Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 5522026, 14 pages
https://doi.org/10.1155/2021/5522026

*Research Article*

# Improve Performance by a Fuzzy-Based Dynamic Replication Algorithm in Grid, Cloud, and Fog

**Mahsa Beigrezaei**,[1] **Abolfazel Toroghi Haghighat**,[1] **and Seyedeh Leili Mirtaheri**[2]

[1]*Faculty of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*
[2]*Electrical and Computer Engineering Department, Faculty of Engineering, Kharazmi University, Tehran, Iran*

Correspondence should be addressed to Abolfazel Toroghi Haghighat; haghighat@qiau.ac.ir

Received 31 January 2021; Revised 25 May 2021; Accepted 12 June 2021; Published 22 June 2021

Academic Editor: Lei Liu



The efficiency of data-intensive applications in distributed environments such as Cloud, Fog, and Grid is directly related to data access delay. Delays caused by queue workload and delays caused by failure can decrease data access efficiency. Data replication is a critical technique in reducing access latency. In this paper, a fuzzy-based replication algorithm is proposed, which avoids the mentioned imposed delays by considering a comprehensive set of significant parameters to improve performance. The proposed algorithm selects the appropriate replica using a hierarchical method, taking into account the transmission cost, queue delay, and failure probability. The algorithm determines the best place for replication using a fuzzy inference system considering the queue workload, number of accesses in the future, last access time, and communication capacity. It uses the Simple Exponential Smoothing method to predict future file popularity. The OptorSim simulator evaluates the proposed algorithm in different access patterns. The results show that the algorithm improves performance in terms of the number of replications, the percentage of storage filled, and the mean job execution time. The proposed algorithm has the highest efficiency in random access patterns, especially random Zipf access patterns. It also has good performance when the number of jobs and file size are increased.

## 1. Introduction

Distributed environments such as Data Grid and cloud deal with data sharing and performing data-intensive tasks [1]. Due to the huge volume of data required by the data-intensive tasks and the high latency in these environments, data availability and quick data access are challenging [2]. Data replication is one of the most important approaches to increase data access performance and data availability [3, 4]. It can provide many advantages. For example, it can decrease data access time and bandwidth consumption. It also can enhance availability, scalability, and load balancing [5]. Replication methods create multiple copies of files and place them near the data requester [6]. Data replication has been widely applied in database management systems (DBMS) [7], distributed database [8], mobile system [9], Data Grid [7], Cloud [10, 11], peer-to-peer (p2p) [12], and Fog [13].

Replication methods consist of three main phases: replica selection, replica placement, and replica replacement [14]. The performance of the replication methods in each part directly impacts the efficiency of data access and execution time of data-oriented jobs in the distributed systems. By looking more closely at replication algorithms, we can understand that replication algorithms in each part can be considered as a multimetric decision problem. Distributed systems have dynamic nature, and their resources are heterogeneous. The resources have various dynamic and static properties. Their specifications have direct and indirect effects on the efficiency of decision-making in these environments. Many dynamic replication algorithms have been proposed till now. These methods often assume that distributed systems are homogeneous or consider only one or a limited number of resource specifications as decision-making parameters. This shortcoming can lead to inefficiencies or reduced efficiency of replication algorithms in

real distributed systems. It is necessary to consider appropriate decision parameters in each part of the replication algorithm to achieve maximum efficiency. Therefore, presenting an efficient replication algorithm that uses instrumental parameters in its different decision-making parts to improve the performance of real distributed systems is still an open issue. In previous work, in each part of the replication algorithm, there is a lack of attention to some influential decision-making parameters. In the following, we discuss them in more detail.

The decision in each part of the replication algorithm depends on different parameters. The file popularity is an effective parameter that most existing replication methods have considered in the replica placement and replica replacement parts. Most of the existing replication methods often consider only the number of file accesses in the current interval in estimating the file popularity in a storage node. Indeed they consider current file popularity instead of future file popularity in decision-making. Replication algorithms should consider the future number of accesses to file (future file popularity) for more improvement. Although predicting the future number of accesses is a challenge, we should note that the file popularity metric is not enough to select a suitable replication place (storage node) because resources are heterogeneous in the real distributed systems. Suppose that only file popularity is considered in decision-making. In this case, most file access requests (locally or remotely) may run at nodes with low storage speed or low response rates or low communication capability and centrality that can dramatically increase job completion time. Besides, replication in the wrong place may lead to overload and inefficiency. Therefore, to select the appropriate node, it is necessary to consider a proper set of static and dynamic characteristics of resources such as communication capacity, workload, bandwidth, centrality, storage speed, and future popularity. Existing methods often assume that the resources are homogeneous or consider a limited number of metrics, such as the number of accesses and last access time and workload. Using an appropriate, effective, and comprehensive set of parameters in decision-making is a challenge in increasing efficiency. The replica selection part is another important part of the replication algorithm. Decision parameters that are considered in this part have a direct impact on replication algorithm performance. Queue waiting time is one of the most influential parameters. The higher the storage queue workload, the longer the access delay. Lack of attention to this parameter when selecting replica for remote access and finding suitable replica location leads to workload imbalance and high access delay. On the other hand, failures in distributed environments such as Cloud and Grids are common [15]. If the data request is sent to a high failure probability node, a delay is imposed on the data access time. Ignoring these metrics can reduce efficiency. In most previous works, less attention has been paid to the delays due to failure and workload in the storage queue.

Another difficulty is how to make decisions with multiple criteria (decision parameters) in each part of the algorithm. In each part, the algorithm must select the most appropriate item among existing items. One of the common methods is to evaluate items based on evaluation criteria by a function. The function is often presented innovatively or modeled using two methods: weight sum model (WSM) and fuzzy-based model. WSM is straightforward and is used in most previous replication algorithms. Determining weights is very important in WSM. When the number of criteria increases, the correct calculation of these weights can be very difficult, so each metric's impact on the valuation may not be determined correctly. Fuzzy set theory can show real-world knowledge in the face of uncertainty. Fuzzy modeling, which has been considered in a small number of previous replication methods, can determine the effect of each metric individually on valuation. The fuzzy-based method models valuation function by a fuzzy inference system. The rules and knowledge of the fuzzy system can be changed with the lowest cost. The use of a fuzzy system can provide a high degree of flexibility. The fuzzy inference system is relatively fast and flexible, and its modeling is not very complex. On the other hand, the existing fuzzy-based replacement algorithm shows the acceptable efficiency of the fuzzy method [16]. The mentioned advantage motivates using the fuzzy-based functions to value items based on multidecision parameters.

This paper proposes a new replication algorithm named effective fuzzy-based replication algorithm (EFRA) to solve the mentioned challenges. EFRA for decision-making in each part uses the valuation of items based on a new set of effective decision parameters. EFRA for decision-making in each part utilizes a decision function. The function values and evaluates the existing items for selection based on a new set of decision parameters. We model the decision function in the replica selection part based on the weight sum model (WSM). We used a WSM-based function in the replica selection part. EFRA in the replica selection part selects a replica with minimum transfer time, queue waiting time, and failure probability in the replica node in a hierarchical manner. Decision parameters are ambiguous, linguistic, and fuzzy in the replica placement and replica replacement parts of the replication algorithm. The relationship between fuzzy rules and valuation is extractable. The fuzzy-based decision is consistent with the conditions of the problem. Due to the mentioned advantage of the fuzzy-based decision, we used fuzzy-based functions to evaluate nodes as a place of replication and evaluate the replicas as a candidate for deletion. To select the appropriate location for creating a new replica, EFRA considers the future number of file accesses, last access time, workload, failure probability, centrality, and communication capacity in the candidate nodes. The main contributions in this paper are as follows:

(i) Considering an inadequate set of decision parameters in each part of the replication algorithm can reduce data-oriented jobs' performance and execution time in a real distributed system. We proposed a new effective fuzzy-based replication algorithm (EFRA) that improves the system's performance by considering a new effective and comprehensive set of decision-making parameters in each phase.

(ii) To utilize the advantage of fuzzy-based decision-making, EFRA in two parts of replica replacement and replica placement uses a fuzzy function to evaluate candidate items and increases decision-making efficiency with several parameters.

(iii) To avoid the delays due to failure and queue workload in the nodes, our replication algorithm (EFRA) is presented aware of the workload and failure. EFRA in the replica selection and replica placement parts considers these two parameters and other effective parameters. It prevents the delay caused by the execution of data access requests in nodes with a high workload and high failure probability.

(iv) In order to increase data locality, our proposed algorithm makes decisions based on the future file popularity in the replacement and placement parts. It uses the Simple Exponential Smoothing method to predict future file popularity. It also presents a metric named communication capacity (CC) that can evaluate the centrality and bandwidth capacity. By storing a new replica in a node with high communication capacity, the algorithm reduces access delay, bandwidth consumption, and mean job execution time.

(v) To evaluate the EFRA replication algorithm, we used the OptorSim simulator. The simulation results showed that EFRA could improve the performance compared to other similar algorithms in terms of mean job execution time, effective network usage (ENU), replication number, and percentage of storage usage. Evaluation of algorithms performance in three access patterns revealed that EFRA has the most significant improvement in random Zipf access patterns and when the number of jobs and file size are large. Besides, when the access pattern is sequential and the storage space's size is large, our proposed replication algorithm does not improve significantly compared to other algorithms. The performance of the EFRA in various metrics proved that the EFRA could make a more accurate decision than other methods by considering more comprehensive and effective parameters and utilizing fuzzy-based valuation in decision-making. Despite the advantages of fuzzy-based valuing, it should be noted that setting correct membership functions in a real system can be difficult. If not done properly by an expert, it can lead to undeniable inefficiency.

The other sections of this paper are organized as follows. Section 2 presents the related works; in Section 3, we describe our proposed method; then, in Section 4, we evaluate the method and show the results of our simulation. Section 5 concludes the paper and explains future works.

## 2. Related Works

Researchers in some previous works addressed data replication. We mention some of them below.

In [17], the Least Frequently Used (LFU) and two economic strategies were proposed. These methods take place replication when the file is accessed remotely. Another similar replication algorithm called Least Recently Used (LRU) was proposed in [18]. LFU and LRU delete the least frequently accessed files and least recently used files, respectively. In [4], a Bandwidth Hierarchy Replication (BHR) algorithm was presented, which works based on the network level locality. BHR significantly reduces the cost of data access by replicating the data required by the job near the place where the job is executed. It reduces interregion transmission and enhances performance. Sashi and Thanamani presented a modified version of the BHR algorithm (MBHR) [19]. It improves the performance compared with the BHR by selecting a node with the maximum number of accesses in the requesting region. In [2], a new method called Hierarchy Replication Strategy (HRS) was proposed. It keeps a copy of needed files in the local cluster. Hence, data requests in a cluster can access the data at a low cost. Rajaratnam proposed a dynamic replica placement algorithm with load balancing (RPLB) [20]. RPLB replicates files in a node with the highest degree and access frequency and the lowest workload. The results showed that RPLB could enhance performance parameters such as the effective network usage (ENU) and job execution time. In [21], the authors presented three algorithms: a replication algorithm, a job scheduling algorithm, and a job migration method. Decision-making in this algorithm is based on the workload parameter. In [22], replication algorithms were introduced, which utilize a hierarchical three-level architecture. It replicates popular files at scheduling time. It also uses a migration method to improve workload. Beigrezaei et al. presented a fuzzy-based replication method named Fuzzy_Rep [23]. It utilizes a fuzzy inference system with three input parameters, bandwidth, the last access time, and the number of accesses, to select a tailored place for replication. Results showed that Fuzzy_Rep algorithm could reduce the mean job time and network usage. Also, a new fuzzy-based replication algorithm (FRA) was presented in [16]. It uses a fuzzy inference system to calculate file value and select a low-value file for deleting. John developed a swarm-based method called D2R-IWD [24]. It determines the number of replicas for each file based on the number of file accesses, file size, the number of available replicas, and a threshold. It also uses the intelligent water drop (IWD) algorithm to select the best replica for user requests. In [13], a Markov-based replication method at edge-cloud computing was proposed. A gray Markov chain prediction model calculates the required number of replicas for each block of files. It determines replica location based on the Fast Nondominated Sorting Genetic algorithm. The algorithm improves response time, write delay, read throughput, and workload balance on nodes. A cost-aware data replication algorithm was provided by Gill and Singh [25]. It calculates the number of replicas based on data available in the system. It also uses the concept of knapsack to improve the cost of replication. For this purpose, replicas are transferred from expensive nodes to low-cost nodes. Sun et al. proposed a dynamic replication algorithm (DARS) under high load Cloud-P2P that reduces

the workload on nodes and access latency [26]. DARS calculates an overheating similarity range using the fuzzy membership function. If the overheating similarity in each node exceeds a predefined similarity, it saves the replica for popular files in the neighboring nodes with the highest node degree and low load. In [14], a centralized replication algorithm named RCP was presented. It predicts the number of needed replicas for each file in each region. When storage space is not enough for replication, RCP deletes the least recently used file.

Resources in distributed environments are heterogeneous. A set of dynamic and static characteristics, directly and indirectly, affects the replication algorithm's performance in the distributed system. In Table 1**,** dynamic replication algorithms are compared in terms of the considered parameters in different phases of the replication algorithm (replica selection, replica placement, and replica replacement). As Table 1 shows, one of the shortcomings of the previous methods is that most of them have considered only one or a limited number of decision-making parameters in each part. This shortcoming can reduce the algorithm's efficiency or lead to proper performance only in certain circumstances. On the other hand, some parameters such as failure probability, communication capacity, file size, and future file popularity (in replica placement and replacement phase) have not been considered.

## 3. Proposed Method

The assumed network structure is explained in this section; then the proposed replication algorithm (EFRA) and its various parts (replica selection, replica replacement, and replica replacement) are described.

*3.1. Network Structure.* Figure 1 illustrates the assumed network structure in our proposed method. In the proposed method, we assume that the distributed system consists of a set of independent nodes $\{node_1, node_2, node_3, ..., node_n\}$ with computing elements (CE) or storage elements (SE) that are connected through communication links with the specific communication bandwidth $BW\{BW_{1,1}, BW_{1,2}, \ldots, BW_{i,j}, \ldots, BW_{n,n}\}$. There are $h$ various files $\{f_1, f_2, \ldots, f_h\}$ in the system stored in different nodes. Each file has an original version and can have zero or several replicas. We also assume that the files are read-only. User requests are given in the form of jobs/tasks to resource brokers (RB). The broker dispatches jobs to the suitable nodes in regions for execution. Each job may request one or more files. As shown in Figure 1, the assumed network structure is a hierarchical architecture consisting of two levels (regions and nodes). The data transfer time within the region is lower than that among the regions. In each region, a node with high computing power and storage is considered as the region header. The region headers monitor nodes within the region and store requirement information for our proposed algorithm, such as communication bandwidth, files, and replicas information, the latency between nodes, file access history, and storage queue workload, as well as physical properties of nodes. The proposed replication algorithm is performed in nodes in a decentralized manner. There are two software components called local job scheduler (LJS) and data scheduler (DS) in each node. Figure 1 illustrates the components and their relationship. DS has two internal components called local replica manager (LRM) and local replica catalog (LRC). LJS and DS are responsible for managing the execution of jobs and managing data in node, respectively. LRM is responsible for managing replicas and access files through their replicas. LJS communicates with the LRM to access the needed data remotely by a replica. LRM executes the proposed replication algorithm. It receives the information needed to run the algorithm from the local replica catalog (LRC). LRC contains the replicas information such as replica addresses, file access history, and other needed information for performing the EFRA. When the information in LRC changes (creating/deleting a replica), the changes are informed to the related region header server (RS) by the LRM; then the RS updates its replica catalog and sends updated replica catalog information to the nodes and other RSs at intervals.

*3.2. The Proposed Algorithm: Effective Fuzzy-Based Replication Algorithm (EFRA).* Remote access increases job execution time and decreases the performance of the distributed system. If a job does not have its required data locally, it must access the data remotely. Data replication is a vital optimization step to manage data access. It places some replicas in geographically distributed data stores near the data requesters. Therefore, to minimize data access time and data-oriented job execution time, dynamic replication is required. This paper proposes a dynamic replication algorithm named effective fuzzy-based replication algorithm (EFRA) that improves job data-oriented execution time.

The resource broker (RB) gets jobs from users, schedules them based on job scheduling policy, and finds suitable computing nodes for job execution; afterwards, the jobs are dispatched to the suitable nodes. When a job is executed in a node, if the job requires a not locally available file, the local job scheduler (LJS) in the node sends the file access request to the local replication manager (LRM) for consideration. The LRM calls the EFRA. LRM uses information in the local replica catalog (LRC) for performing the EFRA. Algorithm 1 presents the proposed replication algorithm (EFRA). EFRA consists of three main parts: replica selection, replica placement, and replica replacement described. In the replica selection part, the best replica among existing replicas for a remote file access request is selected by an algorithm that is shown in Figure 2. The replica placement part determines the appropriate location to store the new replica using Algorithm 2. If there is not enough space to store the new replica, the EFRA in its replacement part provides the required free space by removing the low-value replicas using Algorithm 3. These parts are explained in the following sections.

*3.2.1. Replica Selection.* In the distribution systems that use the data replication method, many replicas may be created for a popular file to increase performance. Hence, when a job

Table 1: Comparison of various replica algorithms and considered parameters in replica selection, replica placement, and replica replacement sections.

| | Replica selection | | | | | Replica placement | | | | | | | Replica replacement | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | TD | SWL | SO | SS | FP | SWL | FP | CC | SS | NA | LA | FPO | NA | LA | RAT | FS | FPO |
| LFU [17] | * | — | — | — | — | — | — | — | — | — | * | — | * | — | — | — | — |
| LRU [18] | * | — | — | — | — | — | — | — | — | — | * | — | — | * | — | — | — |
| BHR [4] | * | — | — | — | — | — | — | — | — | * | — | — | * | — | * | — | — |
| HRS [2] | * | — | * | — | — | — | — | — | — | * | — | — | * | — | * | — | — |
| MBHR [19] | * | — | — | — | — | — | — | — | — | * | — | — | * | — | * | — | — |
| Fuzzy_Rep [23] | * | — | — | — | — | — | — | — | — | * | — | — | — | * | * | — | — |
| PPRA [27] | * | — | — | * | — | — | — | — | * | * | * | — | * | — | * | — | — |
| RPLB [20] | * | — | — | — | — | * | — | — | — | * | — | — | — | * | — | — | — |
| Method in [21] | * | — | — | — | — | — | — | — | — | — | * | — | * | — | — | — | — |
| Method in [13] | * | * | — | — | — | * | * | * | * | — | — | — | — | — | — | — | — |
| CRP [14] | * | — | — | — | — | — | — | — | * | * | * | — | — | * | — | — | — |
| DRLBS [22] | * | * | — | * | — | — | — | — | — | * | — | — | * | * | * | — | — |
| D2R-IWD [24] | * | * | — | — | — | * | — | — | * | — | — | — | * | — | * | — | — |

TD: transfer delay, SWL: storage queue workload, SO: search overhead, SS: the storage speed, FP: failure probability, CC: communication capacity, NA: number of accesses to file, LA: last access time, FPO: future file popularity, RAT: remote access time, and FS: file size.

requests a file that does not exist locally, a most appropriate replica must be selected for efficient remote access. EFRA uses a replica selection algorithm to find the most appropriate replica. For this purpose, EFRA calls a replica selection algorithm. Our proposed selection algorithm uses the replicas list in the LRC to find a suitable replica. Figure 2 shows the flow chart of the replica selection method in the EFRA.

As shown in Figure 2, the EFRA, in the selection part, creates a list of existing replicas in the current region (List1). Then it selects a replica with low file transfer cost (TC), low queue delay, and low failure probability. For this purpose, it evaluates the existing replicas in List1 using equation (1). Then it selects a replica with the maximum value (ReplicaCost). If a replica of file does not exist in the current region, then the EFRA replica selection algorithm searches among replicas within other nearest regions (neighborRegion). Searching for the best replica among a large number of replicas would lead to long latency. The replica selection process in the EFRA has two steps. It uses a hierarchical search. Used hierarchical searching method decreases searching time. In equation (1), TC is file transfer cost, and FP is failure probability in node $g$ with a replica of requested file $f$. Equation (1) models the value of each replica by the WSM model. The TC and FP are disunited by normalizing methods before usage, since the criteria should have the same unit. In equation (1), $w_1$ and $w_2$ are appropriate weights. These weights determine each parameter's impact in determining the replica's value. The weights are calculated empirically. The file transfer cost TC is calculated by equation (2). In equation (2), $\text{size}_r$, bandwidth, $\text{DS}_a$, Delay_Queue, and $\text{PropagationDelay}_{\text{ag}}$ are the size of file "$r$," available bandwidth between node "$g$" and node "$a$," disk speed in node "$a$," the mean waiting time in the disk queue of the node "$a$," and propagation delay between node "$a$" and node "$g$," respectively. $\text{Delay\_Queue}_a$ is estimated by equation (3), where $n$ is the number of file access requests that are waiting in the storage queue and $\text{Size}_z$ is the size of the requested file.

$$\text{ReplicaCost} = w_1 * \text{TC} + w_2 * \text{FP}, \tag{1}$$

$$\text{TC}(f, a, g) = \frac{\text{size}_f}{\min\left(\text{DS}_a, \text{bandwidth}_{\text{ag}}\right)} + \text{PropagationDelay}_{\text{ag}} + \text{Delay\_Queue}_a, \tag{2}$$

$$\text{Delay\_Queue}_a = \sum_{z=1}^{n} \frac{\text{Size}_z}{\text{DS}_a}. \tag{3}$$

*3.2.2. Replica Placement.* In the proposed algorithm, when a job requires a file, which exists in the local storage, it accesses the file locally. If the file does not exist locally, replication takes place. The needed files should be replicated in the suitable node where the file will be accessed soon, have high communication capacity, and have a low workload. The placement part of the EFRA is shown in Algorithm 2. It evaluates nodes in the region with the workload and failure probability less than a threshold. It uses a fuzzy-based valuation function (FuzzyNodeValuFunction). Then, it selects a node with the highest value (NodeValue) as the best place for replication. In this paper, the valuation function is

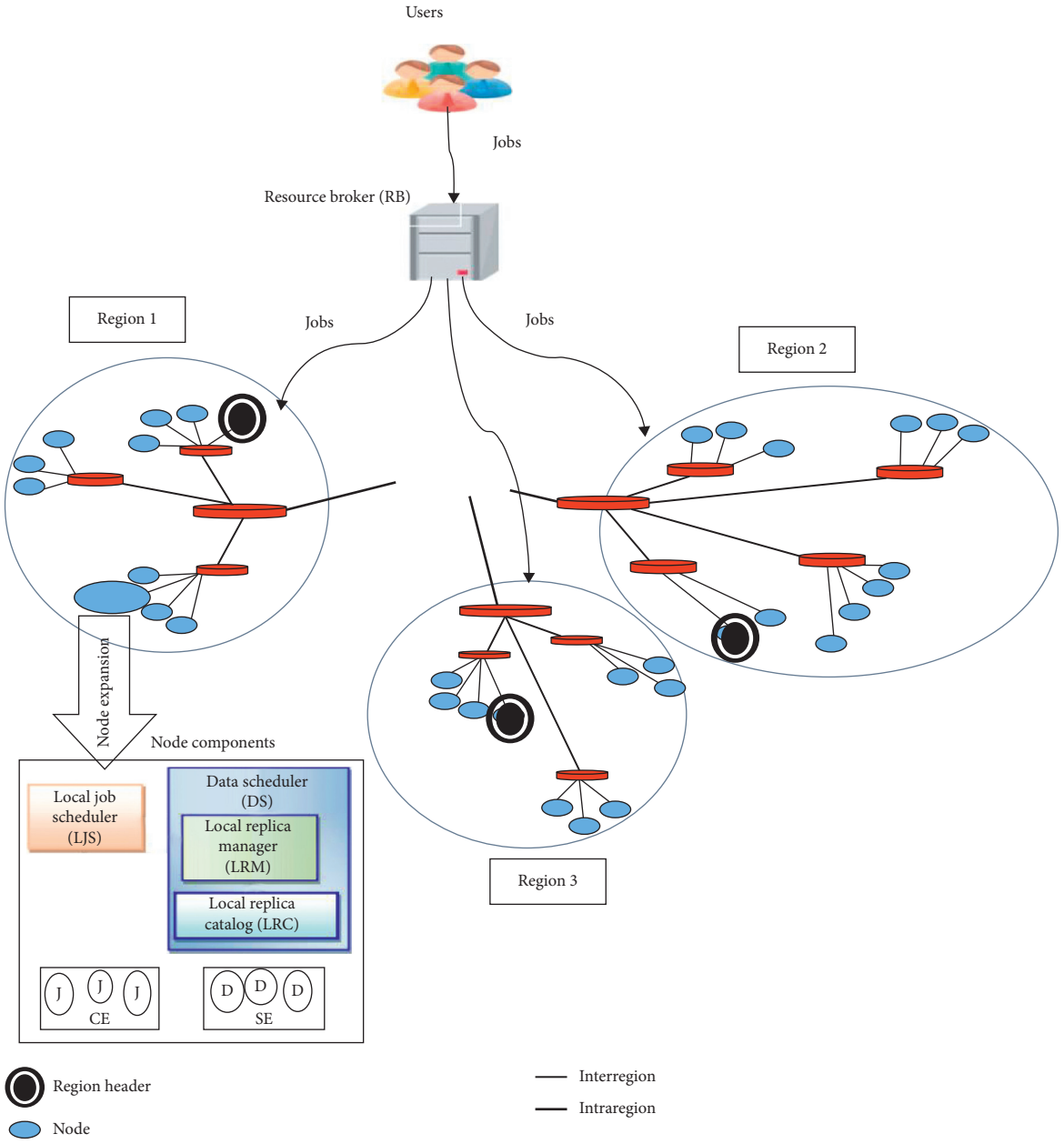


Figure 1: The network structure in the proposed algorithm.

implemented using a fuzzy inference system with four input parameters and one output. In Figure 3, an abstraction of the used fuzzy inference system is shown. The output shows the value of a node (node $i$). The inputs of the fuzzy inference system are as follows.

*(1) The Number of Accesses in the Future (NA).* The number of future accesses (NA) indicates the number of possible accesses to file $f$ in node $i$ in the future. If node $i$ has a high number of accesses to file $f$ in the future, it can be considered a good candidate for replication. This paper uses the Simple Exponential Smoothing method to predict the number of future accesses to a file. This method predicts the number of future accesses based on file access history. It calculates the number of future accesses to file $f$ ($\mathrm{NA}_{t+1}$) in period $t+1$ by equation (4). In this equation, $A_t$ is the actual value of the number of accesses to file $f$ in period $t$, $\mathrm{NA}_t$ is the predicted value of the number of accesses in period $t$, and $\alpha$ is a constant value between 0 and 1.

$$\mathrm{NA}_{t+1} = \alpha.A_t + (1-\alpha).\mathrm{NA}_t, \tag{4}$$

$$\mathrm{NA}_t = \alpha.A_{t-1} + (1-\alpha).\mathrm{NA}_{t-1}, \tag{5}$$

$$\mathrm{NA}_{t-1} = \alpha.A_{t-2} + (1-\alpha).\mathrm{NA}_{t-2}, \tag{6}$$

```
Begin
  If job j in node i request file f, which is not available in node i locally, Then
    BestReplica = EFRA_ReplicaSelection (f, i); //it calls the replica selection algorithm shown in Figure 3, where BestReplica is a
suitable replica for remote access to file f from the node i
  Access to file F from node BestReplica.node;
  BestPlace = EFRAPlacmentAlgorithm (I,f) //it call algorithm 2 for determing best place
      If f exists in BestPlace, Then
        Exit;
      End If
      If BestNode.FreeStorageSize〉 = f.size Then
        Replicate file f to BestNode and exit
      Else
        EFRA_ReplacementAlgorithm(I,f); // it calls algorithm 3 to provide enough free storage space
      End if
    End if
  End if
```

Algorithm 1: EFRA replication algorithm (file $f$, job $m$, and node $i$).

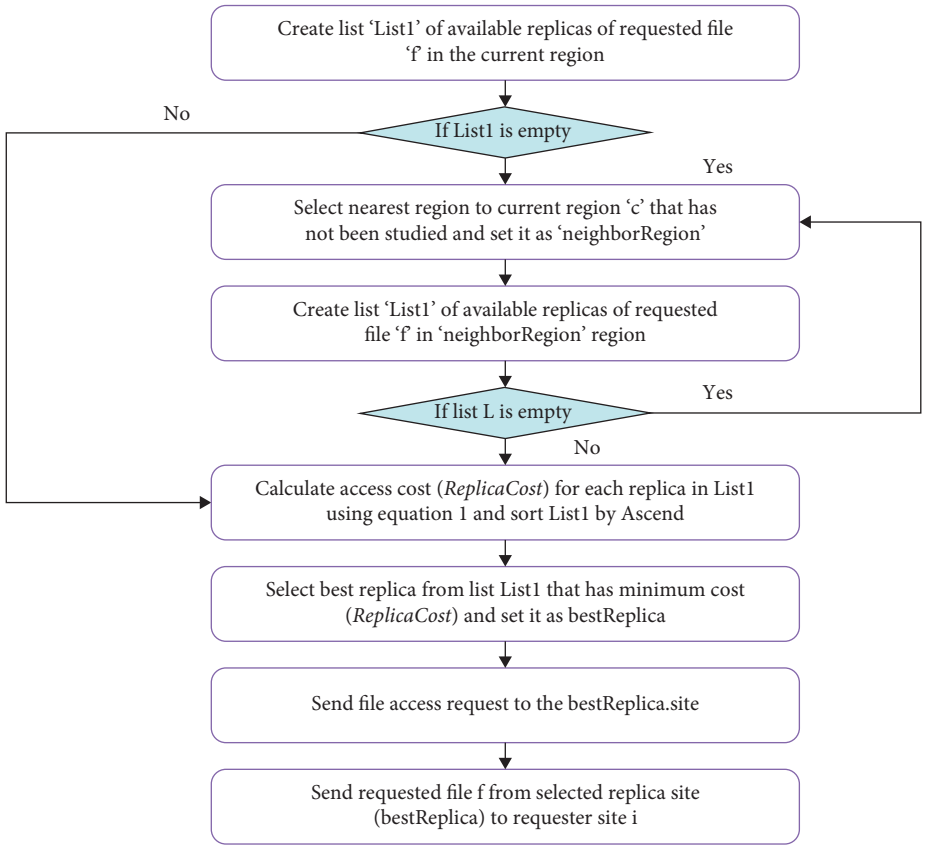


Figure 2: The replica selection algorithm as part of the EFRA.

$$\mathrm{NA}_{t+1} = \alpha.A_t + \alpha.(1-\alpha).A_{t-1} + \alpha.(1-\alpha)^2.A_{t-2} + \alpha\cdot(1-\alpha)^3.\mathrm{NA}_{t-2}. \quad (7)$$

*(2) The Communication Capacity (CC).* The higher the node's communication capacity, the faster the access time. CC can also show the centrality of node $i$. The communication capacity (CC) for node $i$ in region $c$ is calculated by equation (8), where $n$ is the number of the nodes within region $c$, and $\mathrm{bw}_{ij}$ is the available bandwidth between node $i$ and node $j$.

$$\mathrm{BW}_{c,i} = \frac{\sum_1^n \mathrm{bw}_{i,j}}{n-1}. \quad (8)$$

*(3) The Last Access Time Interval (TL).* Based on temporal locality, if a file is recently requested in a node, it will be

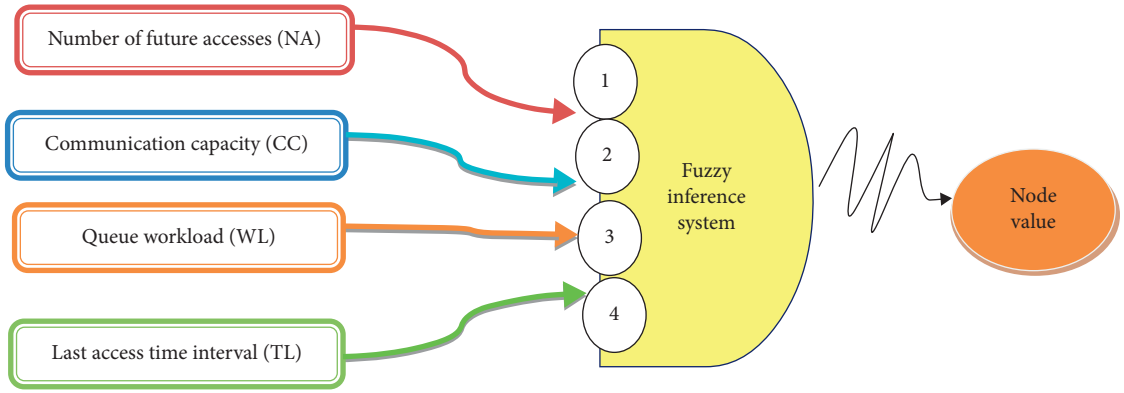


Figure 3: An abstraction of the used fuzzy inference system for determining a node's value.

```
Begin
  For each node i that has Storage Element (SE) in the current region do
    If FP_i < T failurether and workload_i < T workload Then
      Bestnodelist = add node i;
      Bestnodelist[i].NodeValue = FuzzyNodeValuFunction (i, f, CC_i, NA, TL_i,WL_i);
    End If
  End For
  BestNodeList; = sort the BestNodeList list in ascending order according to their node value
  If BestNodeList; ! = empty then
      BestPlace = select a node from top of the BestNodeList;
       Return BestPlace
  End If
End
```

Algorithm 2: EFRA placement algorithm (node $i$ and file $f$).

```
Begin
  recentAccessHistoryList = getRecentAccessHistory (long dt); //Return the recent access history
  For each replica file "f" in storage Node i Do
    If a is removable and has another replica in Region, Then //a is replica file "f"
      a.value = RepFuzzyFunction (TI, NA, RAC); // it calculates value of file 'f' by an fuzzy inference system
      FilesIsInRigon.add(a); //add replica a in the FilesIsInRigon list
    Else
      a.value = RepFuzzyFunction (TI, NA, RAC);
      FilesIsNotRigon.add(a);
    End If
  End For
  Sort FilesInRigon in ascending order according Its File value;
  while "FilesInRigon" ! = empty Do
    Select a replica from top of the "FilesIsInRigon" and delete it from node 'i';
    If i.availableStorageSize(f.size Then
      replicate file 'f' to node 'i' and exit;
  End While
  Sort FilesIsNotRigon in ascending order according Its File value;
  while FilesIsNotRigon ! = empty Do
    Select a replica from top of the "FilesIsNotRigon" list and delete it from node'i' and list;
    IF i.availableStorageSize(f.size Then
      replicate file 'f' to grid node 'i' and exit;
  End While
End
```

Algorithm 3: EFRA replacement algorithm (node $I$ and file $f$).

requested again soon. The last file access time interval is an effective metric for selecting a suitable place for replication. We calculate the last access time interval by equation (9), where TL is the difference between the present time (CT) and the last access time to file $f$ (AT) in node $i$.

$$\mathrm{TL}_f = \mathrm{CT} - \mathrm{AT}. \tag{9}$$

*(4) The Storage Queue Workload (WL).* A node with a high workload should not be selected as a replication place because it imposes a long delay due to waiting time queue to file access time. The storage queue workload (WL) is calculated by equation (10), where NRW is the number of file access requests waiting in the storage queue in node $i$.

$$\mathrm{WL} = \frac{\mathrm{NRW}}{\mathrm{DS}}. \tag{10}$$

In the used fuzzy system, we considered 40 identical rules. Some of them are explained in Table 2.

*3.2.3. Replica Replacement.* After selecting the appropriate replication location, a new replica is created if enough free storage space is available on the selected node. Else, the EFRA replacement algorithm is called to provide enough storage space. Our proposed replacement algorithm provides free space in the node by valuing and deleting low-value files with low remote access time from the replication place. We calculate file value (FileValue) using a fuzzy function (equation (5)) with three input parameters. The inputs include the time interval between the current time and the last file access time (TI), the number of file accesses in the future (NA), and remote access time to the file (RAC), which is calculated from equation (12). In equation (12), $\mathrm{size}_r$, bandwidth, $\mathrm{DiskSpeed}_a$, and $\mathrm{PropagationDelay}_{\mathrm{ag}}$ are the size of file "$r$," available bandwidth between node "$g$" and node "$a$," disk speed in node "$a$," and propagation delay between node "$a$" and node "$g$," respectively. The number of accesses in the future (NA) is predicted based on the Simple Exponential Smoothing method that was described in the previous section (replica placement). Figure 4 shows an abstraction of the used fuzzy inference system for valuing a file. It shows that the fuzzy system has three inputs and one output, which determines the replica file's value in the future. The fuzzy inference system uses 20 rules. Table 3 shows some of them. For example, rule 1 indicates that if the number of file accesses (NA) is high and the last access time interval (TI) is low and the remote access time to the file is high, then FileValue of the replica is very high.

$$\mathrm{FileValue}\,(f,\ g) = \mathrm{RepFuzzyFunction}\,(\mathrm{TI}, \mathrm{NA}, \mathrm{RAC}), \tag{11}$$

$$\mathrm{RAC}_f = \frac{\mathrm{size}_f}{\min\left(\mathrm{DiskSpeed},\ \mathrm{bandwidth}_{ag}\right)} + \mathrm{PropagationDelay}_{ag}. \tag{12}$$

Algorithm 3 shows the pseudocode of the proposed replacement algorithm. It lists all files with at least one other copy in the region and calculates each candidate file (FileValue) using equation (11) (a fuzzy function). It sorts the candidate list based on the value of files in ascending order. Then a replica with the lowest value is selected and deleted from the beginning of the list until enough space is provided. If free space is sufficient, replication is done. Otherwise, the remaining replicas are listed in the second candidate list (FilesIsNotRigon) and are valued using equation (10). The files are sorted based on file value, and low-value files are deleted until enough space is provided. Finally, the new replica is stored.

## 4. Evaluation Results

We use the OptorSim simulator for evaluation. In this part, first, the OptorSim as a used simulation tool is explained; then the simulation configuration input file and eventually simulation results are given.

*4.1. Simulation Tools.* The OptorSim simulator was developed by the European DataGrid (EDG) project. It can evaluate different replication and job scheduling algorithms and simulate different distributed projects like CMS [28]. We should note that other simulation tools such as MicroGrid, ifogsim, SimGrid, GridSim, and cloudSim for evaluating replica optimizer algorithms were also developed. This paper used the OptorSim simulator for simulation and performance evaluation. The OptorSim simulator has four configuration files that must be configured before starting a simulation. These files specify the distributed system configuration settings. These files are the configuration file, the job configuration file, the parameters file, and the bandwidth configuration file. Details of these files are shown in Figure 5.

*4.2. Simulation Parameters.* Distributed environment architecture in our proposed algorithm is a hierarchical architecture and consists of regions. Figure 1 shows the network topology in our simulation, which consists of two regions. Nodes in the regions have computational or storage elements. Most of the existing replication strategies assume that the data is read-only. We also assume that the files are read-only. The used configuration parameters are shown in Table 4. We evaluated our method with the random Zipf, random walk, and sequential access patterns. Access patterns determine the order in which files are accessed in the jobs.

*4.3. Implementation.* We used the fuzzy toolbox in MATLAB software to implement the fuzzy inference systems. We added several classes to OptorSim to connect it to MATLAB software. The input parameters of the fuzzy inference systems are determined during the simulation. The properties of used fuzzy inference systems like the inference engine, DE fuzzy method, implication method, and aggregation are specified in Table 5.

Table 2: Three used rules in the first fuzzy inference system (replica placement phase).

| | Rules |
|---|---|
| 1 | If NA is high, CC is high, TI is low, and WL is low, then node_value is very high |
| 2 | If NA is high, CC is high, and TI is average (WL is low), then node_value is high |
| 3 | If NA is high, CC is average, and TI is low (WL is low), then node_value is high |

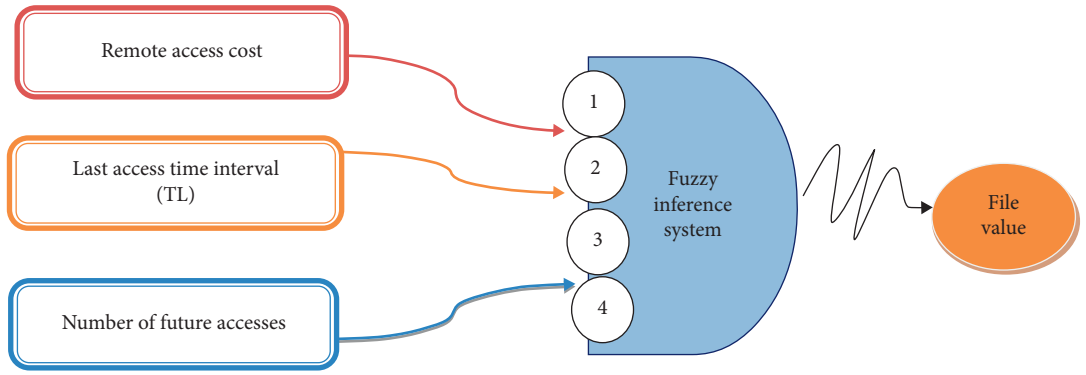


Figure 4: An abstraction of the second used fuzzy inference system.

Table 3: Three used rules in the first fuzzy inference system (replacement part).

| | Rules |
|---|---|
| 1 | If NA is high, TI is low, and RAC is high, then FileValue is very high |
| 2 | If NA is high, TI is average, and RAC is high, then FileValue is high |
| 3 | If NA is low, TI is high, and RAC is low, then FileValue is very low |

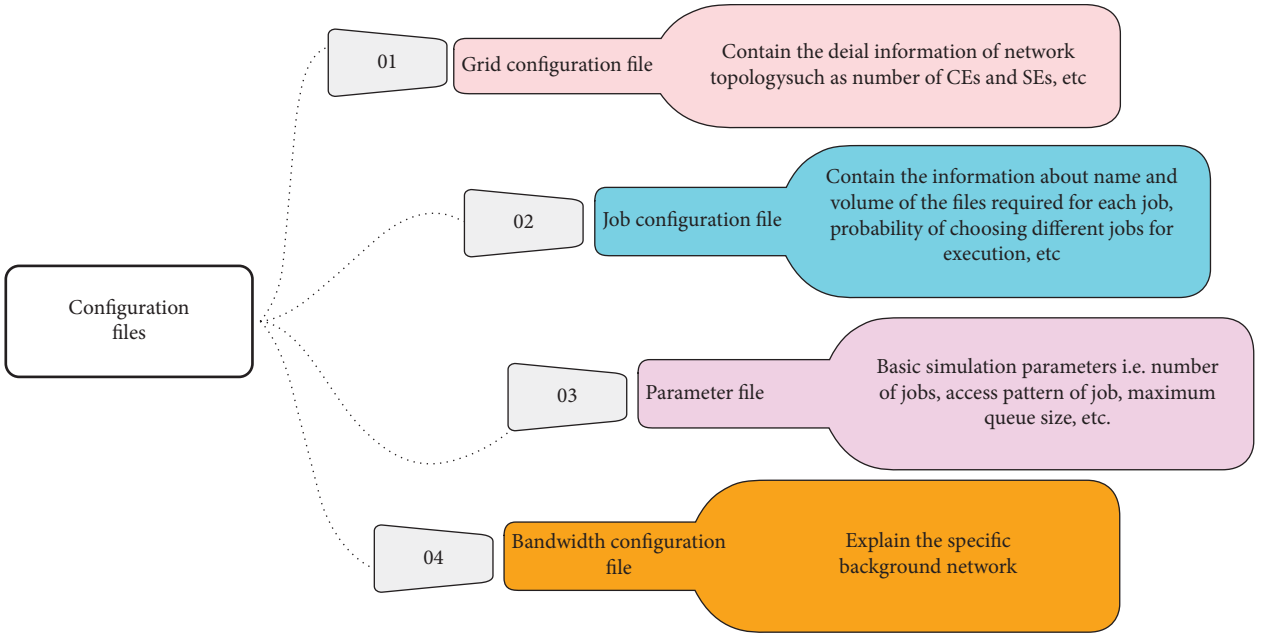


Figure 5: Configuration files of OptorSim simulator.

*4.4. Simulation Results.* In this section, our simulation results are mentioned. The proposed algorithm is compared with the no replication, LRU, FRA, RCP, and modified BHR algorithms.

*4.4.1. The Mean Job Time of All Jobs.* This criterion is defined as the ratio of all the jobs' total execution time to the total number of the jobs. It is considered the most important measure for evaluating an algorithm's

Table 4: The configuration parameters in the simulation.

| Parameter | Value |
|---|---|
| Number of nodes | 33 |
| Number of jobs | 1000 |
| File size (GB) | 1 |
| Failure probability | 0–40% |
| Minimum bandwidth between two nodes (Mbit/s) | 200 |
| Maximum bandwidth between two nodes (Mbit/s) | 1000 |
| Recovery time (ms) | 10000 |
| Access pattern | Random Zipf, random walk, sequential |
| Number of experiments | 20 |

Table 5: The properties of the fuzzy inference system.

| Type system | Mamdani |
|---|---|
| AND method | Min |
| OR method | Max |
| DE fuzzy method | Centroid |
| Implication method | Min |
| Aggregation method | Max |

performance. The lower mean job execution time indicates that the algorithm works better, and the jobs are executed within a shorter time. It is obtained using equation (13), where $t_j(\text{st})$, $t_j(\text{ct})$, and NOJ are the time that job $j$ is submitted, the time that job $j$ is completed, and the total number of the jobs.

$$\text{MJET} = \frac{\sum_{i=1}^{\text{NJ}} t_j(\text{ct}) - t_j(\text{st})}{\text{NOJ}}. \tag{13}$$

We examine the mean job execution time of the proposed replication algorithm in three test cases. In the first test case, the proposed method is evaluated compared to other methods in different access patterns. The second and third test cases evaluate the replication methods based on varying the number of jobs and sizes of files.

*(1) Test Case 1: Replication Strategies and Different Access Patterns.* In the first test, replication algorithms are evaluated with different access patterns. Figure 6 shows that all algorithms have the best results in random access patterns. In these access patterns, a specific set of files is often requested. Hence, most files are available locally because they are already replicated. The test results show that the EFRA has the shortest mean job execution time in all access patterns. The LRU method has the maximum job execution time in all access patterns. It always unreasonably performs data replication. In the absence of enough space, files that have not recently been accessed are candidates for being deleted. It may delete the files with high access frequency or high remote access cost. Indeed, it replicates and deletes the files more unreasonably than other algorithms. In both modified BHR and FRA methods, the LRU method's problems have been reduced because replicas are stored in a high-frequency location in the current region. Hence, access time and mean job execution time are less.

It should be noted that the FRA method further reduces the access time by preventing the deletion of files that have high remote access costs and are recently accessed. As shown

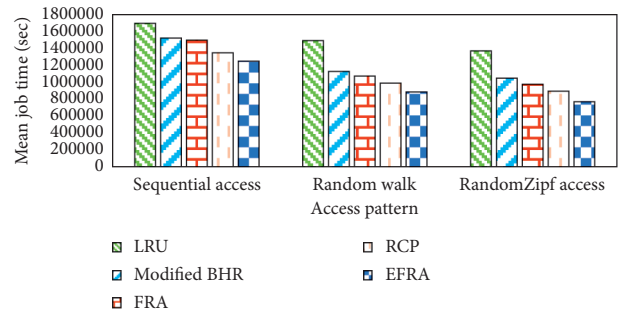


Figure 6: Mean job time of different replication methods with different access patterns.

in Figure 6, the mean job execution time of the presented method is the lowest. Because the EFRA predicts the nodes' future needs based on their previous access, it replicates the next needed files before being requested. Consequently, the files are more probably locally available at the job execution time. Since the required files' unavailability is one of the main reasons that increase the job execution time, this time is considerably lowered in our proposed algorithm. Also, replication in our method (EFRA) is done more reasonably and correctly than the other methods. The right node is selected based on more parameters with a more rigorous logic using Algorithm 2 compared with the other methods such as the MBHR and FRA methods. Other methods only take the number of accesses into account to select the right replication place. Therefore, the files are more probably locally and/or with lower cost available for the jobs during the execution. So, the node where the file is replicated will be more probably the node requiring the file in the future. Hence, the mean job execution time is reduced especially in the random Zipf access pattern. The proposed method also improves the load balancing by considering queue workload and queue delay in the replica section and replica placement phases. Also, using a more suitable replica selection

algorithm, EFRA prevents sending file access requests to the replica nodes with a high queue waiting time. Therefore, remote access is done more quickly, and, consequently, it can have a lower mean job time. On the other hand, the proposed method avoids the delay caused by the failures by considering the failure probability. Regardless of failure probability, other methods select the place of replica for remote access.

*(2) Test Case 2: Replication Strategies and Different Number of Jobs.* Today, the number of users' jobs is increasing in distributed systems. In this test, replication algorithms' mean job time based on varying the number of jobs is evaluated. The results of this test are shown in Figure 7.

The results show that our proposed method (EFRA) has the best results compared to other methods, especially when the number of jobs is high. At a large number of jobs, the storage disk fills up quickly. As a result, an algorithm that prevents valuable files' deletion is more efficient. EFRA in the replacement phase (Algorithm 3) uses a fuzzy inference system to evaluate files with at least one other copy of the file in the current region. This inference system considers comprehensive parameters such as remote access cost, latest file access time, and future file popularity. So it determines valuable files more accurately than other methods. Other replication methods consider only the last file access time or the number of accesses in the past. Hence, they may also delete valuable files. The EFRA replication algorithm in its replacement part (Algorithm 3) further increases accessibility and file locality by preventing the deletion of valuable files compared to other algorithms. Hence, the mean job execution time is less in our proposed method. In a small number of jobs, all algorithms perform similarly and well. When the number of jobs is small, most of the required files are replicated quickly. Hence, the files are locally available in all algorithms.

*(3) Test Case 3: Replication Strategies and the Different Size of Files.* Today the size of required files is increasing. Therefore, in the third test, we evaluated algorithms' performance in the different sizes of files. The results are shown in Figure 8. When the size of the requested files is large, the replica's location, workload, and the replica nodes' communication ability with other nodes are very effective in terms of the mean job time. The EFRA reduces access latency and increases load balancing and availability by placing new replicas (using Algorithm 2) in the nodes with high bandwidth capability, high future file popularity, low failure probability, and workload. Besides, EFRA, by preventing the deletion of files that will be requested soon (using Algorithm 3), increases local data access. Other methods in determining the suitable replica location do not consider suitable parameters. These methods are more likely to access files remotely. Besides, they do not consider the node's failure probability. These methods may also select a node with a high failure probability for replication. Therefore, the nodes' occurrence of failures increases the data access delay and job execution time.

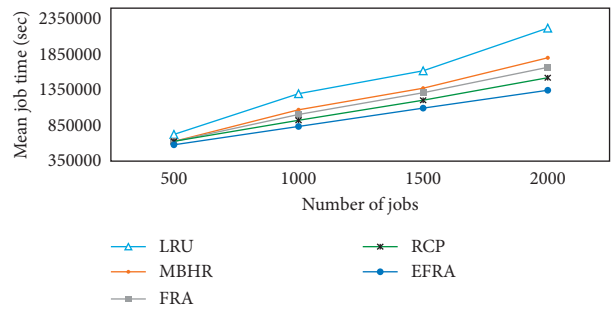


Figure 7: Mean job execution time of replication algorithms based on a varying number of jobs.

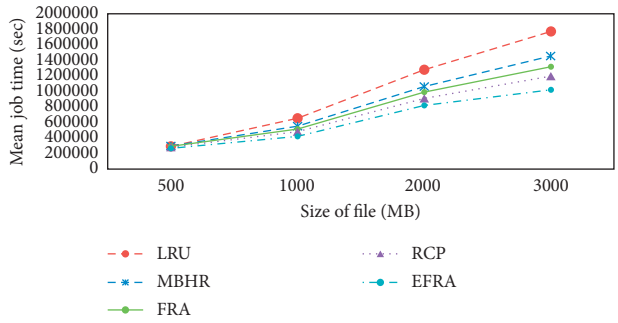


Figure 8: Comparison of replication strategies based on varying sizes of files.

*4.4.2. Total Number of Replications.* Replication consumes different resources such as storage space and bandwidth. Therefore, data replication is an expensive task. As a result, the lower the total number of replications, the lower the system's cost. On the other hand, the low number of replications indicates that the number of local accesses is high. To avoid replication overheads, the number of replication operations should be limited. Therefore, a good algorithm has a low number of replications. Figure 9 shows that the EFRA improves the total number of replications by about 11% and 49% compared to the LRU and RCP. The EFRA has a better total number of replications compared to other methods. Because, in the EFRA, replicas are placed in a suitable location, indeed EFRA in the placement part (Algorithm 2) is more successful in finding a tailed place for replication. It uses more comprehensive parameters compared to other algorithms in selecting the best replication place. Thus, the number of local accesses increases and the number of replications decreases. As shown in Figure 9, the number of replications is zero. We should mention that the noRep (no replication) method does not perform any replication.

*4.4.3. The Percentage of Storage Filled.* The percentage of storage filled metric is the average percentage of storage space used to store files/replicas in the nodes. Figure 10 shows that the noRep (no replication) method has the lowest percentage of storage filled because it does not do any

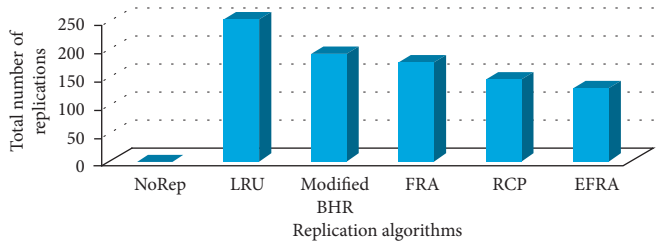


Figure 9: Total number of replications.

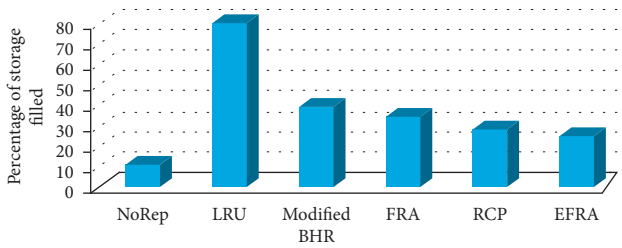


Figure 10: Percentage of storage filled.

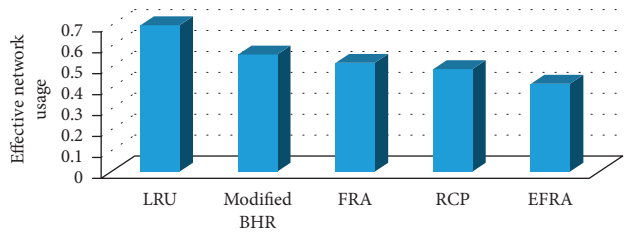


Figure 11: Effective network usage of various replication algorithms.

replication and only saves the original files. After noRep, our proposed method has the best percentage of storage filled. EFRA results show about 13% and 43% improvement compared to the RCP and LRU methods, respectively. In the proposed method, as shown in Figure 9, the number of replications in the EFRA is less than those in the other algorithms. Therefore, as a result, storage consumption is reduced compared to LRU, MBHR, RCP, and FRA.

*4.4.4. The Effective Network Usage (ENU).* Effective network usage (ENU) is a well-known metric for evaluating network resource usage efficiency. It is obtained from equation (14). It is defined as the ratio of files transferred to files requested. A low ENU shows that the used replication algorithm performs better in choosing a suitable place for replication.

$$\text{ENU} = \frac{N_{\text{remote file accesses}} + N_{\text{file replications}}}{N_{\text{remote file accesses}} + N_{\text{local file accesses}}}. \quad (14)$$

In Figure 11, the EFRA is compared with the four replication algorithms in terms of the effective network usage (ENU) with the Zipf access pattern. As shown in Figure 11, the EFRA improves ENU more than the RCP and FRA methods. Because the EFRA reduces the number of remote accesses to the files, it lowers the need for file replication using Algorithm 2. Algorithm 2 places the files in the nodes with a higher probability of future access. Besides, the EFRA prevents deleting popular files by considering the future number of accesses and last access time in the replacement part (Algorithm 3). Other algorithms like RCP only consider the last access time in selecting a low-value file. Indeed, it considers the temporal and geographical locality and has more local accesses. Besides, the most popular files are requested in the zip access pattern. Consequently, in the EFRA, the jobs at the time of execution access the required files locally with more probability. Hence, the number of local accesses increases, and the number of replications reduces, in addition to lowering the ENU.

## 5. Conclusion

Recently, with the advancement of technology and the production of large amounts of data by data-intensive applications, the popularity of distributed systems such as Grid, Cloud, and Fog is increasing. These systems provide infrastructures and platforms for sharing and managing large amounts of data. Delay in accessing data is one of Grid and Cloud's significant problems. Data replication is used as a key method to solve this problem. In this paper, we present a new dynamic fuzzy-based replication algorithm. The proposed algorithm uses the fuzzy decision to select the best place for replication and the best replicas to delete. It considers the parameters of the future number of file accesses, communication capacity in the node, failure probability, storage queue workload, and the latest file access time to select the best replica location. The proposed algorithm reduces the access time and bandwidth consumption by preventing the deletion of popular and valuable files in the absence of storage space. We compare the proposed algorithm with well-known replication algorithms using the OptorSim simulator. Simulation results show that our proposed algorithm can improve performance parameters compared to LRU, Modified BHR, RCP, and FRA methods. Our method can enhance performance by considering more efficient parameters in the replica placement and

replacement phases and using the fuzzy system analysis power. We intend to use more decision parameters such as price and security in future works. In addition, evaluating the proposed method in a real distributed system is another upcoming plan. We also plan to evaluate the proposed method's combination with different popular job scheduling algorithms.

## Data Availability

No data were used to support this study.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## References


[1] M. Beigrezaei, A. Toroghi Haghighat, and S. Leili Mirtaheri, "Minimizing data access latency in data grids by neighborhood-based data replication and job scheduling," *International Journal of Communication Systems*, vol. 33, no. 12, Article ID e4552, 2020.

[2] R.-S. Chang, J.-S. Chang, and S.-Y. Lin, "Job scheduling and data replication on data grids," *Future Generation Computer Systems*, vol. 23, no. 7, pp. 846–860, 2007.

[3] L. Xiong, L. Yang, Y. Tao, J. Xu, and L. Zhao, "Replication strategy for spatiotemporal data based on distributed caching system," *Sensors*, vol. 18, no. 1, p. 222, 2018.

[4] S.-M. Park, J.-H. Kim, Y.-B. Ko, and W.-S. Yoon, "Dynamic data grid replication strategy based on internet hierarchy," in *Proceedings of the International Conference on Grid and Cooperative Computing*, pp. 838–846, Shanhai, China, December 2003.

[5] J.-P. Yang, "Efficient load balancing using active replica management in a storage system," *Mathematical Problems in Engineering*, vol. 2016, Article ID 4751829, 9 pages, 2016.

[6] M. Tang, B.-S. Lee, X. Tang, and C.-K. Yeo, "The impact of data replication on job scheduling performance in the data grid," *Future Generation Computer Systems*, vol. 22, no. 3, pp. 254–268, 2006.

[7] J. M. Pérez, F. García-Carballeira, J. Carretero, A. Calderón, and J. Fernández, "Branch replication scheme: a new model for data replication in large scale data grids," *Future Generation Computer Systems*, vol. 26, no. 1, pp. 12–20, 2010.

[8] K. S. Maabreh, "An enhanced university registration model using distributed database schema," *KSII Transactions on Internet and Information Systems*, vol. 13, no. 7, 2019.

[9] M. Tu, P. Li, L. Xiao, I.-L. Yen, and F. B. Bastani, "Replica placement algorithms for mobile transaction systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 7, pp. 954–970, 2006.

[10] W. Hashem, H. Nashaat, and R. Rizk, "Honey bee based load balancing in cloud computing," *KSII Transactions on Internet and Information Systems*, vol. 11, no. 12, 2017.

[11] X. Fu, W. Liu, Y. Cang, X. Gong, and S. Deng, "Optimized data replication for small files in cloud storage systems," *Mathematical Problems in Engineering*, vol. 2016, Article ID 4837894, 8 pages, 2016.

[12] K. Ranganathan, A. Iamnitchi, and I. Foster, "Improving data availability through dynamic model-driven replication in large peer-to-peer communities," in *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*, p. 376, Berlin, Germany, May 2002.

[13] C. Li, M. Song, M. Zhang, and Y. Luo, "Effective replica management for improving reliability and availability in edge-cloud computing environment," *Journal of Parallel and Distributed Computing*, vol. 143, pp. 107–128, 2020.

[14] P. Vashisht, V. Kumar, R. Kumar, and A. Sharma, "Optimizing replica creation using agents in data grids," in *Proceedings of the 2019 Amity International Conference on Artificial Intelligence (AICAI)*, pp. 542–547, Dubai, UAE, February 2019.

[15] Y. Liu and W. Wei, "A replication-based mechanism for fault tolerance in mapreduce framework," *Mathematical Problems in Engineering*, vol. 2015, Article ID 408921, 7 pages, 2015.

[16] M. Beigrezaei, A. T. Haghighat, and N. Hajizadeh Bastani, "Increasing performance in Data grid by a new replica replacement algorithm," *International Journal of Advanced Computer Research*, vol. 11, no. 2, pp. 1–10, 2020.

[17] K. Ranganathan and I. Foster, "Identifying dynamic replication strategies for a high-performance data grid," in *Proceedings of the International Workshop on Grid Computing*, pp. 75–86, Denver, CO, USA, November 2001.

[18] K. Ranganathan and I. Foster, "Simulation studies of computation and data scheduling algorithms for data grids," *Journal of Grid Computing*, vol. 1, no. 1, pp. 53–62, 2003.

[19] K. Sashi and A. S. Thanamani, "Dynamic replication in a data grid using a modified BHR region based algorithm," *Future Generation Computer Systems*, vol. 27, no. 2, pp. 202–210, 2011.

[20] K. Rajaretnam, M. Rajkumar, and R. Venkatesan, "Rplb: a replica placement algorithm in data grid with load balancing," *International Arab Journal of Information Technology*, vol. 13, no. 6, 2016.

[21] M. Meddeber and B. Yagoubi, "Dependent tasks assignment and data consistency management for grid computing," *Multiagent Grid System*, vol. 15, no. 2, pp. 179–196, 2019.

[22] S. Bakhshad, R. M. Noor, T. Saba et al., "A dynamic replication aware load balanced scheduling for data grids in distributed environments of internet of things," *Ad Hoc & Sensor Wireless Networks*, vol. 40, 2018.

[23] M. Beigrezaei, A. T. Haghighat, and H. R. Kanan, "A new fuzzy based dynamic data replication algorithm in data grids," in *Proceedings of the 2013 13th Iranian Conference on Fuzzy Systems (IFSC)*, pp. 1–5, Qazvin, Iran, August 2013.

[24] S. N. John and T. T. Mirnalinee, "A novel dynamic data replication strategy to improve access efficiency of cloud storage," *Information Systems and e-Business Management*, vol. 18, pp. 405–426, 2020.

[25] N. K. Gill and S. Singh, "A dynamic, cost-aware, optimized data replication strategy for heterogeneous cloud data centers," *Future Generation Computer Systems*, vol. 65, pp. 10–32, 2016.

[26] S. Sun, W. Yao, and X. Li, "DARS: a dynamic adaptive replica strategy under high load Cloud-P2P," *Future Generation Computer Systems*, vol. 78, pp. 31–40, 2018.

[27] M. Beigrezaei, A. T. Haghighat, M. R. Meybodi, and M. Runiassy, "PPRA: a new pre-fetching and prediction based replication algorithm in data grid," in *Proceedings of the 2016 6th International Conference on Computer and Knowledge Engineering (ICCKE)*, pp. 257–262, Mashhad, Iran, October 2016.

[28] I. Foster, "The grid: a new infrastructure for 21st century science," *Grid Computing: Making the Global Infrastructure a Reality*, vol. 55, pp. 51–63, 2003.