Fuzzy FP-Tree based Data Replication Management System in Cloud

P.Elango[#1], Dr. Kuppusamy[*2]
[#]Department of Computer Science, Gobi Arts and Science College, Gobichettipalayam, Erode
Tamilnadu, India
[*]Department Of Computer Science and Engineering, Alagappa University ,karaikudi
Tamilnadu India

Abstract- *Replication is a better way for enhancing high availability and increasing performance of accessing the data in database systems. The purpose of data replication is to improve transaction response time, throughput, and system availability in case of failure. The main aim of this paper is to provide a enhanced technique for solving the shortcomings of the currently existing works of data replica method in cloud environment. This work intends to propose Data Replication System based on data mining techniques. In proposed method, we will use a cloud computing with Combination of replication algorithm and job scheduling policy for data replication in the data cloud environment through monitoring all job process. The data replication will be done by identifying the frequently used data patterns in the large database of a node. This will be done by Fuzzy FpTree based frequent pattern mining algorithm. The system availability and replica is measured and identifying the location in which the replicated data will be stored and its performance shows most promising result compared to the apriori and FPGrowth approaches*

Keywords - *Data Replication, fptree, fpgrowth, cloud storage, data mining.*

I. INTRODUCTION

Data replication, a well-known technique from distributed systems, is the main mechanism used in the cloud for reducing user waiting time, increasing data availability and minimizing cloud system bandwidth consumption by offering the user different replicas with a coherent state of the same service.

Cloud computing is a large-scale parallel and distributed computing system. It consists of a collection of interconnected and virtualized computing resources that are managed to be one unified computing resources. The provided abstract, virtual resources, such as networks, servers, storage, applications and data, can be delivered as a service. Services are delivered on demand to the end-users over high-speed Internet as three types of computing architecture, namely Software as a Service (SAAS), Platforms as a Service (PAAS) and Infrastructure as a service (IAAS). The main goal is to provide users with more flexible services in a transparent manner, cheaper, scalable, highly available and powerful computing resources[1].

As a result, Cloud computing moved away the computation and data storage from the end user and onto large number of data centers infrastructure. This relieves users of theburdens of hardware and application provisioning and management. Hardware and software are delivered to users as on-demand services over the Internet. The Cloud infrastructure can scale out the hardware capacity to meet the required non-functional quality of services (QoS). However, it is challenging to provide high reliability and efficient access to the cloud data centers because of the large scale and dynamic nature of the Cloud. Replication is the process of providing different replicas of the same service at different nodes[2]. Replication is a used technique in the current different clouds architectures, such as GFS (Google file system) and HDFS (Hadoop Distributed File System) [3, 4]. In the cloud, data replication is achieved through data resource pool and by a service provider for the end-user replacing locally-run applications with web services applications.

However, maintaining a large number of data copies in cloud systems are expensive, and therefore, the number of replicas should be bounded. Clearly, optimizing access cost of data requests and reducing the cost of replication are two conflicting goals, so finding a good balance between them is a challenging task. This paper handles the problem of data replication management by using fuzzy FP tree based mining algorithm to determine which data files has to be replicated frequently instead of whole database.

II. *Related Work*

The data replication problem is an extension of the classical file allocation problem (FAP). Replication technology is one of the useful techniques in distributed systems for improving availability and reliability [5]. In Cloud computing, replication is used for reducing user waiting time, increasing data availability and minimizing cloud system bandwidth consumption by offering the user multiple replicas of a specific service on different nodes. For example, if one node fails, a replica of the failed service will be possibly created on a different node in order to process the requests [6]. Many replication techniques have been proposed in the literature, which are classified into static and dynamic replication. In a static replication, the number of replicas and their locations are initially set in advance [7, 8, 9]. On the other hand, dynamic replication dynamically creates and deletes replicas according to changing environment load conditions [10, 9]. There has been an interesting number of works for data replication in the Cloud computing. For example, in [7], a GFS static cloud data replication algorithm is proposed.The GFS adopts a replication strategy where a single master places the selected data chunk replicas on chunk servers with below-average disk space utilization. Further, the number of replication for each data chuck is specified by the users. Similarly, in [8], an application can specify the number of replicas for each file, and the block size and replication factor are configurable per file. In [9], a static centralized data replication algorithm sets a specific number of replicas based on the minimum weighted distance. In [10], a dynamic distributed cloud data replication algorithm CDRM is based on the HDFS file system where the replication site are selected based on the nodes which have low utilization. In [9], six different dynamic data replication algorithms, Caching-PP, Cascading-PP, Fast Spread-PP, Cascading-Enhanced, and Fast Spread-Enhanced are proposed. In [11], a dynamic centralized data replication algorithm is proposed. The algorithm treats uses a weighting factor for the replication, and utilizes different prediction techniques. Kwok et al. [12] proposed several algorithms to solve the data allocation problem in distributed multimedia databases (without replication), also called as video allocation problem (VAP).

Most of the research papers outlined in [13], aim at formalizing the problem as an optimization one, sometimes using multiple objective functions. Network traffic, throughput of servers and response time exhibited by users are considered for optimization. Although a lot of effort was devoted in providing comprehensive models, little attention was paid in proposing good heuristics to solve an often NPhard problem. Furthermore access patterns are assumed to remain static and solutions in the dynamic case are obtained by reexecuting a perhaps time consuming linear programming technique. Some on-going work is related to dynamic replication of objects in distributed systems when the read-write patterns are not known a priori. Awerbuch's et al. work in [14] is significant from a theoretical point of view, but the adopted strategy that before commuting an update replicas of the object must be deleted, can prove difficult to implement in a real-life environment. In [15] Wolfson et al. proposed an algorithm which leads to optimal single file replication in case of a tree network. The performance of the scheme for general network topologies is not clear though. Dynamic replication protocols were also considered under the Internet environment.

III. *Materials and Methods*

Apriori algorithm

The most representative method of associations was proposed as Apriori algorithm [16] by Agrawal et al. in 1994. In Apriori algorithm process, two steps are included. Fist, it finds out the satisfied frequent itemsets of minimum support; second, it finds out the satisfied rules of minimum confidence. In other words, we are used to find out information of frequent itemset and mine all association rules. Apriori algorithm is continuously repeated to scan database, find out all frequent itemsets, until it does not produce new candidate itemsets. Apriroi algorithm does not filter prior candidate itemsets, so that reduces the amount of candidate itemsets to scan. Therefore, it needs many times to complete scanning a database. In implementing efficiency, Apriori algorithm is not completely efficient.

FP-growth algorithm

The new concept of FP-growth algorithm was proposed by Han et al.[17], it can be one of the representations of the itemsets which do not require candidate generations. It does not need association length to proceed phases which generate candidate itemsets in Apriori algorithm. However, mining with Apriori algorithm does not archive the goal efficiently because it may need many times to scan database and generate a lot of candidate itemsets. Therefore, FP-growth proceeds the first scan in transaction database, it then later filters the frequent itemsets and gradually increases support. Next, in the second scan, establish a FP-tree structure by the transaction database. Then, use a Header table to allocate each item node in FPtree, each item of tree will link each other. Last, a Header table mines conditional pattern tree which finds out all frequent itemsets in recursive method. It is a very efficient and memory saving algorithm.

Mining the FP-Tree using FP-Growth

The FP-Tree provides an efficient structure for mining, although the combinatorial problem of mining frequent patterns still has to be solved. For discovering all frequent item sets, the FP-Growth algorithm takes a look at each level of depth of the tree starting from the bottom and generating all possible item sets that include nodes in that specific level. After having mined the frequent patterns for every level, they are stored in the complete set of frequent patterns. FP-Growth takes place at each of these levels. To find all the item sets involving a level of depth, the tree is first checked for the number of paths it has. If it is a single path tree, all possible combinations of the items in it will be generated and added to the frequent item sets if they meet the minimum support. If the tree contains more than one path, the conditional pattern base for the specific depth is constructed. Looking at depth a in the FP-Tree of figure 2, the conditional pattern base will consist of the following item sets: , and . The item set is obtained by simply following each path of a upward

Hybridizing FP-growth algorithm with Fuzzy Logic

Traditional quantitative association rule mining methods partition continuous domains into crisp intervals. When dividing an attribute into intervals covering certain ranges of values, the sharp boundary problem arises. Elements near the boundaries of a crisp set (interval) may be either ignored or overemphasized [18]. Fuzzy association rules overcome this problem by defining fuzzy partitions over the continuous domains. Moreover, fuzzy logic is proved to be a superior technology to enhance the interpretability of these intervals. Hence, fuzzy association rules are also expressions of the form $X \rightarrow Y$, but in this case, X and Y are sets of fuzzy attribute-value pairs and fuzzy confidence and support measure the significance of the rule. FP-Growth algorithm does not produce the candidate item-sets, but it uses growth models to mine frequent pattern. It compresses the database into a frequent pattern tree, but still retains itemsets associated information. Then it will divide such a compressed database into a group of conditional databases, each one of which is related to a frequent item-set

Proposed Fuzzy FP-Tree based Frequent Pattern Mining Algorithm for Data Replication in Cloud

System Architecture for Data Replication in a Cloud

Three main requirements of database replication are the performance, the availability and the consistency of data. These requirements are in conflict with each other because a change for the benefit of one of the criterion implies a change (minimization) at the expense of the other criteria. The access to a replicated entity is typically uniform with access to a single, non-replicated entity. The replication itself should be transparent to an external user. In addition, in a failure scenario, a failover of replicas is hidden as much as possible. The data replication in the cloud can be explained with the help of the architectural diagram as shown in figure 1. The architecture shows that it contains three major section like User, Scheduling manager and Replica manager. The users are normally the clients those who access the cloud from different locations. Each user can access the cloud independently and will provide different data access which has the property of replication. The particular task of each user is first given to the scheduling manager divides the task to the corresponding data centers through the replica manager based on the number of user using the particular data center to access the file without collusion. The dynamic data replication strategy consists of three different stages.

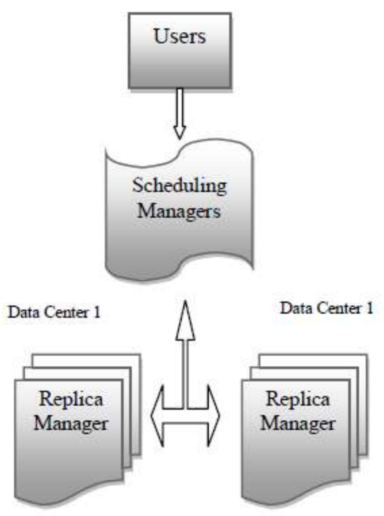
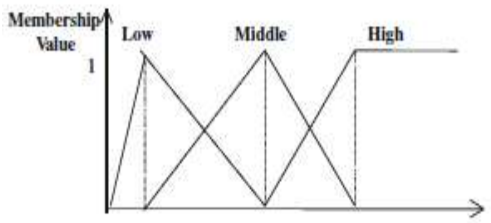


Fig 1: Architectural Diagram of Replication in Cloud Computation

IV. *PROPOSED ALGORITHM*

In this paper, we are interested in finding the frequent patterns of genes and conditions which leads to discover the symptoms of diseases. Though, FP-growth algorithm shows its potential over the Apriori algorithm to find the frequent patterns, we have used this FP-growth algorithm along with the support and confidence is being measured using the membership function of the fuzzy logic to find the frequent patterns which are not only using the crisp nature but also uses the fuzziness to measure the support and confidence to infer the association rules which ill dive us direction to discover the diseased symptoms. Therefore, we have hybridized the FP-growth algorithm with fuzzy logic. Our proposed algorithm is given step wise below.



No.of.data itemset

Fig Membership representation of fuzzy logic

Notations

The notation used in the proposed fuzzy mining algorithm is first described below:

- D - the original set of data items in database
- n - the number of transactions in D
- T - the ith transaction in D, 1 ≤ i ≤ n
- m -the number of items in D
- Ij - the jth item, 1 ≤ j ≤ m
- hj - the number of fuzzy regions for Ij
- Rjl - the lth fuzzy region of Ij; l ≤ l ≤ hj
- vij -the quantitative value of Ij in T
- fijl - the membership value of vij in region Rjl
- countjl - the count of the fuzzy region Rjl in D;
- max-countj -the maximum count value among the fuzzy regions of Ij
- max-Rj -the fuzzy region of Ijwithmax _ countj
- s - the predefined minimum support threshold

Besides, a special notation is often used to represent fuzzy sets. Assume that x1 to xn are the elements in fuzzy set A, and µ1 to µn are, respectively, their membership grades in A. A is usually represented
as follows:

$$A = \frac{\mu_1}{x_1} + \frac{\mu_2}{x_2} + \cdots + \frac{\mu_n}{x_n}.$$

The proposed fuzzy fp-tree mining algorithm

The proposed fuzzy FP-tree mining algorithm integrates the fuzzy- set concepts and the FP-tree-like approach to find frequent fuzzy itemsets from quantitative transaction data. The fuzzy FP-tree construction algorithm is designed to generate the tree structure for frequent fuzzy regions (terms). It first transforms quantitative values in transactions into linguistic terms based on Hong et al.'s approach (Hong et al., 1999a, 1999b). Each term uses only the linguistic term with the maximum cardinality in later processes, thus making the number of fuzzy regions processed equal to the number of the original items for reducing the processing time.

The frequent fuzzy itemsets, represented by linguistic terms, are then derived from the fuzzy FP tree.

The fuzzy FP-tree structure could thus help mine fuzzy association rules from quantitative data efficiently and effectively. However, when extending the crisp FP tree to the fuzzy one, the processing becomes much more complex than the original since fuzzy intersection in each transaction has to be handled. The fuzzy FP-tree construction algorithm is stated below.

Procedure for construction of fuzzy FP- tree to determine frequent data itemset to be replicated

INPUT: A quantitative database consisting of n transactions, a set of membership functions, and a predefined minimum support threshold s.

Steps:

1. Transform the quantitative value v_{ij} of each item I_j in the ith transaction into a fuzzy set fij represented as $(f_{ij1}/R_{j1} + f_{ij2}/R_{j2} + \ldots + f_{ijn}/R_{jhj})$ using the given membership functions, where h is the number of fuzzy regions for I_j; R_{jl} is the lth fuzzy region of I_j; $1 \le l \le h$, and f_{ijl} is v_{ij}'s fuzzy membership value in region Rjl membership functions, where h is the number of fuzzy regions for I_j; R_{jl} is the lth fuzzy region of I_j; $1 \le l \le h$, and f_{ijl} is v_{ij}'s fuzzy membership value in region R_{jl} .
2. Calculate the scalar cardinality of each fuzzy region R_{jl} in the transaction data as:

$$Count\, j_l = \sum_{I=1}^{n} f_{ijl}$$

3. Find max- $count_j$ = Max (count j_1) for j = 1 to m, where h_j is the number of fuzzy regions for dataset item i and , m is the number of items. Let max-R_j be the region with max-$count_j$ for item Ij. It will then be used to represent the fuzzy characteristic of item Ij in the later mining process.
4. Check whether the value *max-countj* of a fuzzy region *max-Rj*, *j* = 1 to *C*, is larger than or equal to the predefined minimum count *G*ƫ* . If a fuzzy region *max-Rj* satisfies the above condition, put the fuzzy region with its count in *L1*. That is

$$L_1 = \{max\text{-}R_j \mid max\text{-}count_j \ge G^* ƫ$$

5. Build the Header_Table by keeping the frequent fuzzyregions in L1 in the descending order of counts.
6. Remove the fuzzy regions of the items not in L1 from the transactions of the transformed database.
7. Sort the remaining frequent fuzzy regions in each transaction by their membership values in a descending order.
8. Initially set the root node of the fuzzy FP tree as {root}.
9. Insert the transactions in the transformed database into the fuzzy FP tree tuple by tuple. The following two cases may exist.

a. If a fuzzy region max-Rj in a transaction has been at the corresponding branch of the fuzzy FP tree for the transaction, add the membership value of max-Rj in the transaction to the node of max-Rj in the branch.

b. Otherwise, add a node of max-Rj at the end of the corresponding branch, set the count of the node as the membership value of max-Rj, and insert a link from the node of max-Rj in the last branch to the current node. If there is not such a branch with the node of max-Rj, insert a link from the entry of max-Rj in the Header-Table to the added node.

The Fuzzy Association Rule Extraction for Classification

For each class Cj :

Step 1: Calculate the minimum support of the class Cj according to eq. (8).

Step 2: Create the levels 0 and 1 of the tree.

Step 3: Create a new level in the tree.

Step 4: Prune nodes.

Step 5: If there are more than 2 nodes in the new level and the depth of the tree is less than Depthmax, go to Step 3.

Step 6: Generate the rules with the class Cj on the right-hand sid

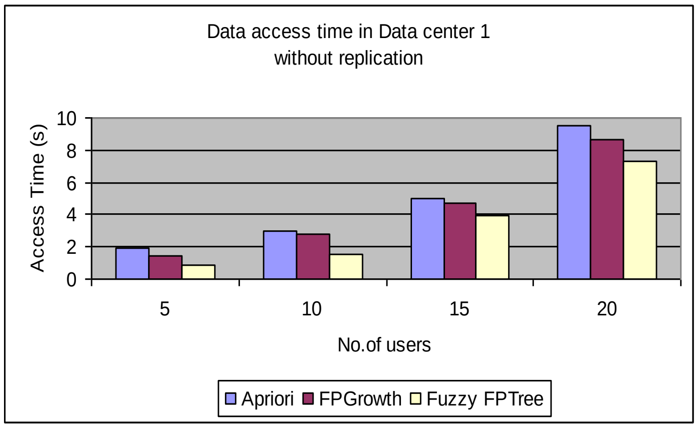
V. *RESULTS AND DISCUSSION*

The proposed method for security model in cloud computing is implemented in the JAVA platform. The cloud data are dived into various datacenters and in each data center the replication data sets are identified and the results we obtain shows that that our method helps in developing an improved data replication system when compare to the existing methods with better efficiency and reduced data loss probability. The Fuzzy FP tree based frequent pattern mining technique employed in our proposed method provides better replication strategies of data's from the various data centers that are being formed.

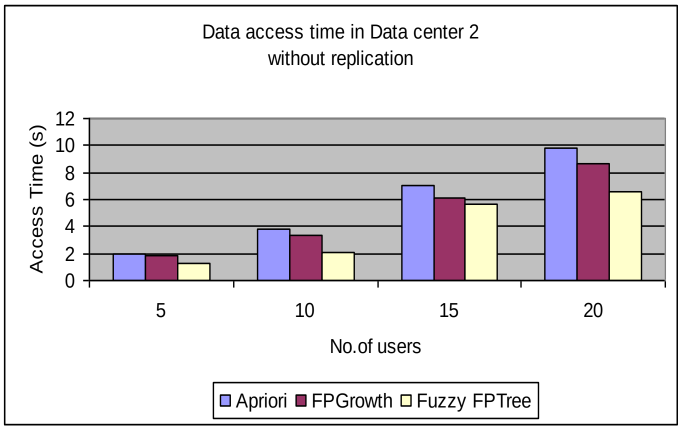
The data access time in cloud computing plays a major role in its effective functioning and the table 1 given below shows the Data access time taken by our proposed method with different users utilizing the data centers.

Number of users	Data Access Time (s)								
	Data Center1			Data Center 2			Data Center 3		
	Apriori	FPGrowth	Fuzzy FPTree	Apriori	FPGrowth	FuzzyFPTree	Apriori	FPGrowth	Fuzzy FPTree
5	1.89	1.45	0.89	1.95	1.86	1.24	2.68	2.14	1.69
10	3.02	2.76	1.52	3.84	3.29	2.09	4.52	3.85	2.72
15	5.02	4.68	3.92	7.02	6.15	5.71	7.94	6.72	4.93
20	9.52	8.61	7.3	9.84	8.7	6.58	10.87	9.02	8.82

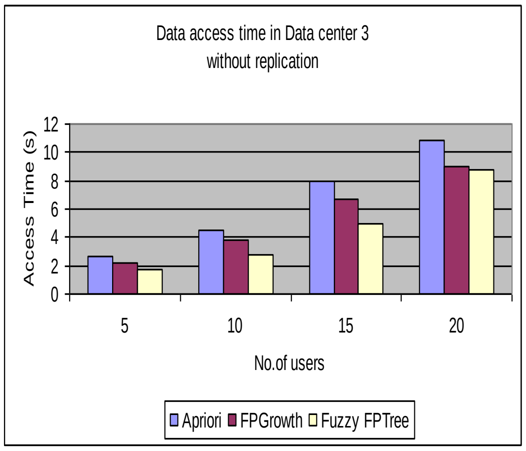
Table 1: Performance comparison of Access time of data by different number of users after performing without Data Replication



(a)



(b)



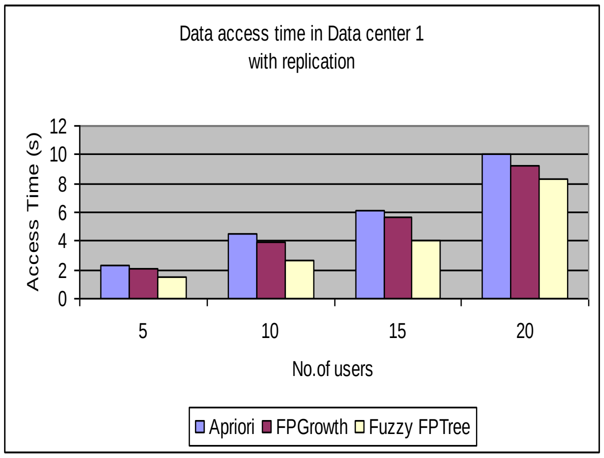
(C)

Graphical Representation of Access time of data by different number of users without Data replication in Data Center 1 (a) Data Center (2) and Data Center (3)

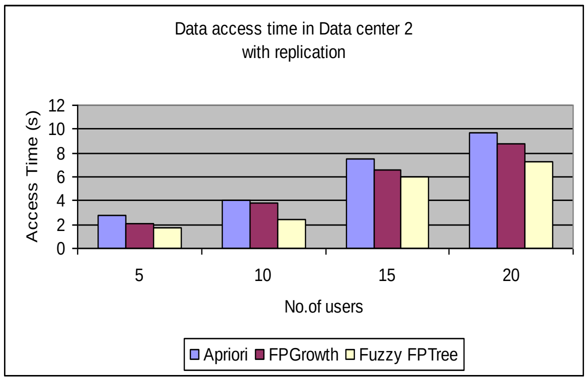
The Table 1 shows the similar tabulation value with access time taken by different users when replication strategy is not performed and the values shows that our proposed method of data replication delivers better data access time without much delay while comparing the existing apriori and fpgrowth.

Number of users	**Data Access Time (s)**								
	Data Center1			**Data Center 2**			**Data Center 3**		
	Apriori	**FPGrowth**	**Fuzzy FPTree**	**Apriori**	**FPGrowth**	**FuzzyFPTree**	**Apriori**	**FPGrowth**	**Fuzzy FPTree**
5	2.34	2.02	1.48	2.75	2.04	1.76	3.08	2.76	1.96
10	4.52	3.98	2.61	4.06	3.78	2.43	5.97	4.16	3.15
15	6.12	5.62	4.02	7.49	6.58	5.98	8.52	7.28	5.42
20	10.08	9.26	8.3	9.75	8.74	7.25	11.53	10.83	9.06

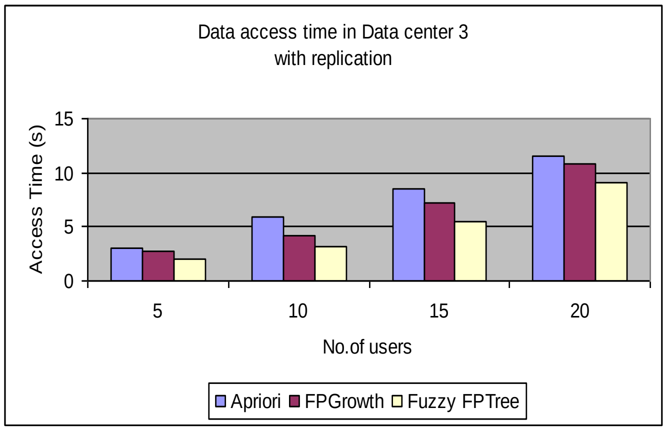
Table 2: Performance comparison of Access time of data by different number of users after performing with Data Replication



a)



b)



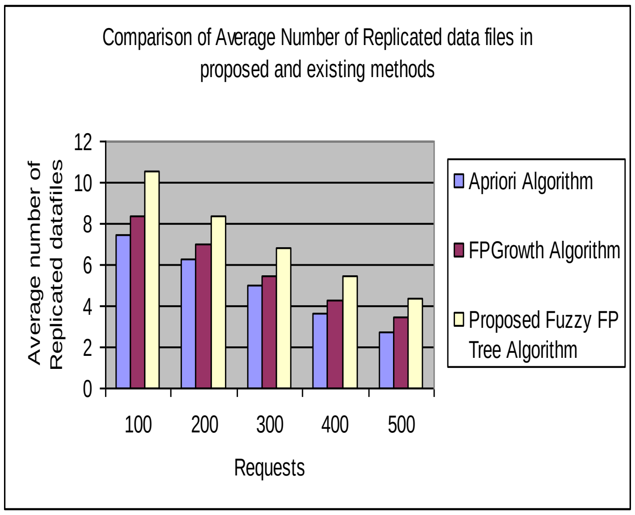
c)

Graphical Representation of Access time of data by different number of users with Data replication in Data Center 1 (a) Data Center (2) and Data Center (3)

The Table 2 and its corresponding graphical representation shows the similar tabulation value

Number of Requests	**Average Number of Replicated Data Files**		
	Apriori Algorithm	FPGrowth Algorithm	Proposed Fuzzy FP Tree Algorithm
100	7.43	8.36	10.56
200	6.27	7.02	8.37
300	4.98	5.47	6.78
400	3.64	4.25	5.49
500	2.69	3.42	4.39

with access time taken by different users when replication strategy is performed and the values



Comparison of Average Number of Replicated data files in proposed and existing methods

CONCLUSION

In this Paper we have proposed an efficient Fuzzy FP Tree based Data Replication Model for Cloud Computing. The Data Replication in Cloud Computing requires an in-depth analysis because it is necessary to provide complete access to the users in cloud systems with less time access. The data replication process can ease out the access process of any kind of data without making delay. Our proposed method we utilized the fuzzy membership function and partition along with FP tree algorithm which proved to be a better mechanism to provide efficient replication process to the cloud system. The results we obtain shows that our proposed method has better results when compared with shows that our proposed method of data replication delivers better data access time without much delay due to its fastness and efficiency of using membership values to determine the frequency of each data itemset.

Table 3: Average Number of Replicated data files corresponding to the user requests

Table 3 shows the average number of replication files that are created by our proposed method Fuzzy FP tree based on the request from the users. These values are compared with that of the existing work and our proposed method proved to be more efficient than the existing one where Tree search method is used for replication. The corresponding graphical representation is shown in figure 4.

the existing methods of data replication in cloud computing.

References

1. Buyya, R., et al., Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst., 2009. **25**(6): p. 599-616
2. Bj, M., et al., Optimizing service replication in clouds, in Proceedings of the Winter Simulation Conference. 2011, Winter Simulation Conference: Phoenix, Arizona. p. 3312-3322.
3. Ghemawat, S., H. Gobioff, and S.-T. Leung, The Google file system. SIGOPS Oper. Syst. Rev., 2003. 37(5): p. 29-43.
4. Shvachko, K., et al., The Hadoop Distributed File System, in Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies *(MSST)*. 2010, IEEE Computer Society. p. 1-10.
5. Sun, D.-W., et al., Modeling a Dynamic Data Replication Strategy to Increase System Availability in Cloud Computing Environments. Journal of Computer Science and Technology, 2012. 27(2): p. 256-272.
6. Nguyen, T., A. Cutway, and W. Shi, Differentiated replication strategy in data centers, in Proceedings of the 2010 IFIP international conference on Network and parallel computing. 2010, Springer-Verlag: Zhengzhou, China. p. 277- 288.
7. Ghemawat, S., H. Gobioff, and S.-T. Leung, The Google file system. SIGOPS Oper. Syst. Rev., 2003. 37(5): p. 29-43.
8. Shvachko, K., et al., The Hadoop Distributed File System, in Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). 2010, IEEE Computer Society. p. 1-10.
9. Dogan, A., A study on performance of dynamic file replication algorithms for real-time file access in Data Grids, Future Gener. Comput. Syst., 2009. 25(8): p. 829-839.
10. Wei, Q., et al., CDRM: A cost-effective dynamic replication management scheme for cloud storage cluster., in 2010, IEEE International on Cluster Computing. 2010. p. 188 - 196
11. Wang, S.-S., K.-Q. Yan, and S.-C. Wang, Achieving efficient agreement within a dual-failure cloud-

computing environment. Expert Syst. Appl., 2011. 38(1): p. 906-915.

12. Y.K. Kwok, K. Karlapalem, I. Ahmad, N.M. Pun, Design and Evaluation of data allocation algorithms for distributed database systems, IEEE J. Selected Areas Comm. (Special Issue on Distributed Multimedia Systems), 14(7) (September 1996) 1332–1348
13. L.W. Dowdy, D.V. Foster, Comparative models of the file assignment problem, ACM Comput. Surveys 14(2) (June 1982), 287–313
14. B. Awerbuch, Y. Bartal, A. Fiat, Optimally-competitive distributed file allocation, 25th Annual ACM STOC, Victoria, BC, Canada, 1993, pp. 164–173.
15. O. Wolfson, S. Jajodia, Y. Huang, An adaptive data replication algorithm, ACM Trans. Database Systems 22(4), 255–314 (June 1997).
16. Agrawal, R. & Srikant, R. (1995). Mining sequential patterns. In The eleventh IEEE international conference on data engineering (pp. 3–14).
17. Han, J., Pei, J. & Yin, Y. (2000). Mining frequent patterns without candidate generation. In The 2000 ACM SIGMOD international conference on management of data (pp. 1–12)
18. F. Javier Lopez, Marta Cuadros, Armando Blanco and Angel Concha:“Unveiling Fuzzy Associations Between Breast Cancer Prognostic Factors and Gene Expression Data”, 20th International Workshop on Database and Expert Systems Application, pp.338-342,(2009).