**Journal of Information Technology Research**
Volume 15 • Issue 1

# Performance Enhancement of Cloud Datacenters Through Replicated Database Server

Sudhansu Shekhar Patra, Kalinga Institute of Industrial Technology, India*
https://orcid.org/0000-0001-9996-7681

Veena Goswami, Kalinga Institute of Industrial Technology, India
https://orcid.org/0000-0002-4260-4721

## ABSTRACT

Cloud computing has risen as a new computing paradigm providing computing, resources for networking, and storage as a service across the network. Data replication is a phenomenon which brings the available and reliable data (e.g., maybe the databases) nearer to the consumers (e.g., cloud applications) to overcome the bottleneck and is becoming a suitable solution. In this paper, the authors study the performance characteristics of a replicated database in cloud computing data centres which improves QoS by reducing communication delays. They formulate a theoretical queueing model of the replicated system by considering the arrival process as Poisson distribution for both types of client request, such as read and write applications. They solve the proposed model with the help of the recursive method, and the relevant performance matrices are derived. The evaluated results from both the mathematical model and extensive simulations help to study the unveil performance and guide the cloud providers for modelling future data replication solutions.

## KEYWORDS

Cloud Computing, Cost Model, Performance Measure, Queue, Replicated Database

## 1. INTRODUCTION

Cloud computing is the increasingly popular platform that offers enormous opportunities for the distribution of online services is an appeal to the ICT service providers. It provides utility services, sharing resources of scalable data centres (Buyya et al. (2008)); Hussain et al. (2013). To some extent, cloud computing is a new distributed Internet technology in the market, providing services such as grid computing, distributed computing, utility computing, and software-as-a-service, that have received many significant research direction and the commercial implementations. There are many companies such as Amazon, Google, which provides the services to their clients through their data centres and

DOI: 10.4018/JITR.299948 *Corresponding Author

This article published as an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0/) which permits unrestricted use, distribution, and production in any medium, provided the author of the original work and original publication source are properly credited.

offers cloud computing services and infrastructure products. The end users get the benefit by accessing data from the gadget of services and global service, by backups managed centrally, high computational power and pay-per-use billing strategies (Khalaj et al. (2016)). It benefits by the efficient utilization of data centres, data centre power strategies, large scale virtualized resources and optimized software flocks. In a survey done in 2010, studies show that the data centres consumed around 1.1-1.5% of global electricity consumption and between 1.7 and 2.2% for the United States. In 2022, datacentre consumption will be almost 175TWh (Ni and Bai (2017)). There need to be an over-provisioning of computing, storage and cooling resources, power distribution to ensure a high degree of reliability in datacentres (Bertoldi et al. (2017)). Power distribution and cooling systems consume around 15 and 45% of the total energy consumption respectively, whereas leaving approximately 40% to the IT equipment (Avgerinou et al. (2017)). This 40% energy which consumed in the IT equipment shared among the computing servers and the networking equipment.

The communication network takes 30% to 50% of the total energy used by the IT equipment depending on the data centre load level (Koomey et al. (2011)). In this paper, we discuss the opportunities and limitations of deploying data management challenges on these emerging cloud computing platforms (e.g., Google web services, Amazon Web Services, etc.). The large scale decision support systems, application-specific data marts and data analysis tasks take advantage of cloud computing platforms instead of operational, traditional transactional database systems. There are many approaches for making the datacenter to consume less power. The two main significant techniques are shutting the components down when there is a minimum load in the data centre or by scaling down their performance. Both approaches can be applied to computer servers (Srikantaiah et al. (2008); Travostino et al. (2006)) as well as to network switches (Cao et al. (2009); Buyya et al. (2009)). The bottom-line performance of cloud computing applications, for example, audio and video conferencing, gaming, online office services, social networking, storage, backup bets mainly on the reliability, availability and efficiency of high-performance network resources (Sindhu and Mukherjee (2011)). To build higher credibility and low latency service provisioning the data resources should be replicated or made nearer to the physical infrastructure, where the cloud applications are running. In the literature, many researchers have proposed a large number of replication strategies for the cloud data centres, and one may refer to Soltesz et al. (2007); Sotomayor et al. (2009). These strategies can be able to optimize the system bandwidth and data availability through geographically distributed data centres. However, none of these articles focuses the energy efficiency or on minimization of the energy consumption or the replication strategies by the data centres. In this paper, we present a data replication technique for cloud computing data centres which minimizes the energy consumption and increases the energy efficiency, optimizes the network bandwidth and decreases the communication delay among the geographically distributed data centres and also inside each datacenter (Sasikumar et al. (2020)). Specifically, our contributions in this can be summarized as follows.

- Optimize the energy consumption of data centres distributed geographically and inside each data centre in the IT infrastructures.
- Develop the data replication strategies for optimizing the energy consumption and also the bandwidth capacity of data centres.
- Minimize the communication delay, which gives a better experience to the end-users of the different cloud applications.
- Carry out the performance evaluation of our developed replicated strategy through mathematical modelling.
- Analyze the correlation among the performance, reliability, serviceability and power consumption.

The organization of the paper is as follows. The literature survey on energy efficiency and replication strategies is described in Section 2. Section 3 shows the different data replication schemes and replication in the cloud is depicted in Section 4. Section 5 presents the model description of

the replicated database analysis in the cloud. Section 6 measures the performance of the proposed replicated database system. Section 7 demonstrates the numerical results. Section 8 concludes the paper.

The organization of the paper is as follows. Section 2 presents a literature survey on energy efficiency and describe replication strategies. Section 3 displays the different data replication schemes, and in Section 4 depicts the replication in the cloud.

Section 5 presents the model description and analysis of the replicated database in the cloud. Section 6 measures the performance of the proposed replicated database system and demonstrates the numerical results in Section 7. Section 8 concludes the paper.

## 2. RELATED WORK

There are two significant alternatives through which one can minimize the data centre energy consumption: (a) shutting down the hardware components whenever and wherever not necessary and/or (b) by scaling down the hardware performance. Both methods are applied to computing servers and/or network switches. The first technique is commonly known as dynamic power management (DPM) applied to the servers (Sotomayor et al. (2009), Ebadi et al. (2019)). DPM saves energy in most extent in the cloud data centres. In combination with the workload consolidation scheduler, the system trying to maximize the number of idle servers which can be put into sleep mode because often the average load of the system stays below 30% Nurmi et al. (2009). The second technique represents the dynamic voltage and frequency scaling (DVFS) technology (Sindhu and Mukherjee (2011); Cao et al. (2009)). DVFS establishes the relation between the power consumption $P$, supplied voltage $V$, and the operating frequency $f$ represented by the polynomial: $P = V^2 f$.

The power consumption can be reduced by reducing the voltage and/or frequency. Effect of DVFS is limited, as for only the CPU the power consumption may be reduced, whereas the system bus, disks, memory, as well as peripheral devices consume at their peak rates. In the similar path of computing servers, many energy-efficient architectures for network communication equipment depends on (i) downgrading the transmission rate or operating frequency or (ii) turn off or power down the entire device, and/or the hardware components of it for energy conservation. Sakr et al. (2011) were first studied the power-aware communication networks. In 2003, using dynamic voltage scaling (DVS) links, the first work was proposed, called power-aware interconnection network Sun and Sugawara (2011). After that, the DVS technology was combined with dynamic network shutdown (DNS) to optimize further energy consumption (Hameed et al. (2016)). Virtualization is another widely used technology which indirectly affects energy consumption. Hu and Veeravalli (2013) showed that through virtualization, multiple virtual machines (VMs) could share the same physical machine in a data centre. As per the client application's need, the server resources are dynamically provisioned to the VM. Similar to DPM and DVFS power management, virtualization is applied in both the computing servers and network switches; however, with different objectives. Logically separate virtualization addressing and forwarding mechanisms are implemented in the field of network, but it may not necessarily achieve energy efficiency (Liu et al. (2009)). Cloud computing which is established on geographically distributed platforms enables the providers to provide immense IT services, offered globally. For the reliability of the system and better performance, the resources can be replicated at redundant locations, including the databases. Since there is an exponential increase in data traffic and a need to optimize the energy and bandwidth in datacenter systems, several data replication approaches have been proposed (Cecchet et al. (2011)).

By maintaining replicas of the databases, resources, as well as various infrastructures at multiple sites of the cloud system, improves the performance of the system by minimizing the remote access delay and smartly handles the single point of failure. However, for maintaining data replicas, there is a need for many infrastructures, storage devices and networking devices. Indeed, replicas must be synchronized such that changes made at a replica of one site need to be reflected altogether at

other sites. This requires an initial communication cost both in terms of network bandwidth and energy. There are many data center resources which remain underutilized but consume a significant amount of energy (Kliazovich et al. (2012)). Underuses resources must be entirely used without any additional costs. There is a variation in the cost of electricity at different geographical locations, and this one must consider while allocating the replicas to the data centres (Agrawal et al. (2009)). In Zhao et al. (2012), the authors have proposed an energy-efficient data replication scheme for data centre storage. They modelled the system by turning off the underutilized storage servers to minimize energy consumption while keeping one of the replica servers alive for the data object to maintain the availability. Chang and Chang (2008) proposed dynamic data replication in a cluster of data grids. This approach motivates the policymaker for replica management. It continuously collects the information from the cluster heads, which is decided with a set of weights chosen to grant the age of the reading. The policymaker further determines the popularity of a file based on the access frequency. The number of replicas for a file is computed in relationship with the access frequency of all other data in the system to achieve load balancing. The solution given maintains a centralized design pattern, thus keeping a single point of failure. The solution provided supports a centralized design pattern, thus preventing a single point of failure.

Wada et al. (2011) suggested a replication strategy across multiple data centres which minimized the power consumption in the central network. This article was based on linear programming and evaluated the optimal points of replication based on data centre traffic requirements and familiarity of the data objects. Since the power consumption of the collection of ports linearly depends on the traffic load, optimization on the traffic demand brings significant power savings. This article was focused on the replication strategies among various data centres, but not inside through data centre. Analytical performance and simulation studies of distributed databases generally apply queuing systems as the underlying models (Baccelli and Coffman (1982); Coffmann et al. (1981); Acharya and Zdonik (1993); Ciciani et al. (1990)). They presented the model of a fully replicated database by using *the* $M / M / m$ queueing system. The write transactions take entire m servers throughout their service time, while the $m$ server's process read transactions in parallel. This model shared exclusive write and read operations. Baccelli and Coffman (1982) discussed the model where writes get preemptive priority over reading operations. Nelson (2013) depicts the case of a database system with two replications of the data. The performance characteristics of a replicated database under synchronous and non-synchronous policies have been studied by Nelson and Iyer (1985). Bondi and Jin (1996) analyzed a performance model to reproduce and distribute customer records database for database-driven telecommunications services. Berral et al. (2010) proposes another way of optimizing power consumption through data replication in data centres. The optimal placement of the replicas is determined by periodically reading the log file of recent data accesses. By applying the weighted k-means clustering of user locations, the replicas are placed nearer to the centroid of each cluster. There will be a need for migrating a replica from one site to another site if the gain in QoS is more than a defined threshold. The data replication based on the cost model is proposed in Lin et al. (2011). They analyzed the data storage failures as well as data loss probability those are in the direct relationship of the performance of the system and can build a reliable model. The time points when to create replication is determined from data storage reliability function. In this article, the technique presented is different from all replication methodologies available in the literature by (i) the scope of implementing a replication within a data centre and by studying the geographical distribution of the data centre, and (ii) optimized replication strategy, which considers the energy consumption of the system, communication delay and network bandwidth to deploy the network strategy of replication.

## 3. DATA REPLICATION

Data Replication or replicating data is the technique through which valuable information stored in more than one location or site or node. Through it, the reliability of the system and the availability of

data increases. It is the process of copying the data from a database from one server into another server so that all the users can share the same data by keeping the consistency of the data. The result of this is the creation of a distributed database through which the end-users can access the data relevant to their tasks without interfering the work of other end users. Through data replication, the transactions must run on an ongoing basis, and the consistency level of the database must be maintained and must synchronize with the source. But, during data replication, though data is available at different locations, a particular relation has to reside at only one place. There are two types of replication strategies we generally apply in replication. The entire database stores at every site, in case of full replication. There may be partial replication, in which the most frequently used fragment of the database replicates at various locations, and the others remain as it is.

### 3.1 Categories of Data Replication

1. Transactional Replication: In this category, the end-users first receive full initial copies of the database and then get the updates as data changes. The data is copied from the publisher to the receiving database or the subscriber as they were published analogously. Therefore, this type of replication guarantees transactional consistency and generally used in the server to server communications. It merely does not copy the data changes; instead, it replicates the changes in the data consistently and accurately.
2. Snapshot Replication: In this type of replication, data is replicated and distributed as it appears at a particular time instance and monitoring the updates of the data does not occur. The whole snapshot is being yielded and being forwarded to the users. In general, in the case of infrequent data changes, snapshot replication is being used. With each try, it moves many records from one site to another site. It is a slower replication than a transactional replication. For initial synchronization, Snapshot replication is the most promising way of replication between the publisher and the subscriber.
3. Merge Replication: This category of replication merges data from two or more databases into a single database. This category of replication is the most complex case of replication as here both the publisher and the subscriber severally do changes to the database they are using. In client-to-server environments, merge replication is typically used. Through multiple replications, multiple subscribers used to update the same data at different times and notify the changes to the publisher and also to the other subscribers. The above schemes illustrate in Fig. 1.

3.2 Replication Schemes

1. Full Replication: In full replication, the entire database is at all the sites of the distributed system. The system can able to continue its operation until a single location is up in the whole system.

Benefits of full replication:

- The data is highly available and is available until one site is up.
- The performance of the retrieval of any global query is improved, and the execution time is the same for any local site.
- The query execution will be very fast.

Drawbacks of full replication:

- Maintaining the concurrency is difficult in full replication.

- Since a single update needs to be performed at different sites of the distributed system, the update process is slow.

2. No Replication: In No replication strategy, each fragment is being stored only at a single site.

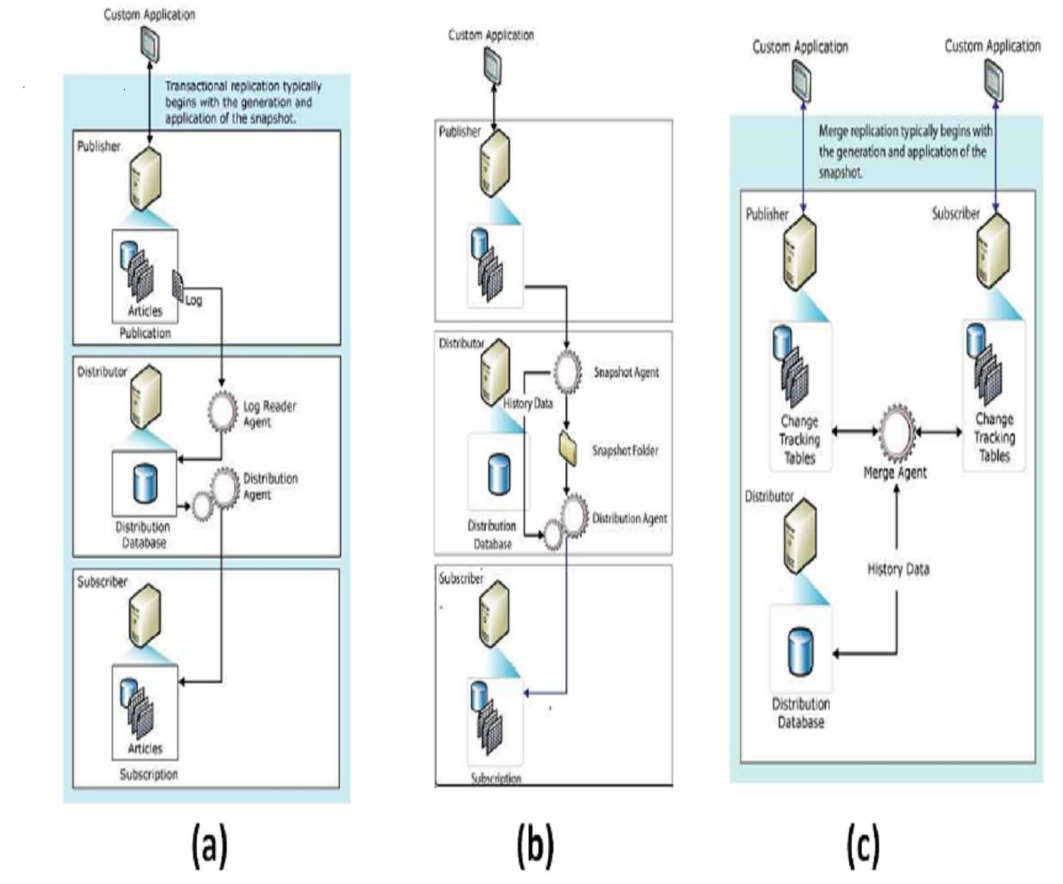
   Benefits of No replication:

- The data is being easily stored in a faster way as there is no replication of the data.
- The information is stored efficiently and more quickly as there is no replication of the data.

   Drawbacks of No replication:

- The query execution is slow as multiple requests from multiple users access the same database server.
- The availability and so the reliability of the system decreases.

**Figure 1. Components used in (a) the Transactional (b) Snapshot (c) Merge Replication**



3. Partial Replication: In partial replication strategy, some of the fragments are being replicated and the others are not replicated. The fragments being replicated to the sites may vary from one to total sites present in the system. The replication schema describes the nature of the sites with the description of the replication fragments.
4. Based on Importance of data: In this category of replication, where to place the fragments depends on the usage and importance of the data.

## 3.3 Replication Protocol

### *3.3.1 Primary Backup Replication*

Fig. 2 shows the principle of the Primary Backup replication and is as follows:

- Primary copy: $q_1$; Backup copies: $q_2$ and $q_3$
- The client sends the Read, writes request to primary copy $q_1$. $q_1$ receives the invocation request and performs the operations required in site $q_1$. At the end of the operation, the change of the state at $q_1$ is forwarded to $q_2$ and $q_3$. The "Ack" msg is received back from the sites $q_2$ and $q_3$ after they have updated their states. The primary site responds to the client after receiving the "Ack" from all the backup sites.

Until the primary does not crash, the order and atomicity of the system is maintained and it is maintained by the primary. The crash of the backups is easy to handle. The crash of the primary is difficult to handle. There are 3 cases to be handled which are shown in Fig. 3:

**Figure 2. Primary Backup Replication**

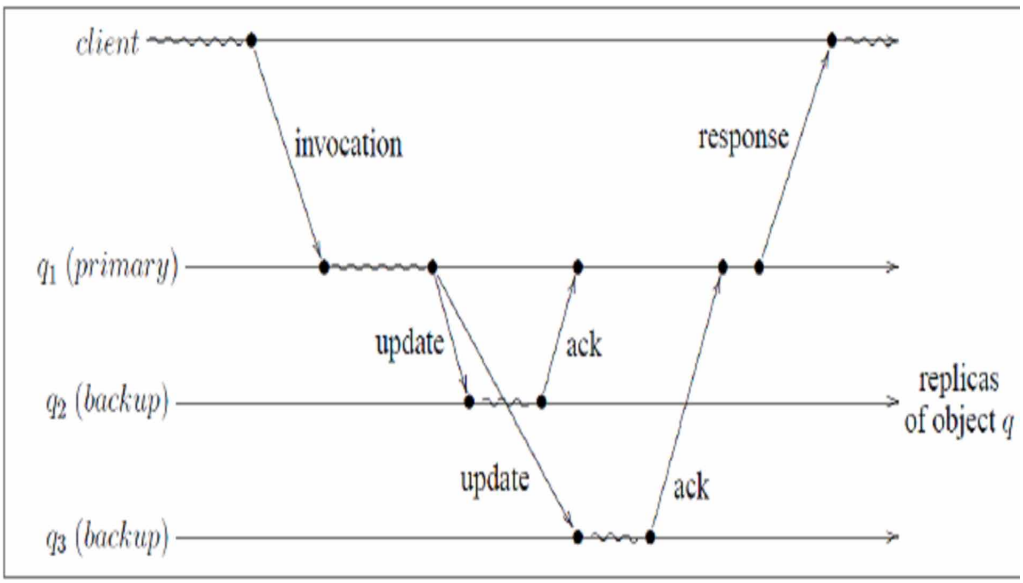
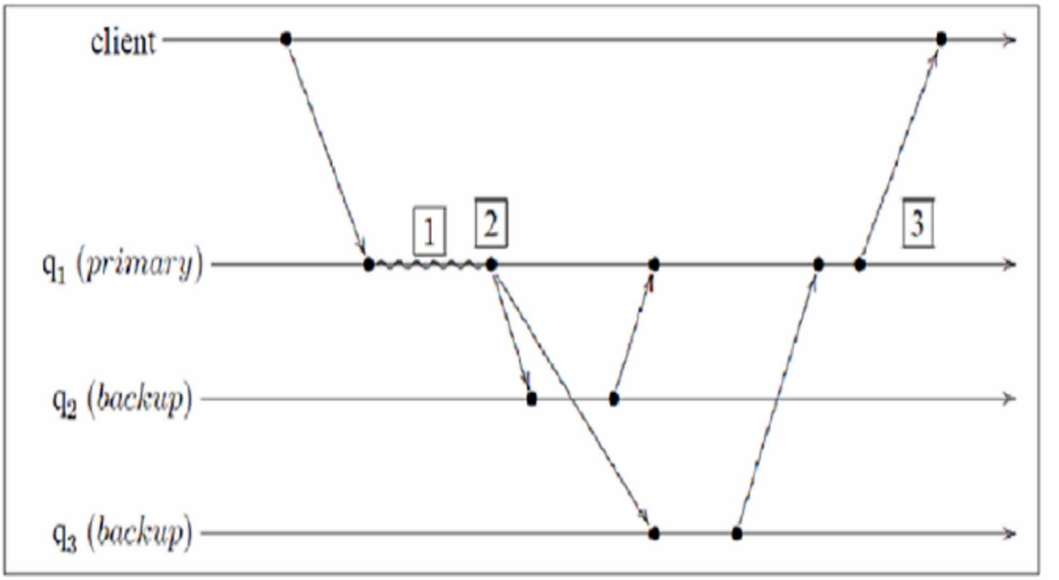


**Figure 3. Recovery of Primary Backup Replication**




1. The primary crashes before sending the "update" msg. In this case, the client will be time-out waiting for the response.
2. The primary crashes while sending the "update" message and before sending the response.
3. After sending the response the primary crashed. In this case, the new primary has to be selected.

### *3.3.2 Active Replication*

With active replication shown in Fig. 4, the client sends its invocation to all replicas:

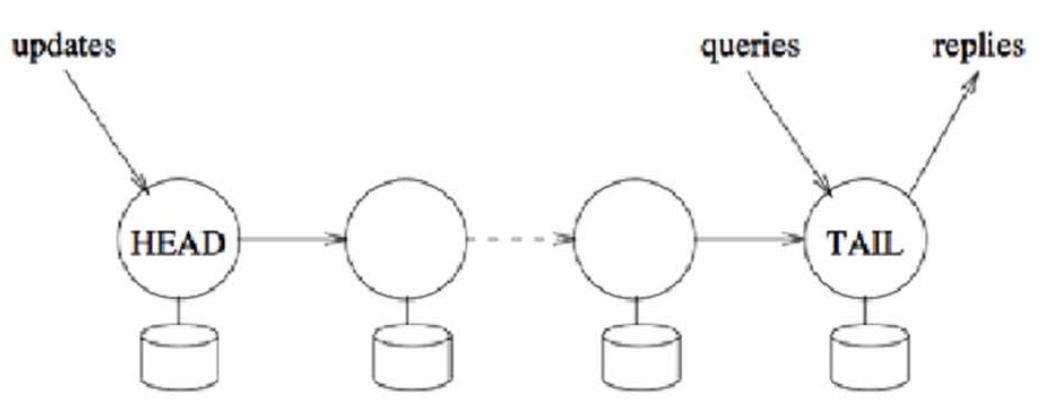
- All replicas handle the invocation and send the response.
- The client waits for the first response since all responses are identical. With the active replication, the crash of a replica is transparent to the client.

### *3.3.3 Chain Replication*

Fig. 5 shows the chain replication. In this protocol, the request types are classified into two types. A query request (read) and an update request (write). The query request is sent to the tail of the chain where it is processed and replied back to the client. Each update request is directed to the head of the chain. The request is processed there and the state changes are forwarded to the next element of the chain. The next element is handled there and then forwarded. This process continues until all the elements of the chain handle it and finally, it is sent to the client by the tail. The protocol considers 3 failure cases:

1. Fail-stop of the head
2. Fail-stop of the tail (primary)
3. Fail-stop of a middle server

**Figure 4. Active Replication**



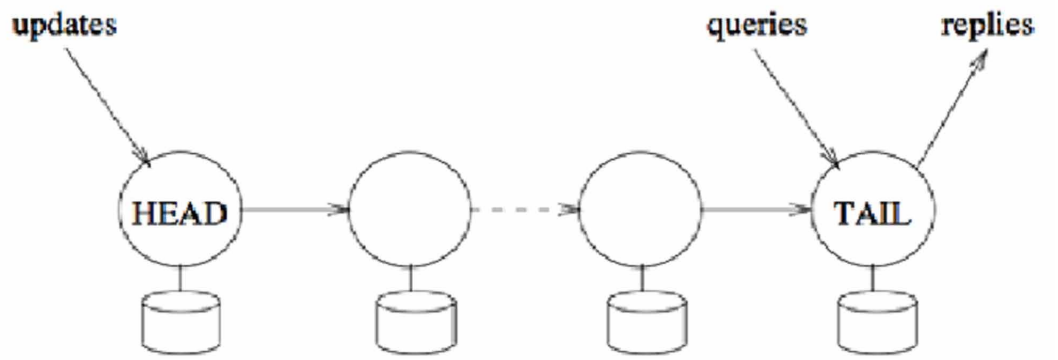

The Paxos service is used for failure detection. The fail-stop of the head is the simplest form of the failure, where the next server in the chain takes the charge as the head. When there is a failure of the tail occurs any other server is allocated as the tail. If other than head and tail server any other server fails i.e., in case of any middle server failure, its two neighbors bypass the middle server and connect. The later server connects with the former server and sends all the requests so that the former server forwards the requests that are dropped during the middle server failed.

### 3.4 Benefits of Data Replication

Through data replication:

- Helps to provide the consistent copy of the database fragments at various sites.
- The availability of the data increase.
- Data replication helps the system's reliability.
- Data replication helps to access the system by multiple users without degrading the system performance.

**Figure 5. Chain Replication**



- If any data redundancy exists in the database, the databases are merged and the slave databases are being updated from incomplete or outdated data.
- Since data fragments are being replicated, if the data is found in the local site where the application is running there is no need of data movement.
- There is a faster execution of queries.

3.4 Drawbacks of Data Replication

- Since the same data is replicated at different sites, more storage is needed for storing the data.
- During the updating of the data, the process is more expensive as it needs to be updated at all the sites where data being replicated.
- Complex measure has to be taken to maintain the consistency of the data at various sites.

## 4. REPLICATION IN THE CLOUD

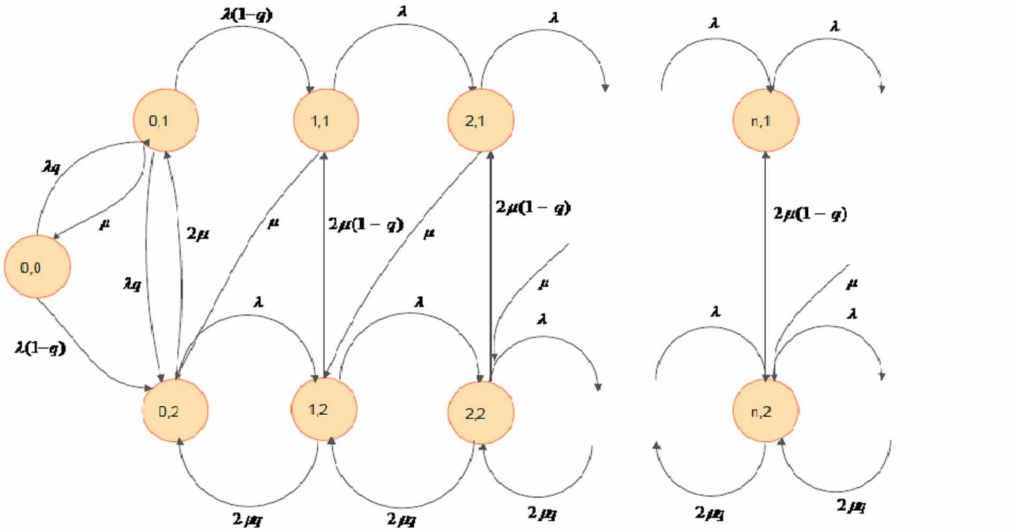
To ensure the better performance of the system, to maintain the SLA of the system; the service providers can take the benefit from a plethora of varieties of choices. Among all these choices, data replication and research data management technologies are well known and used for decades in many distributed systems (Amjad et al. (2012)). The benefits of the data replication, availability, and reliability of the system increase through the replication of the database fragments at multiple sites of the distributed system. The performance of the system also increases through better fault tolerance strategies against possible failures of the system. When a user submits a query to execute the data management system, the execution is carried out by many executions plans (Benoit et al. (2008)). In a large scale environment when all the data are not present in the executing node, the query has to be processed on multiple servers according to inter-operator and intra-operator parallelism. Because of the low network bandwidth and remote data may be dispersed at a geographically separate location, there is a delay in response time may create dissatisfaction among the user. The bottleneck data must be identified using different heuristics and placed before executing the query to maintain the execution time of the query (Bonvin et al. (2011)). There is another crucial decision that must be taken care for the same goal is when to trigger and start the actual replication event. Further, in the data replication decision process, how many replications to create, retrieve the available replicas also has to be dealt with by the replication administrator. The strategic planning for the placement of replicas reduces the latency and improves the response time. All these replication strategies should give economic benefit to the service provider, which is very important as a decisive step for such large scale distributed systems, including cloud computing. A good replication strategy should answer the following questions: (i) Which fragment of data to be replicated (ii) when and where to replicate for better performance of the system (iii) the number of replicas to avoid wasting of storage (iv) When, what and where to replicate so that the performance of the tenant should improve with the economic benefit of the cloud provider. For cloud storage providers, data availability and durability are paramount, as a violation of these SLAs damages the business reputation of the provider to the bottom line (Bai et al. (2013); Tabet et al. (2017)). Data availability and durability typically achieved through under-the-covers replication; that is, data replicated automatically without customer interference or requests. Cloud computing providers provide a high level of fault tolerance by replicating the data at various geographic locations. Amazon S3 cloud storage service replicates data across regions and zones so that applications can even persist even in the case of failures, even at an entire location. The replication layer is being abstracted to the user. Amazon EBS (elastic block store) is more prone to breakdowns as it only replicates data within the same availability zone (Li et al. (2011)).

## 5. MODEL DESCRIPTION OF REPLICATED DATABASE ANALYSIS IN CLOUD

We consider the database system deployed on a cloud where there are replications of data on various sites. The replication on each site is accessible independently and modelled by a virtual machine. The requests to access the database on a website are assumed to be a queueing system in a central location and performs read or write operations. The challenge is to maintain the integrity of the system. We presume that write operations must wait until both copies of the database are available before commencing execution to protect the integrity of both copies of the database. Both copies of the database are supposed to be updated in parallel and released concurrently. Any representation of the database may process the read operation. We assume that both types of requests have to wait in the queue, according to first-come, first-served (FCFS) discipline. Let us consider that requests arrive the system according to Poisson process with the rate $\lambda$ and that the probability a given request is a read (write) is provided by $q\left(1-q\right)$. Service times for both read and write requests are assumed to be exponential with mean $1/\mu$. The read requests can be serviced by any server but write request has to be services by both servers. As we consider that writes are processed in parallel, the total service time equals the maximum of two exponential random variables with parameter $\mu$ for write requests. The cloud system has finite buffer capacity of size $N$. Let $N_n$ be the number of requests waiting in the queue at time $t$ and $I_n$ be the number of replications that are involved in service, $I_n = 0,1,2,$ that are associated in a read or write operation at time $t$. This imply that ( $N_{n,}$ $I_n$ ) forms a bivariate Markov chain with finite state space $\grave{\mathrm{U}} = \left\{\left(0,0\right)\cup\left(i,j\right)|0 \le i \le N, j = 1,2\right\}$. Let $\left(N,I\right)$ be the stationary limit of ( $N_{n,}$ $I_n$ ) and its distribution is denoted as $\pi_{i,j} = P\left\{N = i, I = j\right\}$. Fig. 6 depicts the state transition diagram.

We describe here some of the transitions from the part of the process. In state $\left(0,0\right)$ both servers are idle. In state $\left(2,2\right)$ both servers are busy serving requests, and the request at the head of the queue is corresponding to a not examined request. Hence, it is a read (write) request with probability $q\left(1-q\right)$.

**Figure 6. State Transition Diagram**



Upon service execution, there are two possibilities based on the sort of demand at the head of the queue at rate $2\mu$ . If the head of the queue is a write request, then the next state is $(2,1)$ as the write must wait until all servers are free before beginning execution. If there is a read request, then the next state will be $(1,2)$, and hence the transition rate from $(2,2)$ to $(1,2)$ is given by $2\mu q$ . The transition rate from $(2,2)$ to $(2,1)$ is given by $2\mu(1-q)$. The remaining of the transitions may be interpreted in a similar way. Based on the one-step transition analysis, the steady-state equations can be written as

$$\lambda \pi_{0,0} = \mu \pi_{0,1}, \tag{1}$$

$$(\lambda+\mu)\pi_{0,1} = \lambda q \pi_{0,0} + 2\mu \pi_{0,2}, \tag{2}$$

$$(\lambda+\mu)\pi_{1,1} = \lambda(1-q)\pi_{0,1} + 2\mu(1-q)\pi_{1,2}, \tag{3}$$

$$(\lambda+\mu)\pi_{n,1} = \lambda \pi_{n-1,1} + 2\mu(1-q)\pi_{n,2},\ 2 \le n \le N-1, \tag{4}$$

$$\mu \pi_{N,1} = \lambda\ \pi_{N-1,1} + 2\mu(1-q)\pi_{N,2}, \tag{5}$$

$$(\lambda+2\mu)\pi_{0,2} = \lambda(1-q)\pi_{0,0} + \lambda q \pi_{0,1} + 2\mu q \pi_{1,2} + \mu \pi_{1,1}, \tag{6}$$

$$(\lambda+2\mu)\pi_{n,2} = \lambda \pi_{n-1,2} + 2\mu q \pi_{n+1,2} + \mu \pi_{n+1,1},\ \ 1 \le n \le N-1, \tag{7}$$

$$\mu \pi_{N,2} = \lambda\ \pi_{N-1,2}. \tag{8}$$

To find the performance measures, we need steady-state probabilities at $\pi_{0,0}$, $\pi_{i,1}\ (0 \le i \le N)$ and $\pi_{i,2}\ (1 \le i \le N)$. We may find the steady-state probabilities recursively by solving the system of equations (1) to (8). Solving (8) and (5), we have

$$\pi_{N-1,2} = 2\omega\ \pi_{N,2}\ , \tag{9}$$

$$\pi_{N-1,1} = \omega\ \pi_{N,1} - 2\omega\left(1-q\right)\pi_{N,2}\ , \tag{10}$$

where $\omega = \frac{\mu}{\lambda}$. From equation (7) and (4), we get

$$\pi_{n-1,2} = \left(1+2\omega\right)\pi_{n,2} - 2\omega q\,\pi_{n+1,2} - \omega\,\pi_{n+1,1},\ n = N-1, \ldots, 2, \tag{11}$$

$$\pi_{n-1,1} = \left(1+\omega\right)\pi_{n,1} - 2\omega\left(1-q\right)\pi_{n,2},\ n = N-1, \ldots, 1, \tag{12}$$

We obtain $\pi_{0,1}$ and $\pi_{0,0}$ using equation (3) and (1), as

$$\pi_{0,1} = \frac{1-\omega}{1-q}\pi_{1,1} - 2\,\omega\,\grave{A}_{1,2} \tag{13}$$

$$\pi_{0,0} = \omega\,\grave{A}_{0,1} \tag{14}$$

Applying (2), the probability $\pi_{N,2}$ can be expressed in terms of $\pi_{N,1}$. Finally, the only unknown $\pi_{N,1}$ is found from the normalized condition $\pi_{0,0} + \sum_{n=0}^{N}\pi_{n,1} + \sum_{n=0}^{N}\pi_{n,2} = 1$. This completes the evaluation of steady-state probabilities.

Another way to write the equations (1) to (8) as a quasi-birth death process (QBD) where $\pi Q = 0$. For more details one may refer Nelson (2013). We consider the generator matrix $Q$ with the following structure:

$$Q = \begin{bmatrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ B_{00} & B_{01} & 0 & \cdots & 0 & 0 \\ B_{10} & A_1 & A_0 & \cdots & 0 & 0 \\ 0 & A_2 & A_1 & \cdots & 0 & 0 \\ 0 & 0 & A_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & A_1 & A_0 \\ 0 & 0 & 0 & \cdots & A_2 & C \end{bmatrix}$$

where

$$B_{00}=\begin{bmatrix}-\lambda & \lambda q & \lambda(1-q)\\ \mu & -(\lambda+\mu) & \lambda q\\ 0 & 2\mu & -(\lambda+2\mu)\end{bmatrix},\ B_{01}=\begin{bmatrix}0 & 0\\ \lambda(1-q) & 0\\ 0 & \lambda\end{bmatrix},\ B_{10}=\begin{bmatrix}0 & 0 & \mu\\ 0 & 0 & 2\mu q\end{bmatrix}$$

$$A_0=\lambda I,\quad A_1=\begin{bmatrix}-(\lambda+\mu) & 0\\ 2\mu(1-q) & -(\lambda+2\mu)\end{bmatrix},A_2=\begin{bmatrix}0 & \mu\\ 0 & 2\mu q\end{bmatrix}\text{ and } C=A_0+A_1.$$

To determine the stability condition, let us define $A=A_0+A_1+A_2$ and let $f=(f_1,f_2)$ denote the unique vector satisfying the equations $fA=0$ and $fe=1,$ where $e$ the column vector with all the entries is equal to one of appropriate dimension. Thus, we have $f_1=\dfrac{2(1-q)}{3-2q}$ and $f_2=\dfrac{1}{3-2q}$. The ergodicity condition for QBD process is the drift to higher numbered levels must be strictly less than the drift to lower levels. The following condition must hold for a QBD process to be ergodic if

$$fA_0e<fA_2e\Rightarrow\lambda<\frac{2\mu}{3-2q}.$$

Remark 1 Note that if $q=1$, the model reduces to an $M/M/2$ queue and the traffic intensity is $\rho=\dfrac{\lambda}{2\mu}$. For $q=0$ it is same as an $M/G/1$ queue, that is, $\rho=\dfrac{3\lambda}{2\mu}$, where the average service time is the maximum of two exponential at rate $\mu$.

### 5.1 Computational algorithm

Here, we demonstrate a computational algorithm for finding probabilities at steady-state. The algorithm is based on the analysis of Section 5, that is, we compute all probabilities $\pi_{0,0,}\ \pi_{i,j},\ 0\le i\le N, j=1,2$ in terms of $\pi_{N,1}$. We determine $\pi_{N,1}$ using the normalization condition. The computational algorithm has computational complexity of order $N^3$, where $N$ is the buffer size.
For $n=N,N-1,\ldots,1,$ calculate $\pi_{n,1}$ and $\pi_{n,2}$ in terms of $\pi_{N,1}$ and $\pi_{N,2}$ as follows

$$\pi_{n,1}=g_n\pi_{N,1}+h_n\pi_{N,2},\quad 0\le n\le N, \tag{15}$$

$$\pi_{n,2}=\xi_n\pi_{N,1}+\psi_n\pi_{N,2},\quad 0\le n\le N, \tag{16}$$

where $g_n,h_n,\xi_n,\psi_n$ are computed as follows:

$$g_N=1,\ h_N=0,\ g_{N-1}=\omega,h_{N-1}=-2\omega(1-q),$$
$$\psi_N=1,\psi_{N-1}=2\omega,\ \xi_N=\xi_{N-1}=0,$$
$$g_{n-1}=(1+\omega)g_n-2\omega(1-q)\xi_n,\ n=N-1,N-2,..,2,$$

$$h_{n-1} = \left(1+\omega\right)h_n - 2\omega\left(1-q\right)\psi_n\,, n = N-1, N-2, .., 2,$$
$$\xi_{n-1} = \left(1+\omega\right)\xi_n - 2\omega q\xi_{n+1} - \omega g_{n+1}\,, n = N-1, N-2, .., 2,$$
$$\psi_{n-1} = \left(1+\omega\right)\psi_n - 2\omega q\psi_{n+1} - \omega h_{n+1},\ n = N-1, N-2, .., 1,$$

where $\omega = \dfrac{\mu}{\lambda}$. Calculate $g_0$ and $h_0$ as follows:

$$g_0 = \frac{1+\omega}{1-q}g_1 - 2\omega\xi_1,\quad h_0 = \frac{1+\omega}{1-q}h_1 - 2\omega\psi_1$$

Find $\pi_{0,0}$ in terms of $\pi_{N,1}$ and $\pi_{N,2}$ as follows:

$$\pi_{0,0} = \omega\left(g_0\,\pi_{N,1} + h_0\,\pi_{N,2}\right)$$

Compute $\pi_{N,2}$ in terms of $\pi_{N,1}$ as

$$\pi_{N,2} = K\,\pi_{N,1}$$

where $K = \dfrac{\left(\lambda+\mu-\lambda q\omega\right)g_0 - 2\mu\xi_0}{\left(\lambda q\omega - \lambda - \mu\right)h_0 + 2\mu\psi_0}$

Calculate $\pi_{n,1}$ and $\pi_{n,2}$ in terms of $\pi_{N,1}$ as follows

$$\pi_{n,1} = \left(g_n + Kh_n\right)\pi_{N,1},\ 0 \le n \le N$$
$$\pi_{n,2} = \left(\xi_n + K\psi_n\right)\pi_{N,1},\ 0 \le n \le N$$

Using normalization condition $\pi_{0,0} + \sum_{n=0}^{N}\pi_{n,1} + \sum_{n=0}^{N}\pi_{n,2} = 1$. Determine $\pi_{N,1}$ as

$$\pi_{N,1} = \left[\sum_{n=0}^{N}\left\{\left(g_n + \xi_n\right) + K\left(h_n + \psi_n\right) + \omega\left(g_0 + K\,h_0\right)\right\}\right]^{-1}.$$

The computational complexity of the given algorithm is $\left(N^3\right)$, where $N$ is the maximum capacity of the system.

## 6. PERFORMANCE MEASURES

To investigate the effectiveness of our model, we find some performance measures. The average number of requests in the queue $(L_q)$ and the average number of requests in the system $(L_s)$, respectively, are

$$L_q = \sum_{n=1}^{N} n(\pi_{n,1} + \pi_{n,2}),\ L_s = L_q + \frac{\lambda}{\mu}\left(1 - \pi_{N,1} - \pi_{N,2}\right)$$

The effective arrival rate $\lambda^e$ in the queue is

$$\lambda^e = \lambda\left(\pi_{0,0} + \sum_{n=0}^{N-1}\pi_{n,1} + \sum_{n=0}^{N-1}\pi_{n,2}\right) = \lambda\left(1 - \pi_{N,1} - \pi_{N,2}\right)$$

The probability of loss or blocking is $PBL = \pi_{N,1} + \pi_{N,2}$. Applying Little's rule, the average waiting time of a requests in the system $(W_s)$ and the average waiting time of a requests in the queue $(W_q)$ are obtained as $W_s = L_s / \lambda^e$ and $W_q = L_q / \lambda^e$, respectively. The probability that the number of replication that is involved in a read or write operation is either one or two are respectively, given by

$$P_{B_1} = \sum_{n=0}^{N}\pi_{n,1,} \quad P_{B_2} = \sum_{n=0}^{N-1}\pi_{n,2}$$

## 7. NUMERICAL RESULTS

In this section, we illustrate the numerical tractability of the replicated database system. Fig. 7 depicts the effect of *N* (buffer size) on the probability of blocking (*PBL*) for various values of *q* (the probability that a given request is a read operation). We observe that the loss probability monotonically decreases as buffer size increases and eventually attains to its minimum amount zero as it should be for all values of *q*, as the model turns an infinite-buffer queue. Again one may note that as the probability of reading operations increases, the *P BL* decreases and it yields the lowest when $q = 1$. Figure 8 shows the impact of buffer size (*N*) on the expected waiting time in the system ($W_s$) for the various

**Figure 7.** $N$ **Versus** $PBL$

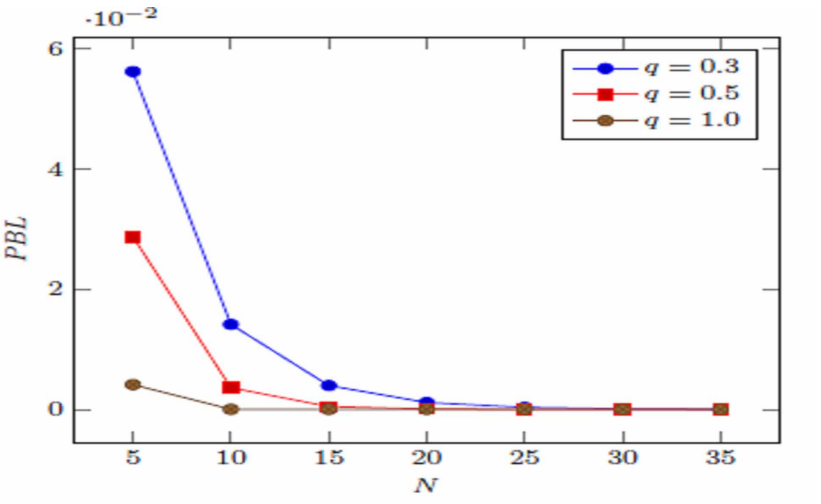


**Figure 8.** $N$ **Versus** $W_s$

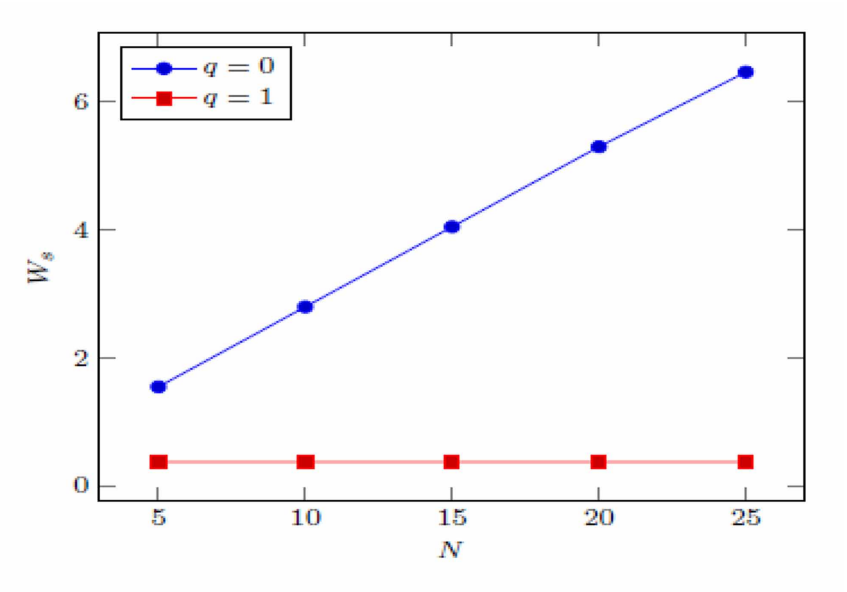
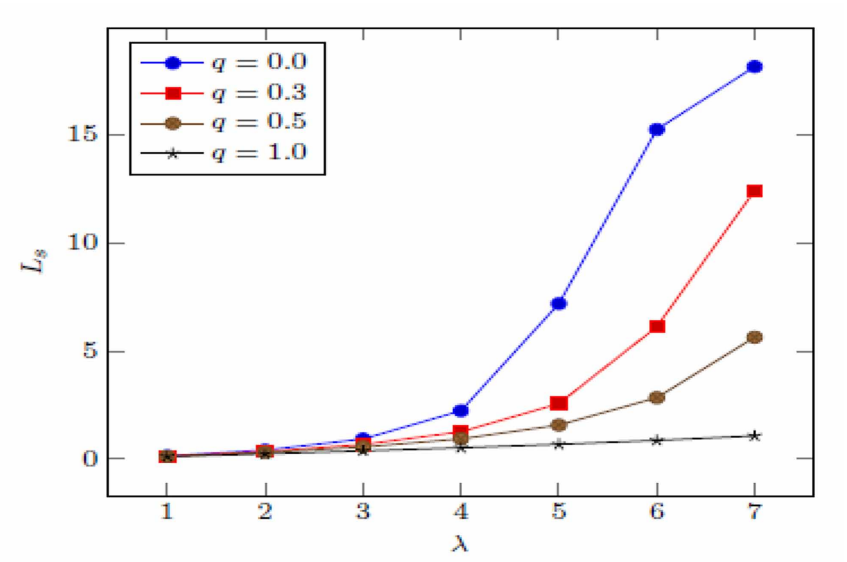



**Figure 9.** $\lambda$ **Versus** $L_s$




probability of reading operations ($q$). One may observe that $W_s$ is almost static for $q = 1$, and it increases monotonically for

$q = 0$.

**Figure 10.** $N$ **Versus** $W_q$

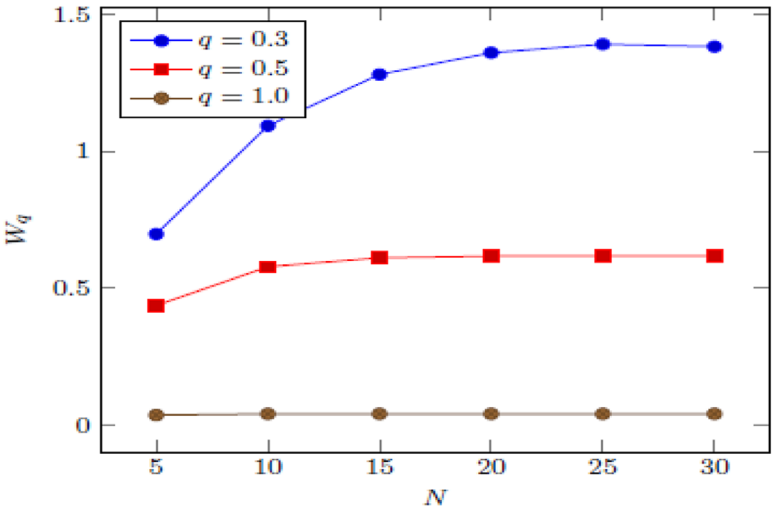



Fig. 9 demonstrates the effect of arrival rate (ë) on the average number of customers in the system ( $L_s$ ) for various values of $q$. For a fixed $q$, $L_s$ increases after the value of $\lambda = 3$ . The average number of customers in the system is almost static when $q = 1$. Fig. 10 illustrates the tradeoff between the buffer size and the average waiting time of requests in the queue ( $W_q$ ) for various values of $q$. For fixed $q$, the average waiting time of requests in the queue increases for lower values of $N$, but for higher values of $N$, the changes are almost stationary. When $q = 1$, the model reduces to an $M / M / 2$ queueing system, that is, the requests arrive in the system is having read options only. From Figs. 7-10, we have seen that for $q = 1$ the performance results are almost static for various parameters.

In Table 1, the results of the queue-length distributions are given for various values of $q$. One may observe from the table that queue-length distributions decrease as several request increases. The variation in the average number of customers in the system ( $L_s$ ) for different values of the buffer size ($N$) and the probability of reading operation ($q$) is shown in Fig. 11. We varied the probability of reading operation $q$ from 0 to 1, while the buffer size $N$ is varied from 5 to 25. For a fixed buffer size, the average number of customers in the system decreases when the probability of reading operation increases. Further, with a fixed number of $q$, the average number of customers in the system increases when the buffer size increases. To accomplish this, we may meticulously set up a suitable buffer size and the probability of reading operation to assure the minimum average number of customers in the system. Fig. 12 presents the dependence of the blocking probability on the buffer size $N$ varying from 5 to 25 and the probability of reading operation $q$ ranging from 0 to 1. We observe that for fixed $q$, the loss probability decreases as the buffer size increases. Further, with a fixed buffer size, it falls when the probability of reading operation increases. Hence we can set up a suitable buffer size and the sufficient probability of reading operation in the system to have a lower blocking probability.

## 8. CONCLUSION

Cloud storage services draw more attention to their high scalability and availability at a low cost because of the increasing popularity of cloud computing. Replication is one of the performance-

**Table 1. Queue-length distributions**

| $\lambda = 2, \mu = 3, N = 30$ | | | | | | |
|---|---|---|---|---|---|---|
| $n$ | $q = 0.3$ | | $q = 0.5$ | | $q = 0.7$ | |
| | $\pi_{n1}$ | $\pi_{n,2}$ | $\pi_{n1}$ | $\pi_{n,2}$ | $\pi_{n1}$ | $\pi_{n,2}$ |
| 0 | 0.121827 | 0.091371 | 0.190474 | 0.111105 | 0.252252 | 0.121922 |
| 1 | 0.078680 | 0.053300 | 0.075395 | 0.062171 | 0.051024 | 0.057649 |
| 2 | 0.063452 | 0.03934 | 0.053242 | 0.03847 | 0.031054 | 0.029567 |
| 3 | 0.050761 | 0.030140 | 0.036239 | 0.024902 | 0.018147 | 0.015905 |
| 4 | 0.039975 | 0.023358 | 0.024345 | 0.016416 | 0.010418 | 0.008774 |
| 5 | 0.031300 | 0.018213 | 0.016276 | 0.010897 | 0.005932 | 0.004902 |
| 6 | 0.024474 | 0.014228 | 0.010863 | 0.007253 | 0.003365 | 0.002756 |
| 7 | 0.019130 | 0.011121 | 0.007245 | 0.004833 | 0.001905 | 0.001554 |
| 8 | 0.014953 | 0.008691 | 0.004830 | 0.003221 | 0.001078 | 0.000877 |
| 9 | 0.011687 | 0.006793 | 0.003220 | 0.002147 | 0.000610 | 0.000496 |
| 10 | 0.009134 | 0.005309 | 0.002147 | 0.001431 | 0.000345 | 0.000280 |
| 15 | 0.002664 | 0.001548 | 0.000283 | 0.000188 | 0.000020 | 0.000016 |
| 20 | 0.000777 | 0.000452 | 0.000037 | 0.000025 | 0.000001 | 0.000001 |
| 30 | 0.000079 | 0.000016 | 0.000001 | 0.000000 | 0.000000 | 0.000000 |
| Sum | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 |
| $\pi_{0,0}$ | 0.182741 | | 0.285710 | | 0.378378 | |

improving techniques for a cloud storage system which used broadly. An efficient replica placement policy balances the load and which significantly boost the overall performance of the cloud system. These motivate us to study the replica placement policy to distribute workload across cloud nodes competently. In this paper, we analyzed the performance characteristics of a replicated database in cloud computing data centres which improves QoS by reducing communication delays. Moreover, our proposed model considers database system deployed on a cloud where there are replications of data on two sites. We formulated a theoretical queueing model of the replicated system by considering the arrival process as a Poisson distribution for both types of the client request, such as read and write applications. We solved the proposed model with the recursive method, and the relevant performance matrices are derived. Numerical results show that to assure the minimum average number of customers in the system, and we may meticulously set up a suitable buffer size and $q$. Furthermore, as a part of future work, we aim to consider the performance of a cloud database system where there are $n$ replications of the data on the frameworks of fog devices for better usage of the computing devices while satisfying the high-reliability constraint.

**Figure 11.** $L_s$ **Versus** $\lambda$ **Versus** $d$

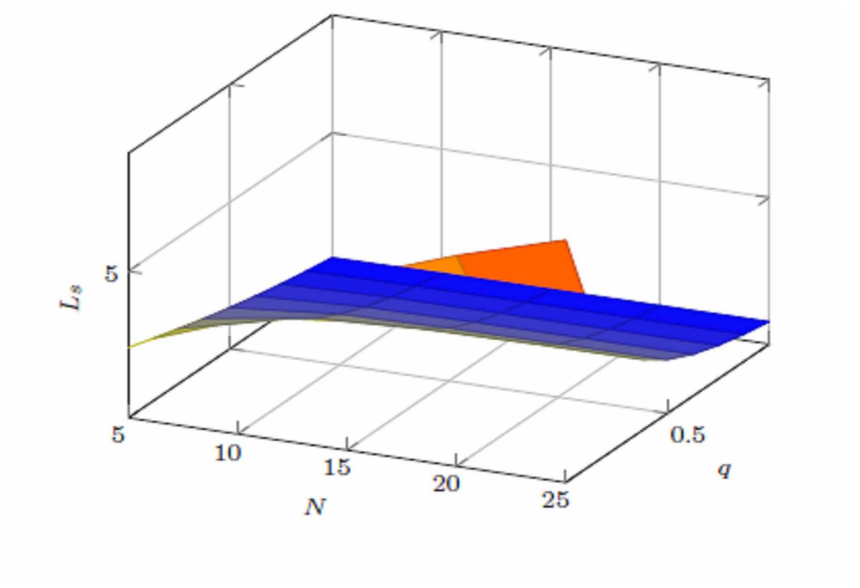
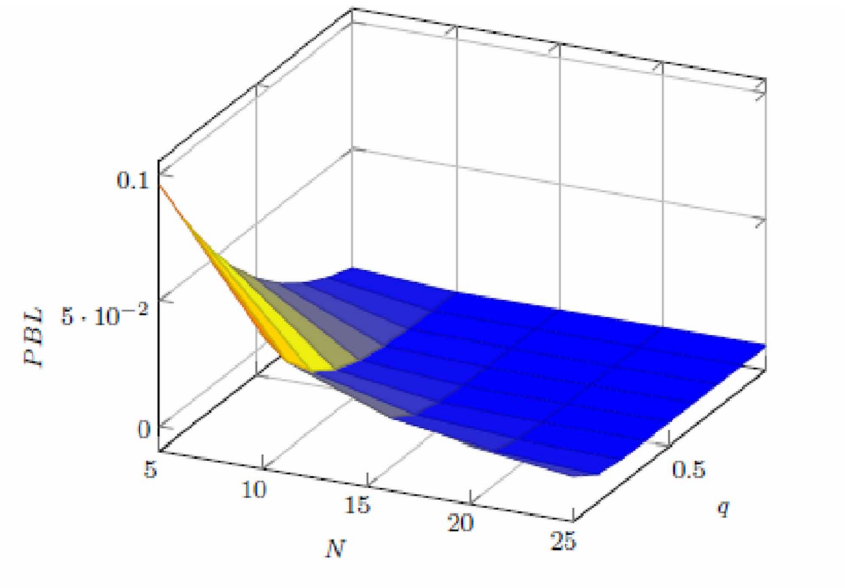



**Figure 12.** $PBL$ **Versus** $N$ **Versus** $q$




## FUNDING AGENCY

The publisher has waived the Open Access Processing fee for this article.

## REFERENCES

Acharya & Zdonik. (1993). *An efficient scheme for dynamic data replication.* Academic Press.

Agrawal, D., El Abbadi, A., Emekci, F., & Metwally, A. (2009) Database management as a service: Challenges and opportunities. In *2009 IEEE 25th International Conference on Data Engineering*. IEEE.

Amjad, T., Sher, M., & Daud, A. (2012). A survey of dynamic replication strategies for improving data avail ability in data grids. *Future Generation Computer Systems*, *28*(2), 337–349. doi:10.1016/j.future.2011.06.009

Avgerinou, M., Bertoldi, P., & Castellazzi, L. (2017). Trends in data centre energy consumption under the European code of conduct for data centre energy efficiency. *Energies*, *10*(10), 1470. doi:10.3390/en10101470

Baccelli, F., & Coffffman, E. G. (1982). A data base replication analysis using an m/m/m queue with service interruptions. In *ACM SIGMETRICS Performance Evaluation Review* (Vol. 11, pp. 102–107). ACM. doi:10.1145/1035293.1035309

Bai, X., Jin, H., Liao, X., Shi, X., & Shao, Z. (2013) Rtrm: A response time-based replica management strategy for cloud storage system. In *International Conference on Grid and Pervasive Computing*. Springer. doi:10.1007/978-3-642-38027-3_13

Benoit, A., Rehn-Sonigo, V., & Robert, Y. (2008). Replica placement and access policies in tree networks. *IEEE Transactions on Parallel and Distributed Systems*, *19*(12), 1614–1627. doi:10.1109/TPDS.2008.25

Berral, J. L., Goiri, I., & Nou, R. (2010). Towards energy-aware scheduling in data centers using machine learning. In *Proceedings of the 1st International Conference on energy-Efficient Computing and Networking*. ACM. doi:10.1145/1791314.1791349

Bertoldi, P., Avgerinou, M., & Castellazzi, L. (2017). *Trends in data centre energy consumption under the european code of conduct for data centre energy efficiency*. European Commission-DG JRC, Publications Office of the European Union.

Bondi, A. B., & Jin, V. (1996). A performance model of a design for a minimally replicated distributed database for database-driven telecommunications services. *Distributed and Parallel Databases*, *4*(4), 295–317. doi:10.1007/BF00119337

Bonvin, N., Papaioannou, T. G., & Aberer, K. (2011). Autonomic sla-driven provisioning for cloud applications. In *Proceedings of the 2011 11th IEEE/ACM international symposium on cluster, cloud and grid computing*. IEEE Computer Society. doi:10.1109/CCGrid.2011.24

Buyya, R., Yeo, C. S., & Venugopal, S. (2008). Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *2008 10th IEEE International Conference on High Performance Computing and Communications*. IEEE.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, *25*(6), 599–616. doi:10.1016/j.future.2008.12.001

Cao, Q., Wei, Z. B., & Gong, W. M. (2009). An optimized algorithm for task scheduling based on activity based costing in cloud computing. In *2009 3rd International Conference on Bioinformatics and Biomedical Engineering*. IEEE. doi:10.1109/ICBBE.2009.5162336

Cecchet, E., Singh, R., Sharma, U., & Shenoy, P. (2011). Dolly: virtualization-driven database provisioning for the cloud. In *ACM SIGPLAN Notices* (Vol. 46, pp. 51–62). ACM. doi:10.1145/1952682.1952691

Chang, R. S., & Chang, H. P. (2008). A dynamic data replication strategy using access-weights in data grids. *The Journal of Supercomputing*, *45*(3), 277–295. doi:10.1007/s11227-008-0172-6

Ciciani, B., Dias, D. M., & Yu, P. S. (1990). Analysis of replication in distributed database systems. *IEEE Transactions on Knowledge and Data Engineering*, *2*(2), 247–261. doi:10.1109/69.54723

Coffmann, E. G., Gelenbe, E., & Plateau, B. (1981). Optimization of the number of copies in a distributed system. *IEEE Transactions on Software Engineering*, *7*(1), 78–84. doi:10.1109/TSE.1981.234510

Ebadi, Y., & Jafari Navimipour, N. (2019). An energy-aware method for data replication in the cloud environments using a Tabu search and particle swarm optimization algorithm. *Concurrency and Computation*, *31*(1), e4757. doi:10.1002/cpe.4757

Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P. P., Kolodziej, J., Balaji, P., Zeadally, S., Malluhi, Q. M., Tziritas, N., Vishnu, A., Khan, S. U., & Zomaya, A. (2016). A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing*, *98*(7), 751–774. doi:10.1007/s00607-014-0407-8

Hu, M., & Veeravalli, B. (2013). Requirement-aware strategies for scheduling real-time divisible loads on clusters. *Journal of Parallel and Distributed Computing*, *73*(8), 1083–1091. doi:10.1016/j.jpdc.2013.03.013

Hussain, H., Malik, S. U. R., Hameed, A., Khan, S. U., Bickler, G., Min-Allah, N., Qureshi, M. B., Zhang, L., Yongji, W., Ghani, N., Kolodziej, J., Zomaya, A. Y., Xu, C.-Z., Balaji, P., Vishnu, A., Pinel, F., Pecero, J. E., Kliazovich, D., Bouvry, P., & Rayes, A. et al. (2013). A survey on resource allocation in high performance distributed computing systems. *Parallel Computing*, *39*(11), 709–736. doi:10.1016/j.parco.2013.09.009

Khalaj, A. H., Scherer, T., & Halgamuge, S. K. (2016). Energy, environmental and economical saving potential of data centers with various economizers across australia. *Applied Energy*, *183*, 1528–1549. doi:10.1016/j.apenergy.2016.09.053

Kliazovich, D., Bouvry, P., & Khan, S. U. (2012). Greencloud: A packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, *62*(3), 1263–1283. doi:10.1007/s11227-010-0504-1

Koomey, J. (2011). *Growth in data center electricity use 2005 to 2010*. A report by Analytical Press.

Li, W., Yang, Y., & Yuan, D. (2011). 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*. IEEE. doi:10.1109/DASC.2011.95

Lin, B., Li, S., Liao, X., Wu, Q., & Yang, S. (2011). estor: Energy efficient and resilient data center storage. In *2011 International Conference on Cloud and Service Computing*. IEEE. doi:10.1109/CSC.2011.6138549

Liu, L., Wang, H., Liu, X., Jin, X., He, W. B., Wang, Q. B., & Chen, Y. (2009). Greencloud: a new architecture for green data center. In *Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session*. ACM. doi:10.1145/1555312.1555319

Nelson, R. (2013). *Probability, stochastic processes, and queueing theory: The mathematics of computer performance modeling*. Springer Science & Business Media.

Nelson, R. D., & Iyer, B. R. (1985). Analysis of a replicated data base. *Performance Evaluation*, *5*(3), 133–148. doi:10.1016/0166-5316(85)90008-2

Ni, J., & Bai, X. (2017). A review of air conditioning energy performance in data centers. *Renewable & Sustainable Energy Reviews*, *67*, 625–640. doi:10.1016/j.rser.2016.09.050

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseffff, L., & Zagorodnov, D. (2009). The eucalyptus open-source cloud-computing system. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE Computer Society. doi:10.1109/CCGRID.2009.93

Sakr, S., Liu, A., Batista, D. M., & Alomari, M. (2011). A survey of large scale data management approaches in cloud environments. *IEEE Communications Surveys and Tutorials*, *13*(3), 311–336. doi:10.1109/SURV.2011.032211.00087

Sasikumar, K., & Vijayakumar, B. (2020). An Efficient Multi-Objective Model for Data Replication in Cloud Computing Environment. *International Journal of Enterprise Information Systems*, *16*(1), 69–91. doi:10.4018/IJEIS.2020010104

Sindhu, S., & Mukherjee, S. (2011) Efficient task scheduling algorithms for cloud computing environment. In *International Conference on High Performance Architecture and Grid Computing*. Springer. doi:10.1007/978-3-642-22577-2_11

Soltesz, S., P¨otzl, H., Fiuczynski, M. E., Bavier, A., & Peterson, L. (2007). Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. In *ACM SIGOPS Operating Systems Review* (Vol. 41, pp. 275–287). ACM. doi:10.1145/1272996.1273025

Sotomayor, B., Montero, R. S., Llorente, I. M., & Foster, I. (2009). Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing*, *13*(5), 14–22. doi:10.1109/MIC.2009.119

Srikantaiah, S., Kansal, A., & Zhao, F. (2008). *Energy aware consolidation for cloud computing*. Academic Press.

Sun, W., & Sugawara, T. (2011). Heuristics and evaluations of energy-aware task mapping on heterogeneous multiprocessors. In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*. IEEE. doi:10.1109/IPDPS.2011.209

Tabet, K., Mokadem, R., Laouar, M. R., & Eom, S. (2017). Data replication in cloud systems: A survey. *International Journal of Information Systems and Social Change*, *8*(3), 17–33. doi:10.4018/IJISSC.2017070102

Travostino, F., Daspit, P., Gommans, L., Jog, C., De Laat, C., Mambretti, J., Monga, I., Van Oudenaarde, B., Raghunath, S., & Wang, P. Y. (2006). Seamless live migration of virtual machines over the man/wan. *Future Generation Computer Systems*, *22*(8), 901–907. doi:10.1016/j.future.2006.03.007

Wada, H., Fekete, A., Zhao, L., Lee, K., & Liu, A. (2011). *Data consistency properties and the trade-offs in commercial cloud storage: the consumers' perspective* (Vol. 11). CIDR.

Zhao, L., Sakr, S., Fekete, A., Wada, H., & Liu, A. (2012). Application-managed database replication on virtualized cloud environments. In *2012 IEEE 28th International Conference on Data Engineering Workshops*. IEEE. doi:10.1109/ICDEW.2012.77

*Sudhansu Shekhat Patra is currently an Associate Professor in the School of Computer Application, KIIT University, Bhubaneswar, India. He received his Master degree in Computer Application from Motilal Nehru National Institute of Technology, Allahabad, India, M.Tech(Computer Science & Engg) from Utkal University, Bhubaneswar, India and Ph.D. in Computer Science from KIIT University, Bhubaneswar, India. His research interests include grid computing, Cloud Computing, Algorithms. He is a life member of Indian Society for Technical Education.*

*Veena Goswami is currently a Professor in the School of Computer Applications, Kalinga Institute of Industrial Technology, Bhubaneswar, India. She received her PhD from Sambalpur University, India and then worked as a Research Associate at the Indian Institute of Technology, Kharagpur for two years. Her research interests include continuous- and discrete-time queues. She has published research articles in INFORMS Journal on Computing, Computers and Operations Research, RAIRO Operations Research, Computers and Mathematics with Applications, Applied Mathematical Modelling, Computers & Industrial Engineering, Applied Mathematics and Computation, etc.*