ISSN 2723-6471

Dynamic Replica Management Strategy Based on Data Accessing Popularity for Load Balancing and Optimizing Network Performance in Cloud Storage

S. Thavamani[1,*], Siti Sarah Maidin[2], S. T. Varun[3]

[1,3]*Sri Ramakrishna College of Arts and Science (Autonomous), Coimbatore-641006, Tamilnadu, India.*

[2]*Centre for Data Science and sustainable Technologies, Faculty of Data Science and Information Technology, INTI, International University, 71800 Nilai, Negeri Sembilan, Malaysia*

(Received: January 22, 2025; Revised: February 25, 2025; Accepted: March 29, 2025; Available online: April 30, 2025)

Abstract

Network performance plays a vital role in organizational efficiency, where large volumes of data, fast transmission, and low latency significantly enhance productivity and reduce downtime. Cloud storage offers a service model that enables remote data management and efficient content distribution. In such systems, data replication is widely used to improve availability, reliability, fault tolerance, and throughput. However, static replication policies often allocate replicas during system initialization, failing to adapt to the dynamic and heterogeneous nature of cloud environments. These environments are susceptible to challenges such as data loss, node failures, and fluctuating demand, which can degrade service quality. To address this, we propose a dynamic replica management strategy that considers data popularity, active peer participation, and peer capacity. Virtual peers are grouped into strong, medium, and weak clusters based on their weight values, which are derived from bandwidth, CPU speed, memory size, and access delay. Content is categorized into Class I, II, and III based on access frequency. Highly popular data (Class I) is replicated in strong clusters, while less frequently accessed data is placed in medium and weak clusters. A hierarchical routing mechanism ensures that queries are directed to the appropriate cluster. The proposed system was implemented and evaluated through simulations. Results show up to 25% improvement in throughput, 20% reduction in packet drops, 97% query efficiency, and decreased bandwidth utilization under high load. By maintaining optimal replica counts without compromising availability, the system supports cloud SLA compliance while minimizing overhead. This solution is aligned with the ninth UN Sustainable Development Goal: Industry, Innovation, and Infrastructure.

Keywords: Cloud storage systems, Dynamic Replication, Weight Value, Classifying Data, Computing, Sustainable Development Goal, Process Innovation, Product Innovation

1. Introduction

Network performance plays a vital role in organizational efficiency. Fast data transmission and low latency contribute to higher productivity, reduced downtime, and improved overall performance in today's fast-paced digital environment [1]. Cloud storage is a service model that operates across multiple network services, connecting devices and systems to remote data centers. It enables users to manage and access data from anywhere, at any time, and using any internet-connected device. Efficient content distribution is essential to ensure optimal performance and resource utilization. However, due to the dynamic nature of cloud environments, data unavailability can negatively impact service quality [2], [3].

Cloud computing refers to the delivery of computing services over the internet, whether centralized or distributed. It can be deployed through large-scale data centers, virtualization, or physical infrastructure. The peer-to-peer (P2P) model is a widely adopted approach for designing cloud applications due to its decentralized nature and ability to facilitate efficient content sharing [4], [5], [6]. Virtual machines (VMs) are virtualized instances of computing systems that can run operating systems and applications in cloud environments. They rely on hypervisor software to allocate computing resources, enabling users to share content, storage, and processing power as part of infrastructure-as-a-

***Corresponding author: S. Thavamani (thavamani@srcas.ac.in)**
DOI: https://doi.org/10.47738/jads.v6i2.674

service (IaaS) models [7], [8], [9]. P2P networks are especially valued for their scalability, fault tolerance, and efficiency in resource sharing [2], [3], [10].

The emergence of internet-scale distributed systems—including storage administration, entertainment file sharing, and large-scale database management—has driven extensive research into efficient distributed computing architectures. P2P computing presents a promising alternative to traditional client-server models, offering decentralized support for file sharing and content delivery. As web content and streaming media grow in importance, ensuring efficient content distribution in cloud storage becomes increasingly critical. Maintaining high Quality of Service (QoS) while managing balanced resources presents a significant challenge [11].

QoS refers to the capability of a network to guarantee performance metrics such as bandwidth allocation, latency, and reliability for specific applications or protocols. It improves availability and reduces access costs through techniques such as replication. Replication is essential for handling node failures, mitigating network congestion, improving system scalability, and balancing loads [2], [3], [10].

This study proposes a dynamic replica placement strategy within P2P networks to enhance QoS. In this model, each peer can act as both a client and a server. Server peers allocate part of their storage to hold replicas. When a request is made, a server will respond immediately if the requested file is available in its local storage. If not, the file is retrieved from another server peer in the group, which incurs higher access costs. If no replica is available within the group, the request is forwarded to the origin server, increasing latency and cost. To reduce load and improve responsiveness, frequently requested content should be replicated closer to the requesting peers.

The challenge lies in determining how many replicas to create and where to place them. The proposed replica management algorithm considers content demand to optimize distribution. Effective replication reduces access latency, minimizes bottlenecks, enhances system performance, and ensures fault tolerance. However, excessive replication may lead to inefficient bandwidth usage, storage waste, and increased operational costs, including server rental, energy consumption, and cooling.

2. Literature Review

Numerous studies have addressed content replication and resource optimization in peer-to-peer (P2P) networks [2], [3], [9], [12], [13]. These works adopt different models and techniques with shared objectives, including reducing bandwidth usage, enhancing data availability, minimizing access latency, and improving load balancing [14], [15], [16], [17], [18].

In [2], the author applies a machine learning classification approach to categorize user tasks based on size and log file data. Virtual machines (VMs) are grouped according to CPU and RAM utilization, enabling various workloads to share VM resources efficiently. This method aims to reduce job rejections, increase dynamic resource allocation, and improve service quality standards, ultimately optimizing resource utilization and user satisfaction.

The work in [12] proposes a distributed approximation technique to cache and replicate the most frequently accessed data items. This technique enhances replication efficiency for popular objects within a distributed group, reducing access latency and network bandwidth usage. Unlike centralized approaches, this method results in only a marginal drop in system performance and avoids the complexity of calculating the optimal number of object copies. By focusing on popular data replication, the system achieves better performance and user experience without introducing excessive computational overhead.

In [8], a proactive replication technique is introduced to improve the discoverability of rare objects. The technique enables peers to probe for content upon joining the network, allowing informed decisions on whether to replicate their data. This approach increases the visibility of less common objects and enhances search efficiency while potentially reducing communication overhead by adjusting detection cycle intervals.

The study in [1] introduces a coverage perception interference model for Wireless Networked Systems (WNS), focusing on the challenges of homogeneous interference in regions of interest. Although the model performs well in high-security environments, its scalability to larger and more complex WNS poses challenges. As the network expands, the computational burden of processing and integrating sensor data increases. To address this, optimization strategies

such as hierarchical or distributed processing could be explored to maintain responsiveness and reliability across large-scale deployments.

In [4], an algorithm is proposed that categorizes content into two groups based on weight values and access patterns, then divides peers into strong and weak clusters. This enhances system scalability while reducing network traffic and access latency. However, the strategy tends to place more copies of certain objects in strong clusters, resulting in increased costs related to memory, bandwidth, and redundancy. In contrast, weak clusters may host fewer replicas, leading to performance bottlenecks and longer response times for some users. Despite this trade-off, the algorithm effectively boosts scalability [20], [1].

The flooding technique, a common initial search method in unstructured P2P networks, generates a high volume of duplicate messages and incurs significant search overhead. It also offers users limited control over the message volume. In comparison, studies suggest that random walk-based search strategies are substantially more efficient than flooding [21], [22], [23].

3. Algorithm Overview and Workflow Architecture of Proposed System

The proposed strategy introduces a dynamic replica management system in a cloud-based peer-to-peer (P2P) environment. Peers in the network are categorized into three distinct clusters—strong, medium, and weak—based on their calculated weight values. These weight values are determined using key performance parameters such as available storage capacity, CPU speed, memory size, and access latency. By evaluating these parameters, the system effectively differentiates the computational and storage capabilities of each peer.

Simultaneously, content objects are classified into three classes: Class I, Class II, and Class III, depending on their access popularity. Class I represents the most frequently accessed data, while Class III includes the least accessed content. Based on this classification, the system replicates Class I content across strong peers, Class II content across medium peers, and Class III content across weak peers. This approach ensures that highly demanded content is hosted on peers with higher performance capacities, thereby optimizing resource allocation and enhancing system responsiveness.

Routing within the cloud network is executed hierarchically, wherein queries are directed only to the relevant cluster (strong, medium, or weak) based on the content class. This targeted broadcasting mechanism minimizes unnecessary data transfer, reduces overall bandwidth consumption, lowers latency, and simplifies system maintenance. As a result, the proposed model enhances several Quality of Service (QoS) metrics, including throughput, query efficiency, bandwidth utilization, and packet delivery reliability.

The workflow architecture underlying this strategy follows a logical sequence of operations. It begins with the computation of peer weight values, followed by the determination of maximum and minimum thresholds used to classify peers into appropriate clusters. Subsequently, the system evaluates the popularity of data objects and assigns them to the corresponding content class. Each object is then mapped to the appropriate peer cluster for replication based on its popularity. Finally, the system serves client requests efficiently by providing access to the closest available replica, while also creating new copies of objects when needed to maintain performance consistency and scalability. This architecture is visually depicted in figure 1, which outlines the end-to-end flow of the dynamic replica management strategy.

 ISSN 2723-6471

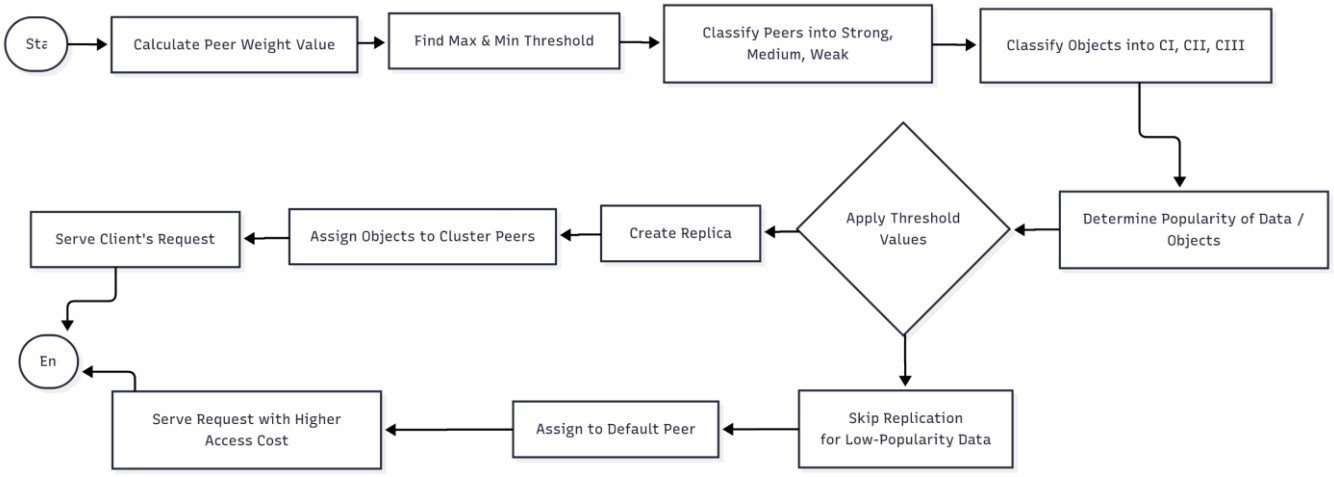

Figure 1. Workflow Architecture of Dynamic Replica Management Strategy

4. Proposed System

The proposed replication strategy groups the virtual machines or peers into strong, medium, and weak nodes according to their weight vector values using an intelligent dynamic clustering algorithm. To suit client needs, queries are broadcast hierarchically, guaranteeing effective data replication and compliance with SLA (Service Level Agreement) contracts. Cloud computing network infrastructure consisting of N virtual machines (VM). Along the part of the overlay, each virtual machine in the network functions as a server while responding to query requests which come from clients outside of the overlay network. As an example, every virtual machine may function as a web server with the overlay linking the servers and the clients being web browsers on remote machines requesting content from the servers. It is assumed that each virtual machine always stores one copy of its own content which it serves to clients and that has additional storage space to store the replicated content from other nodes which can also serve. The object is associated with an authoritative origin server (OS) in the network where the content provider makes the updates to the object. The object copy located at the origin server is called the origin copy and an object copy at any remaining server is called a replica. The steps involved in the proposed replica strategy is as follows

4.1. Initialization of Virtual Peers (Virtual Nodes)

The proposed system begins by initializing all virtual peers with weight vector values. Each virtual peer, denoted as $VPiW$, is characterized by several parameters: bandwidth (BWi), CPU speed ($CPU\ Si$), access delay (ADi), and memory size (MZi). These parameters are used to compute the peer's weight value, which reflects its overall capability in the system. The weight for each peer is calculated using the formula:

$$VPiW = \frac{BWi+CPU\ Si+MZi}{ADi} \quad (1)$$

Here, $VPiW$ represents the calculated weight of the $i-th$ virtual peer. Once computed, the weight values of all virtual peers are sorted in descending order. This ordering helps determine which peers are most suitable for storing frequently accessed content, and forms the basis for dynamic clustering.

4.2. Dynamic Clustering (Grouping Peers) Algorithm

Based on the calculated weight values, the virtual peers are dynamically grouped into three clusters: strong, medium, and weak. This classification is guided by two threshold values. A high-level minimum threshold β1 and a low-level maximum threshold β2 are derived from a base threshold β, such that β1=β+10 and β2=β−10. These threshold values create boundaries to categorize peers with varying performance capabilities.

Peers with weight values greater than β1 are grouped into the strong cluster, representing the most capable nodes in terms of bandwidth, processing power, and storage. Conversely, peers with weight values less than β2 fall into the weak cluster, suitable for handling less critical or rarely accessed data. The remaining peers, with weight values between β2 and β1, are assigned to the medium cluster.

To formally define the peer sets, let $NV\ P$ be the total set of virtual peers. The strong cluster peers ($SCV\ P$) are those satisfying $VPiW > \beta 1$, while the weak cluster peers ($WCV\ P$) satisfy $VPiW < \beta 2$. The medium cluster peers ($MCV\ P$) are defined as the subset of peers not included in either the strong or weak clusters, and whose weight values fall within the inclusive range $\beta 2 \leq VPiW \leq \beta 1$. This clustering mechanism ensures that peer capabilities are aligned with content demands, enhancing system efficiency and contributing to improved QoS through optimized replication, reduced access latency, and better resource utilization.

4.3. Content Classification Algorithm Based on Accessing Popularity

In the proposed system, content is dynamically classified into three categories—Class I (CI), Class II (CII), and Class III (CIII)—based on its access popularity. Class I includes data that is requested frequently by a high number of peers and is therefore replicated in strong clusters, denoted as $\{SCVP\}$. Class II contains data accessed by a moderate number of peers, and it is replicated in medium clusters $\{MCVP\}$. Class III includes rarely accessed data, which is stored in weak clusters $\{WCVP\}$.

A central component of this process is the Query Server (denoted as $QrySrv$), which is responsible for recording every query submitted by client peers. For each peer, the query server maintains cluster-related metadata including the peer ID and the cluster designation—strong, medium, or weak ($SCVP$, $MCVP$, or $WCVP$, respectively). At any given time, let there be mmm client peers generating a set of query requests $\{Qm\} = \{Q1, Q2, Q3, \ldots, Qm\}$, where each query $q\{Pid, ckwd\}$ includes the peer identifier Pid and a content keyword *ckwd*. All incoming queries are registered and managed by the query server *QSR*.

The classification of content is based on the frequency of access observed across these queries. Specifically, if a query for a particular object $Qryj$ has a number of access instances $n(Qryj)$ greater than or equal to a defined high-level threshold β_{max} the object Ob_{req} is classified as Class I (CI). If the access frequency is less than β_{max}ut greater than $\beta_{min}$the object is categorized as Class II (CII). Finally, if the access frequency is less than or equal to β_{min} the object is assigned to Class III (CIII). Formally, the classification algorithm can be expressed as follows:

Given a query Qry_j where $j < m$:

$$\begin{gathered} \text{If } n(Qry_j) \geq \beta_{max} \text{ then } Obr_{eq} \in CI \\ \text{If } \beta_{min} < n(Qry_j) < \beta max, then\ Ob_{req} \in CII \\ \text{If } n(Qry_j) \leq \beta min \text{ then } Ob_{req} \in CIII \end{gathered} \tag{2}$$

In this context, $n(Qry_j)$ represents the total number of times the content associated with the query has been accessed. The values of $\beta_{max}$and $\beta_{min}$are predetermined threshold values that define the upper and lower bounds of access frequency. For example, based on Table 1, β_{max}ay be set to 5 and β_{min} 3, although these values may vary depending on system configuration and workload characteristics. This classification process ensures that content is replicated appropriately according to its popularity, which in turn improves overall system performance and resource efficiency.

Table 1. Content Classification Based on Access Popularity

Access Popularity	Object Request Rate (%)	Number of Files (Objects)	Classified Data Type	Assigned Cluster Group
Low	1	3000	CIII Data	Weak Cluster (WCVP)
Low	2	1500	CII Data	Medium Cluster (MCVP)
Medium	3	750	CII Data	Medium Cluster (MCVP)
Medium	4	300	CII Data	Medium Cluster (MCVP)
High	5	150	CI Data	Strong Cluster (SCVP)

Table 1 shows that, if the request rate of object is low, then the accessing popularity of the object is also low and the numbers of objects are high. If the request rate is medium, then the accessing popularity of the object is also medium

 ISSN 2723-6471

and the numbers of objects are medium. If the request rate is high, then the accessing popularity of the content is also high and the numbers of objects are less.

4.4. Dynamic Replica Creation on Demand Based

Dynamically replicate data based on user interactions and data accessing popularity. Monitor demand and trigger on-demand replication. Replicate Class-I Data in strong cluster peers, Class-II Data in medium cluster peers, and Class-III Data in weak cluster peers.

Algorithm 1. Dynamic Popularity-Aware On-Demand Replica Allocation (DPORA) Algorithm

Create Replica (Ob_{req})

{

1. For each Ob_{req} arriving form Ci // ith client

Classify (Ob_{req} € C_I or C_{II} or C_{III})

2. If (Ob_{req} € C_I) then Assign Ob_{req} to SC_{VP}

Else If (Ob_{req} € C_{II}) then Assign Ob_{req} to MC_{VP}

Else If (Ob_{req} € C_{III}) then Assign Ob_{req} to WC_{VP}

3. If Replica of Ob_{req} then serve request

Else If BWi && CPUSi && MZi available in VPi

then Create replica (Ob_{req})

4. QSR assign replication pattern information to OS

5. OS Broadcast replica (Ob_{req}) with {P_{id}, Clid (SC_{VP}" or" MCVP" or" WCVP"), Cd1, Cd2, Cd3}

}

- Where Cd1, Cd2 and Cd3 are the content database id.

The query server registers queries from virtual peers, assigning objects to clusters based on popularity of the content, and serves them through OB_{req}, which is replicated by VPi. The origin server (OS) distributes queries to different clusters based on their strength, medium, and weak to enhance service quality.

4.5. Dynamic Query Search and Data Retrieval

The Dynamic Route Determination Algorithm is used to locate replicated data in a peer-to-peer network by Optimizing Network performance. This improves data retrieval efficiency and speed. The algorithm uses the Dynamic Query Search and Data Retrieval Algorithm to search, and fetch the replicated content from the nearest cloud peer. So, the content accessing time is reduced and improve the system performance.

The virtual peer $VP1$ receives a query Qry from the client peer C in peer-to-peer network. The virtual peer $VP1$ compares the query Qry with query identifier $Qidkj$, where Qry_{idkj} is the k[th] query of j_{th} peer j. Check whether the query hit is higher than zero ($Qry_{HitK} > 0$), then find the maximum score of the requested query Qry by using the equation (2)

$$\text{Score} = \sum_{k=0}^{m} \text{NR}(\text{VPj},\text{Qry}_k)^{\alpha} \quad (3)$$

Select the best peer for sending the request of virtual peer VP_j, which has the maximum number of results returned for the requested query Qry and forward the query Qry to the best peer. Virtual peer VP_j sends the acknowledgement ACK to client peer C. The client peer C sends a confirm packet to S_{best} node. And S_{best} send requested data for requested query Qry, to the client peer C. Check whether the query hit is greater than zero ($Q_{Hitk} > 0$) for all j, then the client peer C send query request Qry to Server peer. Server peer sends requested data for query Qry, to the client peer C. Client C has the available memory size and bandwidth then the client peer C caches the requested data item. And the client peer C is assigned as a strong cluster peer. The client peer C propagates the object to other peers with the information of node identifier, client identifier and data base identifier like $\{Nid, Cntid\ (\text{“}SCVP\text{”}), d1\}$.

 ISSN 2723-6471

N_{id} is the node identifier, Cnt_{id} is the cluster identifier and $d1, d2$ and $d3$ are the content database identifiers. The hierarchical approach in cloud computing optimizes data replication by implementing node initialization, dynamic clustering, data classification, on-demand replication, hierarchical query broadcast, and dynamic query search and retrieval techniques.

4.6. Matrix Representation of Replica Placement

Virtual peer VPi knows the request rates $r\_ij\ , j = 1, \dots, n$ of its local users for all objects. The vector of request rates of the users at virtual peer VPi is denoted by $r\_i = (\ ri1\ , ri2\ , ri3\ , \dots, rim\ )$ the m X n of request rates of all objects at all peers. For the replica placement matrix (RPM) as $m\ X\ n$ matrix whose entries are represented by (3)

$$RPM_{ij} = \begin{cases} -1, if Object O_j is in Local Catch \\ 0, if Object O_j is replicated at Strong Cluster Peer \\ 1, if Object O_j is replicated at Medium Cluster Peer \\ = 2, if Object O_j is replicated at Weak Cluster Peer \\ > 2, if Object O_j is replicated at Origin Server \end{cases} \quad (4)$$

$i = 1, 2, \dots, m\ and\ j = 1,2, \dots, n$. Replica placement matrix RPM_ij is representing the requested object Oj which is in either Local peer or in Strong group or in medium group or in weak cluster peers or in the origin server. The system's goal is to minimize the access time at each peer.

4.7. Cloud Computing Adaption

Peer-to-peer nodes in the clustering process are replaced by virtual machines in a cloud computing environment. The memory capacity, access latency, CPU speed, and bandwidth of each virtual machine are its defining characteristics. Every virtual machine has a weight, which is determined and used to base the clustering. As a result, virtual machines with comparable performance traits are grouped into strong, weak, and medium clusters. This algorithm enables cloud service providers to dynamically cluster virtual machines based on their resource attributes. The high and low-level threshold values (β1 and β2) allow for the customization of cluster strength, ensuring optimal resource allocation and performance in the cloud computing environment. The significant achievement of this approach enhances the cloud resource management, allowing for the formation of clusters that can be utilized for load balancing, optimization of response times, and efficient resource allocation in a dynamic and scalable cloud computing infrastructure. And also improve the QoS matrices while replicating and transferring the objects from into the cloud by using wireless network

5. Results and Discussion

5.1. Simulation Results and Discussions

In this proposed system the QoS parameters are considered for improving the network performance. The parameters are Bandwidth Utilization. Access Latency, Throughput, Packet Drop. Peer-to-peer nodes in the clustering process are replaced by virtual machines in a cloud computing environment. This section covers the simulation-based experimental performance evaluation of the approaches. The protocol was tested using the Cloud based - NS2 simulator. Using discrete events, user-defined networks can be simulated with Cloud based - NS2, a general-purpose simulation programmed. Only the packet-level simulator makes use of a network topology. The network bottleneck at the user access lines, not the routers, necessitated a more straightforward architecture in the simulations. In order to model the network, the overlay and access connections were useful. Every peer has an asymmetric link connecting it to its access router. All access routers have direct links to all other routers, essentially creating an overlay link. This makes it possible to simulate differences in end-to-end (e2e) latency among different peers as well as variations in upload and download speeds. The performance measures are selected based on two factors.

5.1.1. Performance Based on Load

This analysis shows that the requested content has a load ranging from 2.0 MB to 5.0 MB. The received throughput up to 600 packets are measured. Additionally, contrast the current method with the throughput, packet drop up to 100 packets, and bandwidth utilization up to 1000 KB/S are measured and compared with the existing approach.

5.1.2. Performance Based on Rate

In the second examination, the data transmission rate is increased from 250KB to 1MB. Response time in seconds, number of missing packets up to 100, query efficiency up to 100%, are measured and compared with the existing approach. This document emphasizes the performance metrics taken for comparison between the existing solution and proposed works for the problem specified along with simulation settings and simulation results. The protocol used for routing is Distance Vector Routing Protocol with 64 nodes / terminals. The bandwidth of the wired communication channel is set as 2 MBPS with queue length 250 and packet interval 0.08 milliseconds. The packet size is fixed as 1000 bytes with constant bit rate data transfer fashion.

Figure 2 shows the throughput of each client node that received its assigned amount of data and it makes it clear that the proposed method PS has a higher throughput than the existing technique ES and that the throughput values drop with increasing the load ranging from 2.0 MB to 5.0 MB. Figure 2 compares the performance of the proposed approach PS with the existing method ES with increasing the load ranging from 2.0 MB to 5.0 MB. Figure 3 demonstrates how an increase in load causes an increase in the packet drop ratio. The results make it evident that proposed system PS has a lower packet drop ratio than existing system ES.

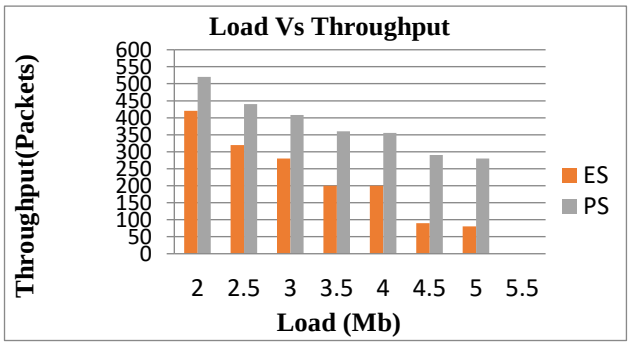

Figure 2. Load Vs Throughput

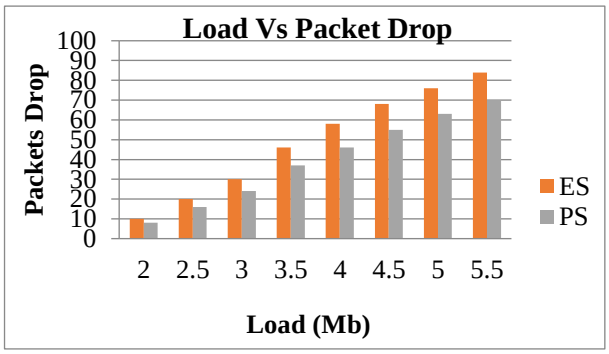

Figure 3. Load Vs Packet Drop

Figure 4 illustrates the query sending rate in this experiment is varied between 250 Kbps and 1 Mb. In figure 4 results make it evident that proposed system PS has a lower packet drop ratio than the existing system ES. Figure 5 displays the query efficiency of the proposed system approach we proposed. The percentage of data queries that are answered is used to calculate a simulation's query efficiency. As the data flow data transfer rate increases, Figure 5 demonstrates that proposed system PS has a higher query efficiency than existing system ES. In this experiment, the query sending rate can be varied between 250 KB and 1 MB.

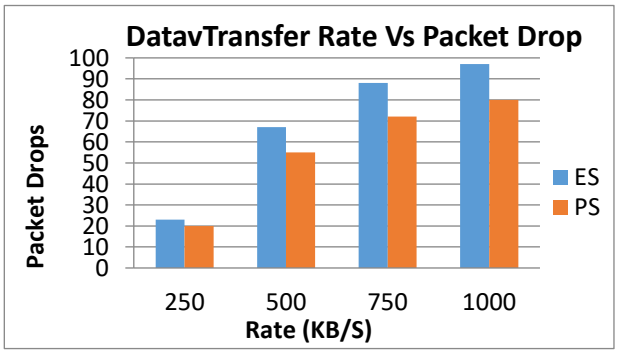

Figure 4. Data Transfer Rate Vs Packet Drop

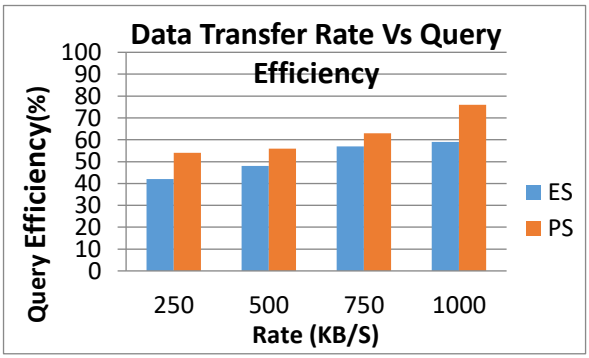

Figure 5. Data Transfer Rate Vs Query Efficiency

Figure 6 shows the bandwidth utilization of our proposed system PS is lower than that of existing system ES when increases at a higher load and the load ranging from 2.0 MB to 5.0 MB.

ISSN 2723-6471

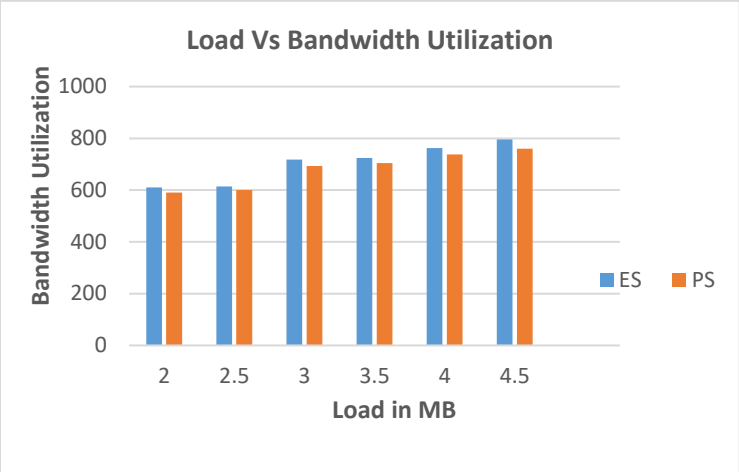

Figure 6. Load Vs Bandwidth Utilization

Robust connectivity, little bandwidth usage, low maintenance costs, and reduced latency are among the results. The method's dynamic and adaptable design meets the required SLA, targets performance, and improves overall system availability. Additionally, the results indicate that PS outperforms proposed system in terms of packet drop, making it a more efficient option for handling increased data flow. These findings suggest that implementing proposed system can lead to improved query efficiency and overall system performance in scenarios with varying query sending rates.

5.2. Scientific Impacts & Practical Utility with Real-Time Applications of Proposed Strategy

The Dynamic Replica Management Strategy (DRMS) based on Data Accessing Popularity plays a crucial role in enhancing the QoS parameters in cloud storage environments. By dynamically replicating data based on its access patterns, this strategy improves load balancing, fault tolerance, scalability, reliability, efficient content distribution, and network performance. Let’s explore its scientific impacts, practical utility, and real-time applications in the context of improving QoS.

5.2.1. Load Balancing to Improve QoS

Dynamic replication plays a pivotal role in achieving optimal load balancing within cloud storage systems. Scientifically, this approach leverages data access frequency to intelligently distribute data replicas across multiple storage nodes. When popular content is accessed more frequently, it is dynamically replicated and placed on less congested or geographically closer nodes. This strategic placement reduces latency and ensures that resources like bandwidth and computational power are utilized more efficiently. As a result, the system can prevent bottlenecks and maintain a balanced workload distribution, which is essential for enhancing the overall QoS metrics.

In practical scenarios, load balancing via DRMS prevents server overloads by dispersing requests to multiple nodes, avoiding degradation in performance during peak access times. For example, platforms like Netflix benefit from such strategies by ensuring that trending shows are mirrored across numerous content delivery servers. This minimizes buffering and ensures that user experience remains seamless regardless of spikes in traffic. Through such dynamic replication, the system becomes more adaptive, responsive, and robust in managing unpredictable user demand while maintaining efficient data accessibility.

5.2.2. Fault Tolerance and Data Availability

Scientifically, DRMS improves fault tolerance by replicating critical and popular data across multiple nodes, ensuring data redundancy and resiliency in case of system failures. When one node fails due to hardware issues or network outages, other replicas serve as immediate backups, allowing uninterrupted data access and maintaining service continuity. This mechanism is particularly crucial for ensuring data survivability in distributed cloud environments where single points of failure can otherwise compromise system integrity and availability.

Practically, this model provides a reliable framework for applications requiring high uptime and constant data access. For instance, in cloud-based financial services, transaction logs and sensitive records are continuously replicated across

different geographical data centers. If one center becomes unreachable, others seamlessly take over operations, guaranteeing that services such as online banking or digital payments remain unaffected. This high-availability model significantly enhances QoS by minimizing service disruptions and protecting critical user data.

5.2.3. System Scalability

From a scientific perspective, DRMS contributes to system scalability by dynamically adjusting the number of data replicas in response to fluctuating access patterns. As the number of users or requests for specific content increases, the algorithm automatically increases the number of replicas without requiring manual intervention. This enables the cloud system to adapt to growing workloads while maintaining performance consistency, effectively supporting the horizontal scaling of cloud infrastructure.

In real-time usage, such scalability is essential for e-commerce platforms like Amazon, especially during events like Black Friday or Cyber Monday. During these periods, the access rate to product pages surges exponentially. DRMS dynamically replicates high-demand product data across numerous servers, allowing the system to manage massive concurrent user traffic. This ensures that pages load quickly, checkout processes remain responsive, and customer experience is not hindered by latency or service crashes, regardless of user volume.

5.2.4. Reliability and Data Integrity

Scientifically, DRMS enhances data reliability and integrity by ensuring that multiple synchronized copies of data are consistently available across the cloud network. This distributed replication not only safeguards against data loss but also ensures that both read and write operations are validated across nodes, preserving the consistency and accuracy of information. Such a mechanism is fundamental for maintaining trust in data-driven systems, especially those with stringent consistency requirements.

In practice, cloud-based healthcare platforms can significantly benefit from this strategy. Patient medical records, diagnostics, and prescriptions are vital data that must remain accessible and accurate at all times. By replicating this information dynamically across multiple clusters, DRMS ensures healthcare providers have uninterrupted access to the latest updates, even during system upgrades, maintenance, or unforeseen outages. This contributes to better patient outcomes and supports real-time clinical decision-making, reinforcing QoS and data dependability.

5.2.5. Optimizing Network Performance for QoS

Scientifically, DRMS improves network performance by minimizing the physical and logical distance between the user and the data. By replicating content closer to where it is most frequently accessed, the system reduces the number of network hops, leading to lower data transfer latency, reduced congestion, and higher throughput. This is especially crucial for applications where delay sensitivity directly affects user satisfaction and operational efficiency.

In real-time applications such as online multiplayer games, this optimization is vital. DRMS ensures that game updates, frequently accessed assets, and real-time data are stored on servers nearest to the players. This geographical replication leads to reduced lag, faster synchronization, and smoother gameplay experiences, which are key to retaining users in competitive gaming environments. Overall, DRMS not only optimizes infrastructure utilization but also enhances the responsiveness and efficiency of interactive services.

5.3. Advantages of Proposed Method

The proposed method is straightforward, dynamic, scalable, and dependable since it divides the peers into three groups based on their capacity and weight values. Peers can dynamically join and exit the cloud network. Based on how popular it is to access, the content is divided into three groups and copied into the appropriate peer groups. By duplicating the material across cloud network environments, it increases the accessibility of often used things. By getting the requested content from the closest peers, this minimizes bandwidth consumption, delays, server peer overhead, and access costs. The clients can effectively access the content from the cloud at any time and from any location. This approach is relatively flexible and efficient, allowing for seamless access to resources regardless of fluctuations in network traffic or demand. Additionally, the decentralized nature of the cloud network ensures high availability and reliability of content delivery. The dynamic replica management strategy has both scientific impact in advancing research on cloud system efficiency, load balancing, and fault tolerance, and practical utility in real-world

applications where it enhances performance, scalability, cost-effectiveness, and user experience. Through real-time replication based on data popularity, this strategy ensures that cloud systems adapt to changing demand, maintaining optimal performance in environments ranging from video streaming to healthcare and financial services.

6. Conclusion

This work addresses the issues of data unavailability and storage node failures by introducing a novel method of data replication in cloud computing. Optimizing load balancing, response time, and availability while adapting to user interactions is the goal of dynamically clustering virtual nodes, data classification, and replicating on-demand. The suggested method is a viable way to improve cloud computing environments because of its hierarchical approach, which ensures service quality while abiding by SLA contracts. When compared to the current method, the proposed work improves system performance and yields high throughput. It improves the efficiency of the system's use of its resources and offers faster query responses. Techniques for caching and replication are combined with the suggested approach. The technique reduces both the amount of bandwidth used and the frequency of packet drops. Subsequent research endeavors could delve into practical applications and other enhancements for wider relevance.

7. Declarations

7.1. Author Contributions

Conceptualization: S.T., S.S.M., and S.T.V.; Methodology: S.S.M.; Software: S.T.; Validation: S.T., S.S.M., and S.T.V.; Formal Analysis: S.T., S.S.M., and S.T.V.; Investigation: S.T.; Resources: S.S.M.; Data Curation: S.S.M.; Writing Original Draft Preparation: S.T., S.S.M., and S.T.V.; Writing Review and Editing: S.S.M., S.T., and S.T.V.; Visualization: S.T.; All authors have read and agreed to the published version of the manuscript.

7.2. Data Availability Statement

The data presented in this study are available on request from the corresponding author.

7.3. Funding

The authors received no financial support for the research, authorship, and/or publication of this article.

7.4. Institutional Review Board Statement

Not applicable.

7.5. Informed Consent Statement

Not applicable.

7.6. Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] G. Khekare, K. P. Kumar, K. N. Prasanthi, S. R. Godla, V. Rachapudi, M. S. Al Ansari, and Y. A. B. El-Ebiary, "Optimizing Network Security and Performance Through the Integration of Hybrid GAN-RNN Models in SDN-based Access Control and Traffic Engineering," *Int. J. Adv. Comput. Sci. Appl. (IJACSA)*, vol. 14, no. 12, pp. 1-12, 2023, doi: 10.14569/IJACSA.2023.0141262.

[2] I. E. Miloudi, B. Yagoubi, and F. Z. Bellounar, "Dynamic Replication Based on a Data Classification Model in Cloud Computing," *in Modelling and Implementation of Complex Systems, S. Chikhi, A. Amine, A. Chaoui, D. Saidouni, and M. Kholladi, Eds. Cham: Springer*, vol. 156, no. Sep., Lecture Notes in Networks and Systems, pp. 3–17, 2021. doi: 10.1007/978-3-030-58861-8_1.

[3] M. Elrotuub and A. Gherbi, "Virtual Machine Classification-based Approach to Enhanced Workload Balancing for Cloud Computing Applications," in *Proc. 9th Int. Conf. Ambient Syst., Netw. Technol.*, vol. 130, no. 1., pp. 683–688, 2018, Elsevier.

[4] S. Ayyasamy and S. N. Sivanandam, "A QOS-Aware Intelligent Replica Management Architecture for Content Distribution in Peer-to-Peer Overlay Networks," *Int. J. Comput. Sci. Eng.*, vol. 1, no. 2, pp. 71–79, 2009.

[5] M. Sozio, T. Neumann, and G. Weikum, "Near-optimal Dynamic Replication in Unstructured Peer-to-Peer Networks," in *Proc. 27th ACM SIGMOD-SIGACT-SIGART Symp. Principles Database Syst. (PODS)*, vol 27, no. 1, pp. 281–290, 2008.

[6] E. Cohen and S. Shenker, "Replication Strategies in Unstructured Peer-to-Peer Networks," in *Proc. ACM SIGCOMM 2002 Conf.*, vol. 32, no. 4, pp. 177–190, 2002.

[7] S. Zaman and D. Grosu, "A Distributed Algorithm for the Replica Placement Problem," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 9, pp. 1455–1468, Sep. 2011.

[8] G. Gao and others, "Proactive Replication for Rare Objects in Unstructured P2P Networks," *J. Netw. Comput. Appl.*, vol. 35, no. 1, pp. 85-96, Apr. 2011.

[9] Y. Chawathe and others, "Making Gnutella-like P2P Systems Scalable," in *Proc. SIGCOMM*, vol. 03, no. Aug., pp. 407-418, Aug. 2003.

[10] D. Tsoumakos and N. Roussopoulos, "APRE: A Replication Method for Unstructured P2P Networks," CS-TR-4817, College Park, MD 2007.

[11] S. Limam and others, "Data Replication Strategy with Satisfaction of Availability, Performance and Tenant Budget Requirements," *Cluster Comput.*, vol. 22, no. 3, pp. 1199-1210, Jan. 2019, doi: 10.1007/s10586-018-02899-6.

[12] S. F. Piraghaj, R. N. Calheiros, J. Chan, A. V. Dastjerdi, and R. Buyya, "Virtual Machine Customization and Task Mapping Architecture for Efficient Allocation of Cloud Data Center Resources," *Future Gener. Comput. Syst.*, vol. 59, no. 2, pp. 208–224, 2017.

[13] Y. Wei and others, "Comparative Analysis of Artificial Intelligence Methods for Streamflow Forecasting," *IEEE Access*, vol. 12, no. Jan., pp. 10865–10885, 2024, doi: 10.1109/ACCESS.2024.3351754.

[14] S. Narkhede, T. Baraskar and D. Mukhopadhyay, "Analyzing web application log files to find hit count through the utilization of Hadoop MapReduce in cloud computing environment," *2014 Conference on IT in Business, Industry and Government (CSIBIG), Indore, India*, vol. 2014, no. 1, pp. 1-7, 2014, doi: 10.1109/CSIBIG.2014.7056950.

[15] S. Sun, W. Yao, and X. Li, "DARS: A dynamic adaptive replica strategy under high load Cloud-P2P," *Future Generation Computer Systems*, vol. 78, no. 1, pp. 31–40, 2018, doi: 10.1016/j.future.2017.07.046.

[16] A. S. Vijendran and S. Thavamani, "Analysis Study on Caching and Replica Placement Algorithm for Content Distribution in Distributed Computing Networks," *Int. J. Peer-to-Peer Netw.*, vol. 3, no. 6, pp. 13–21, Nov. 2012.

[17] S. F. Piraghaj, R. N. Calheiros, J. Chan, A. V. Dastjerdi, and R. Buyya, "Efficient Virtual Machine Sizing for Hosting Containers as a Service," in *Proc. IEEE*, vol. 2015, no. Aug., pp. 31-38, 2015, doi: 78-1-4673-7275-6.

[18] T. Hariguna and A. Ruangkanjanases, "Assessing the impact of artificial intelligence on customer performance: a quantitative study using partial least squares methodology," *Data Science and Management*, vol. 7, no. 3, pp. 155–163, 2024, doi: 10.1016/j.dsm.2024.01.001.

[19] I. G. A. Purnamawati, A. Y. Oudah, H. B. Othman, I. B. Ibrahim, A. H. Iswanto, A. Komariah, and Y. F. Mustafa, "A Comprehensive Optimization Approach Based on Cloud Computing for Logistic Sharing System Planning," *Ind. Eng. Manag. Syst.*, vol. 21, no. 3, pp. 468–474, 2022.

[20] Q. Liu, Q. Liu, and M. Wang, "Quantifying the Effects of Homogeneous Interference on Coverage Quality in Wireless Sensor Networks," *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 8, pp. 1-20, 2024.

[21] C. Gkantsidis, M. Mihail and A. Saberi, "Random walks in peer-to-peer networks," *IEEE INFOCOM 2004, Hong Kong, China*, vol. 2004, no. 1, pp. 130-130, 2004. doi: 10.1109/INFCOM.2004.1354487.

[22] C. Gkantsidis, M. Mihail, and A. Saberi, "A Hybrid Search Scheme for Unstructured P2P Networks," in *Proc. INFOCOM*, Miami, FL, vol. 2005, no. Mar., pp. 1-12, Mar. 2005.

[23] A. S. Vijendran and S. Thavamani, "Survey of Caching and Replica Placement Algorithm for Content Distribution in Peer-to-Peer Overlay Networks," in *Proc. 2nd Int. Conf. Comput. Sci. Eng. Inf. Technol. (CCSEIT 2012)*, Coimbatore, India, vol. 2, no. Oct., pp. 248-252, Oct. 2012. ACM.