Ensuring performance and provider profit through data replication in cloud systems

Uras Tos, Riad Mokadem, Abdelkader Hameurlain, Tolga Ayav, Sebnem Bora

Ensuring performance and provider profit through data replication in cloud systems

Uras Tos[1,2,3] · Riad Mokadem[1] · Abdelkader Hameurlain[1] · Tolga Ayav[2] · Sebnem Bora[3]

Abstract
Cloud computing is a relatively recent computing paradigm that is often the answer for dealing with large amounts of data. Tenants expect the cloud providers to keep supplying an agreed upon quality of service, while cloud providers aim to increase profits as it is a key ingredient of any economic enterprise. In this paper, we propose a data replication strategy for cloud systems that satisfies the response time objective for executing queries while simultaneously enables the provider to return a profit from each execution. The proposed strategy estimates the response time of the queries and performs data replication in a way that the execution of any particular query is still estimated to be profitable for the provider. We show with simulations that how the proposed strategy fulfills these two criteria.

Keywords Cloud computing · Data replication · Performance · Economic benefit

1 Introduction

In the last decade cloud computing has been in the spotlight both as a research area and the implementations in the industry. Cloud providers are able to provide vast computational resources for an affordable sum by delivering abstracted resources that is provided by relatively cheap hardware. Providers rent these shared resources to the tenants in an economy based-manner [9], with the ultimate aim of returning profit. As the load generated by tenant applications change, cloud resources assigned to the tenants can be elastically adjusted. Using a pay-as-you-go pricing model [3], tenants are billed for only the portion of the cloud resources that are consumed by them. While the providers pursue profit, tenants expect the providers to deliver a quality of the service that will serve the tenants' needs. This economic relationship between the provider and the tenant is clearly defined in the *service level agreement* (SLA) [32], which is a legally binding contract between the two parties. Among the terms of this contract, an agreed upon set of *service level objectives* (SLO) are included to define the quality of the service. Breach of the SLA terms often result in some consequences for the provider, including monetary compensation as a penalty.

Increased data availability and improved level of performance are some important issues that exist in the cloud systems. Data replication has been around for decades that addresses these issues as well as providing other benefits. Data replication has been studied in many traditional systems, including (i) database management systems (DBMS) [14], (ii) parallel and distributed systems [23], (iii) mobile systems [11] and (iv) other large-scale systems including P2P [31] and data grid systems [35].

A common way of replicating data in the traditional systems is creating as many replicas as possible to attain maximum resource utilization to provide best performance. In cloud systems, such a data replication strategy may not

✉ Uras Tos
urastos@gmail.com; uras.tos@irit.fr

Riad Mokadem
riad.mokadem@irit.fr

Abdelkader Hameurlain
abdelkader.hameurlain@irit.fr

Tolga Ayav
tolgaayav@iyte.edu.tr

Sebnem Bora
sebnem.bora@ege.edu.tr

[1] Institut de Recherche en Informatique de Toulouse (IRIT), Paul Sabatier University, 118 Route de Narbonne, 31062 Toulouse, France

[2] Department of Computer Engineering, Izmir Institute of Technology, 35430 Urla, Izmir, Turkey

[3] Department of Computer Engineering, Ege University, 35100 Bornova, Izmir, Turkey

be economically beneficial for the provider, since creation of a large number of replicas can result in a wasteful resource utilization, degraded performance and reduced profit. Therefore, dealing with data replication, three major questions of what to replicate, when to replicate, and where to replicate [26] must be answered in such a way to satisfy the performance in an economically feasible way [36].

In the literature, current efforts on data replication in cloud systems are mainly interested in satisfying the objective of a given level of availability for the tenants [29,33]. On the other hand, satisfying performance objectives has been comparatively a less visited issue with only a handful of studies being particularly interested in improved performance [4,12,37,39]. This may be particularly attributed to the lack of performance guarantees being a de facto part of the SLA until recently [16,28]. Moreover, even fewer of the proposed strategies consider the economic impact of data replication on the cloud provider [7,10].

We propose *PErformance and Profit oriented data Replication strategy, version 2* (PEPRv2) that aims the simultaneous satisfaction of both the SLA terms, e.g. availability and performance and ensuring the economic profit of the cloud provider. We treat the response time as the main performance metric, which is also included in the SLA. PEPRv2 estimates the response time of the queries when they arrive at the cloud. If the estimated response time is greater than the SLO response time threshold, profitability of that particular query is then estimated. The profit estimation includes any possible replication event that will be necessary during execution. Moreover, PEPRv2 also has the ability of dealing with a group of queries that are executed in parallel to achieve a task. Replication is carried out only when both the response time and economic benefit of the provider are estimated to be satisfied. Placement of the new replicas are also performed considering these two criteria. The number of replicas is dynamically adjusted as the SLOs are satisfied over time. Furthermore, a minimum number of replicas are always kept to ensure a minimum given availability [37].

Predecessor of PEPRv2 has been introduced in our earlier paper [36]. While the two strategies bear some similarities, PEPRv2 is proposed with several novel contributions including (i) a response time estimation model that regards both the computational load and data intensiveness of queries separately, (ii) consideration of distributed execution of queries on any number of virtual machines, and in turn inclusion of this aspect in the proposed replication decision algorithms, (iii) a detailed economic model that estimates the monetary correspondances of the cloud resources expected to be consumed by each query (iii) a realistic performance evaluation study performed on a heterogeneous cloud environment that includes a comparison with another data replication strategy for cloud systems, as well as the predecessor of PEPRv2.

The performance evaluation study aims to show the importance of data replication in satisfying response time and to demonstrate how PEPRv2 not only satisfies the SLA, but also ensures profitability of the cloud provider. In a simulation environment, we design a scenario that compares the performance of PEPRv2 against another data replication strategy for cloud systems, as well as the original PEPR strategy.

The organization of the rest of the paper is as follows. Section 2 provides a summary of related work on data replication strategies in the cloud. Section 3 gives some background information and describes the cloud environment in which PEPRv2 operates. Sections 4 and 5 describe the details of PEPRv2 and the implemented economic cost model, respectively. Section 6 evaluates the performance of PEPRv2 in a simulation study. Section 7 concludes the paper and discusses some possible future studies.

2 Related work

In data replication context, several existing strategies already deal with some of the issues addressed in this paper. Most notably, the consideration of the economic impact of data replication is particularly interesting for our work. This section briefly discusses the strategies according to whether they take the economic aspect of data replication into account.

2.1 Data replication without considering economic impact

Cost-effective dynamic replication management (CDRM) strategy [37] highlights that having too many replicas does not increase availability but instead results in diminished returns. CDRM calculates and maintains a minimum number of replicas to satisfy a given level of availability. In the considered cloud storage, data sets are fragmented into several blocks. Authors calculate the availability of data sets depending on the availability of each of their fragments. With this information, CDRM calculates a required number of replicas for each fragment. Placement of replicas is performed in such a way that balances the load on all sites. Reducing access skew ensures that all of the fragments are served without causing a bottleneck. As a consequence of balanced load, the authors also observe increased access performance. Investigating the relationship between the number of replicas and the level of availability has been the focus of other studies as well [22]. Another approach [33] that is very similar to CDRM also deals with finding minimum number of replicas from a data popularity point of view in cloud storage clusters. In this work, not all data but only a subset of files which exceed an access threshold are considered for replication.

Increasing data locality, more specifically spatial locality through data replication yields increased performance with

strategically placing replicas closer to the requestor sites. Even though some strategies do not directly consider performance as an objective, increased performance is observed as a consequence of data replication [17]. Another form of increasing locality of data for improved performance is focusing on temporal locality. In this manner, some strategies argue that frequently accessed data sets will likely stay popular in the future [13] and decide what to replicate according to the mentioned popularity assessment. This is a simple but effective strategy for satisfying performance guarantees as evidenced by RTRM strategy [4]. When accessing popular data is causing a higher-than-desired average response time, these data sets are replicated to resolve the performance issue. What is interesting about RTRM is that it replicates not all but just popular data in order to conserve resource usage.

Some other strategies deal with availability as a reliability issue regarding the cloud sites [18,19]. While availability is a frequently considered objective, what sets these two strategies apart is their consideration of cost-effectiveness in satisfying availability. By determining the duration to keep replicas according to a reliability metric, these strategies can optimize the number of replicas for each file, hence saving provider costs on storage resource.

2.2 Data replication with considering economic impact

Modeling a data placement strategy to improve availability can be also be achieved through custom economic cost models. Some present an auction model to implement a replica placement policy in a large-scale cloud storage environment [39]. If the desired availability level cannot be maintained, a bidding is held to determine some placement for a new replica. Bidding price is dependent on several properties of the nodes including failure probability, network bandwidth and available space. Some performance increase is also observed as a consequential benefit of data replication. Others deal with a similar problem in a multi-cloud context [1], where tenants receive services from multiple cloud providers. These examples use a custom economic model that is tailored to take into account the heterogeneity caused by resource and pricing variations among multiple cloud providers. Similarly to existing strategies, this economic model is used in order to minimize a cost model [21].

In another similar effort, Skute [7] is introduced as a cost-efficient data replication strategy based on a virtual economy that takes into account marginal utility, storage usage and query load. Virtual nodes act autonomously and they periodically announce their rent to other nodes. Moreover, nodes also accumulate wealth by answering queries and spend this wealth on other nodes to store replicas on them, according to their rent. Skute self organizes replica configuration among the virtual nodes to minimize communication cost while maximizing economic benefit. Even though the strategies of this type utilize some form virtual economy, it is not an actual monetary cost model of the provider–tenant relationship [6].

Increasing availability by replicating popular data closer to the locations that originate most amount of requests is also a frequently researched problem. Some strategies even propose mechanisms to predict future accesses based on historical access records to preemptively replicate data to meet the increased demand [27]. Moreover, some other strategies deal with data popularity that peaks for a short amount of time, and then tapers off [25]. In order to deal with the quick burst of popularity, these data replication strategies quickly respond and change the replica configuration accordingly. Increasing data locality based on popularity also decreases network consumption [20].

Tos et al. [36] deals with some of the concepts that are proposed in this paper. Proposed work deals with performing data replication in the cloud with respect to the performance metric in the SLA as well as estimating the provider profit in each billing period. Response time estimation for tenant queries are calculated when the queries are arrive at the cloud. If the estimation indicate that a desirable performance cannot be satisfied, data replication is performed, but only when it is economically feasible for the provider. However, in the mentioned work, inter-query parallelism is not considered. Moreover, the performance evaluation considers a comparison with a strategy that is targeted at the data grid systems.

3 Background

While cloud providers offer numerous services to the public, they are also business entities that operate for generating profit. Regulated by the SLA, the relationship between provider and tenant revolves around the quality of the service and monetary cost of these services. Providers take advantage of cheaper hardware, resource sharing, and other factors to maximize profit and offer elastically scalable resources simultaneously in a large scale.

Providing computing resources in a global scale with a realistic quality of service pose significant challenges for the providers. Often times providers establish facilities in multiple geographical regions to achieve this goal. By having facilities in various regions, providers aim to handle queries from tenants with a large diversity of location. Evidently, having facilities in a number of different regions may also result in operating costs of the provider to vary from region to region, depending on the resources consumed (e.g. power costs). This introduces a heterogeneous service supply by the provider. In a particular region, the bandwidth may be cheaper than the storage, while the inverse might be true for another region. This is another issue that should be taken

into account by any data management strategy operating in the cloud.

The heterogeneity is not only present for the costs of the provider, but also the performance of the cloud services as well. As an example of this phenomenon, all regions of the cloud are interconnected via network links. It can be expected that, while the network links are more abundant and less expensive inside the subregions; bandwidth is less abundant and more expensive upper in the hierarchy, i.e. between different geographical regions [24]. Inside the subregions, network links are dedicated to the provider's use. On the other hand, usually the Internet infrastructure is used between different geographical regions, therefore bandwidth is more precious in that case.

Tenants consume the services they obtained from the provider by submitting queries to the cloud in order to process their data. The data to be processed is typically globally distributed in the cloud. Queries may require one or more data sets during execution. Considering the network capabilities between regions and load variations in the nodes that will execute the queries, meeting response time objective is non-trivial for the provider.

Breach of SLA is handled relatively straightforward from the provider's perspective. For example, in a case when the tenant does not pay the rent, the provider can simply stop providing the services. On the other hand, it is more difficult to keep track of SLA breaches from the tenant's perspective. Usually the provider monitors the quality of the service, and when the service quality falls below a threshold, provider pays a monetary sum to the tenant [16]. Obviously this approach involves a certain amount of trust between the parties.

4 Data replication strategy

Data replication involves several operation steps to be carried out in a meaningful way to achieve desired benefits. These steps are finding *what* data to be replicated, *when* the replication event should be carried out, *how many* replicas to create and *where* to place these replicas [26]. As a result, PEPRv2 makes each of these decisions to determine a near-optimal replica configuration. Instead of gathering popularity data and performing a periodical replica reconfiguration, PEPRv2 performs the necessary replication tasks before executing the queries. Hence, what to replicate is always the data associated with each query being executed at a particular time.

4.1 When to replicate

In PEPRv2, we follow an on-arrival decision approach for whether to replicate. That is, when a query arrives at the cloud, all replication decisions are made before executing that query. However, going through with the decision to replicate depends on the satisfaction of two criteria. A replication event occurs if and only if it will satisfy the response time guarantee and it is a profitable course of action for the provider.

Initially a query $\mathcal{Q}$ is submitted to the cloud that contains a number of nodes $\{n_1, n_2, \ldots, n_i\}$. Assuming that $\mathcal{Q}$ is delegated to node n_i that will execute it, in the first step a response time estimation for that particular query $\mathcal{T}_{\mathcal{Q},n_i}$ is calculated. If the estimated $\mathcal{T}_{\mathcal{Q},n_i}$ is determined to be greater than the service guarantee given to the tenant ($\mathcal{T}_{SLO}$), it would pose a breach of contract and cause the provider to pay a penalty to the tenant. Therefore, provider should perform data replication beforehand to prevent this problem.

The response time estimation is the mechanism that ensures provider to decide whether to replicate or not to meet tenant requirements. However, response estimation is not enough on its own to carry on with the replication. Economic benefit of the provider must also be ensured for the provider to keep the business afloat and continue serving the tenants.

Algorithm 1 Replication decision algorithm.

```
1: T_{Q,n_i} ← estimated response time of executing Q on node n_i
2: if Availability(Q, d_j) < MinAvailability_SLO then
3:     PlaceReplica()
4: end if
5: if T_{Q,n_i} > T_SLO then
6:     p ← estimated profit generated by the execution of Q
7:     if p > 0 then
8:         PlaceReplica()
9:     end if
10: end if
```

Executing query $\mathcal{Q}$ costs some monetary sum for the provider. Placement of some replica that is associated with $\mathcal{Q}$ on a node n_j that may or may not be the same node as n_i is included in this cost. Ideally, provider wants to return some profit from the execution of each query. Therefore a profit estimation is also performed before the execution to evaluate the profitability of the potential replication decisions. Replication is performed only if the result of these two estimations are acceptable as described in Algorithm 1. Of course, replication decision algorithm also checks the availability of the data associated with $\mathcal{Q}$ and replicates the concerned data set if the availability is below the threshold defined in the SLA.

4.1.1 Response time estimation

When a query $\mathcal{Q}$ is submitted to the cloud for processing, an available node n_j takes it up for execution. Processing of $\mathcal{Q}$ requires data sets of $\{d_1, d_2, \ldots, d_j\}$ during execution. Some of this data may already be present on n_i or it may be necessary for them to be read from remote nodes $\{n_1, n_2, \ldots, n_k\}$.

Some queries may read data sets only once while other queries may require data sets to be accessed many times, depending on the data intensiveness of any particular query. This variations between queries in terms of data intensiveness is included in the response time estimation with the factor of β. During execution, the portion of data that resides on remote nodes would be required to transferred to the executing node n_i. This transfer took place over the I/O throughput capability t_{n_k} of the remote node n_k. Moreover, network link between n_i and n_k is also expected to be consumed. Since the network links other than the ones inside a particular subregion usually established over the Internet, they are heterogeneous. Therefore, our estimation model takes into this account by using the expected available bandwidth between the nodes at the time of execution.

Total data involved with the $\mathcal{Q}$ is expected to be processed on n_i by consuming local I/O throughput of t_{n_i}. Of course, there may already be some other query load on n_i that may slow down the execution of $\mathcal{Q}$. This situation is handled by taken l_{n_i} into account, which is the expected query load on n_i during execution. Every single query execution may be different as some queries may have more computational complexity than others. We use α as a factor to differentiate these computational variations between queries.

Response time estimation models for similar scenarios has been studied in traditional systems [23], whereas Equation 1 shows our response time estimation for the aforementioned query execution scenario.

$$\mathcal{T}_{\mathcal{Q},n_i} = \alpha \sum_{1}^{j} \frac{d_j l_{n_i}}{t_{n_i}} + \beta \sum_{1}^{j} \sum_{1}^{k} \left(\frac{d_j}{b_{n_i,n_k}} + \frac{d_j}{t_{n_k}} \right) \phi_{j,k} \quad (1)$$

$$\phi_{j,k} = \begin{cases} 1 & \text{if } d_j \text{ migrated from node } n_k \text{ to } n_i \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

As depicted, the response time estimation mainly consist of the sum of two parts, namely processing and network transfers. As a result, $\mathcal{T}_{\mathcal{Q},n} > \mathcal{T}_{SLO}$ condition may occur in two situations that are relevant to those parts. (i) If the remote data is in a low bandwidth region relative to the executing node or the node hosting the remote data is having low I/O performance, this may have an adverse affect on response time. (ii) Ideally the queries should be processed by a node with a suitable computational load. However, if the CPU load on the executing node is high during the execution, this would also cause not meeting the response time guarantee. Therefore, it is beneficial to replicate remote data to more suitable locations as well as migrating the tasks to the nodes with enough computational resources to prevent over-utilization.

In some contexts, tenant may require the completion of a task that involve parallel execution of multiple independent queries [15]. The execution of these queries naturally takes place in parallel on a multitude of cloud sites. The completion of that particular task depends on the completion of all of its queries. In this case, due to parallel execution, the response time estimation should be taking each of the concerned queries into account. Rather than the sum, the logical approach is to estimate which of the mentioned queries is going to take the longest amount of time to produce a response. PEPRv2 also handles this parallel execution model.

Let K be a tenant task that consist of a number of queries, $K = \{Q_1, Q_2, \ldots, Q_i\}$ that are executed in a parallel. We assume that a query Q_i in K requires a data set R_n for processing, and all queries of the same task begins at the submission of the task. Given that all the R data sets as $R = \{R_1, R_2, \ldots, R_n\}$ are distributed in the cloud with replicas on j sites as $N = \{N_1, N_2, \ldots, N_j\}$; queries will be executed on the replicas in parallel as Q_i^j. In this case, the estimated response time $\mathcal{T}_K$ of the task K is determined by the longest estimated amount of time it takes for Q_i to execute on a certain particular data set R_n^j as shown in Eq. 3.

$$\mathcal{T}_K = \max_j \mathcal{T}(O_i^j) | O_i^j \in K \quad (3)$$

4.1.2 Profit estimation

While the function of the cloud provider is to create and distribute services for the tenants, the purpose of the cloud provider is to obtain economic benefit, in other words monetary profit. Since PEPRv2 makes the replication decision on a per-query basis, ideally every replication decision should at least on average be profitable for the provider.

In PEPRv2, we regard that the tenant pays for a particular service with an agreed upon quality. This service quality may include the total size of the hosted data and the arrival rate of the queries. In case the tenant requirements change over time, the rent is elastically scaled as well. Profitability of the provider depends on how much of this revenue is spent on providing the service to the tenants. In other words, ensuring profitability for a certain amount of revenue, the provider must pursue the most economical routes of providing the services in order to decrease expenditures. As PEPRv2 deals only with the data replication portion of these services, it aims to make the replication decision that would yield the least amount of monetary cost.

$$profit = revenues - expenditures \quad (4)$$

Similar to the idea of estimating the response time, a profit estimation is calculated for each query. In the profit estimation (Eq. 4), how much revenue is expected to be generated for each query is compared with the expenditures of the resources that are estimated to be used during the execution of each particular query. The estimated profit is used alongside the response time estimation to make replication

decisions in a way that only the profitable replication options are carried out. The details such as how the expected revenue and estimated resource consumptions are revisited in Sect. 5.

4.2 How many replicas

Keeping how many replicas of any particular data set is another issue to be addressed by the replication strategy. While PEPRv2 focuses on the performance aspect of the provider responsibilities, availability is also an important objective of the provider. Therefore, PEPRv2 keeps each data set at a minimum given availability through replication. Satisfying availability is a widely researched topic in cloud computing as there are already many works focusing on that particular issue [34]. In our study we do not visit this issue in depth, and design PEPRv2 in such a way to not let the number of replicas to not fall below a number that satisfies a minimum given availability level [37].

In the matter of satisfying response time, the number of replicas can be varied between the minimum number of replicas to satisfy a given availability and a number that is possible for the provider to physically create. In other words, the maximum number of replicas for any particular dataset has not a predefined limit, PEPRv2 can create seemingly infinite number of replicas as it deems necessary. This necessity is limited by determining the satisfaction of the response time estimation and profit estimation models.

Creation of replicas is performed in an incremental manner. At each query execution, the number of replicas of relevant data to that particular query is incremented by one. PEPRv2 assumes that, by placing the new replica in a near-optimal manner, the response time should already be satisfied. Therefore incrementing the number of replicas by a number greater than one is should not be necessary.

Algorithm 2 Replica retirement algorithm.

```
1: slaViolation ← false
2: logQueryCompletionStatistics()
3: T_Q^actual ←measureResponseTime(Q)
4: if T_Q^actual > T_SLO then
5:     slaViolation ← true
6: end if
7: if ExecutedQCount = QPerEpoch and
   slaViolation = true then
8:     slaViolation ← false
9:     ExecutedQCount ← 0
10:    removeReplicasByLRU(Num_LRU)
11: end if
```

Removal of unnecessary replicas is the other issue regarding the number of replicas maintained in the cloud. Over time, some replicas may get infrequent accesses simply due to changing demand caused by the tenant queries. In these cases, it may be beneficial to retire those replicas from the system to free up storage space for future replicas and reduce storage costs of the provider. Of course, the retirement operation must be done in a way to make the cloud still satisfy the response time guarantees after the removal of the unnecessary replicas. Evidently, there is a need for a decision mechanism to determine which replicas to be removed from the cloud and when.

First step in replica retirement is the identification of some replicas as unnecessary. While other replication decisions in PEPRv2 is made on-the-fly before the execution, replica retirement operation requires some a priori information. This a priori information mainly consists of access histories of each individual replica. Using this information, PEPRv2 can determine the list of replicas that are least frequently accessed during any particular window of time in the past.

Let $QPerEpoch$ show a system parameter that indicates the number of queries that defines an epoch. After every $QPerEpoch$ queries executed, if there are no SLA breaches observed in that epoch, a predefined number of least recently used replicas, denoted by Num_LRU, are removed from the system. The retirement algorithm is invoked at each query execution, however it only performs the actual cleanup after each $QPerEpoch$ queries. Algorithm 2 describes the retirement algorithm in more detail.

4.3 Where to replicate

The final replication decision is the placement of the new replica. Similarly to what has been done in the previous sections, this decision is also made by the response time and profit estimations. A suitable placement for a new replica must satisfy the response time requirement and should still be profitable for the provider after the placement. Moreover, the destination node should have enough storage space and relatively low load to avoid overutilization.

Ideally, finding an optimal placement requires evaluation of all nodes in the cloud for the necessary criteria. However, as seemingly infinite number of nodes possibly existing in a globally distributed cloud system, this approach is not feasible. The search duration itself would be enough to violate the SLA response time guarantee. As a result, a relatively straightforward heuristic is employed to determine a placement for the new replica. This heuristic would not look for the best placement, but a near-optimal placement that satisfies both the response time and the profit.

$$\exists R \subset C | \mathcal{T}_{R_i} \leq \mathcal{T}_{SLO}$$
$$\exists S_{i,j} \in R_i | \mathcal{P}_{S_{i,j}} > 0 \wedge S_{i,j} \leftarrow \max \mathcal{P}_{S_{i,j}} \quad (5)$$

In the first step, each region R_i in the cloud $C = \{R_1, R_2, \ldots, R_i\}$ is evaluated whether a placement to R_i would satisfy the response time guarantee or not. Then,

among the regions that satisfy response time (Eq. 5), the subregion that belongs to that particular region $R_i = \{S_{i,1}, S_{i,2}, \ldots, S_{i,j}\}$ and nets the highest amount of provider profit is selected. From the selected subregion, a node with acceptable storage and load is selected for placement. Bandwidth for reaching out of the subregion is the same for each node inside that particular subregion, as long as storage and load criteria are met. Therefore choosing one node from another would not have any significance as long as it is in the determined subregion.

Algorithm 3 Replica placement algorithm.

```
for each region R in the cloud C do
  if T_R ≤ T_SLO then
    for each subregion S in the region R do
      if P_Th ∧ S ← max P_S then
        for each node N in the region S do
          if N.freeSpace ≥ sizeOf(data) ∧ N.load ≤ loadThreshold then
            place on N
          end if
        end for
      end if
    end for
  end if
end for
```

Algorithm 3 depicts the placement algorithm as described. While the placement algorithm searches for the most profitable subregion, it does not look for the lowest possible response time. As the provider's guarantee is not the best performance but instead satisfaction of a threshold; it would be economically more beneficial for the provider to focus on profit once the response time is satisfied.

It is important to mention that, at any time some data migrate, there is a cost associated with the migration. In data migration, bandwidth is consumed during transfer and storage is consumed at the destination. Monetary aspect of the migration cost is dealt with the profit estimation by regarding migration cost a part of bandwidth and storage costs. In addition, data migration caused by pre-replicating data also results in an adverse effect on response time of execution of a query.

5 Economic cost model

Another important consideration in PEPRv2 replication management is the economic impact of data replication on provider's profit. The economic model presented in this section estimates provider revenue and expenditures in order to assess the estimated profit from the tenant queries.

5.1 Provider's revenue

As a part of the service agreement, tenants are obligated to pay for the rent associated with the services acquired. Considering the elastically scaled services, the total amount of rent can vary during a billing period. The exact amount of payment is determined by measuring the utility metrics that are defined in the SLA. This rent is the only revenue source for the provider in our economic model.

In PEPRv2, the economic part of whether to replicate decision is also depends on the profit of the provider. However, in this case we are interested in the revenue and the profit generated by each query, instead of focusing on profitability in a certain period of time. Therefore, it is important to make the transition between the actual revenue to the expected revenue generated from each query. Service details included in the SLA contains the limits for maximum arrival rate of the tenant queries. Therefore, it is possible for the provider to expect how many queries may be received from the tenant in a billing period. Of course, reaching the limits of the service is the worst-case scenario for the provider, but it may be sensible for the provider to be prepared for that case as well. The actual economics of operating a cloud is beyond the scope of this paper, therefore we assume that the provider is capable of calculating an expected, on average revenue generated from the execution of each query.

It is worthy of noting that, data replication is the responsibility of the service provider in PEPRv2. The tenant pays for a service and expects this service to be supplied at the agreed upon quality against the changing workload of the tenant queries. It is therefore the provider's responsibility to maintain the replicas in the cloud to satisfy the SLA terms. The tenant pays for the utilized service, and does not care about how many replicas the provider needs to create or maintain to meet the SLA.

5.2 Provider's expenditures

Meeting the computational demands of the tenants and elastically adjusting the properties of the services provided, cloud providers face the continually changing costs associated with these services. A cloud operator deals with many types of costs ranging from the paychecks of the employees to the power consumption of the facilities. While some of these costs are directly affected by data replication (e.g. storage costs), others may not be concerned by replication (e.g. on-site physical security).

Obviously it is not feasible to take into account every type of cost during a replication decision. In response time estimation model, we estimated the amount of time spent using some specific types of resources during the execution of a query. During estimation of the provider cost, we follow a similar path and estimate the monetary cost of the resources

consumed by each query execution. Consumption of each type of computational resources is estimated before the execution and this estimation is compared against the profit generated by the execution of that particular query to predict the profit. This profit value is used in the replication decision as mentioned in the earlier sections. It should be noted that, while estimating the expenditures, the costs include the possible replication events to evaluate the profitability with the inclusion of the new replicas.

During execution of the query $\mathcal{Q}$, a CPU cost of $\mathcal{C}_c$ is incurred as a result of the expected consumption of the CPU resource. A query may involve a number of nodes, including the executor node and other nodes that may contribute during migrating the relevant data to that particular query. As a result, the cost of CPU (Eq. 6) is estimated by the consumption of CPU time (T_i) on all nodes that contribute to the execution [30]. Conversion of CPU time to monetary cost is done by the factor of unit cost of CPU resource denoted by $\mu_{t,i}$ that is dependent on the node itself to take heterogeneity into account.

$$\mathcal{C}_c = \sum_{1}^{i} T_i \mu_{t,i} \tag{6}$$

Data transfer over the network (Eq. 7) is another cost item ($\mathcal{C}_n$) the provider deals with. Following a similar assumption as in the response time estimation model, a query $\mathcal{Q}$ requires data sets of $\{d_1, d_2, \ldots, d_j\}$ during execution. Some of this data may migrate from remote nodes $\{n_1, n_2, \ldots, n_k\}$ to the executor node n_i. Data migration may occur between nodes that reside in various parts of the cloud hierarchy. Moreover, bandwidth cost between any two node can vary due to heterogeneity in the cloud. Therefore, ν denotes the unit cost of bandwidth that is specific between any two nodes in the cloud.

$$\mathcal{C}_n = \sum_{1}^{j} \sum_{1}^{k} d_j \nu_{n_i,n_k} \phi_{j,k} \tag{7}$$

$$\phi_{j,k} = \begin{cases} 1 & \text{if } d_j \text{ migrated from node } n_k \text{ to } n_i \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Final main cost item that is relevant to data replication is the storage cost, $\mathcal{C}_s$, as depicted in Eq. 9. Each data set can be used by any number of queries during the lifetime of the replicas. It is therefore not accurate to bill the queries by the amount of storage space occupied. For this reason, we estimate the storage cost by the amount of storage I/O is expected to be performed by any particular query. This way, we establish that the queries that cause more I/O operations result in more monetary cost of storage. The possible heterogeneity of different cloud regions having different storage costs is handled by the factor σ, which denotes the unit cost of storage on any particular cloud node.

$$\mathcal{C}_s = \sum_{1}^{j} \sum_{1}^{k} d_j \sigma_{n_k} \chi_{j,k} \tag{9}$$

$$\chi_{j,k} = \begin{cases} 1 & \text{if } d_j \text{ is read from node } n_k \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

As mentioned before, provider expenditures may include other cost items that may not be directly caused by data replication. Providers can simply include these costs in the economic model as other costs ($\mathcal{C}_{others}$) or impose a margin on the expected profit to cover these expenses. Furthermore, the provider pays penalties to the tenants if one or more SLOs are not satisfied. As a result, in addition to other costs, penalties ($\mathcal{C}_p$) are also included in the provider cost. Through data replication, the provider aims to minimize the penalties paid to reduce cost [38]. In regard to all of the described cost types, the estimated provider cost, i.e. expenditures, is given in Eq. 11.

$$expenditures = \mathcal{C}_c + \mathcal{C}_n + \mathcal{C}_s + \mathcal{C}_p + \mathcal{C}_{others} \tag{11}$$

Since the expected revenue of each query is a known amount for the provider, at this point, it is a simple subtraction to calculate the estimated profit using the total amount of expenditures caused by the execution of any particular query.

6 Performance evaluation

6.1 Simulation environment

Realizing an entire cloud topology is a very hard and expensive way of verifying the performance of any study. Acquiring and configuring many hosts to achieve a model of a desired cloud system is indeed non-trivial. Furthermore, this way of testing makes it harder to control all the variables involved in a scenario. In order to address these concerns, several cloud simulators have been proposed in the literature [2] to accurately implement performance evaluation scenarios without dealing with the burden and cost of an actual cloud environment. While every proposed simulator focuses on one aspect of the cloud, e.g. resource provisioning, as of writing this paper, no particular cloud simulator specializes on data replication. Among the available simulators, CloudSim [8] is noticeably widely used in the literature. CloudSim is also an open source project, therefore it is possible to tailor the simulation suite to meet the the requirements of our performance evaluation scenarios.

In its standard form, CloudSim is targeted for creating datacenters and allocating virtual machines on physical hosts. It allows creation of simple tasks called cloudlets and models the execution of cloudlets on virtual machines. Therefore it is more suitable towards scenarios that involve virtual machine

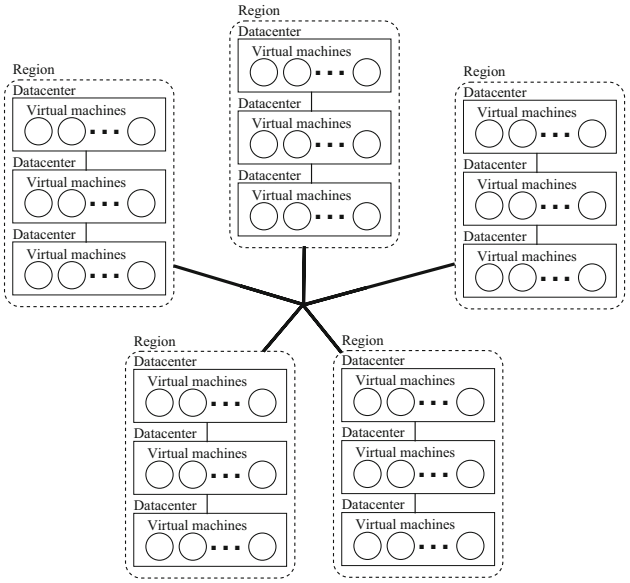

Fig. 1 Simulated cloud topology

(VM) provisioning, task distribution, and power management. However, mechanisms for performing data replication is not present in the original version. As a result, we extended CloudSim to include data replication functionality, as well as other aspects that our simulation scenarios require, including supporting a cloud topology with heterogeneous sites and network links and measuring resource consumptions and conversion of consumed resources into monetary counterparts.

In the simulations, we realized a cloud environment with hierarchical network links as described at the beginning of Sect. 3. Simulated cloud consists of five geographical regions. Each region contains thre subregions represented as datacenters. In turn, each datacenter contains ten nodes realized as VMs. Network topology is established in such a way that, the bandwidth capacity is more abundant and cheaper inside the datacenters but less abundant and more expensive towards the inter-region level. Moreover, computational capabilities and storage capacities are also varies from datacenter to datacenter to simulate a cloud environment with heterogeneous sites and network links. This heterogeneity is also extended to the costs of the CPU, storage and network resources. In every region, the provider costs of these resources are varied to achieve a more realistic cloud system. A schematic overview of this cloud topology is depicted in Fig. 1. Also a complete list of simulation parameters, including in what ranges the heterogeneous cloud resources are varied can be found in Table 1. We chose the simulation parameters to be in accord with existing studies [5] to realistically represent a typical cloud environment.

Each VM has the necessary computational resources to perform the execution of queries. These resources include CPU, RAM, storage, and network connectivity. During execution, VMs can access the data sets in other VMs by remote reads or by replicating them to local storage. Optimal distribution of queries to virtual machines, and other related issues including task migration is not a focus this study. When a query is received by a datacenter, a virtual machines is randomly assigned to handle the execution of that particular query. Queries randomly arrive to the cloud system as the average arrival rate indicates. Each query requires a dataset that is randomly determined when the query is created. Moreover, computational load generated by each query is also randomly varied from one query to another to simulate the computational variations between queries, e.g. a simple projection versus an aggregate function.

The duration of the simulation is regarded as one billing period. For that period, the tenant is charged for the services obtained, and provider profit is determined by the expenditures of the provider during this billing period.

Table 1 Simulation parameters

Parameter	Value
Regions	5
Datacenters per region	3
VMs per datacenter	10
VM processing capability	1000–2000 MIPS
VM storage capacity	15–25 GB
Intra-datacenter bandwidth	4–6 Gbit/s
Intra-region bandwidth	1.25–1.75 Gbit/s
Inter-region bandwidth	0.15–0.25 Gbit/s
Avg. intra-datacenter delay	5–10 ms
Avg. intra-region delay	25–50 ms
Avg. inter-region delay	100–150 ms
Response time SLO	180 s
Simulation duration	10 min
Number of queries processed	10,000
Query arrival rate	16.67 query/s
Computational load of a query	1000–7500 MI
Number of data sets	30
Avg. size of a data set	620 MB
Intra-datacenter transfer cost	$0.005/GB
Intra-region transfer cost	$0.025/GB
Inter-region transfer cost	$0.25/GB
Storage cost	$0.20 to $0.30/GB
Processing cost	$1.25 to $2.25/$10^9$ MI
Penalty cost	$0.0025 per violation
Assumed revenue	$0.004 per query
Num_LRU	1
$QPerEpoch$	10
$loadThreshold$	90%

Table 2 Simulation results

Replication strategy	PEPRv2	PEPR	CDRM	No replication
Avg. response time (s)	45.56	36.41	203.54	343.27
SLA violations	337	298	3609	7303
Number of replications	1040	1226	797	0
Storage usage (%)	21.96	26.97	16.33	3.81
Inter-region data transfer (GB)	26.87	28.03	294.90	499.98
Intra-region data transfer (GB)	295.80	250.64	164.97	62.84
Intra-datacenter data transfer (GB)	301.94	349.77	173.08	65.26
Total expenditures of the provider ($)	26.94	33.81	97.08	154.46

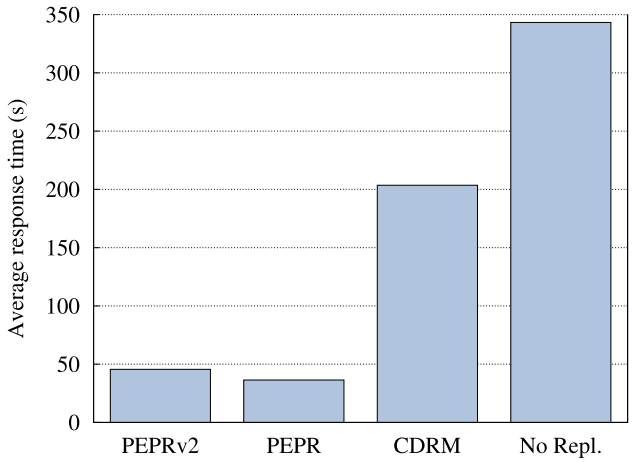

Fig. 2 Average response time in seconds

6.2 Simulation results

6.2.1 Compared data replication strategies

The aim of the simulation study is to monitor how data replication enables providers to cope with the query load, in order to keep the response time SLO. Simultaneously, we observe the provider profit in the same billing period. With a goal of demonstrating how the satisfaction of these two criteria is achieved by PEPRv2, we compared it to its predecessor, PEPR. This enabled us to highlight the improvement of the newer iteration. Moreover, in the comparative simulations, we also employed a more traditional strategy, *Cost-Effective Dynamic Replication Management* (CDRM) [37]. CDRM is another data replication strategy for cloud systems. It considers the availability and balanced load objectives. However, it does not take economic benefit of the provider into account. Finally, we also run the simulations without using any data replication scheme, in order to show a baseline to demonstrate the impact of data replication.

6.2.2 Measured metrics

In Cloudsim, we made the necessary effort to log the response time, the average of actual response times of all queries handled in the billing period. Additionally, number of SLA violations, total number of replications, storage usage, and the amount of network data transfers are also logged during the simulations. The corresponding monetary cost of each of these metrics are calculated by using the associated unit costs. Since the cloud environment in the simulation is heterogeneous, the unit cost of resources are dealt with the consideration of variations between regions. These measured metrics and the monetary cost for the provider are listed in Table 2. The table shows the results for a single billing period, not just one single query.

6.2.3 Average response time and SLA violations

Both PEPRv2 and PEPR managed to satisfy the response time objective with a great margin (Fig. 2). Having a much lower average response time than the SLA threshold also meant that these two strategies satisfied the response time for a greater percentage of the queries processed. On the other hand, satisfying availability and balanced load does not ensure performance for CDRM. While the average response time with CDRM is close to the SLA threshold, response time is not satisfied for a great amount of queries. The results also indicate that, it simply was not possible to meet the performance SLO without using a data replication strategy.

Since SLA violation count is a direct measure of the response time satisfaction, we see a similar view on this metric as well. Both PEPRv2 and PEPR kept the number of violations to a minimum, albeit with some breaches during the initial replica reconfiguration. On the other hand, CDRM suffered heavy violations, however it still provided a significant advantage over not using data replication.

6.2.4 Number of replications and storage usage

In terms of the number of replications, PEPRv2 made slightly less replications compared to its predecessor. This difference can be attributed to the more intricate replica placement decision on PEPRv2's part which takes heterogeneous properties of the cloud into account. Still, both PEPRv2 and PEPR created more replicas compared to the number of replicas created by CDRM. This difference in the number of replication events also made an impact on the storage consumption of these strategies as well. This different outlook on the replication event totals and storage usages are due to the different objectives pursued by these strategies, especially with the case of CDRM. Obviously, without any data replication, the

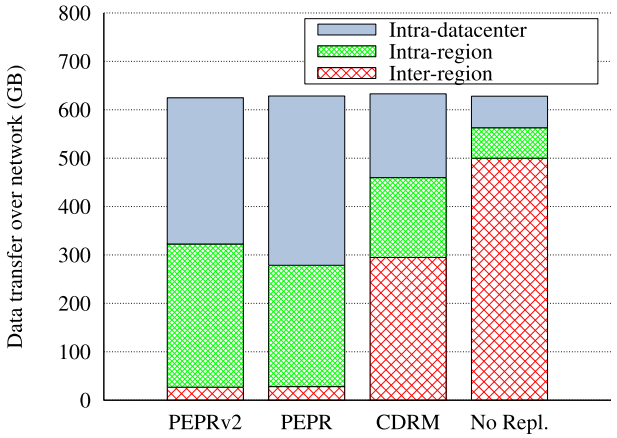

Fig. 3 Breakdown of total data transfers with respect to network hierarchy

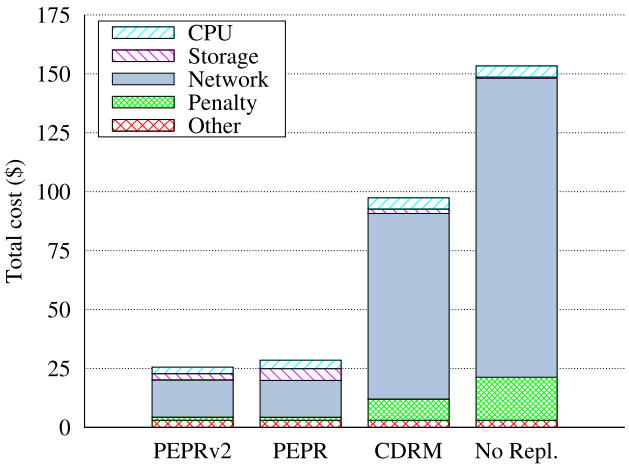

Fig. 4 Total cost (monetary expenditure) of the provider during one simulated billing period

final simulation scenario only used the storage necessary to host the initially placed data sets.

6.2.5 Network bandwidth consumption

PEPRv2 and PEPR, by having multiple replicas of each data set strategically placed throughout the cloud, made almost the entire network transfers inside either the local datacenter or inside the same region. An almost negligible fraction of the transfers are done at the inter-region level. These two strategies only chose to remotely read, i.e. without replication, the data sets that are not presently available in the same datacenter as the requestor but still available in the same region; as long as it satisfies the performance SLO. Of course the decision behind this behavior lies in the profitability. If accessing a data set in another datacenter is more profitable than creating a new replica in local datacenter, PEPRv2 and PEPR took such an action to improve the economic benefit of the provider. Having said that, there is still a difference in the intra-region and intra-datacenter transfers of these two strategies. This difference is mainly due to the improved data placement heuristic of PEPRv2. Compared to its predecessor, PEPRv2 has a better way of determining the placement for new replicas by evaluating each subregion for response time and profit satisfaction instead of former's simplistic way of evaluating closest-to-requestor first approach. This way, PEPRv2 provided a better trade-off between the storage and network utilization.

CDRM also placed its replicas throughout the cloud, however not following an economic objective, its replication decisions yielded more usage of the expensive inter-region bandwidth (Fig. 3). Compared to the evaluated replication strategies, using no data replication obviously made a significant portion of the network transfers over the relatively slow and expensive inter-region bandwidth, hence the failure in satisfying the performance SLO.

6.2.6 Monetary cost

Figure 4 depicts the total monetary cost accumulated for the provider during simulations. Each data replication strategy is evaluated by processing the same number of queries generated with the same arrival rate. Moreover, the computational load of these queries are also the same on average. Therefore, the CPU cost is very similar for each strategy and difference between them is at a negligible level. Also, the unit storage costs are relatively cheap, therefore all strategies impact the provider at an acceptable level with PEPR accumulating a slightly more storage cost due to having more replicas. However it should be noted that, slight difference in storage cost between PEPRv2 and PEPR is due to the better-established tradeoffs on PEPRv2's part as explained in the previous subsection. PEPRv2 managed to escape unnecessary storage costs by performing replication to cheaper-than-local remote subregions that can still satisfy the response time objective. In the scenario that uses no replication, only storage cost is due to having the initial placement of data sets.

Compared to CDRM, PEPRv2 and PEPR accumulated noticeably less network costs. This is caused by taking measures to place the replicas in a cost-effective way when a replication is required for performance satisfaction. Of course, this is a trade-off move towards utilizing the cheaper network resource more intensively by sacrificing some storage space. In the scenario without replication, frequent use of inter-region network links caused the provider costs to rise beyond acceptable levels.

6.2.7 Overall remarks

It is evident from results that, a data replication strategy is crucial to meet the desired quality of service. On the other hand, while a traditional strategy can satisfy the performance

SLO through load balancing, it may not be enough to satisfy the economic benefit of the provider. A traditional strategy can extravagantly utilize cloud resources in an effort to maximize the performance. However, the economic impact of replicating data in an eager manner can be too much for the cloud providers. Therefore, it is a better idea to focus on lowering provider costs, once the performance SLO is satisfied. PEPRv2 achieves this goal of simultaneously satisfying both the performance SLO and economic benefit. Moreover, it also provides improvement on some key areas such as replica placement, compared to its predecessor.

7 Conclusion

In this paper, we proposed PEPRv2, *PErformance and Profit oriented data Replication strategy* for cloud systems. PEPRv2 satisfies both performance and minimum availability guarantees for the tenant while taking the profitability of the cloud provider into account.

Tenants place queries on the cloud for processing. These queries may be single individual queries or part of a set of queries that are executed in parallel to achieve a common task. When a query arrives at the cloud, PEPRv2 estimates the response time of each query and finds out if the response time of the query can satisfy service quality that the tenant expects from the provider. If the response time guarantee cannot be satisfied, i.e. estimated response time is greater than the response time SLO, PEPRv2 considers replicating the relevant data. Then, PEPRv2 calculates a profit estimation. This profit estimate includes any possible data replication planned to enable the provider to execute the query with an acceptable response time. Only if the execution is estimated to be profitable, the considered replication is performed. Moreover, as the performance SLO is satisfied over time, older replicas that are not used are retired from the cloud to further improve the economic benefit of the provider.

We validated PEPRv2's ability to simultaneously satisfy both the performance SLO and provider profit with an experimental evaluation. With a simulation tool, we created a simulated cloud system that processes tenant queries. PEPRv2 is pitted against its predecessor, PEPR as well as another data replication strategy, CDRM, which does not take the economic aspect of the cloud into account. Moreover, a scenario without a replication strategy is also evaluated to demonstrate the impact of data replication on both the response time and provider profit. Simulation results shows that PEPRv2 satisfied the response time guarantees with less expenditures for the cloud resources. Most notably, an improvement over its predecessor is observed in terms of monetary cost while still satisfying SLA response time. Overall results also indicate that, taking economics of the cloud must also be considered while ensuring the performance guarantees for a data replication strategy aiming to operate in the cloud.

A possible future direction for this study is to take queries that include dependent operations, e.g. join operations, into account. This way, PEPRv2 would also be applicable to database queries. Moreover, implementing the proposed strategy in a real cloud environment may also present an interesting research opportunity.

Acknowledgements The work presented in this paper is supported in part by TUBITAK.

References

1. Abouzamazem, A., Ezhilchelvan, P.: Efficient inter-cloud replication for high-availability services. In: 2013 IEEE International Conference on Cloud Engineering (IC2E), pp. 132–139 (2013). https://doi.org/10.1109/IC2E.2013.27
2. Ahmed, A., Sabyasachi, A.S.: Cloud computing simulators: a detailed survey and future direction. In: Souvenir of the 2014 IEEE International Advance Computing Conference, IACC 2014, pp. 866–872 (2014). https://doi.org/10.1109/IAdCC.2014.6779436
3. Armbrust, M., Stoica, I., Zaharia, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A.: A view of cloud computing. Commun. ACM **53**(4), 50 (2010). https://doi.org/10.1145/1721654.1721672
4. Bai, X., Jin, H., Liao, X., Shi, X., Shao, Z.: RTRM: a response time-based replica management strategy for cloud storage system. Grid Pervasive Comput. **1**, 124–133 (2013)
5. Barroso, L.A., Clidaras, J., Hölzle, U.: The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, 2nd edn. Morgan and Claypool Publishers, San Rafael (2013). https://doi.org/10.2200/S00516ED2V01Y201306CAC024
6. Bonvin, N., Papaioannou, T.G., Aberer, K.: An economic approach for scalable and highly-available distributed applications. In: IEEE 3rd International Conference on Cloud Computing, CLOUD 2010, pp. 498–505 (2010). https://doi.org/10.1109/CLOUD.2010.45
7. Bonvin, N., Papaioannou, T.G., Aberer, K.: A self-organized, fault-tolerant and scalable replication scheme for cloud storage categories and subject descriptors. In: Proceedings of the 1st ACM symposium on Cloud computing-SoCC'10, pp. 205–216 (2010)
8. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software **41**(1), 23–50 (2011). https://doi.org/10.1002/spe.995
9. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud computing and grid computing 360-degree compared. In: 2008 Grid Computing Environments Workshop, pp. 1–10 (2008). https://doi.org/10.1109/GCE.2008.4738445
10. Ghanbari, H., Simmons, B., Litoiu, M., Iszlai, G.: Feedback-based optimization of a private cloud. Fut. Gener. Comput. Syst. **28**(1), 104–111 (2012). https://doi.org/10.1016/j.future.2011.05.019
11. Guerrero-Contreras, G., Rodriguez-Dominguez, C., Balderas-Diaz, S., Garrido, J.: Dynamic replication and deployment of services in mobile environments. In: New Contributions in Information Systems and Technologies, pp. 855–864. Springer, Berlin (2015)
12. Janpet, J., Wen, Y.F.: Reliable and available data replication planning for cloud storage. In: IEEE 27th International Conference on

Advanced Information Networking and Applications (AINA), pp. 772–779 (2013). https://doi.org/10.1109/AINA.2013.125
13. Jayalakshmi, D.S., Rashmi Ranjana, T.P., Srinivasan, R.: Dynamic data replication strategy in cloud environments. In: 2015 Fifth International Conference on Advances in Computing and Communications (ICACC), pp. 102–105 (2015). https://doi.org/10.1109/ICACC.2015.79
14. Kemme, B., Jimenez-Peris, R., Patino-Martinez, M.: Database Replication. Morgan and Claypool Publishers, San Rafael (2013)
15. Kloudas, K., Mamede, M., Preguiça, N., Rodrigues, R.: Pixida: optimizing data parallel jobs in wide-area data analytics. Proc. VLDB Endow. **9**(2), 72–83 (2015)
16. Kouki, Y., Ledoux, T., Sharrock, R.: Cross-layer SLA selection for cloud services. In: First International Symposium on Network Cloud Computing and Applications, pp. 143–147 (2011). https://doi.org/10.1109/NCCA.2011.30
17. Lee, J., Chung, J., Lee, D.: Efficient data replication scheme based on hadoop distributed file system. Int. J. Softw. Eng. Appl. **9**(12), 177–186 (2015)
18. Li, W., Yang, Y., Yuan, D.: A novel cost-effective dynamic data replication strategy for reliability in cloud data centres. In: 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, pp. 496–502 (2011). https://doi.org/10.1109/DASC.2011.95
19. Li, W., Yang, Y., Yuan, D.: Ensuring cloud data reliability with minimum replication by proactive replica checking. IEEE Trans. Comput. **65**(5), 1494–1506 (2016). https://doi.org/10.1109/TC.2015.2451644
20. Mengxing, H., Xianglong, Y., Sanpeng, W., Donghai, Z.: A strategy of dynamic replica creation in cloud storage. In: Proceedings of the 1st International Workshop on Cloud Computing and Information Security, Ccis, pp. 389–392. Atlantis Press, Paris (2013). https://doi.org/10.2991/ccis-13.2013.89
21. Miglierina, M., Gibilisco, G.P., Ardagna, D., Di Nitto, E.: Model based control for multi-cloud applications. In: ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, pp. 37–43 (2013). https://doi.org/10.1109/MiSE.2013.6595294
22. Myint, J., Naing, T.T.: Management of data replication for pc cluster based cloud storage system. Int. J. Cloud Comput. **1**(3), 31–41 (2011). https://doi.org/10.5121/ijccsa.2011.1303
23. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems, 3rd edn. Springer, New York (2011). https://doi.org/10.1007/978-1-4419-8834-8
24. Park, S.M., Kim, J.H., Ko, Y.B., Yoon, W.S.: Dynamic data grid replication strategy based on internet hierarchy. In: Grid and Cooperative Computing, pp. 838–846. Springer, Berlin (2004)
25. Qu, Y., Xiong, N.: RFH: A resilient, fault-tolerant and high-efficient replication algorithm for distributed cloud storage. In: 2012 41st International Conference on Parallel Processing, pp. 520–529 (2012). https://doi.org/10.1109/ICPP.2012.3
26. Ranganathan, K., Foster, I.: Identifying dynamic replication strategies for a high-performance data grid. In: Proceedings of the International Grid Computing Workshop, vol. 2242, pp. 75–86. Springer, Berlin (2001)
27. Ridhawi, I.A., Mostafa, N., Masri, W.: Location-aware data replication in cloud computing systems. In: 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob) pp. 20–27 (2015). https://doi.org/10.1109/WiMOB.2015.7347936
28. Sakr, S., Liu, A.: SLA-based and consumer-centric dynamic provisioning for cloud databases. In: IEEE Fifth International Conference on Cloud Computing, pp. 360–367 (2012). https://doi.org/10.1109/CLOUD.2012.11
29. Silvestre, G., Monnet, S., Krishnaswamy, R., Sens, P.: AREN: a popularity aware replication scheme for cloud storage. In: IEEE 18th International Conference on Parallel and Distributed Systems, pp. 189–196 (2012). https://doi.org/10.1109/ICPADS.2012.35
30. Sousa, F.R.C., Machado, J.C.: Towards elastic multi-tenant database replication with quality of service. In: IEEE/ACM 5th International Conference on Utility and Cloud Computing (UCC 2012), pp. 168–175 (2012). https://doi.org/10.1109/UCC.2012.36
31. Spaho, E., Barolli, A., Xhafa, F., Barolli, L.: P2P data replication: techniques and applications. In: Modeling and Processing for Next-Generation Big-Data Technologies, pp. 145–166. Springer, Berlin (2015)
32. Stantchev, V., Schröpfer, C.: Negotiating and enforcing QoS and SLAs in grid and cloud computing. In: Advances in Grid and Pervasive Computing, pp. 25–35 (2009)
33. Sun, D.W., Chang, G.R., Gao, S., Jin, L.Z., Wang, X.W.: Modeling a dynamic data replication strategy to increase system availability in cloud computing environments. J. Comput. Sci. Technol. **27**(2), 256–272 (2012). https://doi.org/10.1007/s11390-012-1221-4
34. Tabet, K., Mokadem, R., Laouar, M.R., Eom, S.: Data replication in cloud systems: a survey. Int. J. Inf. Syst. Serv. Sect. **8**(3), 17–33 (2017). https://doi.org/10.4018/IJISSC.2017070102
35. Tos, U., Mokadem, R., Hameurlain, A., Ayav, T., Bora, S.: Dynamic replication strategies in data grid systems: a survey. J. Supercomput. **71**(11), 4116–4140 (2015). https://doi.org/10.1007/s11227-015-1508-7
36. Tos, U., Mokadem, R., Hameurlain, A., Ayav, T., Bora, S.: A performance and profit oriented data replication strategy for cloud systems. In: International IEEE Conference on Cloud and Big Data Computing (CBDCom), pp. 780–787 (2016). https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0125
37. Wei, Q., Veeravalli, B., Gong, B., Zeng, L., Feng, D.: Cdrm: a cost-effective dynamic replication management scheme for cloud storage cluster. In: IEEE International Conference on Cluster Computing, pp. 188–196 (2010). https://doi.org/10.1109/CLUSTER.2010.24
38. Xiong, P., Chi, Y., Zhu, S., Moon, H.J., Pu, C., Hacigumus, H.: Intelligent management of virtualized resources for database systems in cloud environment. In: International Conference on Data Engineering, pp. 87–98 (2011). https://doi.org/10.1109/ICDE.2011.5767928
39. Zhang, H., Lin, B., Liu, Z., Guo, W.: Data replication placement strategy based on bidding mode for cloud storage cluster. In: 11th Web Information System and Application Conference, pp. 207–212 (2014). https://doi.org/10.1109/WISA.2014.45

Uras Tos received his Ph.D. degree in Computer Science from a joint supervision program between Paul Sabatier University and Ege University in 2017. He was most recently employed as a research assistant at Izmir Institute of Technology. His research interests include large-scale data management, data replication and cloud computing.

Riad Mokadem is currently an Associate Professor in Computer Science at Paul Sabatier University, Toulouse, France, and a member of the Institute of Research in Computer Science of Toulouse (IRIT). He received his Ph.D. degree in Computer Science from Dauphine University in Paris, France in 2006. His main research interests are data management and query optimization in large-scale distributed environments, scalable distributed data structures and database performance.

Abdelkader Hameurlain is full professor in Computer Science at Paul Sabatier University, Toulouse, France. He is a member of the Institute of Research in Computer Science of Toulouse (IRIT). His current research interests are in query processing and optimization in parallel and large-scale distributed environments, mobile databases, and database performance. Prof. Hameurlain has been the general chair of the International Conference on Database and Expert Systems Applications (DEXA'02, DEXA'2011 and DEXA'2017). He is co-editors in Chief of the International Journal "Transactions on Large-scale Data and Knowledge Centered Systems" (LNCS, Springer). He was guest editor of three special issues of "International Journal of Computer Systems Science and Engineering on "Mobile Databases", "Data Management in Grid and P2P Systems", and "Elastic Data Management in Cloud Systems".

Tolga Ayav received his B.Sc. degree in Electronics Engineering in 1995. After a three year industry experience, he received his M.Sc. degree in 1999 and Ph.D. degree in 2004 in Computer Engineering from Ege University. He worked as a post-doctoral fellow at INRIA Rhone-Alpes in 2005. He is now with the Department of Computer Engineering in Izmir Institute of Technology as Assistant Professor. Mr. Ayav's research interests lie in formal methods for embedded software verification, testing and reliability, real-time systems and fault-tolerant systems.

Sebnem Bora received her Ph.D. degree from Ege University, Turkey in 2006. She is currently an assistant professor in the Computer Engineering Department at Ege University. Her research interests include fault tolerance in multi-agent systems, self-adaptive systems, complex-adaptive systems, and agent-based modeling and simulation.