Leveraging a Multi-Objective Approach to Data Replication in Cloud Computing Environment to Support Big Data Applications

Mohammad Shorfuzzaman[1], Mehedi Masud[2]
Department of Computer Science, Taif University, Taif, Saudi Arabia

***Abstract*—Increased data availability and high data access performance are of utmost importance in a large-scale distributed system such as data cloud. To address these issues data can be replicated in various locations in the system where applications are executed. Replication not only improves data availability and access latency but also improves system load balancing. While data replication in distributed cloud storage is addressed in the literature, majority of the current techniques do not consider different costs and benefits of replication from a comprehensive perspective. In this paper, we investigate replica management problem (which is formulated using dynamic programming) in cloud computing environments to support big data applications. To this end, we propose a new highly distributed replica placement algorithm that provides cost-effective replication of huge amount of geographically distributed data into the cloud to meet the quality of service (QoS) requirements of data-intensive (big data) applications while ensuring that the workload among the replica data centers is balanced. In addition, the algorithm takes into account the consistency among replicas due to update propagation. Thus, we build up a multi-objective optimization approach for replica management in cloud that seeks near optimal solution by balancing the trade-offs among the stated issues. For verifying the effectiveness of the algorithm, we evaluated the performance of the algorithm and compared it with two baseline approaches from the literature. The evaluation results demonstrate the usefulness and superiority of the presented algorithm for conditions of interest.**

***Keywords*—Big data applications; data cloud; replication; dynamic programming; QoS requirement; workload constraint**

I. Introduction

Lately, cloud computing has become an attractive and mainstream solution for data storage, processing, and distribution [1]. It provides on-demand and elastic computing and data storage resources without the large initial investments usually required for the deployment of traditional data centers. Cloud computing facilitates the delivery of storage and computing resources as services without any restriction on the location [2]. It offers three types of architecture for service delivery such as SaaS (Software as a Service), PaaS (Platforms as a Service) and IaaS (Infrastructure as a Service) [3]. End users can use the software applications hosted by the cloud providers in a SaaS architecture. In case of IaaS architecture, virtualized storage and computing resources are provided by the cloud providers. Customers of IaaS can then access these resources and other services to complete the application stack such as to create virtual machines and to deploy operating systems and middleware. As for PaaS architecture, the cloud providers allow users to develop, run and customize applications while the providers host them on their own hardware infrastructures.

Due to the benefits offered by cloud computing, organizations are now moving to the cloud. Some of the top cloud providers such as Amazon S3 [4], Google Cloud Platfom [5], App iCloud[1], IBM Cloud[2], Microsoft Azure[3], and DropBox[4] provide cloud services to thousands of millions of users by means of geographically dispersed data centers across the globe. As such, end users are relieved of purchasing traditional expensive hardware (data centers) to process huge amount of data. Consequently, growing interests are shown in an effort to develop data-intensive applications that access massive amount data sets. To name a few, smart city applications and Healthcare Information Systems (HISs) are struggling with data bombardments that need immediate solutions to process these data for knowledge acquisition. These data-intensive applications demand for large-scale computing and storage resources and recent cloud computing advancement suits to meet these challenges. Lately, popular big data applications such as Facebook, Twitter, and Human Genome Project[5], are making most use of computing framework such as MapReduce and Hadoop [6] for petabyte-scale data processing to extract insight.

The performance of these big data applications on cloud depends largely on the reliability, availability and efficient access to the data centers [7]. To address these issues data can be replicated in various locations in the system where applications are executed [8], [9], [10], [11]. This means that data partitions can be replicated across different data centers in the cloud. Replication not only improves data availability and access latency but also improves system load balancing. Many existing cloud systems adopt static replication strategies by replicating data in some physical locations without considering the issue of replication cost and terrestrial diversity. For instance, GFS (Google File System) [5], Amazon S3 [4], and HDFS (Hadoop Distributed File System) [9] implement a replication strategy where three copies of data

[1] App iCloud, https://www.icloud.com/
[2] IBM Cloud, https://www.ibm.com/cloud/
[3] Azure: Microsoft's Cloud Platform, https://azure.microsoft.com/
[4] Dropbox, https://www.dropbox.com/
[5] Human Genome Project, http://www.ornl.gov/hgmis/home.shtml

are created at one time to increase reliability of target data. This replication strategy would result in increased (twice) replication cost which will adversely affect the effectiveness of the cloud system.

There are a number of critical issues that need to be addressed to achieve big data replication in cloud storage: *i*) Determining the degree of data replicas that should be created in the cloud to meet reasonable system and application requirements. This is an important issue for further research. Since excessive replication would increase the replica maintenance and storage cost creating too many or fixed replicas are not a good choice. *ii*) Distribution of these replicas to achieve higher efficiency and better system load balancing. *iii*) Maintaining consistency among replicas due to replica update or delete. These three correlated problems are jointly referred to as the replica management problem. In addition to that, some data replication strategies in the cloud consider energy efficiency issue to optimize the energy consumed by the data centers.

While data replication in distributed cloud storage is addressed in the literature, majority of the current techniques do not consider different costs and benefits of replication from a comprehensive perspective. Most of them provide emphasis on high availability, fast response and high efficiency. Even though critical, these average performance measures do not tackle the quality requirements demanded by various data-intensive applications. The performance of these algorithms gets better as the number of replicas increases. However, the increased number of replicas incurs higher degree of replication cost associated with storage and energy. Hence, the goal should be to minimize the required number of replicas to avoid high replica creation and maintenance cost. Accordingly, a good replica management strategy should be designed to balance a variety of tradeoffs. Moreover, as in the case of existing replication strategies, the increased number of replicas is cost prohibitive due to consistency management.

Given the issues and trends stated above, in this paper, we investigate replica management problem in cloud computing environments to support big data applications from a holistic view. To this end, we provide cost-effective replication of large amount of geographically distributed data into the cloud to meet the quality of service (QoS) requirements of data-intensive (big data) applications while ensuring that the workload among the replica data centers is balanced. Targeting a typical cloud platform that encompasses disparate data centers, we formulate cost-minimizing data replication problem, and present an effective distributed algorithm that optimizes the choice of energy efficient data centers into the cloud for allocating replicas taking into account the consistency among replicas due to update propagation. Thus, we build up a multi-objective optimization approach for replica management in cloud that seeks near optimal solution by balancing the trade-offs among the stated issues. Hence, we make the following contributions:

- Analyze the replica management problem to formulate mathematical models to describe the stated objectives such as application QoS requirements, system load variance, and replica consistency and come up with a replication model considering overall replication cost.
- Propose a highly distributed offline replication scheme that computes data center locations in the cloud by minimizing overall replication cost to meet the above-mentioned stated objectives.
- Assess the benefit and applicability of our scheme using extensive simulation experiments over a wide range of parameters (e.g., no. of data centers, size and access rate of each data file, data center workload capacity constraint, traffic pattern, application QoS requirements, etc.).

The remainder of the paper is presented as follows. QoS-aware replica placement techniques are reviewed in Section 2. Section 3 presents the system design and architecture. The proposed replication algorithm is provided in Section 4. Sections 5 and 6 present simulation methods and obtained results, respectively. We conclude in Section 7 with directions of future work.

II. Related Work

To date, a number of replication strategies have been adopted in many application areas such as cloud storage, large data storage, data grids, distributed systems and so on. These replication strategies are mainly divided into static and dynamic categories. The number of replicas created is fixed in static categories. As mentioned already, GFS [5] and HDFS [6] are adopting this strategy. However, this technique is lacking flexibility even though replica management is straightforward. Most of the current research work is focusing on dynamic replica placement strategies where the number and location of replicas created can vary depending on the application requirements.

Huang et al. [12] propose a reliability model for providing data service in cloud storage systems. The authors present the service model which sets off replica creation and selects storage site based on the relationships among the access reliability of the application nodes. The evaluation results show that the proposed method increase data service reliability with a significant decrease in the count of replicas created. The authors in [13] present a file replication mechanism in P2P file sharing systems which works based on swarm intelligence. The idea is to exploit the collective behavior of swarm nodes which share common interest and in close proximity. In contrast to other similar methods, this approach determines the replica locations according to the accrued query rates of nodes in the swarm instead of only one node. This results in fewer replicas and improved querying efficiency. The proposed technique also considers replica consistency using a message update mechanism in a top to bottom fashion.

Liu et al. [14] tackle the data replication problem in Online Social Network (OSN) by means of a careful data replication technique in dispersed datacenters (SD^3). The goal is to lessen inter-datacenter communication while improving service latency. The replication technique takes into account update rates and visit rates to determine the user data for replication.

To ensure reduced inter-datacenter communication, the technique atomizes users' different type of data for replication. Experimental results demonstrate higher efficiency of SD^3 compared to other replication methods. Yet, the authors in [15] come up with a data replication strategy called Cadros, for decentralized online social network (DOSN) to increase data availability. First, they have done a quantitative analysis of cloud storage to retrieve knowledge about storage capacity and then propose a model that predicts the level of data availability that Cadros can attain. The technique takes into account data partitioning in the cloud in such a way that minimizes the communication overhead and still aims at achieving desired level of data availability. At the same time, the paper also presents data placement strategies that aim to satisfy the stated objective in terms of other performance metrics.

Sun et al. [16] address the problem of an increase in overload nodes which causes poor service performance in cloud-P2P systems through a Dynamic Adaptive Replica Strategy (DARS). Special attention is given to reduce the number of overload nodes by determining appropriate time for replica creation. The replication strategy works based on the node's overheating similarity and exploits a fuzzy-logic based clustering algorithm to determine nodes for optimal replica placement. Experimental results reveal that DARS significantly reduces the number of overload nodes facilitating low replica access delay with high load balance.

A dynamic data replication for hierarchical cloud storage is proposed in [17]. The authors have used temporal locality of a data file to determine its popularity which means that recently used files are likely to be accessed again in near future. The replication of a certain data file is triggered when its popularity crosses a predefined threshold value. Moreover, the newly created replicas are stored in data centers that are directly connected. In a subsequent effort [18], this replication strategy is improved by the introduction of checkpoint mechanism to enhance data availability. Another cloud data replication strategy called CDRM (Cost-effective Dynamic Replication Management) [19] is proposed for heterogeneous environments. CDRM develops a system model to formulate the relationship between data availability and replication factor. It consistently maintains minimum number of replicas in the system and ensures proper balance of workload across the datacenter nodes and reduces access delay. The authors in [20] proposed a replication mechanism called RFH (Resilient, Fault-tolerant and High efficient) which is able to replicate data based on the varying query load. They have taken into account failure rate, size of the partitions, link capacity and the distance in replication cost calculation.

Gill and Singh [21] presented a dynamic replication algorithm for heterogeneous cloud data centers that ensures minimum number of replicas while maintaining desired level of data availability. Their algorithm works based on the concept of knapsack to minimize the replication cost while considering re-replication by moving replicas from high-cost data centers to low-cost ones. The algorithm is found to be effective in minimizing replication cost with satisfactory level of data availability. Boru et al. [22] propose a replication solution which optimizes energy efficiency of the system. GreenCloud simulator is used for performance assessment for the replication technique. The simulation results show that the proposed replication strategy significantly improves data access time, network resource usage, and energy consumption. Long et al [23] propose a replica management strategy called MORM (Multi-objective Optimized Replication Management) to be used in cloud data storage. The idea is to make a trade-off among different important factors affecting replication decision. Another dynamic replication strategy called dynamic popularity aware replication strategy (DPRS) is proposed by Mansouri et al [24] which works based on the access popularity of data and considers parallel downloading of data segments from multiple data centers to improve access performance. Sun et al [25] addressed the replication problem from a different perspective where they handled the trade-off between access performance and consistency dynamically. The authors came up with a replication protocol called CC-Paxos to adaptively balance trade-off between latency and consistency which is independent of any underlying data stores. In addition to that, a number of other researchers [26-31] have proposed various dynamic replica creation and maintenance (e.g., consistency) techniques in distributed cloud storage systems.

At this point, it is evident that there has been reasonable effort in addressing data replication problem in cloud computing environment. Nevertheless, the current research has not adequately addressed the issue of QoS requirements. Few researchers in the literature have worked in this area. Among them, Lin et al. [32] proposed two algorithms that take into account QoS requirements while replicating data in cloud computing systems. The first algorithm is based on a greedy approach which adopts the concept of high-QoS first-replication (HQFR). In this case, the applications requiring QoS are ranked in the order of highest priority to lowest priority applications. However, this algorithm is not able to produce optimal solution. Hence, a second algorithm called minimum-cost maximum-flow (MCMF) is used to minimize replication cost and to maximize the number of replicas satisfying the specified QoS requirements. This algorithm produces near-optimal solution to the replication problem in polynomial time. Varma and Khatri [33] investigated the QoS-aware data replication problem in Hadoop Distributed File System (HDFS). In the original HDFS, a replication technique is used to copy data blocks without any quality restriction between client and the service provider. Hence, the authors consider replication time of an application as the QoS parameter and present an algorithm which can reduce replication cost compared to the existing algorithm. Yet another recent work [34] addresses the QoS aware replication problem for multimedia data in cloud. Naturally, multimedia data has stringent QoS requirement and hence replication of such data often requires fulfillment of QoS requirement from the users. Hence, the authors propose an algorithm to replicate multimedia data considering QoS requirement such as access delay, jitter, bandwidth usage, and loss or error rate implemented in an extended HDFS architecture. The simulation results demonstrate an important reduction in terms of number of replicas that violate the QoS requirement in contrast to the existing replication strategy adopted by Hadoop. In a recent effort, Shorfuzzaman [35] presents a

dynamic replica maintenance technique called DRMS in a multi-cloud environment taking into account various QoS requirements from users. Periodically replicas are relocated to new locations upon significant performance degradation. The simulation results show that the proposed technique improves response time for data access in contrast to it static counterpart where no maintenance takes place.

In addition to that, a number of QoS aware replication strategies are also available in data grid systems [36], [37], [38]. A least value replacement (LVR) strategy is proposed for data replication in data grids by the authors of [36] by taking into account user QoS and storage capacity constraint. The authors devised a replica replacement strategy that determines which replica should be replaced whenever the replica site is found to be full based on the access frequency and future value for a particular data file. Cheng et al. [37] address the replica placement problem in data grids based on general graphs by means two heuristic algorithms namely, Greedy Remove and Greedy Add which take care QoS requirements. The authors consider replica access cost and workload capacity of the replica servers in their cost model. Furthermore, the authors in [38] propose a dynamic QoS-aware replication scheme showing a complete lifecycle for determining positions for replica creation. The replication scheme also articulates how old replicas are relocated and replaced. The data is replicated based on its importance which is determined by its access popularity.

A number of replica placement algorithms are also available in other areas of interest such as distributed and mobile databases and P2P networks. As a whole, these replication strategies are not directly applicable to the target environment in our paper and thus we did not present them in the scope of this paper. In this paper, we present a fully distributed approach to data replication which aims at using a multi-objective model in cloud that seeks near optimal solution by minimizing total replication cost and by balancing the trade-offs among the stated objectives such as QoS requirements from applications, workload of replica data center nodes, consistency of created replicas.

III. System Design and Architecture

Our multi-objective data replication technique is designed based on the HDFS (Hadoop distributed file system) architecture and it is assumed that different cloud computing datacenters are placed in different geographical locations (Fig. 1). There is a three tier topology in HDFS architecture which consists of only one NameNode and a number of DataNodes arranged within multiple racks. The primary task of NameNode is to administer the file system namespace and to maintain a virtual map of how different data blocks are associated with different DataNodes. In Hadoop, DataNodes are meant for the execution of applications.

The request for data block (read and update) goes from an HDFS application to the NameNode which examines the pre-stored mapping for data blocks to DataNodes to find an appropriate DataNode to process the request. Each rack is equipped with an Ethernet switch for facilitating communication between the data nodes within the rack. For inter-rack communication, aggregated Ethernet switches are used. Thus, a logical network topology based on tree structure is built with different switches which use the prevalent communication protocol called spanning tree protocol (STP). To this end, as shown in Fig. 1, core network, aggregation network and access network layers constitute the interconnect network in this scenario.

The three-tier cloud computing architecture as shown in Fig. 1 maintain a number of databases among which the central database (Central DB) is stationed in the highest layer. This database stores entire datasets that are accessed by the applications residing in the cloud. The other type of databases hosted by data centers is primarily used to improve data access efficiency and they are called datacenter database (Datacenter DB). These databases are intended for copying the most often accessed data items from Central DB. Furthermore, each rack is equipped with a rack-level database (Rack DB) which replicates data from datacenter databases.

A replica manager module in the *NameNode* performs analysis of data accesses periodically and determines which data items should be replicated and in which data nodes. The goal is to improve data access efficiency and balance workload of the data centers by spreading data replicas from central database to appropriate data centers down the hierarchy. Replica updates are only transferred from the central database to the databases in the data centers in the lower tiers. In addition to replica managers (as illustrated in Fig. 2), a scheduling broker and data centers constitute the system of could data service. The system is managed centrally by the scheduling broker whereas the locations of the replicas in different data centers are stored in replica managers.

A set of network links (E) interconnecting a set of data centers (V) forms the targeted cloud topology as an undirected tree $T = (V, E)$. The data transfer cost is calculated as the sum of the cost associated with each link along the path. Initially, all data are stored in the central DB and are disseminated down the cloud hierarchy in different data centers upon request from the users in the lowest tier. Over an interval of time, each user carries a count representing the access frequency of a specific data file which indicates the popularity of the file. Data consistency is maintained by propagating updates from the central DB to the replica server nodes. In this case, the associated cost is considered as maintenance cost for previously created replicas.

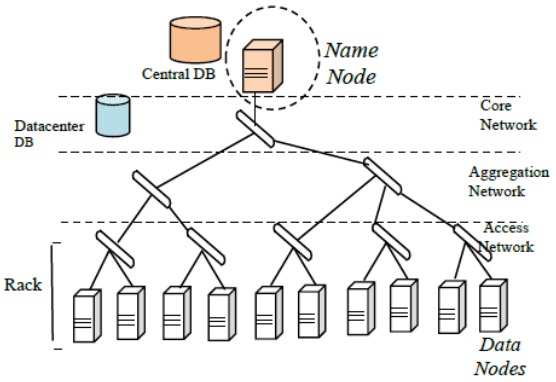

Fig. 1. Three-Tier Cloud Computing DataCenter Architecture based on HDFS [6].

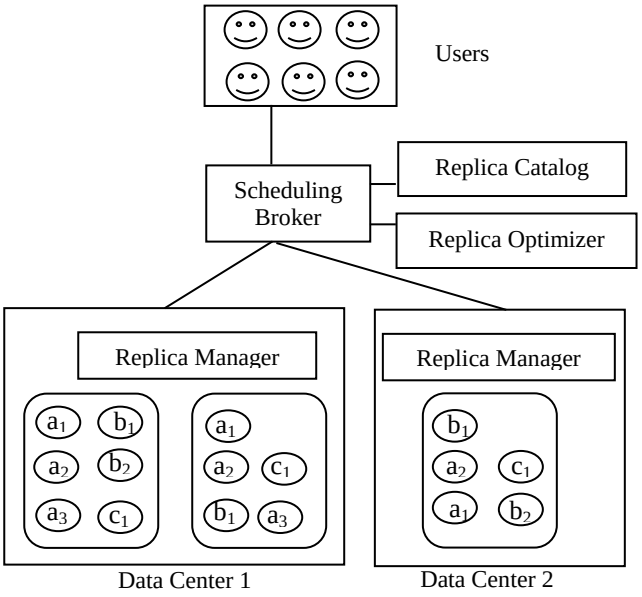

Fig. 2. Cloud Data Service Architecture.

Each data center node is associated with a workload constraint. The workload constraint refers to the maximum number of user requests that is served by a data center server for a certain period of time. It is expected that the incoming data requests from the users are satisfied while not exceeding the workload capacity of replica data centers each having a different workload constraint. If the aggregated workload of a data center server exceeds its capacity, the data center will be considered as overloaded.

Whenever a user needs to access data, it sends the request to the closest data center server along with the QoS constraints. A user (v) specifies his/her QoS requirement, $q(v)$, by means of an upper bound on some criteria. For example, a user may request for a data item within a specific time or from a specific distance in terms of network hops. The replication strategy then confirms that there is a data center server within this $q(v)$ to satisfy the request. If the request is not satisfied due to the absence of a data center server or any other reason the requirement is said to be violated.

A. Data Access Cost

As mentioned above, whenever a datacenter server receives a request for data it serves the request locally if data is present in the server. Alternatively, the request is served by any other closest replica server that holds the requested data. In our data cloud architecture, each user $v \in V$ carries an aggregated count, $count(v)$ which represents the total number of accesses for a particular data item over a time interval. This aggregated count determines the weighted communication cost for a user requesting the targeted data item. It is worthwhile to note that we have excluded the communication cost among the users and their directly associated datacenters as this does not affect the replica placement decision. The overall cost for data access is given as follows:

$$Total_{cost} = \sum_{f \in F} S_{cost} + C_{cost} + CMC_{cost} \quad (1)$$

where C_{cost} is the communication cost, CMC_{cost} is the data consistency maintenance cost and S_{cost} is the storage cost.

Given the set of replica data center servers R and the set of users requesting the data file (f) Q, the overall communication cost (C_{cost}) is computed as follows:

$$C_{cost} = \sum_{f \in F} \sum_{v \in R \cap u \in Q} count(u, f).d(u, v) \quad (2)$$

where F denotes different data file types and $count(u, f)$ represents the data access count of user u for file f.

The storage cost (S_{cost}) is determined as follows. Let $SC_{u,f}$ be the cost of storage associated with a file f stored at replica server (data center) u. Suppose f is replicated over R sites. The total cost for storage for f is

$$S = \sum_{u \in R} SC_{u,f} \quad (3)$$

where, $SC_{u,f}$ denotes the cost of storage of a file with type $\boldsymbol{f}$ in the data center u and the total cost for storage for different types (F) of files will be:

$$S_{cost} = \sum_{f \in F} \sum_{u \in R_f} SC_{u,f} \quad (4)$$

where, R_f denotes the set of replica servers that stores the file of type f.

The consistency maintenance cost for the replicas of a data file f with the update frequency $\mu(f)$, is formulated as follows:

$$CMC_{cost} = \sum_{f \in F} \sum_{T_v \cap R \neq \emptyset} \mu(f).d(v, p(v)) \quad (5)$$

where, $p(v)$ is the parent of node v and T_v is the sub-tree rooted at node v in the update distribution tree. The communication link (v, $p(v)$) participates in the update multicast process if $T_v \cap R \neq \emptyset$. Hence, the cost related to consistency maintenance is determined by aggregating the data transfer costs over the communication links (v, $p(v)$), when $v \neq r$.

B. Problem Definition

We now devise the replica placement problem in a large-scale cloud computing environment. In practice, the replica placement problem can be regarded as an optimization problem:

$minimise\ Total_{cost}$ and maximizing QoS

The goal is to select R data center locations from M probable data centers ($M > R$) by optimizing the cost (storage, communication, and consistency maintenance) of overall replication and satisfying specified QoS requirements of the users and the systems. More specifically, the goal is to identify a suitable replication strategy (i.e., the set of replica servers, R), with the minimal data access cost where the requests for data can be met by a datacenter server while meeting the user quality requirements and not exceeding the concerned data center's capacity limits.

IV. QoS-Aware Replica Placement Algorithm

This section first presents our base distributed heuristic technique of multi-objective replica placement for cloud computing environment (RPCC). The proposed QoS-aware strategy will then be devised by extending the base algorithm

and incorporating both the users' and system's QoS requirements.

We demonstrate that our multi-objective RPCC can be designed as a dynamic programming problem and a distributed technique can be adopted to find its solution in a hierarchical cloud topology. In essence, the proposed technique breaks the overall reapplication problem into various sub-problems and advocates solving these sub-problems separately and combining the solutions to achieve the overall solution. Using the data access cost function, each data center node in the hierarchical cloud topology can determine the cost for creating a replica locally and transferring a replica from another data center up in the hierarchy as well. In RPCC, each data center needs to decide (based on the cost) whether it should create a replica locally or fetch the data from any replica server up in the hierarchy. A parent data center node accomplishes this by accumulating the results provided by its children. In reality, this is a bottom-up approach which begins from the lowest tier users and stops at tier-0. On the other hand, replica placement is done starting from the tier-0 and stopping at the lowest tier users (based on the previously calculated results). The details of the technique is discussed below.

We also define a cost vector to be used in calculating replication cost with respect to a particular data center node in the cloud hierarchy. Let v be a data center node and there is a sub-tree rooted at v. Now, let $\mathrm{Cost}(v, rd)$ be the cost for replication contributed by the above sub-tree where rd denotes the replica distance from v towards the root data center. So, if this distance is zero (i.e., the replica is in v itself) the replication cost will include the communication cost for all descendants of v, the consistency maintenance cost and the storage cost at v. Moreover, if this distance is greater than zero *(i.e.,* the replica is located at any data center sitting on the path from v towards the root*)*, the replication cost will include the communication cost for all the descendants of v only.

A. Calculation of Replica Cost and Location

Now, each data center's replication cost is calculated by considering the location of a replica anywhere in the path from the node itself towards the origin server or root. The replication cost, $Cost(v, rd)$, for each data center node v is calculated based on the condition that the replica is located at some distance towards the root. As mentioned below, the optimum position of a replica is also calculated for each case. Once the cost vectors for all the children of a data center are calculated, the cost vector for the data center itself is calculated.

Given data center v, leaf (user), when $rd = 0$, the data center contains a replica in itself and thus QoS is satisfied. $Cost(v, 0)$ is calculated as the sum of the cost of storage at v and the cost of update to maintain replica consistency ($CMC_{cost}(v) + S_{cost}(v)$). Replica location is set to v. When $rd > 0$ we have two scenarios regarding the user QoS requirement, $q(v)$. First, if $rd \leq q(v)$ it means that the replica server (data center) is located at a distance which meets the user QoS requirement. The cost of replication of sub-tree T_v (i.e., v only for this case) contains the read cost for v only. Second, if $rd > q(v)$, the QoS requirement is not satisfied by the replica and hence the communication cost is assigned to infinity. Now, the cost for $rd = 0$ is checked against the cost obtained with all possible distance, $rd \geq 1$, to identify the minimal cost for replication and location of replica. If for $rd = 0$ the cost ($Cost(v,0)$) is smaller than the cost for greater values of rd, $Cost(v,rd)$ is assigned to $Cost(v,0)$ and the location of the replica is set to the node itself. Otherwise, the replica will be created somewhere on the path towards the root and the location becomes rd-th ancestor of v. Replication cost will only be v's communication cost (CC). Accordingly, the minimal replication cost $Cost(v, rd)$ and the respective replica server location (RSL) for v using each distance possibility ($rd \geq 0$) can be determined as follows:

$$Cost(v, rd) = \begin{cases} Cost(v, 0), & if\ Cost(v, 0) \leq CC(v, rd) \\ CC(v, rd), & otherwise \end{cases} \quad (6)$$

$$RSL(v, rd) = \begin{cases} v, & if\ Cost(v, 0) \leq CC(v, rd) \\ node\ at\ distance\ of\ rd & otherwise \end{cases} \quad (7)$$

Lemma 1. RPCC optimally places replicas in the sub-tree $T_v = T\text{-}r$ where r is the origin server or root and v is the leaf of the targeted cloud tree T.

Proof. To determine the replica placement for a sub-tree T_v where v is a leaf data center node in the tree T, we consider two possibilities. First, when d (i.e., distance between the user and the root) = 0, T consists of only one node which is the root. This results in no communication, storage, or consistency maintenance cost for replicas and hence the optimality of the algorithm trivially holds true. Second, when $d \geq 1$(i.e., replica server is up on the way to the root) we need to compare the replication cost obtained for $d = 0$ (i.e., replica is in v itself) with the replication cost calculated for each value of d having $d \geq 1$ to decide on the optimal replication cost and locations of replicas. Now, if the cost for $d = 0$ (i.e., replica storage and update cost in v) is less than the cost for $d \geq 1$ (i.e., data access cost of v having a replica server up on the way to the root), placing a replica in v itself would be cheaper. On the contrary, the replica is placed at a higher data center node on the path to the root based on the value of d, which is optimal.

For a data center node v, non-leaf (i.e., non-user), when $rd = 0$, the node should contain a replica in itself. In this situation, the cost of replication, *rep_cost* will include the cost of all its children for rd=1, and the cost of storage and update for replica consistency at v (*rep_cost*

$= \sum_{j \in \mathrm{child(v)}} Cost(j, 1) + CMC_{cost}(v) + S_{cost}(\mathrm{v}))$

However, we have to check whether it is less expensive to make a replica in each child of v by calculating the sum of costs of all its children, $Cost_{child}(v) = \sum_{j \in \mathrm{child(v)}} Cost(j, 0)$ and comparing this with *rep_cost*. If $Cost_{child}(v)$ is less than *rep_cost* the replicas are placed in v's children and the replica location is set to "children". Otherwise, the replica is created locally at v. Thus, for optimal replication cost and replica server locations, the dynamic programming equations are:

$$Cost(v,0) = \begin{cases} Cost_{child}(v), if\ Cost_{child} < rep\_cost \\ rep\_cost, \qquad otherwise \end{cases} \quad (8)$$

$$RSL(v,rd) = \begin{cases} v, \qquad if\ Cost_{child} < rep\_cost \\ children, \qquad otherwise \end{cases} \quad (9)$$

To find *Cost*(*v*, *rd*) when *rd*>0, we determine the sum of replication costs of all the *v*'s children considering replica at a location of *rd*+1 ($child_{cost} = \sum_{j \in \text{child(v)}} Cost(j, rd+1)$). If *Cost*(*v*,0) is less than this sum, *Cost*(*v*, *rd*) is assigned to *Cost*(*v*,0) and the location of replica is assigned to the site obtained from RSL(*v*, 0*)*. On the contrary, the replica is created somewhere on the path towards the root and the location is *rd*-th ancestor of *v*.

$$Cost(v,rd) = \begin{cases} Cost(v,0), \qquad if\ Cost(v,0) < child\_cost \\ child\_cost, \qquad otherwise \end{cases} \quad (10)$$

$$RSL(v,rd) = \begin{cases} RSL(v,0), \qquad if\ Cost(v,0) < child\_cost \\ node\ at\ distance\ of\ rd, \qquad otherwise \end{cases} \quad (11)$$

Lemma 2. RPCC optimally places replicas in the sub-tree $T_v = T\text{-}r$ where *r* is the origin server or root and *v* is a non-leaf data center node of the targeted cloud tree *T*.

Proof. As stated earlier, when the children of a data center node complete the calculation of replication costs, the node itself starts to calculate the cost functions. In Lemma 1, we verified that replicas are allocated optimally in a leaf node for all replica distance values. Thus, we can infer that the non-leaf data center nodes one hop up from the bottom of *T* contain children (leaf nodes) whose calculated replication costs are optimal. Now, it remains to show that RPCC places replicas optimally in the sub-tree with a root being any internal data center, *v*, considering each possible value of *d*. First, when the value of *d* equals to 0, we observe by considering Equations (7) and (8) that the replication cost associated with sub-tree rooted at *v* is the least of the following two scenarios:

1) v itself contains a replica. The cost becomes the aggregate of the costs of the sub-trees having root as v's children and the value of d equals to 1 and moreover storage and consistency maintenance cost at v.

2) v does not contain a replica. The replication cost can be obtained by aggregating the costs incurred from the sub-trees having *v*'s children as roots with *d* = 0.

Second, when $d \geq 1$(i.e., replica server is up on the way to the root) we need to compare the replication cost obtained above for *d* = 0 (i.e., replica is in *v* itself) with the replication cost calculated for each value of *d* having $d \geq 1$ to decide on the optimal replication cost and locations of replicas. Now, if the cost for *d* = 0 is below the cost for $d \geq 1$ (i.e., communication cost of the sub-trees rooted at the children of *v* having a replica server up on the way to the root), placing a replica in *v* itself will be cheaper. Otherwise, it will be preferable to create a replica at any upper node on the path to the root based on the value of *d*, which is optimal. The minimum cost thus calculated is optimal for any node *v*.

Theorem 1. RPCC optimally allocates replicas for the targeted cloud Tree, *T* for a given traffic pattern.

Proof. We provide the proof for our generalized replication problem with the targeted cloud tree topology. The proof is carried out based on induction where Lemma 1 is the induction base and Lemma 2 is the induction step.

B. Placing Replicas

Placement of replicas starts at the root of the targeted cloud tree and stops at the lowest-tier users. In this process, every data center finds whether or not a new replica will be created locally based on the calculated cost and location vector. Once the bottom-up calculation is done, the root or the origin server holds the optimum cost for data replication. *Cost*(*root*, 0) of the whole cloud and the replica location *RSL*(*root*, 0). The value of *RSL*(*root*, 0) can be either *r* (the origin server itself) or "children" (the origin server's children nodes) which indicate that the replica is zero or one hop away (towards the users) from the origin server. So, the origin server sends a message *rd* = 0 or *rd* = -1. A data center node, *v*, which receives the message being one hop away increases the value of *rd* by one and investigates the valued of *RSL*(*v*, *rd*). If *v* becomes *RSL*(*v*, *rd*)'s value, a replica is created in *v* itself and the message *rd* = 0 is passed the children of *v*. If *RSL*(*v*, *rd*) value is -1 it assigns *rd* = -1 and forwards it to the children. Finally, if the *RSL*(*v*, *rd*) value is a data center node *rd* away up in the tree it sends *rd* message to its children without changing the value. When all the users at the leaves receive the message *rd* the replica placement process terminates.

C. Complexity Analysis

Analysis of the computational and message complexity of our algorithm is completed by performing the computation of cost and location vectors and the placement of replicas in the entire cloud. For every data center node *v*, we compute its *Cost*(*v*, *rd*) for its sub-tree for each value of *rd* between 0 and distance to the origin server or root by merging the results of all its children. The computation of data center *v* is done by adding |*child*(*v*)| rudiments of count (*x* + 1), where *x* represents the distance between *v* and the origin data center and |*child*(*v*)| represents the children count for data center *v*. Hence, the number of computations for data center *v* appears as (*x* + 1). |*child*(*v*)|. The total number of computations for all the data centers in the cloud is:

$$\sum_{v \in V} (x_v + 1).|child(v)|$$

where x_v represents the distance between *v* and the origin server.

Given, the number of data centers, |*V*| = *N*, we can observe that $x_v \leq N - 1$ for each value of *v*. Hence, the following can be deduced:

$\sum_{v \in V}(x_v + 1).|child(v)| \leq$ N.$\sum_{v \in V}|child(v)|$ = N (N-1)

This equality holds since there are $N-1$ children nodes in the cloud. This is because except the origin server, each datacenter node has a parent. Consequently, $O(N^2)$ is the overall computing cost of *Cost* vectors for all the nodes. As for the message, the first is the cost vector that is sent by each node to its parents. The other message is *rd* sent by each node to its children. If $|V| = N$ is the total number of data center nodes and each node sends two messages, the message complexity is $O(N)$.

V. Simulation Setup

For the performance evaluation of our proposed replication algorithm we have leveraged the use of a Java based simulator program. Our hierarchical cloud structure consists of four tiers having each tier with data centers. Each data center has five children and thus the total number of data centers becomes 155 including the users in the lowest tier. Requests for data come from the users only. Uniform distribution is used to model the available link bandwidth with the range [0.622, 2.5] (Gbps). According to the same distribution, data center storage capacities are also modeled. The simulation experiments use 2500 data files where the size of each data file is 10 GB. Hence, the total data size becomes nearly twenty-five tera byte (TB). To measure the efficacy of our system, we utilized five diverse storage settings of data centers which are created in accordance to the relative storage capacity of data centers. The relative storage capacity (RSC) is determined by a percentage of total storage size of all data centers compared to the overall data size in the system. For our experiments, we use RSCs ranging from 13% to 75%. The use of relative storage capacity is justified by the fact that it affects the decision to create replicas on data centers in contrast to their absolute storage capacities. Different storage settings as discussed above are shown in Fig. 3.

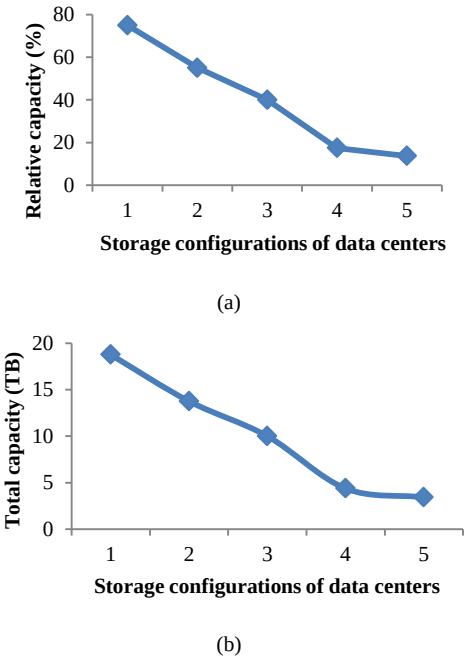

Fig. 3. Storage Configurations of Data Centers based on Relative (a) and Total (b) Capacity.

Upon requests for data from the users, each data center replica server tries to meet the request. However, the number of access requests served by each replica server is limited by its workload constraint. In our experiments, six different workload configurations according uniform distribution are used as shown in Fig. 4. Simulation experiments are done by submitting 50 different jobs each one having a fixed probability of being submitted.

The simulation experiment allows each job to access a sequence of data files. User requests for data files come according to Poisson distribution and each request is issued in an interval of 2500 milliseconds. Moreover, selected access patterns determine the sequence of files that will be accessed. In our experiment, two data access patterns namely Gaussian and Zipf distributions are used. The Zipf distribution is expressed as $Pi = K$, where Pi denotes the count for ith ranked file, K represents the most popular data file (by means of frequently accessed items) and s specifies the distribution shape. Also, the temporal locality present in the data access pattern is measured by this parameter s having values ranging from 0.65 to 1.24. The level of locality in the data is indicated by the value of s. We have experimented with a value of 0.85 for s and call it as Zipf-0.85 distribution. Besides, Gaussian distribution also known as normal distribution is used in our experiment. It is an important distribution in statistics and is frequently used in natural and social sciences to characterize real-valued random variables. Previously, other replication techniques [36], [37] in the literature have used similar data access patterns for their evaluation in data grids.

Our replication strategy was evaluated using the performance metrics which include job execution time, mean bandwidth use, storage utilization, number of replicas created, and rate of satisfaction for users. Job execution time refers to the overall time needed to execute the whole set of jobs and also takes into account the data access time. replica maintenance algorithm. The bandwidth usage for a data transfer is the data size times the aggregate costs of the data transfer route. The average cost for bandwidth usage is calculated by dividing the overall bandwidth usage by total data access counts. Storage consumption is the proportion of the data center storage occupied by the replicas in the system. Finally, user satisfaction rate represents number of users whose QoS constraints are met compared to the total number of users who requested for data access with some QoS constraints. The target is to reduce total job execution time and minimize average bandwidth and storage consumption while maximizing the user satisfaction rate.

Config.	Workload capacity constraint (GB)
1	[250-350]
2	[200-300]
3	[150-250]
4	[100-200]
5	[80-300]
6	[50-150]

Fig. 4. Workload Configuration of Data Centers.

VI. Performance Results

In this section, we discuss the experimental results of our proposed replication technique (RPCC) and compare it with Greedy Add and Greedy Remove [37] protocols from the literature. The QoS requirement of a user issuing a data access request is specified by a range in terms of distance from the user to the closest replica data center. This range is formulated using a uniform distribution. For instance, if a user specifies its QoS requirement of [1-2] it means that the replica data center containing the requested data is expected to be one or two hops away from the user.

A. Job Execution Time

Fig. 5 shows the job execution times based on different data center workload configurations for RPCC, Greedy Add, and Greedy Remove algorithms. In the experiment, the relative storage capacity is set to 75%, user QoS requirements of [1-3] and [0-1] are specified from a uniform distribution to permit both relaxed and relatively more constrained distance ranges respectively. For both the data access patterns (Zipf-0.85 and Gaussian), the total job execution times required by RPCC is shown to be lower than the other two algorithms. This is attributed to the fact that RPCC creates a moderate number of replicas in appropriate locations in the cloud hierarchy in contrast to Greedy Add and Greedy Remove techniques which in turn reduces access times for the data requests. Accordingly, this cuts down the overall job execution time. Fig. 6 shows the number of replicas created by all three algorithms during a sampling simulation period for the same workload and storage resource configurations. With the decrease in workload capacity of replica servers, an increased number of replicas are created (Fig. 6) and consequently job execution times drop in most cases but by varying amounts as shown in Fig. 6. Greedy Add mostly exhibits lower execution time than Greedy Remove. For relatively more constrained QoS requirement ([0-1]) and Gaussian access pattern, Greedy Remove unexpectedly shows lower job execution time compared to the other two algorithms.

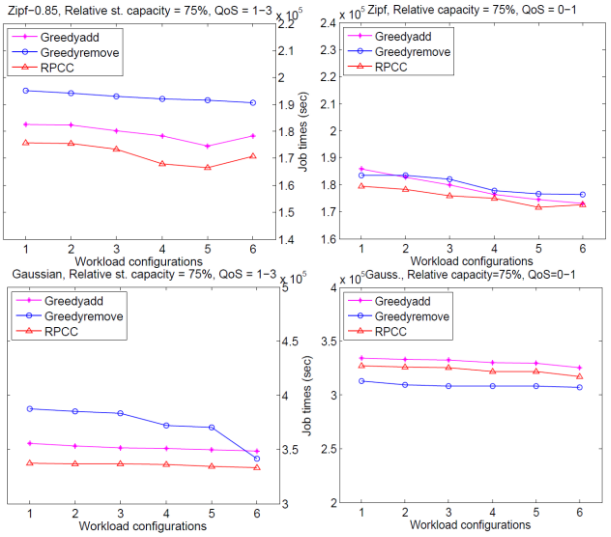

Fig. 5. Comparison of Job Times with 75% Relative Storage Capacity of Data Centers.

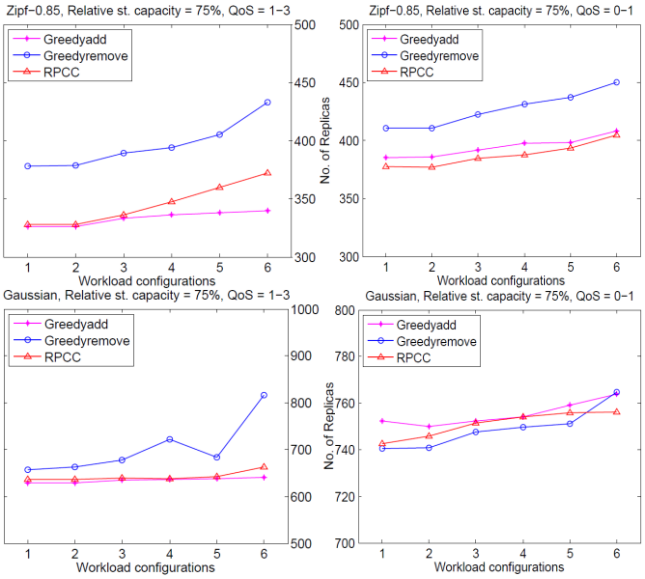

Fig. 6. Comparison of No. of Replicas for Varying user QoS Requirements.

Generally, the performance gain of RPCC over Greedy Add and Greedy Remove turns out to be more evident when user QoS constraints of broader ranges are used.

The job execution times using relative storage capacity of 17.5% are shown in Fig. 7 for the same data access patterns as before. Generally, RPCC exhibits shorter job times compared to Greedy Add and Greedy Remove. However, the use of more constrained storage size results in only a meager benefit for RPCC in terms of job times in most cases. Furthermore, job times for all three algorithms in this case got an increase by varying amount compared to the case when relative storage capacity of 75% is used.

Fig. 8 shows satisfaction rates of users for all methods using relative storage capacity of both 17.5% and 75%. Mostly, RPCC outperforms the other two algorithms. Particularly, the performance gain of RPCC over Greedy Add and Greedy Remove is more when the storage capacity of replica data centers (17.5% relative capacity) is restricted. Nevertheless, satisfaction rates of users decrease in case of constrained storage space of replica data centers regardless of quality requirements from users and the patterns used for data access as shown in Fig. 8.

B. Average Bandwidth Use

Both network providers and end-users deem bandwidth consumption as a key issue since undue bandwidth use can cause slowdowns due to network congestion. We include two different types of costs namely replication (create and update) cost and read cost to measure average bandwidth use.

Fig. 9 displays the average bandwidth cost in terms of varying workload for RPCC, Greedy Add, and Greedy Remove algorithms. As before, the relative storage capacity is set to 75% and 17.5%, user QoS constraints on replica server distances of [1-3] and [0-1] are specified from a uniform distribution to allow both relaxed and relatively more constrained ranges respectively. RPCC mostly shows moderate bandwidth consumption rate compared to Greedy

Add and Greedy Remove algorithms. The reason is that both data access (read) cost and replication cost are reduced on account of the placement of a modest number of well-placed replicas down the cloud hierarchy. On the other hand, Greedy Remove mostly exhibits the lowest bandwidth cost among the three algorithms. The reason is that more replicas are created in the upper part of the cloud hierarchy. The replication cost involved in this case is comparatively lower than the elevated data access (communication) cost. With the decrease in workload capacity of data center servers, a higher number of replicas are created and consequently the bandwidth cost increases for all three algorithms. For more constrained relative storage capacity (17.5%) of data centers, RPCC exhibits moderate bandwidth consumption compared to Greedy Add and Greedy Remove as before. Greedy Remove performs better than the other algorithms due to reduced replication cost with Gaussian access pattern irrespective of the user QoS ranges.

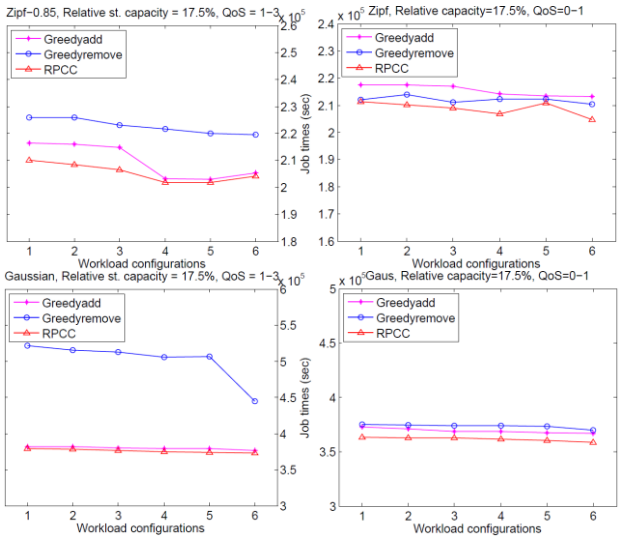

Fig. 7. Job Times with Relatively More Constrained Data Center Storage Capacity (17.5%).

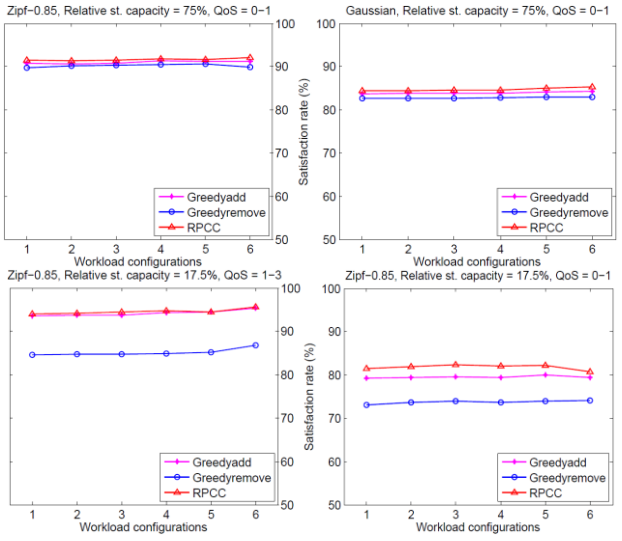

Fig. 8. Comparison of Satisfaction Rates for Varying Relative Storage Capacity of Data Centers.

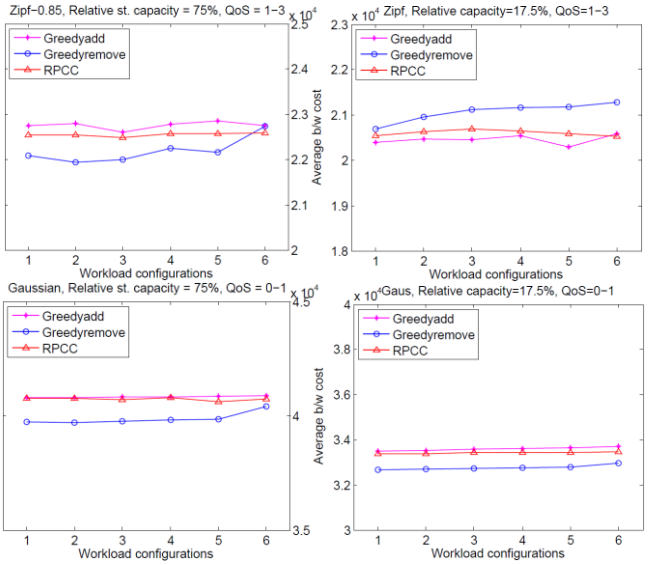

Fig. 9. Comparison of Average b/w Cost with Relative st. Capacity of 75% and 17.5%.

C. Storage Use

The storage resources used in the system is vital to grid providers. Since storages are relatively cheaper, we can come to a trade-off in case improvements in job execution times and network bandwidth consumption are achieved.

Fig. 10 shows the storage usage (y-axis) as a function of varying workload (x-axis) for all algorithms with a relative storage capacity of 75% and 17.5%. RPCC shows moderate storage usage compared to Greedy Add and Greedy Remove algorithms in all cases. When data centers' capacities in terms of workload decreases, the number of replicas created increases (Fig. 6) and accordingly storage overhead increases in most cases but by varying amounts.

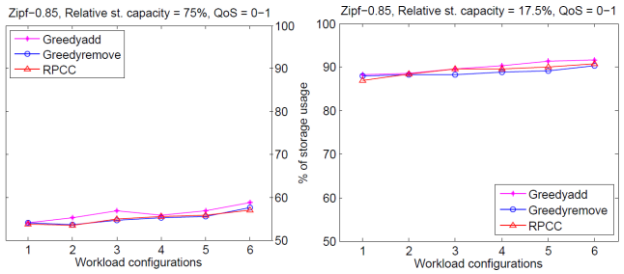

Fig. 10. Comparison of Storage Cost with More Constrained user QoS Range [0-1].

VII. Conclusions and Future Work

In this paper, we studied the data replication problem in data cloud considering the QoS requirements from users to support big data applications. Aiming to put forward a multi-objective solution to the replication problem, user QoS constraints in terms of distance to replica data center servers and workload constraints of replica servers are considered. First, we formulate the replica placement problem as a dynamic programming problem. Second, we propose a novel distributed replica placement algorithm (RPCC) for a multi-tier cloud platform so as to avoid the limitations usually found

in centralized algorithms such as scalability, reliability, and performance bottlenecks. Performance analysis of the proposed algorithm was done in terms of job execution time, mean bandwidth usage, storage resource utilization, total number of replicas that are created during a simulation period, and satisfaction rates of users. The simulation results showed that RPCC can considerably reduce job execution times which include data access time while incurring modest bandwidth and storage costs compared to two other algorithms. These results are obtained by utilizing a variety of storage and workload setting of data center servers and data access patterns with a degree of temporal locality and randomness.

In the future, we envision to implement our proposed data replication technique in a physical cloud infrastructure. Besides, we plan to extend our replication technique to deal with the bandwidth constraints imposed on the network links.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," Technical Report UCB/EECS-2009-28, Dept. of EECS, California Univ., Berkeley, Feb. 2009.

[2] J. M.D. Dikaiakos, D. Katsaros, P. Mehra, G. Pallis, and A. Vakali, "Cloud Computing: Distributed Internet Computing for IT and Scientific Research," IEEE Internet Computing, vol. 13, no. 5, pp. 10-13, Sept. 2009.

[3] I.S. R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the fifth Utility," Future Generation Computer Systems, vol. 25, no. 6, pp. 599-616, June 2009.

[4] Amazon. Amazon simple storage service (Amazon S3). Available: http://aws.amazon.com/s3

[5] Ghemawat, S., H. Gobioff, and S.-T. Leung, The Google file system. SIGOPS Oper. Syst. Rev., 2003. 37(5): p. 29-43.

[6] Hadoop at Twitter, http://www.slideshare.net/kevinweil/hadoop-at-twitter-hadoop-summit-201.

[7] Bessani, A., Correia, M., Quaresma, B., Andr'e, F. and Sousa, P., DepSky: Dependable and Secure Storage in a Cloud-of-clouds, In Proc. of the 6th Conference on Computer Systems, pp. 31-46, 2011.

[8] F. Wang, J. Qiu, J. Yang, B. Dong, X. Li, and Y. Li, "Hadoop High Availability through Metadata Replication," Proc. First Int'l Workshop Cloud Data Manage, pp. 37-44, 2009.

[9] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop Distributed File System," Proc. IEEE 26th Symp. Mass Storage Systems and Technologies (MSST), pp. 1-10, June 2010.

[10] A. Gao and L. Diao, "Lazy Update Propagation for Data Replication in Cloud Computing," Proc. Fifth Int'l Conf. Pervasive Computing and Applications (ICPCA), pp. 250-254, Dec. 2010.

[11] W. Li, Y. Yang, J. Chen, and D. Yuan, "A Cost-Effective Mechanism for Cloud Data Reliability Management Based on Proactive Replica Checking," Proc. IEEE/ACM 12th Int'l Symp. Cluster, Cloud and Grid Computing (CCGrid), pp. 564-571, May 2012.

[12] C.Q. Huang, Y. Li, H.Y. Wu, Y. Tang, X. Luo, Modeling and maintaining the reliability of data replica service in cloud storage system, Journal of Communication. 36 (3), 2014.

[13] H. Shen, G.P. Liu, H. Chandler, Swarm intelligence based file replication and consistency maintenance in structured P2P file sharing systems, IEEE Trans.Computing, 64 (10), pp. 2953–2967, 2015.

[14] G. Liu, H. Shen, H. Chandler, Selective data replication for online social networks with distributed datacenters, IEEE Trans. Parallel Distrib. Syst. 27 (8), pp. 2377–2393. 2016.

[15] S. Fu, L. He, X. Liao and C. Huang, Developing the Cloud-integrated data replication framework indecentralized online social networks, Journal of Computer and System Sciences 82 (2016) 113–129, 2015.

[16] S. Sun, W. Yao and X. Li, DARS: A dynamic adaptive replica strategy under high load Cloud-P2P, Future Generation Computer Systems, 78 (2018) 31–40, Aug 2017.

[17] Sun D.W, Chang G.R. and Gao S.,"Modeling a dynamic data replication strategy to increase system availability in cloud computing environments," *Journal of Computer Science and Technology*, vol. 27, no.2, pp. 256-272 Mar. 2012.

[18] D.-W. Sun, G.-R. Chang, C. Miao, L.-Z. Jin, X.-W. Wang, Analyzing modeling and evaluating dynamic adaptive fault tolerance strategies n cloud computing environments, J. of Supercomputing, 66 (2013), 193–228, 2013.

[19] Wei, Q., Veeravalli B., Bozhao G., Lingfang Z. and Dan F., CDRM: A cost-effective dynamic replication management scheme for cloud storage cluster., in 2010 IEEE International on Cluster Computing. 2010. p. 188 – 196

[20] Y. Qu, N. Xiong, RFH: A resilient, fault-tolerant and high-efficient replication algorithm for distributed storage, in: 41st Int. Conf. Parallel Process., 2012, pp. 520–529.

[21] N. Gill and S. Singh. A dynamic, cost-aware, optimized data replication strategy for heterogeneous cloud data centers. Future Generation Computer Systems, 65 (2016) 10–32, 2016.

[22] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Y. Zomaya, Energy-Efficient Data Replication in Cloud Computing Datacenters, In Globecom 2013 Workshop- Cloud computing systems, networks, and applications, pp. 445-450, 2013.

[23] Long Sai-Qin, Zhao Yue-Long, Chen Wei. MORM: A Multi-objective Optimized replication Management strategy for cloud storage cluster. Journal of Systems Architecture, 60 (2):234–44, 2014.

[24] N. Mansouri, M. K. Rafsanjani, M.M. Javidi, DPRS: A dynamic popularity aware replication strategy with parallel download scheme in cloud environments, Simulation Modelling Practice and Theory 77 (2017) 177–196, 2017.

[25] H. Sun, B. Xiao, X. Wang and X. Liu, Adaptive trade-off between consistency and performance in data replication, Softw. Pract. Exper. 47:891–906, 2017

[26] L. Chen and D. B. Hoang, Adaptive data replicas management based on active data-centric framework in cloud environment, In Proceedings of IEEE International Conference on High Performance Computing and Communications, pp. 101-108, 2013.

[27] Z. Ye, S. Li and J. Zhou, A Two-layer Geo-cloud based Dynamic Replica Creation Strategy, Applied Mathematics & Information Sciences, Vol. 8, No. 1, pp. 431-440, 2014.

[28] J. Janpet and Y. Wen, Reliable and Available Data Replication Planning for Cloud Storage, In Proc. of the IEEE 27th International Conference on Advanced Information Networking and Applications, pp. 772-779, 2013.

[29] S. Kirubakaran, S. Valarmathy and C. Kamalanathan, Data Replication Using Modified D2RS in Cloud Computing for Performance Improvement, Journal of Theoretical and Applied Information Technology, Vol. 58, No.2, pp. 460-470, 2013.

[30] Li, L.L. , Wang, Q.C., Han, Y.J., Ma, X.H., Jiao, Y., A dynamic replication algorithm based on access popularity in cloud storage environment, In Proc. of International Conference on Materials Science and Computational Engineering, 2014.

[31] N. Mansouri, Adaptive data replication strategy in cloud computing for performance improvement, Front. Comput. Sci., 10(5): 925–935, 2016.

[32] J. Lin, C. Chen and J. M. Chang, QoS-Aware Data Replication for Data-Intensive Applications in Cloud Computing Systems, IEEE Transactions On Cloud Computing, Vol. 1, No. 1, pp. 101-115, 2013.

[33] S. Varma and G. Khatri, QoS-Aware Data Replication in Hadoop Distributed File System, Int. J. Advanced Networking and Applications 7(3), pp. 2741-2751, 2015.

[34] Kumar, P.J. and Ilango, P. MQRC: QoS aware multimedia data replication in cloud. International Journal of Biomedical Engineering and Technology, Vol 25, 2017.

[35] M. Shorfuzzaman, "On the Dynamic Maintenance of Data Replicas based on Access Patterns in A Multi-Cloud Environment", International

Journal of Advanced Computer Science & Applications (IJACSA), Vol.8. No. 3, pp. 207-215, 2017.

[36] A.M. Soosai, A. Abdullah, M. Othman, R. Latip, M.N. Sulaiman, and H. Ibrahim, "Dynamic Replica Replacement Strategy in Data Grid," Proc. Eighth Int'l Conf. Computing Technology and Information Management (ICCM), pp. 578-584, Apr. 2012.

[37] C. Cheng, J. Wu, and P. Liu, "Qos-aware, access-efficient, and storageefficient replica placement in grid environments," Journal of Supercomputing, vol. 49, no. 1, pp. 42–63, 2009.

[38] V. Andronikou, K. Mamouras, K. Tserpes, D. Kyriazis, T. Varvarigou, Dynamic QoS-aware data replication in grid environments based on data ''importance'', Future Gener. Comput. Syst. 28 (2012) 544–553, 2012.