AGENT BASED OPTIMIZED RÉPLICA MANAGEMENT IN DATA GRIDS

Priyanka Vashisht[*], Vijay Kumar[**]
[*]Department of Computer Science and Engineering, NorthCap University, Gurugram, India.
[**]Amity Institute of Applied Sciences, Amity University, Noida, India.

ABSTRACT
Sharing of data in a distributed environment such as grid leads to various design issues such. One of the effective measures for accessing the distributed data is réplication. This paper addresses réplica management issues such as availability, placement and consistency. Proper management of réplicas in a distributed grid environment improves the accessibility of updated réplica. A strategy namely, Réplica Creation, Placement and Consistency of data (RCPC) is proposed. RCPC dynamically creates an optimal number of réplicas and places them on nodes having minimum placement cost. An agent-based adaptive threshold approach is used efficiently for maintaining consistency among the réplicated data. RCPC is simulated using *Optorsim* simulator and the results are evaluated in terms of job execution time, the effectiveness of network usage, hit ratio, transfer rate, storage utilization and computing usage.

KEYWORDS: Data Réplication, Réplica Creation, Réplica Placement, Réplica Consistency

MSC: 68M20

RESUMEN
Compartirdata en un ambiente distribuido como mallas lleva a varios aspectos de diseño. Una de las medidas efectivas para acceder a la data distribuida es replicar. Este paper trata del manejo de aspectos de la réplicacomo la disponibilidad, ubicación y consistencia. Un apropiado manejo de las réplicas en un ambiente distribuido como mallas mejora la accesibilidad de la réplicaactualizada. Una estrategiallamada, Réplica Creation, Placement and Consistency of data (RCPC) es propuesta. RCPC crea dinámicamente un numero optimal deréplicas y las pone sobre nodosque tienen un mínimodel costo de ubicación. Un enfoque basado en agentes de umbral adaptativoes usado eficientemente para mantener la consistencia entre la data réplicada. RCPC es simulada usando el simulador *Optorsim* simulator y los resultados son evaluados en términos del tiempo de ejecuciónde la tarea, la efectividad del uso de la red, la tasa de "hit", la tasa de transferencia, lautilizacióndel almacenaje y el aprovechamiento de computación.

KEYWORDS Replicación de la Data, Creación de la Réplica, Ubicación de la Réplica, Consistencia de la Réplica

1. INTRODUCTION

Grids are geographically distributed computing structure that comprises a large number of computational and storage resources. Grids offer transparent access to distributed data files which ensures easy access to data globally. A data grid deals with the applications that are involved in the exploration and manipulation of extensive data sets [1][4][7]. Data grids comprise of a complete dynamic life cycle for the service organization, distribution, administration and decomposition [18]. The massive data sets and their computation generate new issues regarding data access, handling and distribution [3]. One of the major concerns of the data grid is data management, which helps in optimizing data access in a distributed grid environment. The optimization of data can be attained by duplicating the data files. Réplication of data files at geographically distributed nodes is one of the main features affecting the performance of data grids [23]. Primary aspects of dynamic réplication process are to create réplica then place them on a suitable location and ensure the consistency among the files.

The purpose of creating a réplica is to enhance the performance of the system. Réplication is advantageous if the ratio of the number of read-only query to the number of queries for updating the réplicas is greater than or equal to one, otherwise, réplication may cause problems [28]. When creating a réplica, a choice has to be made on the optimal number of réplica, location of the réplica and their relevant consistency. The choice should be based on certain criteria such as a number of requests made for a file, storage capacity, processing capacity of a node, etc. After creating a réplica, an important issue is to place the réplica at a suitable location in order to get the optimized results. The basic idea is the réplica of a file should be placed in the near locality of the process using them, which will help in reducing the access time of the data file [37]. There exist numerous réplica creation and placement policies which are advantageous in attaining significant results.

Despite the advantages of data réplication technique, the main challenging issue is consistency maintenance of geographically distributed data. With increasing dimensions of data grids, if réplica

Email of correspondance author: Priyanka.vashisht@gmail.com

consistency policies of data files are inadequate, some réplica updates may not be distributed timely and thus inconsistency may occur between the primary copies of a file and its réplicas. This can have an evident consequence on the results of users and the problem will become worse as the magnitude of data grid grows. Therefore, the key issue in data grids is how the files and its réplicas can be used effectively and efficiently to reduce inconsistency.
Selection of an appropriate number of réplicas, placing them on suitable nodes and maintaining réplica consistency is crucial for réplica management approaches in data grid environment [43]. To overcome the issues, a strategy, namely RCPC, has been proposed which dynamically creates and places the réplicas on optimal nodes by maintaining the consistency adaptively while considering all key constraints such as access frequency, placement cost, and réplication cost, etc.

2. RELATED WORK

This section includes the study of réplica management in data grids. From the study, it is evident that the key challenges that are to be dealt with in a grid environment are related to réplica creation, placement, and selection and consistency maintenance. Ranganathan et al. [17] proposeda decentralized peer-to-peer network for creating réplicas automatically. A probabilistic measure is used for ensuring réplica availability. The decision for réplicating a file is taken by acquiringpartial information from the system, which at times leads to unnecessary réplication.
Li et al. [20] suggested a cost-effective mechanism, namely, PRCR, which proactively checks the réplicas of a file to ensure the data reliability. This approach assures the availability of a minimum number of réplicas required for the execution of the job. Additionally, the réplication cost is being controlled by managing a large number of data files on the cloud. The cloud environment helps in providing the storage space at a minimum cost. The major drawback PRCR is it does not consider the appropriate location for placing the réplicas.
Tang et al. [38] proposed two algorithms for réplicating data: Aggregate Bottom Up (ABU) and Simple Bottom Up (SBU) for multi-tier data grids. The objective of both strategies is to position the réplica in the proximity of the client. In SBU a threshold value is calculated without considering access records of arrival time whereas in ABU historical records of data access plays a vital role. The results depict that these strategies decrease the average response time of data access as compared to other static réplication algorithm.
Park et al. [29] introduced an algorithm, namely, Bandwidth Hierarchy Réplication (BHR), which take advantage of "network-level locality," for placing the crucial file on a node having a large amount of bandwidth connecting the job execution site and node with the primary réplica. The strategic placement of file in the vicinity of the user helps in reducing the time for accessing the file. BHR algorithm is modified by Horri et al. [15] and Sashi and Thanamani [35], where numbers of réplicas are increased at each level of network hierarchy, resulting in better performance. Further, P Vashisht et al. [39] customized BHR by considering the availability of réplicas and efficient scheduling strategy consequently leads to better execution of jobs. Sonali Warhade et al. [41] proposed a similar algorithm to BHR but uses graph topology to define its grid structure. The réplication process takes place when the file is not available on the nodes whose likelihood is more to be accessed in the near future. The results depict that it reduces access cost and time as compared to BHR.
F B Charrada et al [11] proposed a réplication strategy which enhances the response time and availability of file in a dynamic grid. Moreover, it selects the best candidate files for réplication and the best node for placement based on the number of requests. The placement and selection are exploited in order to acquire the load balancing in the global grid. S Nasser et al. proposed an agent based algorithm for replica placement and selection. Selection of appropriate candidate site is based on certain factors like CPU load, CPU rating, baud rate, storage capacity and demand of the file. Placement of a réplica at a suitable node reduces the network traffic, access cost, and aggregated response time for the applications. [33]. Souravlas et al. [36] introduced an algorithm based on the estimation of file requirement on the node. It uses a binary tree structure to keep track of increasing fluctuating demand of user for a file. The files are categorized as reading file and write files. The algorithm calculates the number of files which need to be accessed more frequently on the basis of temporal locality principle. The simulation results suggested that the proposed algorithm provide better data access of files in terms of the average job execution time and hit ratio, compared to other state-of-the-art strategies.
N Mansouri [22] proposed threshold-based data réplication (TDDR) along with parallel job scheduling (PJS). Storage capacity and the request arrival rate is used to calculate threshold value by TDDR which helps in deciding which file to réplicate. PJS uses file location, network characteristics, disk speed, number of waiting jobs to analyse the hierarchical structure for the scheduling. Simulation results show

that TDDR outperforms eleven existing algorithms in terms of mean job time, network usage, number of inter-communication, resource usage and number of réplications.

N Mansouri [24] proposed two algorithms namely combined scheduling strategy (CSS) and modified dynamic hierarchical réplication algorithm (MDHRA). CSS considered computational capacity, location of the site containing file and waiting time of job for reducing search time of a computational site. The MDHRA is a modification of dynamic hierarchical réplication (DHR) algorithm which reduces the access time of a file. Results show that access latency, response time and bandwidth consumption decreases and overall system performance has increased. Rashedur et al. used the multi-objective approach in the p-center and p-median models, to address the issue of réplica placement. Current status of the network and the pattern of data requests are two crucial parameters which deal with data placement issue. To locate p réplica placement nodes, p-median and p-center modelsare used. The p-center model minimizes the response time between réplica server and user node additionally, the p-median model helps in reducing the total response time between the réplication node and requesting node[32], [26]. M Shorfuzzamanet. al. proposed a threshold based algorithm namely, Adaptive Popularity Based Réplica Placement (APBRP) which positions the replica based on "file popularity". The threshold value is formulated dynamically based on the arrival of the request rate which helps in determining the node for réplica placement. Réplicas are positioned close to the clients so as to reduce the data access time whereas using storage and network resources reasonably. [25].

N N Dang et al. [27] introduced placement strategies in data grids that classify the data based on their characteristics. Data is arranged into various biological classes to which it belongs. The classification and number of accesses is the deciding factor for réplica creation and job scheduling. Réplica placement is based on the distance and category to which the file belongs. The simulation results show that the proposed strategy reduces the file transfer cost by placing a réplica close to the user. Lin et al. [16] tackle the problem of database réplica placement in a dynamic grid environment with locality assurance. In this approach when the request is made it has to specify its requirements in terms of workload and the distance within which the réplicas can be located. For balancing the workload among the nodes strategic placement of réplicas is done along with the guarantee of locality service required by each request. Authors proposed two algorithms, namely, MinMaxLoad and FindR. The MinMaxLoad algorithm is used for placing réplicas at proper server locations so as to balance the workload on all servers. However, FindR algorithm is proposed which determine an optimal number of réplicas and also offers service at the local level.

Al-Mistarihi and Young [2] suggested selection and placement policy for réplicas in the data grid. The proposed system, "Réplica Management in Grid" (RmGrid), consists of global and local optimizers. The global optimizer collects information from the local optimizer and information provider such as réplica requests demand, network status, etc. In order to get an optimal number of réplicas and suitable location for the file global optimizer prompts for placement function. The decision for replica decision is based on location cost which is dependent on three constraints, namely transfer time, site power and allocation of réplicas among nodes. Best replica is selected by the local optimizer. There proposed system resulted in a reduction of storage cost, job turnaround time and bandwidth usage.

C. Yang et al. [42] have suggested a model namely, One-way Réplica Consistency (ORCS). In ORCS nodes are categorized as supernode (SN), the master node (MN), and a child node (CN). SN can store primary files only, which can be réplicated to MN. The CN can store réplica if there is enough storage space and high access frequency of request file. Consistency of the files is maintained by automatically propagating updates from highest level SN to lowest level CN. The simulations demonstrated significant stability between the consistency of the files and performance of the system.

Perez et al. [30] proposed Branch Réplication Scheme (BRS) which enhances the performance, scalability and fault tolerance of the system using a centralized hierarchical approach with the parallel transmission of fragmented data. It is accomplished by generating sub réplicas and using parallel I/O techniques while maintaining consistency. The experimental results represent that the proposed BRS scheme reduces access time of read and write operations for a file. Aggressive Update Propagation (AUP) [31] uses a push approach where updates are transmitted to all copies by the nodes containing the file. The ratio of reading and update file in AUP is relatively high than pull based protocols. Aggressive protocol continuously needs to retain the consistency of réplicas. Spigot [12] is another example of push based, fragment level réplica consistency.

R Grace et al. suggested that in the course of data réplication, identical copies of the data are formed then positioned at suitable nodes and selection can be done while executing the job [14]. At any time, write operation is accomplished on réplicas of file; it must be propagated to all existing réplicas to ensure consistency of data on the data grid. R. Chang and J.S. Chang [10] offered Adaptable Réplica Consistency Service (ARCS) which is a weighted load balancing and consistency service. Weights rely on access frequency of the file, which helps in outperforming it from Lazy services in terms of average

file access delay and total job time. In [9] an enhancement of ARCS is provided, namely, RCDM_NBC, which uses Naive Bayesian Classifier for consistency maintenance. This approach is more flexible and flexible to other states of art.
H. E. Chihoub [13] proposed a cost efficient consistency approach called Harmony. In this approach, an estimated predictable cost and probabilistic assessment of consistency is provided. This helps in enhancing the system by maintaining desired consistency and reducing the economic cost for the user. T. Kraskaet al. [19] presented a consistency management model that provides consistency by rationalizing data. Data is separated into three categories i.e. A, B, C. Data with the highest level of consistency is reserved for category A. Category B select the level of consistency dynamically based on a calculated threshold value. The threshold value is formulated by considering the probability of the monetary cost of pending operations in the update queue and the number of update conflicts which happens during the transaction. Data having weak consistency is enclosed in Category C.

3. RÉPLICA CREATION, PLACEMENT AND CONSISTENCY (RCPC) STRATEGY

The proposed RCPC strategy provides an efficient management technique for the réplicas in a data grid environment. RCPC generates and places the réplicas on optimal nodes while maintaining the consistency adaptively. The RCPC strategy methodically organizes the available nodes of the system into discrete regions. Grid nodes are selected for optimal placement of réplicas. In addition, RCPC maintains the consistency of the réplica placed at the distributed location. RCPC allocate file to the request in accordance with their access frequency. RCPC is an extension of our RCP [40] where optimal numbers of réplicas are created in the distributed data grid environment. The main contributions of the current study are as follows:

i. In our previous work [40] réplica creation strategy based on the popularity of the file. The popularity of certain file can be derived through the frequency by which a file has been accessed in the past. To save storage space on a node, numbers of réplica are restricted by considering réplication cost and access frequency of file.
ii. Réplica placement issue has been addressed in RCPC, taking into account parameters such as réplication cost, the processing capacity of the node, etc., which helps in enhancing the overall system performance.
iii. Reduction in outdated copies of file has been attained by achieving consistency due to hybrid update approach of files after every modification.
iv. Consistency in RCPC can be achieved using a threshold value, which is calculated after every predefined time interval. This process reduces the overhead due to an immediate update of files after every modification.
v. Investigating various trade-off in terms of mean job execution time, number of réplicas, effective network usage, percentage of accessing updated data, number of update transactions.

Various components and terminologies of RCPC strategy as shown in Figure. 1 is:

- *Grid Node: It is the l*owest level entity in the proposed environment which is composed of three components: (i) *computing element* (CE) serves as a computational resource for the execution of the jobs by using data which is kept on storage resources, (ii) *storage elements* (SE) offers storage resources for storing data for fulfilling the requirement of the submitted jobs, and (iii) *réplica manager* (RM) manages communication between various components of the system. The nodes are further classified as the *super node*, *head node*, and *réplica node.*
- Super Node: It provides an interface for job submission and also facilitates scheduling and monitoring of the jobs.
- *Resource Broker (RB):* The Resource Broker (RB) is an integral part of the Super Node, which optimizes the mapping process of resource and job by obtaining information of all possible resources which can satisfy the requirements of the job.
- *Super node Analyzer (SA):* The request made by the user is analyzed and mapped to the node on various regions by considering the access frequency of the file. SA is the component of RB.
- *Job Scheduler (JS):* It is responsible for generating an optimal schedule for requests which is to be executed.
- *Region Allocator (RA)*: Allocation of the region to the request are done by RA based on the access frequency of the file.
- Universal Information Collector (UIC): Information probe (IP) provides information for every region to UIC agent and stores it on the global database (GDB).

- *Global Database (GDB):* It contains the information regarding the physical and logical file names PFN and LFN respectively and their mapping in different regions for future request allocation.
- *Access Information Database (AID):* It stores information related to each request.
- *Head Node:* At the local level, the head node is responsible for maintaining the information regarding the files, their mapping, request and their allocations etc. within the region. RM component is the primary component of the head node and the super node. The agents of head node assist in collecting information of the node on Information Probe. Réplica node contains only the SE.
- *Information Probe (IP):* Resource utilization of the system is monitored by IP. It monitors parameters such as storage space, number and names of files, the processing capacity of nodes and the resultant information is stored in the local database (LDB).
- *Local Database (LDB):* Current status of all the available nodes in the region is stored on LDB.
- *Local Node Controller (LNC):* LNC is an agent who analyzes the request scheduled to the head node by super node and maps it to a node locally.
- *Agent:* Software entity which manages and manipulates information intelligently.
- *Adaptive Consistency Component (ACA):* Threshold value is calculated for consistency management which deals with access frequency of the file between two intervals.

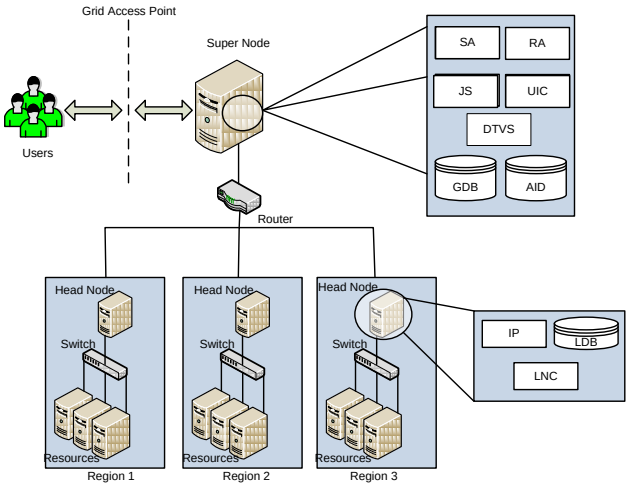

Figure 1:The working of RCPC strategy

While initializing the simulator, the super files are distributed randomly on the nodes of various regions. An agent on GDB is responsible for registering the distributed files and their location with them. The jobs are submitted by a user and are picked by the Resource Broker which is responsible for scheduling the jobs to the node which contains the desired file. The files can be accessed locally or remotely by various jobs using their logical filename (LFN). The logical filename is the name given to a file for the reference of the client irrespective of its physical location. Physical filename (PFN) represents a specific file with respect to its physical location. The LFN is mapped to its respective PFN by Réplica Location Service (RLS). There can exist multiple physical locations of a logical file. To acquire the exact physical location of a file, the Réplica Manager (RM) is consulted by the CE. The RM, in turn, consults the Réplica Catalogue (RC) for acquiring the information regarding the physical location of the file. In general, the selection of the best réplica of a file relies on the minimum transfer time with some network constraints such as bandwidth, latency etc. The request is forwarded to the appropriate region from a higher level to lower level, where the nodes are identified at the local level, which contains a file for execution. Due to the dynamic nature of the grid, sometimes the files or nodes are busy or not available.
The request is transferred to one of the neighbouring nodes which are holding the required replica and then a total number of réplica present in the region at that instance of time is checked. If the available réplica in the region is less than the number which was calculated initially, more réplicas are produced. Once the decision to create a réplica has been made, determining the place for data réplica is the prime concern. Optimal placement of data is achieved by acquiring the current status of the system. The réplica placement algorithm selects the node with lower réplication cost. On occasion, it is not reasonable to always réplicate a file on node due to insufficient available free resources. When the placement of

réplicas is not feasible due to lack of resources some replacement strategy is used. In this paper, some old files are substituted with new files using Least Recently Used (LRU) algorithm [39].
ACA helps in maintaining the consistency of the réplicas but at the cost of slower job execution. It is applied if the threshold value has changed for two consecutive intervals. For implementation and subsequent evaluation of RCPC, mathematical equations for réplica creation, placement and threshold value are formulated and discussed in the forthcoming subsections.

3.1. Modelling of RCPC Strategy

While developing a réplica management algorithm, significant decisions are to be made such as the number of réplicas needed, the node where the réplicas will be placed, how the consistency of the réplicas is determined in order to accomplish the aims of data réplication. To address the above issues a strategy namely, Réplica Creation, Placement and Consistency (RCPC), has been proposed, which comprises of following steps:

i. Data grid regions (system model) are created where réplicas should be placed.
ii. Creation of réplicas.
iii. Determine a node in the data grid for placing the réplica. This is attained by calculating the placement cost of réplica and processing speed of the node.
iv. Finally, the consistency of réplicas is achieved by periodically calculating the threshold value.

3.1.1. System Model

The data grid *D* considered here, is a set comprised of discrete nodes $D= \{n_1, n_2, \ldots, n_k\}$having data user $U = \{ u_1, u_2, \ldots, u_m \}$ connected by some communication link. Initially data files $f = \{ f_1, f_2, \ldots, f_n\}$ can be produced at any node and réplica of the file $R = \{r_1, r_2, \ldots , r_n\}$ can also be stored at any node $n_i \in D$. f_{pi} ,$i = 1, 2, \ldots, n$, is a primary copy of each file. Each node has the following characteristics:
$n_i \in D :< P_{cap}, RC_i, S_i >$ where, $P_{cap}$is the processing capacity of the node,$RC_i$is the name of the réplica catalog to which node n_i is connected and S_i is the total storage capacity of the node. A number of grid nodes in a group form one region as shown in Figure 1. The data nodes are clustered together based on the physical proximity of nodes to form a region. In all, there are *N* regions whose nodes are partitioned into several non-overlapping regions, $Reg = \{Reg_1, Reg_2, \ldots, Reg_N\}$such that, $Reg_1 \cap Reg_2 \cap \ldots \cap Reg_N = \emptyset$. The nodes are connected to each other directly or indirectly through some communication link. Nodes within a region have low latency and maximum bandwidth. One node in a region is called head node comprising of réplica catalog $RC = \{RC_1, RC_2, \ldots, RC_N \}$, which provides the mapping of the file and its physical location locally. The local réplica catalog is connected to global level réplica catalog such that $GRC = \{ RC_1 \cup RC_2 \cup \ldots \cup RC_N\}$which offers a global view of data grid at the super node.

3.1.2. Réplica Creation

Data réplication comprises decisions like how many copies of file are required and when to create a réplica. In our previous work [40] the RCP strategy creates réplicas if the number of réplicas is less than required réplicas in the region. The number of required réplica in a region is created based on the access frequency of the file. More access frequency of a file determines its popularity in a region. Popularity is the deciding factor for creating réplicas of a file in the region. Access frequencies can be determined from the log histories of the file. Réplicas are created in RCP strategy in two phases:

a. A number of réplicas needed for the entire data grid.
b. Establish a number of réplicas required for the region in the data grid.

The average access frequencies of the files are determined within the region and the entire grid environment. The average access frequencies $Freq(f)$ of the files are determined in our previous work [40]:

$$Freq(f) = \frac{\sum_{i=1}^{n} Freq(f_i)}{F_{num}}, \quad (3.1)$$

where, F_{num} is sum of all files that have been requested and $Freq(f_i)$ is number of times a file is requested by user . Réplica creation increases system performance in terms of reliability and availability but proper placement is also very crucial parameter to increase the overall efficiency of the system. In the next section, the réplica placement algorithm has been discussed.

3.1.3. Réplica Placement

Réplicas should be placed at an appropriate location so as to reduce the access cost of replica effectively and enhance the overall performance of the system. The precondition for placing the replica is to calculate placement cost (P_{cost_i}) of réplica (r_i), and is calculated as:

$$P_{cost_i} = \frac{\alpha_{f_i} \times S_f}{P_{cap_i}} \times Rep_{cost_i}, \quad (3.2)$$

where, α_{f_i} is a number of times the request for a replica ($r_i \in Reg_i$) is made by the user, S_f is the size of the file, P_{cap_i} is the processing capacity of the node (n_i) containing réplica ($r_i \in f_i$) and $\text{Rep}_{\text{cost}_i}$ is the replication cost. Further, replication cost is governed mainly by two factors i.e. *communication cost* and the *accessing cost of storage media*. The communication cost(C_{cost}) is determined by:

$$C_{cost} = \sum_{i=1}^{\delta} \sum_{j=1}^{k} C_{ij} + P_{network}, \quad (3.3)$$

where, C_{ij} is communication cost between two node n_i to n_j such that $n_i \neq n_j$ but are connected to each other via some communication link in the network, δ is number of nodes holding the replica of file and k is number of nodes connecting link between n_i and n_j and $P_{network}$ is the propagation delay of the network. The access cost of storage media $C_{storage}$ iscalculated as:

$$C_{storage} = \frac{S_f}{D_{Rate}} + L_{storage}, \quad (3.4)$$

where, D_{Rate} is thetransfer rate of storage media and $L_{storage}$ is the storage media latency. Storage media latency occurs due to reading and writing to and from different blocks of memory. Consequently, the comprehensive replication cost (Rep_{cost_i}) of a file (f_i) can be formulated as storage access cost ($C_{storage}$) of a file and communication cost (C_{cost}) during the transfer of the file from one node to another. Hence, the replication cost (Rep_{cost_i}) is determined as:

$$Rep_{cost_i} = \beta\, C_{cost} + \alpha\, C_{storage}, \quad (3.5)$$

where, α, β are the relative weight such that, $0 \leq \alpha, \beta \leq 1$, and $\alpha + \beta = 1$. The value of weights depending on the access count of the file f_i. Réplicas are stored on the nodes having minimum placement cost and availability of sufficient storage capacity. After calculating placement cost the storage capacity of the node is also investigated. If available storage space ($Avail_{str}$) of the node is less than the size of the file (S_f), then replacement algorithm namely, Least Recently Used (LRU) algorithm is used [39]. LRU strategy compares the size of the file to be replicated with the available space on the selected node if enough space is not available then the older file is removed from the SE. However, if the deleted file is less in size than the required space, the second oldest file is deleted and so on.

3.1.4. Adaptive Consistency

According to G. Belalem, consistency is defined as the relative connection between the file and their distributed replica with some degree of similarity among them [5]. In RCPC, when a node receives write request, the access count (*enu*) will record a request for a file. Recording the write request frequency aims to initiate the consistency mechanism. The proposed consistency approach is determined by the access counts of the file (f_i)for each time interval (τ). For every time interval, a threshold value is determined as:

$$Threshold\ (\tau) = Freq(f_i)\ (\tau) + Freq(f_i)(\tau - 1), \quad (3.6)$$

where, $\text{Freq}(f_i)(\tau) = \frac{\pi_{f_i}}{\sum_{n=1}^{L\tau} \sigma f_i}$, (3.7)

The number of calls to réplica (r_i) is determined by π_{f_i} in last time interval and $\sum_{n=0}^{L\tau} \sigma f_i$ is a number of calls to réplicar_i in time interval τ.

$$Freq(f_i)(\tau) = \begin{cases} 0, & for\ the\ 1st\ interval\ of\ access\ history \\ Freq(f_i)(\tau) - Freq(f_i)(\tau - 1), & otherwise \end{cases} \quad (3.8)$$

If the previous threshold value is equal to the preset threshold value i.e. no written request has been generated and hence no update propagation is initiated, hereafter saving network resources. In the next section, pseudo-code for Réplica Creation, Placement and Consistency (RCPC) are given.

3.2. Algorithms for RCPC

The pseudo code for RCPC strategy is shown in Algorithm 1, is composed of three phases namely; *replica creation, replica placement* and *consistency maintenance*. As discussed earlier RCPC is applied to schedule the request to a region and at the second level the desired file is allocated to the request made by the user.

Algorithm 1 RCPC

1: **procedure** RCPC
2: *initialization*
3: **while** not terminated condition **do**
4: *réplica creation and placement*
5: *consistency maintenance*
6: **end while**
7: **return** consistent result
8: **end procedure**

An efficient scheduling of a request in RCPC strategy is determined by selecting the most appropriate region and node respectively. An appropriate region is the one which is close to the user and holds the most of the requested files which considerably reduces the transfer time. To resolute which region contains most of requested file a record is being kept by a counter (*enu*) for read request as shown in Algorithm 2.

Algorithm 2 Initialization

1: **procedure** *initialization* (Request for file)
2: *enu* ← 0
3: **If** read request received **then**
4: [*enu*] ← [*enu*] + 1
5: **end if**
6: **If** r_i exists on scheduled $n_i$of Reg_i**then**
7: process request;
8: **else**
9: reschedule request to nearest réplica, **goto** 6
10: *réplica creation* and *placement*
11: **endif**
12: **end procedure**

The given pseudo code can be easily modified for the write request to keep track of access frequency of a file.

1: **procedure** replica creation
2: **input:** node id (n_{ij}), primary file(f_{pi}) and region (N)
3: Evaluate $Num_{Reg_i} = \left\lceil \frac{Num_{f_i}}{N} \times \frac{Freq_{Reg_i}(f_i)}{Freq_{Reg_i}(f)} \right\rceil \forall Reg_i \in Reg$
4:**if** $Num_{Reg_i} = \sum r_i$ **then**
5: do nothing
6: **else**
7: **call** *réplica placement*
8: **end if**
9:**for** $(N_{reg} = 1; N_{reg} \leq N ; N_{reg}$ **++)**
10:**for**$\left(n_{ij} = 1; n_{ij} \leq Num_{Reg_i}; n_{ij} + +\right)$
11: Evaluate$P_{cost_i=} \frac{\alpha_{f_i} \times S_f}{P_{cap_i}} \times Rep_{cost_i}$
12: arrange list with minimum placement cost
13: **if**$Avail_{str} \geq S_f$ then
14:**for** each selected n_{ij}
15:n_{ij}← **GridFTP**(f_{pi})
16:GRC_{ij}← **updateLocation**(f_{pi},n_{ij})
17:RC_{ij}← **updateLocation**(f_{pi}, n_{ij}))
18:**end for**
19:**else if**$Avail_{str} < S_f$ then
20: delete another réplica using LRU
21: **end if**
22:**end if**
23:**end for**
24:**end for**
25:**end procedure**

In the *replica creation and placement* phase as shown in Algorithm 3, the new réplicas are produced and every region provides some nodes for the replica placement. As discussed in the previous section the

appropriate location is based on placement cost.N_{reg} represents number of regions in data grid,$Num_{reg_i}$is a total number of replicas in a region and n_{ij} represent i^{th} node and j^{th} region. The function *GridFTP ()* copies updated primary copy to all the replicas of a file. Finally, the location of newly placed replicas is updated by using updatelocation() function.

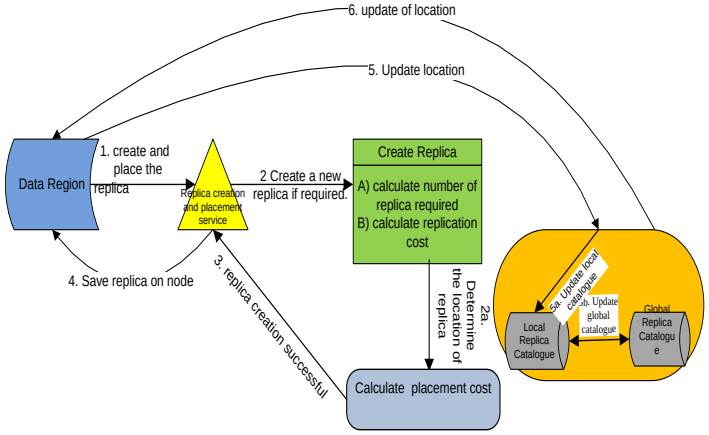

Figure2: Working of replica creation and placement service.

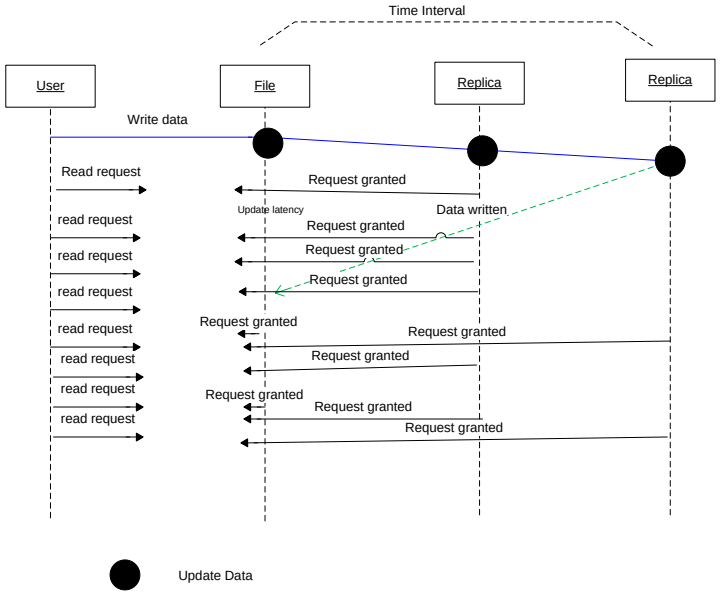

Figure 3: Sequence diagram of Adaptive Consistency Service

Algorithm 4 outlines the consistency maintenance phase. It maps the similarity between the primary copy and its replica on a node. The process is repeated for each region until all the nodes are mapped. The mapping of replicas locally in a region decreases network distance from the user which consequently reduces access latency of the system .We have used lazy consistency strategy in order to avoid wasting network resources by eluding unnecessary broadcast of the update message. User can make read and write requests for any replica of a file. The read and write request frequency is tracked by the access count for each checkpoint time interval τ, which helps in determining the threshold value. The threshold value is dependent on the access frequency of current and previous checkpoint time interval. If the threshold value for the current time interval is equal to the previous interval, no messages will be broadcasted. If the value of the threshold is different only then the consistency mechanism is initiated. All the réplicas stored across the regions of the grid system will be updated. Working of algorithm 3 and 4 are shown in Figure 2 and Figure 3 respectively.

Algorithm 4 Consistency Maintenance

1: **procedure** Consistency Manintenance
2**: input:** *Replica Location*
3: **if** Write request received **then**
4:[*enu*] ← [*enu*] + 1;
5:Execute write operation on node n_{ij}
6: **end if**
7: **if**(time interval finished)
8: Compute $T\_res\_old\ (\tau)$
9: **if** $T\_res\_old(\tau) > T\_res\_old\ (\tau - 1)$
10 **for**$(N_{reg} = 1; N_{reg} \leq N\ ; N_{reg}$ **++)**
11: **for**$\left(n_{ij} = 1; n_{ij} \leq Num_{Reg_i}; n_{ij} + +\right)$
12: Conduct réplica synchronization
13: **else**
14: do nothing
15: **end for**
16: **end for**
17: **end if**
18: **end if**
19**: return**(consistent replicas)
20: **end procedure**

3. EXPERIMENTAL TEST BED

We evaluated the RCPC algorithm using Optorsim simulator [6]. For RCPC, European Data Grid EU testbed architecture has been considered [8]. For simulation total 52 nodes are considered during the setup of the network. Two nodes are considered as super nodes, which act as an interface between user and network. Further, the network is divided into 7 regions. The nodes are assigned to each region in a random fashion. Requests made by the user are executed on the nodes having sufficient computing and storage capacity. The storage capacity of super nodes and other nodes are 10PB and 100 GB respectively. The file size varies from 10 MB-1GB. Each node can execute a maximum of 100 requests, which require file ranging from 1-10 in number. Network latency of the system varies from 10ms-20ms while the latency to access storage media is 2ms-5ms. To simulate the real-world grid environment, random traffic patterns are generated and introduced as background traffic in a simulated environment. Various simulation parameters are used for the execution of requests are shown in Table 1.

Table 1 Various Parameters Considered during Simulation

Parameters	**Values**
Size of File	10 MB - 1GB
Files in Number	1000
Number of Request/Job	100
Network Latency	10ms-20ms
Storage Media Latency	2ms-5ms
Transfer Rate	50 Mb/s
Storage Capacity	100GB,10PB

The grid topology used for the proposed algorithm is specified in a grid configuration file of Optorsim. Various parameters, such as access pattern of jobs, number of job, etc. are set in the parameter configuration file. The background traffic is introduced generated using a bandwidth configuration file. Various performance metrics presented in Table 2.

Table 2 Performance Metrics

Performance Metrics	**Description**
Mean Execution Time	Total execution time + Total Waiting time/Number of completed requests
Effective Network Usage	It specifies the estimation of effective usage of network resources.
Hit Ratio	Specifies the number of réplica that exists locally.
Transfer Rate	Time to transfer file from one location to another.
Storage Utilization	Measure the effective utilization of storage resources.
Computing Usage	Measure the effective utilization of computing resources

The efficiency of RCPC strategy is evaluated by considering various scheduling and réplication strategies. Five scheduling strategies, namely, Shortest Queue, Queue Access Cost, Access Cost, Random and HS have been considered for evaluation. Additionally, some strategies for réplication have also been considered for assessment. These réplication strategies are:

- Least Recently Used (LRU) [39]: In this strategy, réplications always take place at the node where the execution of job is done. If the available space for the new réplica is not sufficient, the oldest file is removed from the SE.
- Least Frequently Used (LFU) [39]: In this strategy, réplications always take place at the node where the execution of job is done. If the available space for the new réplica is not sufficient, the file which is accessed for lest number of times is replaced from the SE.
- Bandwidth Hierarchy Réplication (BHR) [29]: In this strategy, the probability to réplicate a file is more likely which is connected to the node having highest bandwidth in the network.
- Modified BHR [35]: In Modified BHR, access frequency is the decisive parameter while réplicating a file.
- 3-Level Hierarchical (3LHA) [24]: In this strategy, the network layout along with the bandwidth plays a pivot role during réplication process.
- No Réplication [42]: In this strategy, whenever a request for a file is made only then réplicas are created.
- Réplica Creation and Placement (RCPC): In RCPC strategy the réplicas are placed at appropriate nodes with highest access frequency of the file. This strategy reduces the average access latency of the file by selecting the best node for placing the réplica considering the access frequency, storage and transfer time. Moreover, the consistencies of the files are also maintained using asynchronous approach.

Next section will discuss the investigational results for RCPC strategy with various parameters such as execution time, effective network usage, number of réplicas, percentage of accessing updated data, number of update transactions.

4.1. Execution Time

The meanjob execution time is calculated as the ratio of total duration in which jobs are executed completely and waiting time of the job in the waiting queue to the total number of jobs. Meanjob Execution time is calculated as:

$$Meanjob\ Execution\ Time = \frac{\sum_{n=1}^{i}(T_i+W_i)}{n}, \tag{4.1}$$

Where, n is the amount of jobs submitted to the system,T_i is time taken by the system to complete the i^{th} job and W_i is total time spent by i^{th}job in a waiting queue.For evaluating the mean job execution time three scenarios are considered:

4.1.1. Mean Job Execution Time using Various Réplication and Scheduling Strategies

Figure 4 represents that No réplication has performed worst. The result predicts that LRU and LFU strategies have almost the same execution time. Both strategies performed better than No Réplication but their execution time is very high as compared to other strategies. BHR algorithm performed better than LRU and LFU. By avoiding the network congestion in data grid BHR is able to reduce the data access time. The performance of 3LHA is further improved as it considers the variation in inter and intra region communication. RCP performed best in the scenario as it considers the access frequency, waiting time, réplication cost and placement cost for creating and placing the réplicas on a node.

The time to execute a job increases while using Random scheduling strategy, as it does not consider any factors. In Shortest Job Queue Scheduling each node receives approximately the same number of jobs. The overall job execution time increases if the network has low bandwidth as it takes more time to transfer jobs or the réplicas to the desired node. In Access Cost Scheduling strategy access cost is the imperial factor for scheduling the jobs in the network. Nodes having lower access cost may receive higher number of jobs to execute which affect the load balancing factor. So, overall performance of the system will reduce. The Queue Access Cost considers access cost along with shortest job queue; as a result, the total time to execute a job reduces significantly. The HS strategy schedules the request in the vicinity of the data. It also guarantees load balancing by ensuring that the nodes with excellent network connectivity are not overloaded and nodes with poor connectivity are not left idle.

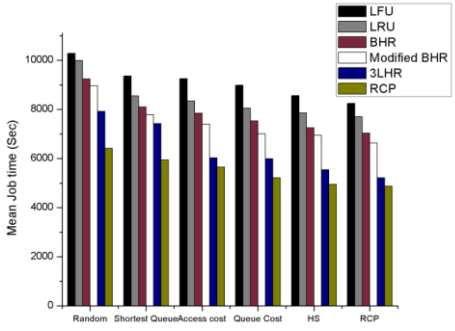

Figure 4: Meanjob Execution Time with Respect to Varying Strategies

4.1.2 Mean Job Execution Time using Various Access Patterns

Figure 5 shows that RCPC has the performed best in terms of mean job execution time when requests are executed using various access patterns for the files. In No Réplication strategy, no file is réplicated, and all the requests have to be transferred to super file hence overloading the node containing the file. This leads to increased job execution time and in this scenario, no réplication has performed the worst. The Random access pattern comprises Random, Gaussian random walk and Unitary random walk. Here few files are more likely to be requested by the users, so a large percentage of requested files have been réplicated before. In case of RCPC strategy, the requests are scheduled to the file in the vicinity of the user which has made the request. Therefore, RCPC strategy has more improvement for the execution of the file.

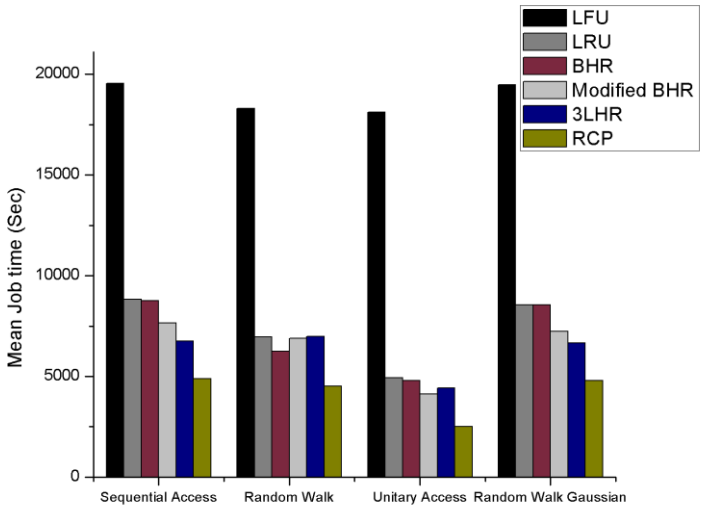

Figure 5: Mean Job Execution Time using Various Access Patterns

4.2 Effective Network Usage (ENU)

The purpose of using ENU is to estimate network resources efficiency. The ENU is the ratio of the number of times the file is accessed remotely and the amount for which the file has been réplicated to the number of files that have been accessed locally. For a particular network topology, if the value of ENU is less, it indicates better optimization strategy and higher effectiveness in réplicating the files at a suitable location. Effective network usage (ENU) is defined as [28]:

$$ENU = (N_{rem} + N_{rep})/N_{loc} \quad (4.2)$$

where $N_{rem}$is a number of access times file is read from a remote site, $N_{rep}$is number of times the file is réplicated and $N_{loc}$is the number of files accessed locally. Figure 6 illustrates the comparative analysis of various replication strategies with effective network usage. The network resource utilization is done efficiently in RCPC strategy. The No Réplication strategy performs worse as it utilizes maximum number of network resources. 3LHA shows improved performance as compared to BHR and modified BHR.

LRU has the least job execution time and highest effective network usage, showing that it is poor at making réplication decisions.

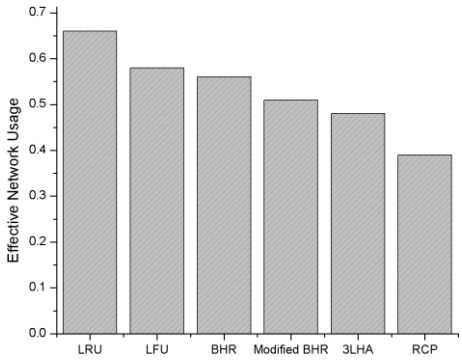

Figure 6: Effective Network Usage for Various Strategies

4.3.Hit Ratio

Hit ratio is considered to check the locality of file locally or remotely. It is dependent on the total number of replications. More number of replications indicates that the file is not available locally for execution. If number of replications is large, hit ratio of the file is less which indicates for executing the job, file are accessed remotely. No Replication strategy has very few files located locally, resulting in lower hit rate. Figure 7 illustrates the hit ratio pattern for various strategies. It is observed that proposed strategy has the highest number of hits for almost all access patterns.

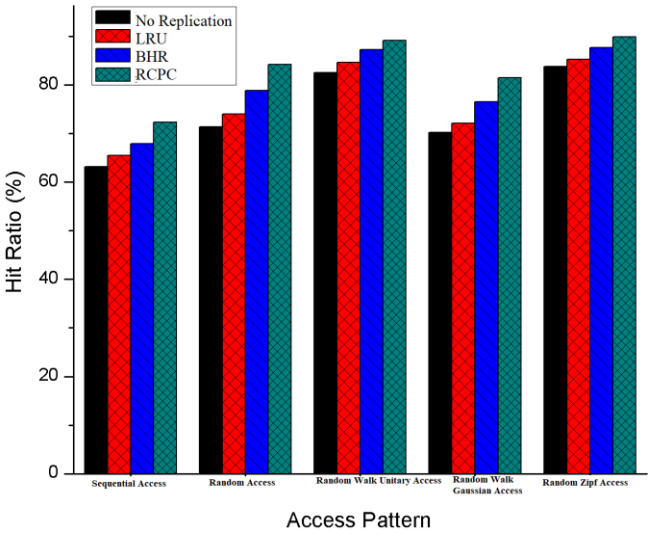

Figure 7: Access Pattern vs. Hit Ratio

4.4.Transfer Rate

Transfer rate is measured from the time when the file starts transferring from one node to another till the time when all transmission of files is finished. The transfer rate is dependent on C_{cost} and $C_{storage}$, which are derived from equation (3.3) and (3.4) respectively. The data rate of transfer of proposed strategy from the data site to the requesting site is compared with LRU, BHR, and No Replication strategies. The data transfer rate is measured by finding the time required for transferring the number of bytes sets from one site to another site. Figure 8 shows the comparison of mean data transfer rate. The mean data transfer rate was reduced in case of RCPC, as the files are located very close to the client. No Replication strategy perform worst, whereas, LRU and BHR take more time to transfer files than RCPC.

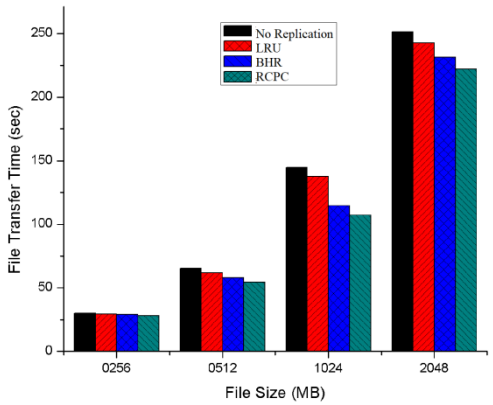

Figure 8: File Transfer Rate Vs Size of File

4.5. Storage Utilization (SU)

It is the measure for estimating the effective utilization of the storage resource in the proposed system. The SU is calculated as amount of storage space occupied by files or its replicas in a region. It is dependent on the storage capacity of the region and the number of files. SU is calculated as:

$$SU = \frac{(S_t - \sum_{i=1}^{N} Aval_{str})}{S_t} * 100 \quad (4.3)$$

Where S_t is total storage capacity of a region and $Aval_{str}$ is calculated from equation

$$Aval_{str} = S_t - S_{usage} \quad (4.4)$$

Where S_{usage} is storage space consumed by the files in the region.

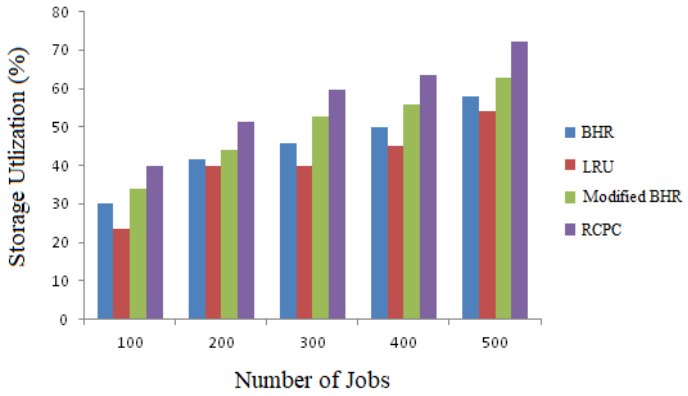

Figure 9: Comparison of Storage Usage

The storage resource utilization of LRU strategy is best less replicas are created. Performance of RCPC is worst as the number of replicas increased. The BHR and Modified BHR strategies use moderate storage space. The results are depicted in Figure 9.

4.6. Computing Usage (CU)

Monitoring the usage of computing power of data grid resources is a valuable source of information. This can be defined as the percentage of time the computing element is active or executing a job. The total computation power is derived as:

$$CU = \frac{(Comp_{node} - \sum_{i=1}^{N} Aval_{cap})}{Comp_{node}} * 100 \quad (4.5)$$

More computational usage indicates that the uniform distribution of jobs among the regions, reduces the execution time of the job. The computing usage is compared with three optimizing strategies; BHR, LRU, and RCPC.

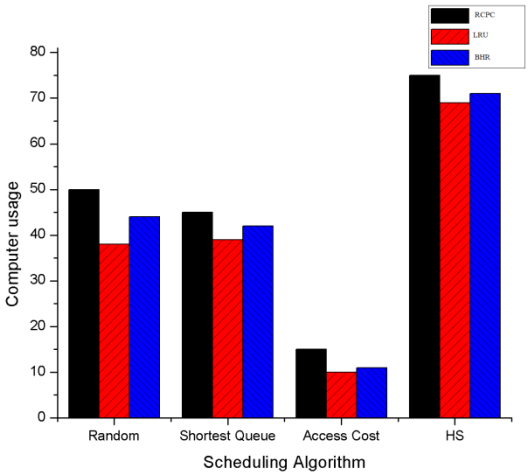

Figure 10: Comparison of Effective Network Usage

Figure 10 show that the scheduling strategy mainly Access Cost Scheduling has lowest computing usage as in this strategy jobs are scheduled to regions with high network bandwidth. In case of HS scheduling strategy, the jobs are scheduled to the regions with high network bandwidth and lower load gauge, which helps in ensuring the uniform distribution of the job in the existing system. The Random and Shortest Queue strategies show similar behaviour and perform better than Access Cost Scheduling.

5. CONCLUSIONS

In this paper, a popularity-driven réplica creation, placement, and consistency (RCPC) strategy is discussed. RCPC strategy is based on a distributed model which dynamically adapt to change in both network and user behavior while improving the performance of the overall system. RCPC strategy is divided into two phases. First replicas are created for the frequently accessed file by calculating the access frequency of the file and then are placed at an appropriate location based on minimum placement cost. Secondly, this approach is extended by using dynamic threshold value at every predefined time interval for data consistency in the data grid environment.The proposed strategy increases data availability and also reduces unnecessary réplication by restricting a number of réplicas in a region. It places the réplica at an appropriate location so as to reduce the placement cost. Additionally, the consistencies of réplicas are maintained using the lazy approach. A rigorous examination of the proposed strategy on Optorsim simulator has been carried out in order to judge the effectiveness of the strategy. The simulation results have established the effectiveness of the proposed strategy over various state of art.

Future work can be extended towards regulating the balanced threshold function and making it more flexible to varying time intervals. Some selection and scheduling algorithm can also be included using multi-master approach.

RECEIVED: MAY, 2019.
REVISED: OCTOBER, 2019.

REFERENCES

[1] ABDULLAH, M., OTHMAN, MD., SULAIMAN N., IBRAHIM, A. and OTHMAN, T. (2005): Data Discovery Algorithm for Scientific Data Grid Environment. **Journal of Parallel and Distributed Computing**, 65,1429–1434.

[2] AL MISTARIHI, H.H.E. and YONG, C.H., (2008): Réplica management in data grid. International **Journal of Computer Science and Network Security**, 8, 22-32.

[3] AMJAD, T., SHER, M., and DAUD, A. (2012): A Survey of Dynamic Réplication Strategies for improving Data Availability in Data Grids. **Future generation Computer Systems**, 28,337-349.

[4] BAKER, M., BUYYA, R., and LAFORENZA, D. (2002): Grids and Grid Technologies for Wide Area Distributed Computing. **International Journal of Software: Practice and Experience**, 32, 1437–1466.

[5] BELALEM, G.BELAYACHI, N.BEHIDJI, R., and YAGOUBI, B. (2010): Load Balancing to Increase the Consistency of Réplicas in Data Grids**. International Journal of Distributed Systems and Technologies,** 1, 42-57.

[6] BELL, W.H., CAMERON, D.G., MILLAR, A.P., CAPOZZA, L., STOCKINGER, K. and ZINI, F., (2003): Optorsim: A grid simulator for studying dynamic data réplication strategies. **The International Journal of High Performance Computing Applications**, 17, 403-416.

[7] BSOUL, M., KHASAWNEH, A., ABDALLAH, E., and KILANI, Y. (2011): Enhanced Fast Spread Réplication Strategy for Data Grid. **Journal of Network and Computer Applications**, 34, 575–580.

[8] CAMERON, D., CASEY, J., GUY, L., KUNSZT, P., LEMAITRE, S., MCCANCE, G., STOCKINGER, H., STOCKINGER, K., ANDRONICO, G., BELL, W. and BEN-AKIVA, I., (2004): Réplica management in the European DataGrid project. **Journal of Grid Computing**, 2,341-351.

[9] CHANG, J.S., and CHANG, R. S. (2008): An Innovative Réplica Consistency Model in Data grids. **International Symposium on Parallel and Distributed Processing with Applications (ISPA'08),** Sydney, NSW: 10-12, 3-10.

[10] CHANG, R. and CHANG, J.S.(2006): Adaptable Réplica Consistency Service for Data Grids, **3rd International Conference on Information Technology: New Generations (ITNG 2006)**, Las Vegas, NV,646-651.

[11] CHARRADA, F.B., OUNELLI, H. and CHETTAOUI, H., (2010): November. An efficient réplication strategy for dynamic data grids". IEEE, **International Conference on P2P, Parallel, Grid, Cloud and Internet Computing,**50-54.

[12] CHEN, P. C., CHANG, J. B., YANG, J. H., ZHUANG, Y. C., and SHIEH, C. (2009): Spigot: Fragment-Level File Sharing and Consistency on Grids. **International Conference on Advanced Information Networking and Applications (AINA'09)**, Bradford: 26-29, 884-891.

[13] CHIHOUB, H. E. (2013): Self-Adaptive Cost-Efficient Consistency Management in the Cloud**. International Symposium on Parallel & Distributed Processing Workshops and PhD Forum(IPDPSW)**, Cambridge, MA, 2290-2293.

[14] GRACE, R. K, and MANIMEGALAI, R., (2014): Dynamic Réplica Placement and Selection Strategies in Data Grids—A Comprehensive Survey. **Journal of Parallel and Distributed Computing**, 74, 2099-2108.

[15] HORRI, A., SEPAHVAND, R. and DASTGHAIBYFARD, G.H., (2008): A Hierarchical Scheduling and Réplication Strategy. **International Journal of Computer Science and Network Security,** 8, 30-35.

[16] JAN-JAN, W., YI-FANG, L., and PANGFENG, L., (2008): Optimal Réplica Placement in Hierarchical Data Grids with Locality Assurance. **Journal of Parallel and Distributed Computing,** 68, 1517-1538.

[17] KAVITHA R., ADRIANA IAMNITCHI, and FOSTER, I. (2002): Improving Data Availability through Dynamic Model-driven Réplication in large peer-to-peer communities**, Cluster Computing and the Grid. 2nd IEEE/ACM International Symposium**, 376-376.

[18] KHANLI, L., M.ISAZADEH, A., and SHISHAVAN, T. N. (2011): PHFS: A Dynamic Réplication Method to Decrease Access Latency in Multi-Tier Data Grid. **Future generation Computer Systems**,27, 233-244.

[19] KRASKA, T.HENTSCHEL, M. ALONSO, and G.KOSSMANN, D. (2009): Consistency Rationing in the Cloud: Pay only when it matters. **Proceeding of VLDB Endowment,** 2, 253–264.

[20] LI, W., YANG, Y., and YUAN, D. (2016): Ensuring Cloud Data Reliability with Minimum réplication by proactive réplica checking. **IEEE Transactions on Computers,** 65, 1494-1506.

[21] LIAO, J. LI, LI. CHEN, H. and LIU, X., (2015): Adaptive Réplica Synchronization for Distributed File Systems. **IEEE System Journal,** 9,865-877.

[22] MANSOURI, N., (2014): A threshold-based dynamic data réplication and parallel job scheduling strategy to enhance Data Grid. **Cluster Computing**, 17, 957-977.

[23] MANSOURI, N., and ASADI, A. (2014): Weighted Data Réplication Strategy for Data Grid Considering Economic Approach. **World Academy of Science, Engineering and Technology, International Science Index, Computer and Information Engineering,** 1,542.

[24] MANSOURI, N., DASTGHAIBYFARD, G.H. and MANSOURI, E., (2013): Combination of Data Réplication and Scheduling Algorithm for Improving Data Availability in Data Grids. **Journal of Network and Computer Applications**, 36,711-722.

[25] MOHAMMAD, S., (2012): Placement of Réplicas in Large-Scale Data Grid Environments. **PhD thesis, University of Manitoba (Canada).**

[26] NASEERA, S. and MURTHY, K.M., (2009): Agent based Réplica Placement in a Data Grid Environment. **International Conference on Computational Intelligence, Communication Systems and Networks**, 426-430.

[27] NHAN, N. D., SANG, B. L., and YEO, C. K., (2007): Combination of Réplication and Scheduling in Data Grids. **International Journal of Computer Science and Network Security**, 7,304-308.

[28] OZSU, T.M. and VALDURIEZ, DISTRIBUTED DATABASE MANAGEMENT SYSTEM. (2018) https://www.cs.purdue.edu/homes/bb/cs54 2-06Spr/week3_lecture2.ppt (Accessed 26 June, 2018)
[29] PARK, S.M., KIM, J.H., KO, Y.B. and YOON, W.S., (2003): December. Dynamic Data Grid Réplication Strategy based on Internet Hierarchy. **International Conference on Grid and Cooperative Computing**, 838-846. Springer, Berlin, Heidelberg.
[30] PEREZ, J. and M.CARBALLEIRA, F. G.(2010): Branch Réplica Scheme: A New Model for Data Réplication in Large Scale, **Future Generation Computer Systems**, 26,12–20.
[31] RADI, JM., (2014): Improved Aggressive Update Propagation Technique in Cloud Data Storage. **International Journal of Emerging Trends & Technology in Computer Science,**3, , 102-106.
[32] RAHMAN, R. M., BARKER, K., and ALHAJJ, R. (2006): Réplica Placement Design with Static Optimality and Dynamic Maintainability. Sixth **IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06),** 1-4.
[33] RAHMAN, R.M., BARKER, K. and ALHAJJ, R., (2008): Réplica Placement Strategies in Data Grid. **Journal of Grid Computing**, 6, 103-123.
[34] SASHI, K.and THANAMANI, A S. (2010): Dynamic Réplica Management for Data Grid. **International Journal of Engineering and Technology,**2, 329-333.
[35] SASHI, K. and THANAMANI, A.S., (2011): Dynamic Réplication in a Data Grid using a modified BHR region-based algorithm. **Future Generation Computer Systems**, 27, 202-210.
[36] SOURAVLAS, S. (2017); Sifaleras, A. Binary-tree based Estimation of File Requests for Efficient Data Réplication. **IEEE Trans. Parallel Distributed Systems,** 28, 1839–1852
[37] TANENBAUM, A. S., and VAN STEEN, M. (2007): **Distributed systems: principles and paradigms**. Prentice-Hall, Amsterdam.
[38] TANG, M., LEE, B.S., YEO, C.K. and TANG, X., (2005): Dynamic réplication algorithms for the multi-tier data grid. **Future Generation Computer Systems**, 21,775-790.
[39] VASHISHT, P., KUMAR, R. andSHARMA, A., (2014): Efficient dynamic réplication algorithm using agent for data grid. **The Scientific World Journal,**1-10, DOI:10.1155/2014/767016.
[40] VASHISHT, PRIYANKA, VIJAY KUMAR, R KUMAR and A SHARMA.(2019): Optimizing Réplica Creation using Agents in Data Grids. **International Conference on Artificial Intelligence (AICAI). IEEE**, 542-547.
[41] WARHADE, S., DAHIWALE, P. and RAGHUWANSHI, M.M., (2016): A dynamic data réplication in grid system. **Procedia Computer Science,** 78, 537-543.
[42] YANG, C. TSAI, W. CHEN, T.and HSU, C. (2007): A One-way File Réplica Consistency Model in Data Grids, **Proc IEEE Asia-Pacific Services Computing Conference**, Tsukuba Science City, 364-373.
[43] YANG, Y. LI, D. (2004): Separating Data and Control: Support for Adaptable Consistency Protocols in Collaborative Systems, **Conference on Computer Supported Co-operative work (CSCW'04), Chicago,** Illinious, USA, 11-20.