. M. Almuttairi, R. Wankar, A. Negi, C. R. Rao, A. Agarwal, and R. Buyya, "A two phased service oriented Broker for replica selection in data grids," *Future Gener. Comput. Syst.*, Vol. 29, pp. 953–72, Sept. 2013.
11. R. Kingsy Grace, and R. Manimegalai, "Dynamic replica placement and selection strategies in data grids — A comprehensive survey," *J. Parallel Distrib. Comput.*, Vol. 74, pp. 2099–108, Nov. 2013.
12. T. Ma, Q. Yan, W. Tian, D. Guan, and S. Lee, "Replica creation strategy based on quantum evolutionary algorithm in data gird," *Knowl.-Based Syst.*, Vol. 42, pp. 85–96, Apr. 2013.
13. N. Mansouri, and G. H. Dastghaibyfard, "A dynamic replica management strategy in data grid," *J. Netw. Comput. Appl.*, Vol. 35, pp. 1297–303, Feb. 2012.
14. V. S. Agneeswaran, and D. Janakiram, "Node-capability-aware replica management for peer-to-peer grids," *IEEE Trans. Syst. Man Cybern.*, Vol. 39, no. 4, pp. 807–18, Jul. 2009.
15. E. Laure, H. Stockinger, and K. Stockinger, "Performance engineering in data grids," *J. Concurrency Comput.: Pract. Exp. - Grid Perform.*, Vol. 17, no. 2–4, pp. 171–91, Feb. 2005.
16. S. Vobugari, D. V. L. N. Somayajulu, B. M. Subaya, and M. K. Srinivasan, "A roadmap on improved performance-centric cloud storage estimation approach for database system deployment in cloud environment," in *IEEE 14th International Conference on Mobile Data Management*, 2013, pp. 182–7.
17. A. Doğan, "A study on performance of dynamic file replication algorithms for real-time file access in Data Grids," *Future Gener. Comput. Syst.*, Vol. 25, pp. 829–39, Feb. 2009.
18. K. Sashi, and A. S. Thanamani, "Dynamic replication in a data grid using a modified BHR region based algorithm," *Future Gener. Comput. Syst.*, Vol. 27, pp. 202–10, Feb. 2011.
19. L. M. Khanli, A. Isazadeh, and T. N. Shishavan, "PHFS: A dynamic replication method, to decrease access latency in the multi-tier data grid," *Future Gener. Comput. Syst.*, Vol. 27, pp. 233–44, Mar. 2011.
20. V. Andronikou, K. Mamouras, K. Tserpes, D. Kyriazis, and T. Varvarigou, "Dynamic QoS-aware data replication in grid environments based on data ''importance''," *Future Gener. Comput. Syst.*, Vol. 28, pp. 544–53, Mar. 2012.
21. T. Amjad, M. Sher, and A. Daud, "A survey of dynamic replication strategies for improving data availability in data grids," *Future Gener. Comput. Syst.*, Vol. 28, pp. 337–49, Feb. 2012.
22. D. W. Sun, G. R. Chang, and S. Gao, "Modeling a dynamic data replication strategy to increase system availability in cloud computing environments," *J. Comput. Sci. Technol.*, Vol. 27, no. 2, pp. 256–72 Mar. 2012.
23. M. -C. Lee, F. -Y. Leub, and Y. -P. Chen, "PFRF: An adaptive data replication algorithm based on star-topology data grids," *Future Gener. Comput. Syst.*, Vol. 28, no. 7, pp. 1045–57, Jul. 2012.
24. M. -K. Hussein, and M. -H. Mousa, "A light-weight data replication for cloud data centers environment," *International Journal of Engineering and Innovative Technology (IJEIT)*, Vol. 1, no. 6, Jun. 2012.
25. S. Kamali, P. Ghodsnia, and K. Daudjee, "Dynamic data allocation with replication in distributed systems," *IEEE 30th International Performance Computing and Communications Conference (IPCCC)*, 2011, pp. 1–8.
26. Z. Wang, T. Li, N. Xiong, and Y. Pan, "A novel dynamic network data replication scheme based on historical access record and proactive deletion," *J. Supercomput.*, Vol. 62, no. 1, pp. 227–50, Oct. 2012.
27. R. -S. Chang, and H.-P. Chang, "A dynamic data replication strategy using access-weights in data grids," *J. Supercomput.*, Vol. 45, no. 3, pp. 277–95, Sept. 2008.
28. N. Mansouri, "An effective weighted data replication strategy for data grid," *Australian J. Basic Appl. Sci.*, Vol. 6, no. 10, pp. 336–46, Oct. 2012.

Authors

Sreekumar Vobugari is currently pursuing a PhD program in the Department of Computer Science and Engineering at the National Institute of Technology, Warangal, AP, India. He holds a Master of Engineering degree in computer science from Jadavpur University, Kolkata, 1998 and a Bachelor of Engineering in computer science and engineering from Gulbarga University, 1992. He has over 21 years of IT industry experience and has played various roles such as programmer, module lead, database architect, project manager, etc. for various global customers during his stint working for some of the major Software consulting companies in India. He has published a conference paper titled “An approach for Database Size Estimation” in *IEEE-International Advance Computing Conference*, 2009 Patiala India, “Index Tuning through Query Evaluation Mechanism Based on Indirect Domain Knowledge” in UKSim 14th International Conference on Computer Modeling and Simulation, 2012, Cambridge, UK, “A Roadmap on Improved Performance-centric Cloud Storage Estimation Approach for Database System Deployment in Cloud Environment” in MDM 14th International Conference on Mobile Data Management, 2013, Milan, Italy. He has filed a defensive publication on “System and Method for defect tracking in software projects”. His research interest is around Performance Engineering, Software Architectures and Large scale distributed data processing.

E-mail: Sreekumar_vobugari@nitw.ac.in

D. V. L. N. Somayajulu is a professor in Department of Computer Science and Engineering at the National Institute of Technology (NIT), Warangal, AP, India. He holds a PhD in computer science and engineering from the Indian Institute of Technology, Delhi, India. He has over 28 years of academic and industry experience. He headed the Computer Center at NIT Warangal and has been head, Department of Computer Science for over four years. He has been awarded the Best Engineer of the Year in 2007. He is the “Dean Academics”, National Institute of Technology, Warangal, AP. His area of interest is around Database Performance, Data mining, and eLearning. He has carried out various Software projects sponsored by Government of India and has organized various conferences and workshops. Currently he is guiding and mentoring six PhD students in various fields of databases. Some of his publications are “Ontology Matching schema integration using node ranking” at International Conference on Semantic Web and Web Services, June, 2006, “Sentiment Classification of text reviews using novel feature selection with reduced over-fitting” at International Conference on Internet Technologies and Secured Transactions, November, 2010.

E-mail: somadvln@nitw.ac.in

B. M. Subaraya is a vice president at the Education and Research Unit of Infosys Limited, India. He holds a PhD degree in computer science and engineering from the Indian Institute of Technology, Delhi, India. He has over 28 years of experience in academic and research areas. He has spearheaded the inception of the foundation program for the fresh engineers entering into Infosys at their Global Education Center at Mysore. He has several International publications to his credit and has authored a booked titled *An Integrated Approach to Web Performance Testing, A Practitioner’s Guide*.

E-mail: Subaya@gmail.com

DOI: 10.1080/03772063.2014.988757; Copyright © 2015 by the IETE