Received: 10 February 2022 | Revised: 23 January 2023 | Accepted: 25 January 2023

DOI: 10.1002/cpe.7672

RESEARCH ARTICLE

WILEY

A novel hybrid meta-heuristic-oriented latency sensitive cloud object storage system

N. Nataraj | **R. V. Nataraj**

Department of Information Technology, Bannari Amman Institute of Technology, Tamil Nadu, India

Correspondence
N. Nataraj, Department of Information Technology, Bannari Amman Institute of Technology, Tamil Nadu, India.
Email: n.nataraj08@gmail.com

Summary

Cloud providers must find out how to properly arrange data in a limited count of servers while ensuring latency assurances to reduce total storage expenses. Timeout is also important to consider because it has a substantial impact on response latency. The core aim of this task is to implement a new cloud object storage system strategy that handles challenges like "latency-sensitive data allocation, latency-sensitive data re-allocation, and latency-sensitive workload consolidation." The main contribution here is that distributing the latency of the cloud object storage system allows for better data allocation, data reallocation, and workload consolidation. The primary aim is to use the fewest number of servers feasible to fulfill all requests while maintaining their latency requirements, lowering the overall data transmission cost. As a consequence, Whale Butterfly Optimization Method (WBOA) is a novel hybrid meta-heuristic algorithm that solves NP-hard problems by combining baseline advanced algorithms. The simulation outcomes reveal that the offered paradigm consistently provides the greatest outcomes regarding throughput utilization, lower latency, higher storage, and number of used nodes when compared to competing techniques.

KEYWORDS

artificial intelligence, data allocation, data reallocation, latency sensitive cloud object storage system, Whale butterfly optimization algorithm

1 | INTRODUCTION

The data associated with the upper-tier services as well as applications are stored and managed by the CLOUD storage system. Modern web-oriented applications rely heavily on cloud technologies like Open-Stack Swift[1] and Amazon S3.[2] Industrials construct and handle their self-storage systems apart from the public cloud object storage systems, such as Facebook's utilization of Haystack for stockpiling photos, LinkedIn's utilization of Ambry for retaining media objects, and Wikipedia's utilization of OpenStack Swift clusters as media object-store locations for scalability and efficiency.[3] The cloud object storage system, as one of the most basic cloud services, stores as well as retrieves millions or billions of different data objects (also known as blobs), such as images, videos, audio, documents, and so on.[4] Furthermore, it may immediately service millions of Internet users who are latency-sensitive.[5] The traditional object storage systems, on the other hand, disperse data objects throughout the system without taking into account the heterogeneity of supporting hardware or asymmetric connectivity patterns. Hybrid object storage technologies enable business data centers to accomplish energy efficiency and performance at a low cost.[6]

Furthermore, millions of latency-sensitive Internet users may be served directly by the cloud object storage system.[7] An analytic-oriented performance method for the cloud object storage system based on the event-driven programming paradigm was created to better understand the link between resource allocation and response latency distribution.[8,9] In the case of a cloud object storage system, the response latency percentage outperforms the average metrics for the below reasons.[10] Initially, response latency is an important performance indicator for cloud object storage

systems because it has a significant influence on user experiences, which are linked to revenue.[11] Next, even 1% of traffic equates to a large number of user requests for cloud object storage systems.[12] The waiting time contains a substantial influence on the cloud object storage system's response latency.[13]

The quality of latency information determines the usefulness of latency-driven approaches in enhancing application performance.[14] A simple method is to probe every delay that the program requires regularly.[15] Continuous testing of pair-wise latencies is not practicable when the count of nodes is extremely great,[16] hence quite a technique stands to reason only in very small systems. Redirecting customers to their neighboring data centers, for instance, would need Google maintaining latency statistics from practically every web client on the internet to each one of its data centers.[1] Furthermore, due to the highly dynamic nature of the Internet, newly recorded latencies are not necessarily a reliable indicator of their contemporary equivalents, since one latency-measured value is often not a strong predictor of a similar measurement. These two issues necessitate the development of accurate latency and scalable detection algorithms.[17]

The core improvements of this task are:

- To offer the latest methodology for cloud object storage systems focusing the problems like latency-sensitive data allocation and latency-sensitive data re-allocation.
- To carry out the proper data allocation and data re-allocation by distributing the latency of the cloud object storage system.
- To utilize a lower count of a subset of servers in terms of the complete requests without violating their latency needs owing to this the total data moving cost is not higher.
- To adopt the latest hybridized optimization algorithm called WBOA for deriving the objective functions of the data allocation and reallocation of the considered latency-sensitive cloud object storage systems.

The leftover of this task is given below: Section 1 gives the introduction associated with cloud object storage systems. Section 2 provides the conventional approaches to the recommended method. Section 3 offers the developed framework for the latency-sensitive cloud storage system. The hybrid meta-heuristic-based latency-sensitive data allocation is given in Section 4. Section 5 gives the latency-sensitive data reallocation utilized for the hybridized optimization algorithm. Section 6 visualizes the findings of the offered method. At last, the task is concluded in Section 7.

2 | LITERATURE SURVEY

2.1 | Related works

In 2019, Su et al.[18] created a performance method for the cloud object storage system based on the assumption that no timeouts would occur. In the setting of event-driven programming, sophisticated disc operations, and requests awaiting acceptance, this method forecasted the proportion of requests fulfilling an SLA. Second, a mechanism was presented for determining if the model was appropriate by anticipating when timeouts would occur. This method was tested on a real-world trace having a production system. It has decreased prediction errors by up to 90% in several circumstances when compared to baseline methods, and its overall average error was 2.63%. Furthermore, the method's applicability was forecasted precisely.

In 2021, Kou et al.[19] developed a new node clustering approach based on graph convolutional networks (GCNs). It could use the excellent quality of pseudo clustering tags to self-supervise the training of node depictions by providing a self-supervision component. Furthermore, a latent distribution preservation term was used to aid the latent models of the similar sample to attain a constant allocation in the original dimension space and dimension space, as evaluated by KL divergence. The two above problems have been combined into a single optimization method. When compared to conventional node clustering approaches, experimental findings on many public datasets showed that the approach was successful.

In 2021, Arora et al.[20] presented an intelligent hybrid disc storage system that was both energy efficient and intelligent. The suggested method was able to detect recently accessed data from application traces. To allocate frequent records to hot discs and certain other files to cold discs, data layout, and replica management were utilized. The request was carried out by an advanced arrangement mechanism that looked for and picked the disc depending on its present status. The suggested system was deployed in the cloud environment by integrating disc management, which was demonstrated to be extremely efficient for producing power savings.

In 2017, Wu et al.[21] examined a variety of real-world company workloads and discovered that read and write requests were not spread evenly among data objects. A biased object storage strategy (BOSS) was developed based on the findings, to decrease SSDs and increase the efficiency of the system for hybrid OSS. Unlike traditional uniform and rigid data distribution systems, the BOSS could continuously migrate and distribute data objects to several forms of devices based on online data access patterns. The BOSS could minimize 64% of writes on

SSDs and enhance the network efficiency by 29.51% on average while providing a degree of load balancing, according to the findings of the experiments.

In 2019, Tao Shen and Yukari Nagai[22] presented a technique for developing distributed cloud storage systems. Initially, the availability of data must be evaluated in the study of the algorithmic model, which was produced in diverse contexts. Monte Carlo was an excellent approach for document analysis since the estimation of data availability became extremely complex as the storage node grew. Its great benefit was that it could minimize the complexities of the analysis framework simulation analysis. Next, while determining the best redundancy allocation on every host was problematic, a particle Swarm optimization (PSO)-oriented allocation approach was presented. Furthermore, the key characteristic of this method in the evaluation of the data availability stage was that its redundancy index was quite low, which improved the program's effectiveness. The suggested strategy could minimize storage costs and data redundancy, according to experimental outcomes.

In 2017, Baun et al.[23] documented the creation and deployment of OSSperf, a lightweight software solution for investigating the efficiency of object-oriented public cloud storage systems such as "Google Storage, Amazon S3, and Microsoft Azure Blob Storage, and private cloud re-implementations." This document also included a description of the tool's outcome as well as a few lessons learned during development. Distinct methodologies and current solutions for assessing the results of object-oriented storage services were assessed in this work, and a novel solution dubbed OSSperf was implemented. This would cover a wide variety of private and public cloud services, as well as their various APIs. It has also looked at the efficiency of the major significant elements of object-oriented storage systems, and to mimic summaries with varying levels of usage, the tool has to offer a parallel mode of operation for object upload as well as download.

In 2017, Yin et al.[24] presented ASSER, an ASsembling chain of Erasure coding and Replication, as a unique storage system. The basic protocols were used as a foundation. Across similar fault tolerance as well as consistency levels, ASSER surpassed pure erasure coding and N-way replication in I/O throughput below a variety of workload and system configurations, demonstrating improved performance reliability. Most significantly, ASSER provided consistently effective I/O efficiency at a far lower storage cost than its competitors.

In 2019, Mohammed[25] established a unique approach to network latency estimate that, owing to its accuracy and effectiveness. To forecast the amount of the end-to-end latency within any particular pair of nodes, a variety of machine learning approaches were applied, including traditional linear regression, CNNs, and SVMs. The iConnect-Ubisoft and Ubique datasets were used to train as well as test the machine learning algorithms in this system.

2.2 | Review

As the primary cloud service, the cloud object storage systems store and retrieve a huge number of reading and heavy data objects. A huge count of requests is served every day, which makes the response latency the major section of client experiences. The response latency suffered from timeout problems owing to the lack of appropriate perception of the distribution. Numerous cloud object storage systems have been developed in recent years, which have diverse superiorities and downsides as given in Table 1. Queuing theory[18] efficiently predicts the applicability and addressed the complexity of different disk operations, and improves the performance of the developed cloud object storage model. However, the computation time of Queuing theory is low. GCN[19] improves the performance with better clustering. Though, an insufficient distribution representation affects the accuracy. The scheduling technique[20] efficiently decreases the execution time and achieves better power savings. On the other hand, the cost consumption of the developed model is more. The data placement strategy[21] guarantees performance, durability, and availability. Conversely, this model suffers from little overhead. Monte Carlo and PSO[22] reduce the storage cost and data redundancy and increase the complexity of the analysis. However, sometimes, little observation is required for constructing the object cloud storage systems. OSSperf[23] improves the performance of object-based cloud storage services. On the other hand, this model lacks implementation. MPL (Multiversional Parity Logging)[24] increases stability and efficient performance. This model does not establish the load balancing strategy. Linear regression, CNN, and SVM[25] efficiently predict the value of the end-to-end latency. Though, prediction approaches are affected by several challenges like network latency, the size of the system, and poor performance. Therefore, there is a need of developing cloud object storage systems with latency distribution.

3 | DEVELOPED FRAMEWORK FOR LATENCY-SENSITIVE CLOUD STORAGE SYSTEM

3.1 | Problem formulation

Suppose that a data chunk represents the fundamental data holding unit, with a size of c, and that it cannot be subdivided anymore. The length of whatever data file e may be expressed in this situation as $c \cdot |e|$, where $|e|$ denotes the count of pieces that make up the file. For the

TABLE 1 Superiorities and downsides of the existing cloud object storage systems.

Author [citation]	Frameworks	Superiorities	Downsides
Su et al.[18]	Queuing theory	• It efficiently predicts the applicability and addressed the complexity of different disk operations. • It improves the efficiency of the offered cloud object storage method.	• However, this method takes more time to compute.
Kou et al.[19]	GCN	• It improves performance with better clustering.	• Though an insufficient distribution representation affects the accuracy.
Arora et al.[20]	scheduling technique	• It efficiently decreases the execution time. • It achieves better power savings.	• On the other hand, the cost consumption of the developed model is more.
Wu et al.[21]	data placement strategy	• It guarantees performance, durability, and availability.	• Conversely, this model suffers from little overhead.
Tao Shen and Yukari Nagai[22]	Monte Carlo and PSO	• It reduces storage costs and data redundancy. • It increases the complexity of the analysis.	• However, sometimes, little observation is required for constructing the object cloud storage systems.
Baun et al.[23]	OSSperf	• It improves the efficiency of object-based cloud storage services.	• This model lacks implementation.
Yin et al.[24]	MPL (Multiversional Parity Logging)	• It increases stability and efficient performance.	• This model does not establish the load balancing strategy.
Mohammed[25]	linear regression, CNN, and SVM	• It efficiently predicts the value of the end-to-end latency.	• Though prediction approaches are affected by several challenges like network latency, the size of the system, and poor performance.

sake of simplicity, consider a single information document that can save on only one server and that cannot be split up and saved on several servers.

A heterogeneous cloud storage system with a group of $|T|$ servers T is considered. $M(t)$ defines the maximum feasible I/O rates or high workload without deteriorating efficiency for a certain server $t \in T$ in the cloud storage system, and $D(t)$ specifies the storage limit. Assume that there exist h different data accesses I/O rates, where $b_1, b_2, \cdots, b_h$. The data access (PUT or GET) time is assumed to follow a predefined distribution. As a result, the Cumulative Density Function (CDF) $g_t^c(y)$ shows the likelihood of retrieving one data chunk in the most y time units. The CDF is in some ways based on the server load underneath the server's maximum workload, although it has not altered considerably, for example, as the server load grows, the latency may be a little greater. To address this problem, the "conservative" CDF is only used when the server's workload is near its maximum, that is, when the latency distribution function's "upper bound" is reached. By doing so, (1) the latency restriction may be ensured to the greatest extent possible, with the minimum server utilization cost. (2) The issue of complexity is much decreased; alternatively, delay probability functions are required for various levels of server load. A variable "session" refers to a single GET/PUT thread or procedure that distributes a request of data from an information document on one server to the other user's end. Multiple periods can be created in this scenario to supply one or multiple data requests on a single server (namely, t); however, the average absorbed I/O rates should not surpass $M(t)$.

The term U denotes the requested data access time for a data file e with size $|e|$, δ denotes the requested probability of recovering e under time U, and α denotes the requested I/O rate for a data request $s(e, U, \delta, \alpha)$. Assume that l servers (replicas) have been issued s and that O_t sessions have been formed on the server t. Let us call the chance of accessing e the server t during time y as $g_t^e(y)$. As a result, the overall chance of reaching e inside U is shown in Equation (1).

$$1 - \prod_{t=1}^{l} \left(1 - g_t^e(U)\right)^{O_t}. \tag{1}$$

The above equation $\left(1 - g_t^e(U)\right)^{O_t}$ shows the failed distribution of accessing E during the time U on the server t for O_t sessions and $\prod_{t=1}^{l} \left(1 - g_t^e(U)\right)^{O_t}$ describes the failed probability of accessing E during the time U on l servers for the entire respective sessions. As a consequence, the chance that one period on one server may activate E during time U is $1 - \prod_{t=1}^{l} \left(1 - g_t^e(U)\right)^{O_t}$

Let us look at an example where there is no discrimination between GET and PUT for the sake of simplicity. Assume there are two servers like *B* and *C*, and the following are their probability density functions (PDF) for acquiring a data file $e(|e| = 20Gb)$ during the time *y* (in ms) as in Equation (2).

$$g_B^e(y) = \begin{cases} 0.9 : y \le 10 \\ 0.05 : 10 < y \le 15 \\ 0.05 : y > 15 \end{cases}, \quad g_C^e(y) = \begin{cases} 0.75 : y \le 6 \\ 0.2 : 6 < y \le 10 \\ 0.05 : y > 10 \end{cases}. \tag{2}$$

Initiate by assuming that the server *B* contains 100 GB of storage and a server load of 60 Mb/s, whereas the server *C* contains 120 GB of storage and a server load of 70 Mb/s. Assume a request *s* is received to retrieve a data file *e* at a rate of $\alpha = 50$ Mb/s, $U = 10$ ms, and $\delta = 0.995$. As per their PDFs, putting *e* it on the server *B* alone can ensure a probability of 0.9 having a latency of no less than 10 ms, while putting *e* it on the server $e(|e| = 20Gb)$ alone can ensure a probability of 0.95 having a delay of no more than 10 milliseconds. As every server can only have one session (either $60 < 2.50 = 100$ or $70 < 2.50 = 100$), installing *E* on either server *B* or *C* will not meet the demand δ. Figure 1 illustrates that sequentially placing *e* on both *B* and *C* results in a probability of 1-(1–0.9) (1–0.95) = 0.995 having a delay of no more than 10 ms, which meets the necessary probability.

Take the issue in which every server may handle several data access sessions at the same time. As CDF $g_t^e(y)$ is dependent on server *t* latency performance in the past, it already accounts for the case in which numerous sessions use shared resources like buffers on a similar server. As a result, every session on the server *B* may explicitly guarantee a probability of at least 0.9 of retrieving the data file during time 10. Equation (1) remains true in this scenario when numerous sessions are to be formed on the same server. Let us pretend that both servers *B* and *C* contain a 180 Mb/s available server load. In this case, there is no requirement to put *e* on two servers to meet the necessary latency probability. Alternatively, the request may be sent with only one copy saved on any server, but there is a need to create multiple sessions. For example, on the server *B*, three sessions are required to accept the request concurrently as in Figure 2A, because the odds of utilizing two and three sessions are $1 - (1 - 0.9)^2 = 0.99 < \delta$ and $1 - (1 - 0.9)33 > \delta$. On the server *C*, two periods are required (as illustrated in Figure 2B), since $1 - (1 - 0.95)^2 = 0.99759975 > \delta$. It can be seen from the preceding example that to meet the latency probability limitation, several sessions from separate data copies/locations or the similar data positions are often required. This last case can save a lot of storage space.

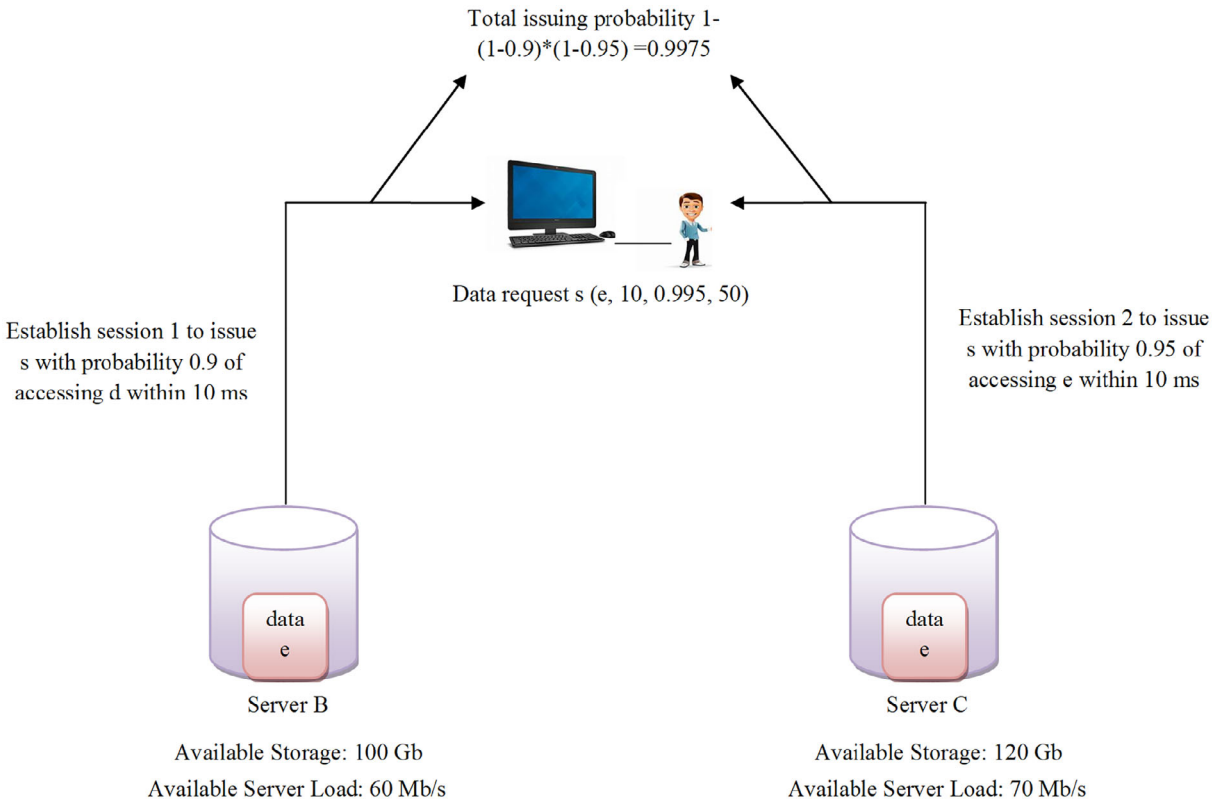

FIGURE 1 Data file placement on two servers by one session on every server to fulfill accessing data latency probability.

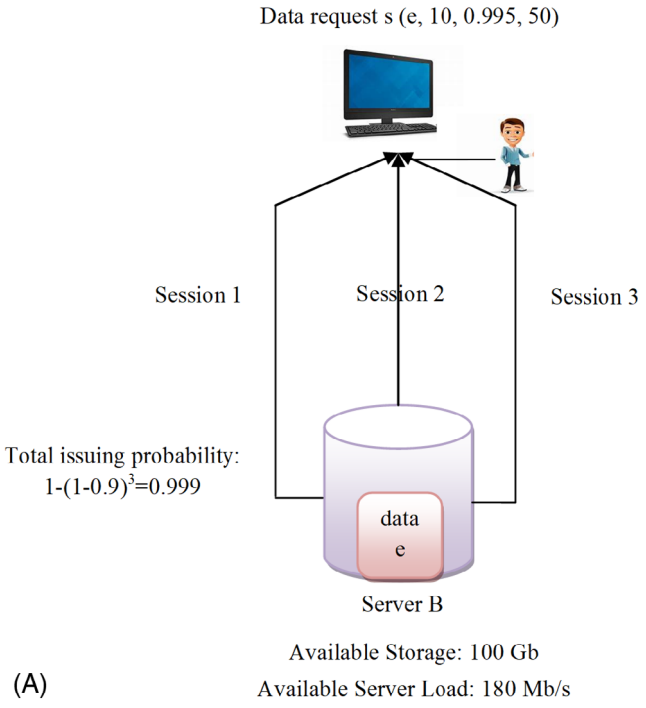

FIGURE 2 Data file placement on a single server by multiple sessions to fulfill accessing data latency probability.

The first optimization objective of this work is the data allocation concentrates on the active server minimization with the help of the recommended WBOA method. The second optimization objective of the data reallocation is to decrease the number of running servers by reallocating data to underutilized servers using the offered WBOA method.

3.2 | Data allocation model

Cloud computing represents a popular computing approach because it permits on-demand resource provisioning. The fundamental of meeting unforeseen needs and optimizing the return on investment from the cloud infrastructure defines a technique of resource allocation as well as

reallocation. Data allocation is the method of assigning an available resource to client-hosted apps in the cloud. The services will starve if the resources are not allocated and handled effectively. The resource provisioning manager tackles the problem by allowing the service provider to allocate resources for every discrete component with the help of a variety of resource allocation strategies. When distributing resources for incoming requests, how the resources are designed is critical. The services that a cloud may offer for developers can be abstracted at several layers, and various factors can be optimized throughout allocation. In most cases, resources can be shared across several customers in a data center and should be constantly allocated and changed to meet demand.

It is vital to realize that clients, as well as developers, may perceive those limiting resources as limitless, and the allocation schedule is the tool that will allow them to do so. The allocation schedule should respond to these erratic demands flexibly and openly. This flexibility should permit the dynamic utilization of physical resources, preventing both under and over-provisioning. Here, the data allocation focuses on active server minimization as the objective, which is done by the proposed WBOA. The diagrammatic model of the data allocation for the latency-sensitive cloud object storage systems is shown in Figure 3.

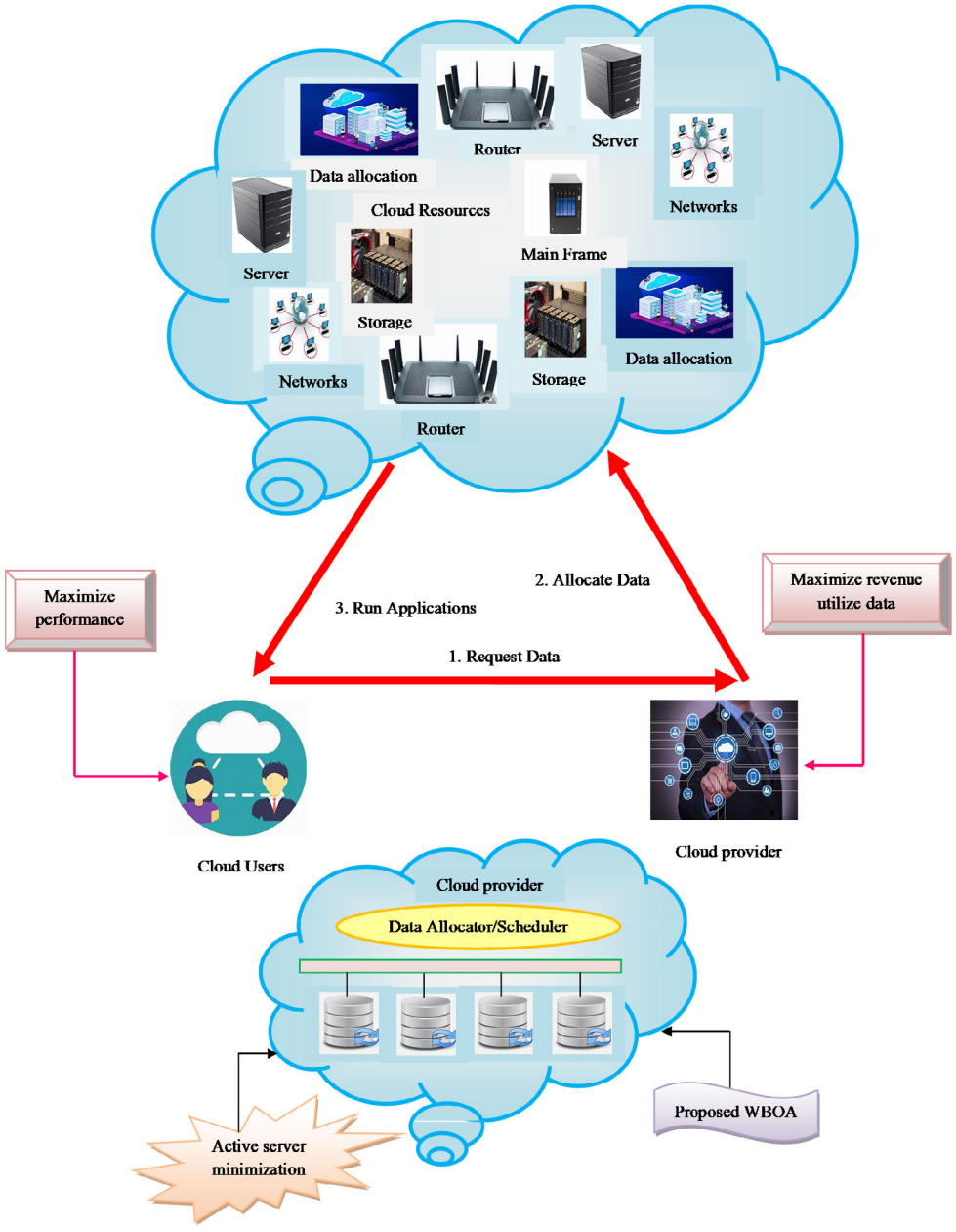

FIGURE 3 Data allocation of latency sensitive cloud object storage systems.

3.3 | Objective model for data allocation

A precise solution is presented to the data allocation problems. It begins with some basic notations and variables as follows. The group of the requested data records is shown by Δ, the group of requests is shown by $S(U, e, \alpha, \delta)$, the group of data records cached in the previous servers T_f, in which $T_f \subseteq T$ is shown by Δ_f, the group of servers are shown by T, CDF of server t for accessing data having a size e for I/O rate α is shown by $CDF_t^{e,\alpha}(y)$, maximum workload as well as storage limit of server t, in which $t \in T$ is shown by $M(t)$ and $D(t)$, Boolean array value representing whether the server $t \in T$ has stored data file e already is shown by $B[t][e]$, maximum session count for establishing on the server $t \in T$ for a single request is shown by O_t, a Boolean value representing whether data $e \in \Delta$ is stored in the server $t \in T$ is shown by Z_t^e, and a Boolean value representing whether the request s is stored by locating its requested data e on the server t and is served by j^{th} session is shown by $Q_{t,j}^s$. Thus, the objective of data allocation is shown in Equation (3). This reduces the overall count of servers utilized. For example, the greatest value Z_t^e is computed for the server $t \in T$, and if $Z_t^e = 1$ for some $e \in \Delta$, it is known that the server t is being used to store data e. Then, for every server $t \in T$, the total of $\max_{e \in E} Z_t^e$ is considered, and strive to minimize this result.

$$fit1 = \arg\min \sum_{t \in T} \max_{e \in E} Z_t^e. \tag{3}$$

The various constraints utilized are shown below. The data request time probability constraint is shown in Equation (4). Equation (4) assures that the chance of accessing data e during time U is at least δ for every request $s(U, e, \alpha, \delta)$. $Q_{t,j}^s \cdot CDF_t^{|e|,\alpha}(U)$ specifies the probability of accessing E the server t during time U by creating the j^{th} session. In this view, Equation (4) shows that at least one session on every server $t \in T$ has a chance of accessing E during the time U no less than δ by putting into consideration all O_t potential sessions on every server $t \in T$.

$$1 - \prod_{t \in T} \prod_{j=1}^{O_t} \left(1 - Q_{t,j}^s \cdot CDF_t^{|e|,\alpha}(U)\right) \geq \delta \forall s(U, e, \alpha, \delta) \in S. \tag{4}$$

The I/O rate constraint is shown in Equation (5). Equation (5) assures that every server's $t \in T$ maximum workload is not exceeded by the total consumed I/O rates.

$$\sum_{U(U,e,\alpha,\delta) \in S} \sum_{j=1}^{O} Q_{t,j}^U \cdot \alpha \leq M(t) \quad \forall t \in T. \tag{5}$$

The data allocation is described on a particular server as in Equation (10). Whether a data file $e \in \Delta$ is stored on the server $t \in T$ is determined by Equation (6).

$$Z_{e'}^t = \max_{1 \leq j \leq O, s \in S} Q_{t,j}^s \quad \forall e' \in \Delta, t \in T, where s \cdot e = e'. \tag{6}$$

The server capacity constraint is shown in Equation (7). It assures that every server's storage limit is not exceeded.

$$\sum_{e \in \Delta} Z_e^t \cdot |e| \leq D(t) \quad \forall t \in T. \tag{7}$$

The server allocation constraint is shown in Equation (8). Equation (8) assures that the server that contains the previous data file is employed first to fulfill the request.

$$B[t][e] \cdot Z_t^e \geq Z_{t'}^e \quad \forall e \in \Delta_f, t \in T_f, t' \in T \backslash T_f. \tag{8}$$

When the existing loaded servers T_f are insufficient to issue the requests, Equation (3) assures that the servers T_f are initially allocated to handle as many more requests as feasible. Equations (3)–(7) will then allocate (minimum) fresh empty servers to handle the remaining requests. When the traffic requests may be provided by fewer than $|T_f|$ servers, Equation (8) assures that some of the current $|T_f|$ servers are assigned to service the entire requests and that the variable $Z[e][t]$ for the other servers T_f is set to 0. If the above conditions are not satisfied under the data allocation phase, then a high penalty will be added to the fitness to avoid that solution.

3.4 | Proposed model

The cloud object storage system, representing a core cloud service, saves and recovers millions, if not billions, of read-heavy data items. Due to the large number of requests received every day, response latency is an important part of the user experience. Timeout is also an important

consideration because it contains a noteworthy influence on response latency. The present system is to overprovision resources to satisfy a service-level agreement (SLA) on response latency owing to a lack of accurate consideration of the prevalence of response latency and the frequency of timeouts. Modern web-oriented applications rely heavily on cloud object storage technologies such as OpenStack Swift and Amazon S3. Industrials create and manage their self-object storage systems apart from the public cloud object storage systems. For example, "Facebook utilizes Haystack for archiving photos, LinkedIn employs Ambry for retaining media objects, and Wikipedia utilizes OpenStack Swift clusters as media object stores for effectiveness and scalability." The hybrid meta-heuristic-based cloud object storage model is the basic cloud service, that saves and recovers even billions or millions of different data objects (also known as blobs), such as videos, images, documents, audio, and so on. Furthermore, the hybrid meta-heuristic-based cloud object storage model may instantly service millions of latency-sensitive Internet users. Cost efficiency is one of the key challenges for a hybrid meta-heuristic-based cloud object storage model, given the enormous amount of data objects and the long tail data access dispersal. Attaining cost efficiency requires a verified performance measure of the hybrid meta-heuristic-based cloud object storage model, which serves as the foundation for capacity planning. Data allocation in cloud computing is the technique of assigning available resources to required cloud benefits via the internet. If resource allocation is not managed precisely, services are starved. The reallocation of resources from less competitive to more key activities is a second key source of productivity improvement. The architecture of the introduced latency-sensitive cloud object storage system model is given in Figure 4.

This architecture provides a new approach for cloud object storage systems that address issues such as "latency-sensitive data allocation, latency-sensitive data re-allocation, and latency-sensitive workload consolidation." The primary contribution here is that effective data allocation,

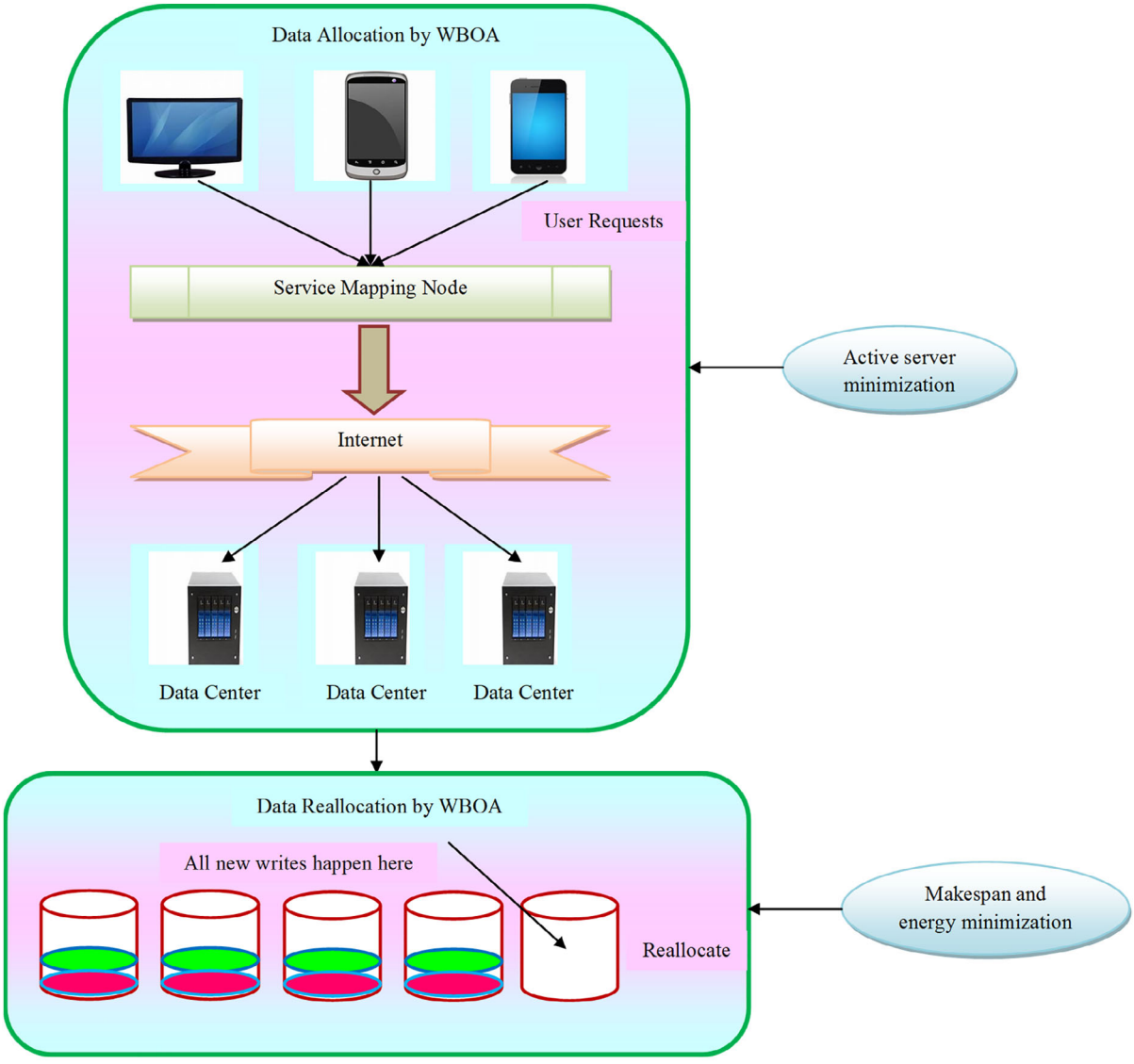

FIGURE 4 Proposed architecture for latency-sensitive cloud storage system.

data reallocation, and workload consolidation are performed by dispersing the latency of the hybrid meta-heuristic-based cloud object storage model. The main goal is to employ the smallest count of servers possible to fulfill the entire requests without breaking their latency needs so that the overall data transfer cost is kept low. As both are NP-hard issues, this is done by introducing a unique hybridized optimization algorithm termed WBOA. The data allocation focuses on the reduction of active servers as the objective function, while the data reallocation is concerned with the reduction of makespan and energy.

4 | HYBRID META-HEURISTIC-BASED LATENCY-SENSITIVE DATA ALLOCATION

4.1 | Proposed WBOA

The proposed WBOA is used for enhancing the data allocation and the data reallocation phases of the proposed cloud storage system. BOA[26] describes the features of the butterfly's food gathering. The optimization is carried out by the BOA's butterfly candidates. A lower-intensity fragrance is generated by the butterfly. Its fitness is linked to its intensity. The gain in fitness is due to the butterfly's movement. As the aroma travels such great distances, other butterflies may sense it, permitting them to share personal information. When a butterfly discovers the fragrance of other butterflies, it embarks on a worldwide hunt. The BOA shows several advantages such as handling a wider range of optimization problems, solving real-world problems, etc. but, it limits by drawbacks like it cannot handle combinatorial problems, cannot handle discrete as well as multi-objective issues, etc. thus, to avoid these drawbacks, WOA is merged into it and the resulting algorithm[27,28] is considered as WBOA. This WBOA shows advantages like lessening computation time, handling multi-objective optimization problems, etc.

The WOA[29] is a humpback whale-like creature. The bubble-net hunting approach is used to inspire it. "Bubble-net feeding is a unique activity seen only in humpback whales. The spiral bubble-net feeding technique is mathematically used to optimize it." In the proposed WBOA, the algorithm is modeled by two constants a and b based on the fitness procedure as in Equations (9) and (10).

$$a = |worstfit - fit(i)|. \tag{9}$$

$$b = |fit(i) - bestfit|. \tag{10}$$

In the above equations, the fitness is shown by $fit(i)$, the best fitness is shown by $bestfit$, and worst fitness is shown by $worstfit$, respectively. Next, the condition $a > b$ is checked. If this condition is satisfied, then the spiral upgrading location of WOA is as in Equation (11).

$$\vec{Y}(u+1) = \begin{cases} \vec{Y}^*(u) - \vec{B} \cdot \vec{D} & if\, q < 0.5 \\ \vec{D'} \cdot e^{cm} \cdot \cos(2\pi m) + \vec{Y}^*(u) & if\, q \geq 0.5 \end{cases}, \tag{11}$$

Here, the random number is shown by q, $\vec{D'} = \left|\vec{Y}^*(u) - \vec{Y}(u)\right|$ shows the length of the j^{th} whale to the prey, $\vec{B}$ defines the coefficient vector, the position vector of the best solution is shown by Y^*, the current iteration is shown by u, the position vector is shown by $\vec{Y}$, element-by-element multiplication is shown by $\cdot$, c defines a constant, and m defines a random number, respectively.

Otherwise, if $a \leq b$, then the updates are carried out during the global search stage of BOA as in Equation (12).

$$Y_j^{u+1} = Y_j^u + \left(s^2 \times h^* - y_j^u\right) \times g_j. \tag{12}$$

The solution vector is shown by Y_j^u for Y_j for the j^{th} butterfly in u iteration. The current best solution is shown by h^*, the random number is shown by s, and the fragrance is shown by g_j, respectively (Figure 5). The pseudo-code of WBOA is shown in Algorithm 1 and its flowchart is in Figure 5.

Algorithm 1. Proposed WBOA

Start
Population and parameter initialization
Compute Fitness Function
Describe index i and population size as $Npop$
For $u \rightarrow 1$ to u_{max}

For $i \rightarrow 1$ to $Npop$
Determine a and b by novel equations as in Equations (9) and (10)
If $a > b$
Update by the spiral updating position of WOA as in Equation (11)
else
Upgrade by the global search stage of BOA as in Equation (12)
End if
End for
End for
Stop

5 | LATENCY-SENSITIVE DATA REALLOCATION USING THE HYBRIDIZED META-HEURISTIC ALGORITHM

5.1 | Data reallocation model

From the standpoint of the providers, the most important problem to address is maximizing usage while lowering essential costs. "Deciding what, how many, where, and when to make a possible resource to a user" is known as resource reallocation. Users typically choose the kind and quantity of resource containers they want, and providers subsequently install the containers on nodes in their data centers. The kind of resource container should be adequately suited to the workload characteristics, and the amount should be essential to satisfy the limitations, that is, the task must be done before the deadline. It's equally vital to evaluate whether to make such modifications "in an elastic environment like the cloud, where users might request or return resources dynamically". The total reallocation procedure may be done by using a co-allocation and systematic allocation approach to reallocate the data that is available in the system.

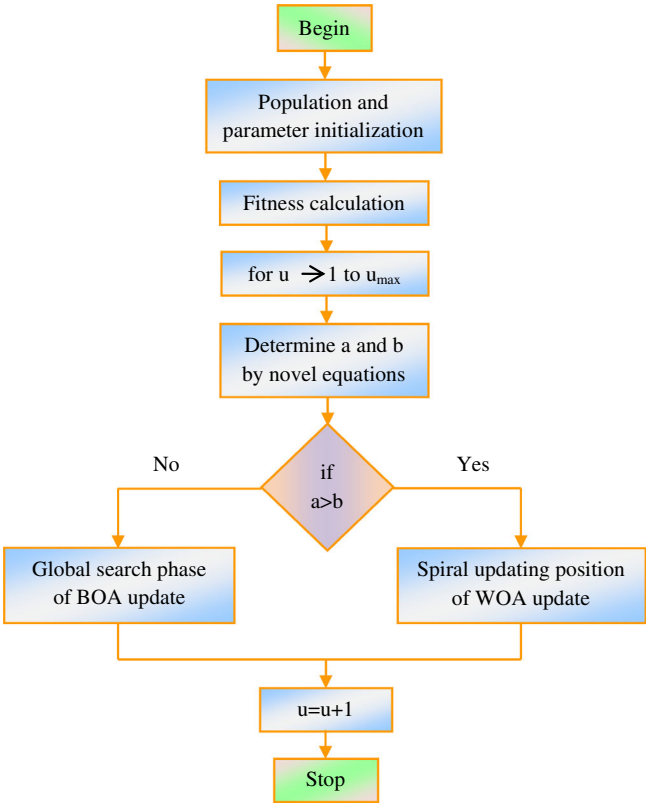

FIGURE 5 Flow diagram of offered WBOA.

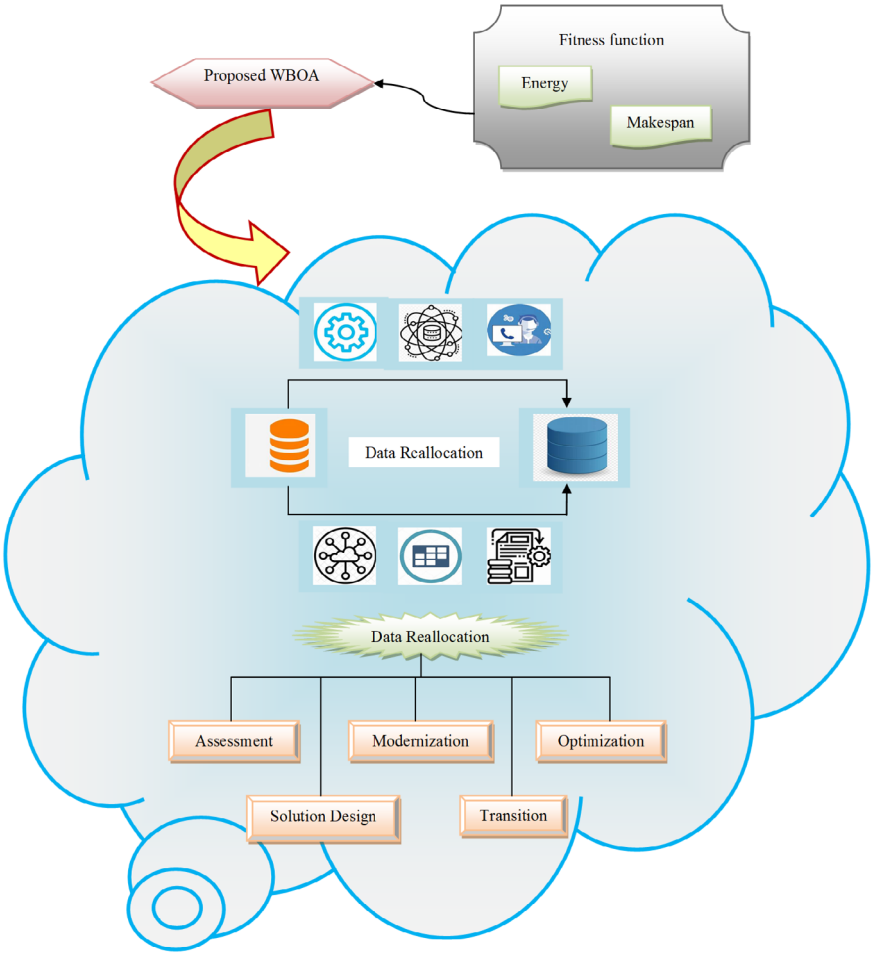

FIGURE 6 Data reallocation of latency sensitive cloud object storage systems.

The goal of the data reallocation is to reduce the number of running servers by reallocating data to underutilized servers, which is done with the WBOA. The data are transported to the server utilizing WBOA-assisted techniques. New techniques are developed, and data reallocation performance is improved by reducing the makespan and energy usage. As a result, after data allocation, the overloaded data in every server are taken for data reallocation and then relocated to the underloaded servers utilizing WBOA, therefore improving task completion efficiency in the cloud sector, thus reducing latency. If the energy and makespan are not minimized for deriving the objective function for the data reallocation, then a penalty will be added. The diagrammatic representation of the data reallocation for the latency-sensitive cloud object storage systems is shown in Figure 6.

5.2 | Solution optimization for data reallocation

Figure 7 shows the solution encoding of the data reallocation model using WBOA. The count of overloaded data files is 1 to the count of underloaded servers, and the count of servers is 1 to the count of underloaded servers. Furthermore, the solution length may be adjusted based on the results of trials with the count of overloaded servers and underloaded servers.

5.3 | Objective model for data reallocation

The second major goal of this suggested cloud computing paradigm is data reallocation to degrade the makespan MT and energy consumption Eo of the cloud storage system, which is accomplished via WBOA, which is given in Equation (13).

$$fit2 = \underset{\{N_1 \ldots N_N T_1 \ldots T_T\}}{\arg\min} \; (MT + Eo). \tag{13}$$

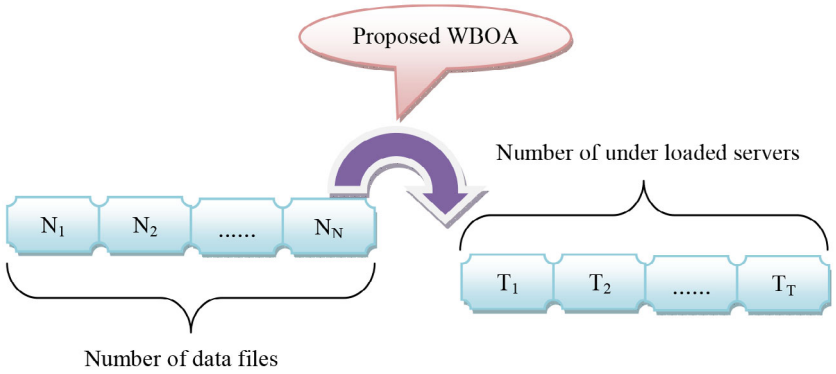

FIGURE 7 Solution encoding of data reallocation model using WBOA.

After executing the entire tasks, terminating servers, and hosts, the sum of energy spent by the hosts computing the entire tasks in the data center for a particular session will describe the data center energy consumption *Eo*, where energy consumption is obtained in Equation (14).

$$Eo = \int_k PQ(zt(k)). \tag{14}$$

As seen in $zt(k)$, CPU use represents a function of time. It is well understood that CPU utilization is proportional to the amount of power consumed by servers. The power is also taken by an active but idle server; therefore, the power model is written as Equation (15).

$$PQ(zt) = I \cdot PQ_{\max} + (1 - m) \cdot PQ_{\max} \cdot \xi. \tag{15}$$

In the above equation, the server's CPU utilization ξ is indicated as $\xi \in [0, 1]$, the proportion of power spent in an inactive state is indicated as m, and the maximum power utilized by a fully loaded server is defined as $PQ_{\max}$. This study aims to lower the count of lively servers to save energy by reducing the amount of energy consumed by the inactive state. The makespan MT is calculated using the memory and bandwidth needed for data reallocation as outlined in Equation (16).

$$MT_{tn} = \frac{MF_n}{BX_n}. \tag{16}$$

In Equation (21), the available network bandwidth is denoted by BX_n; the amount of memory used by VN_n is denoted by MF_n; and the time needed to complete the reallocation is denoted by MT_{tn}.

6 | RESULTS

6.1 | Experimental evaluation

The designed WBOA data allocation and reallocation in latency-sensitive cloud object storage systems were implemented in MATLAB 2020a and CloudSim, and a simulation study was conducted. Through the analysis of distinct constraints, the performance of the suggested method was compared to that of traditional models such as PSO,[30] GWO,[31] WOA,[29] and BOA.[26] Furthermore, the tests were carried out with a population of 10 and maximum iterations of 100. Each experiment in the designed method is run 10 times. The introduced model was carried out by considering various tests based on the number of files and servers, as given in Table 2.

6.2 | Convergence validation on data allocation

The convergence validation of the suggested data allocation on latency-sensitive cloud object storage systems is represented in Figure 8 when the average of servers is 100 and the average of files is 500, 160 and the count of files is 800, 200, and the count of files

TABLE 2 Experimentation description for the introduced model.

Experiments	Number of servers	Number of files
1	100	500
2	120	600
3	140	700
4	160	800
5	180	900
6	200	1000
7	220	1100
8	240	1200
9	260	1300
10	280	1400

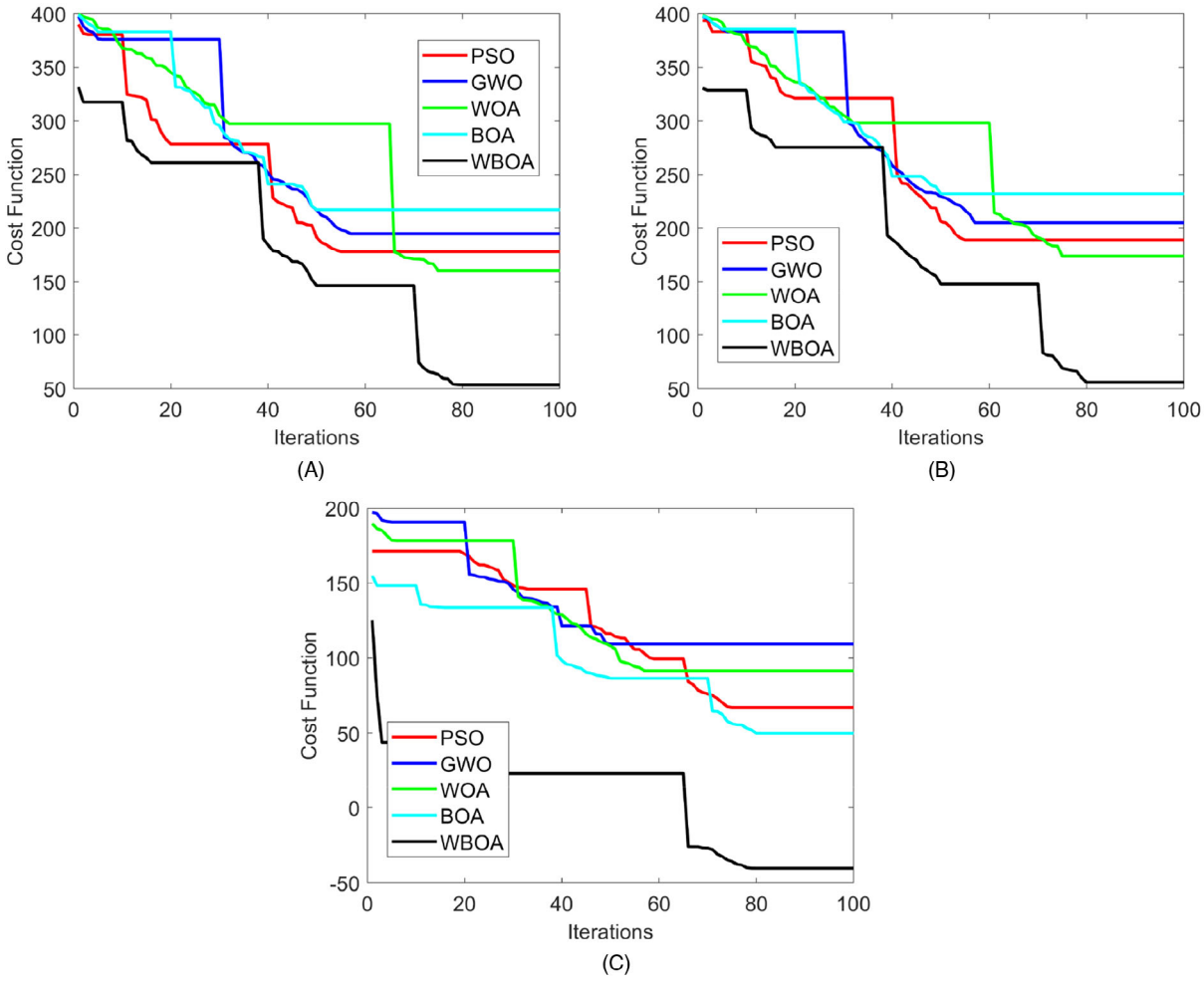

FIGURE 8 Convergence validation of the offered data allocation in latency-sensitive cloud object storage systems with consideration of "(A) No of servers as 100 and No of files as 500, (B) No of servers as 160 and No of files as 800, and (C) No of servers as 200 and No of files as 1000."

is 1000. Over early iterations, the cost function of the offered method is reduced and tested with alternative techniques, demonstrating the improved efficiency of the data allocation. From Figure 8A, at the 20th iteration, the cost function of WBOA is 8.93%, 27.12%, 22.85%, and 18.03% better than PSO, GWO, WOA, and BOA. While taking Figure 8B, at the 60th iteration, the cost function of WBOA is 46.47%, 31.04%, 31.16%, and 37.8% more advanced than PSO, GWO, WOA, and BOA. Based on this efficiency improvement, the designed and existing approaches have been arranged as WBOA, GWO, WOA, BOA, and PSO by computing the cost function values. As a result, when compared to various current methodologies, the created data allocation model outperforms them all.

6.3 | Convergence analysis on data reallocation

As shown in Figure 9, the convergence behavior of data reallocation in latency-sensitive cloud object storage systems utilizing WBOA is examined in various situations. In Figure 9 B, at the 90th iteration, the cost function of WBOA is 96.61%, 95.34%, 93.33%, and 90.90% higher than PSO, GWO, WOA, and BOA. While taking Figure 9C, at the 80th iteration, the cost function of WBOA is 96.71%, 95.01%, 94.48%, and 91.95% more than PSO, GWO, WOA, and BOA. While analyzing the performance enhancement of the offered and baseline techniques has been ordered as WBOA, BOA, WOA, GWO, and PSO using the computation of cost function values. As a result, the convergence of the recommended data reallocation by WBOA has demonstrated that the suggested model is more efficient than previous approaches.

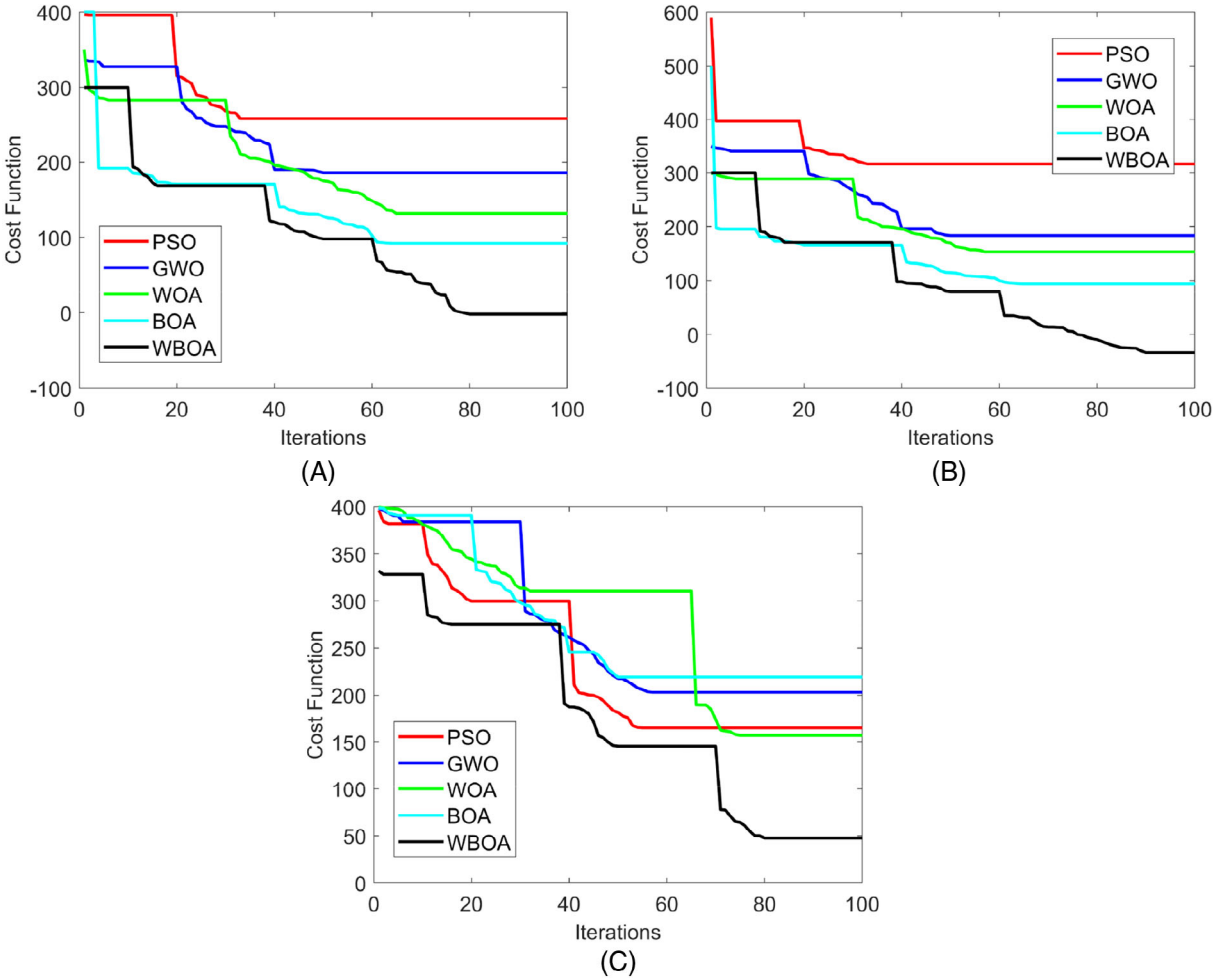

FIGURE 9 Convergence validation of the offered data reallocation in latency-sensitive cloud object storage systems with consideration of (A) no of servers as 100 and no of files as 500, (B) no of servers as 160 and no of files as 800, and (C) no of servers as 200 and no of files as 1000.

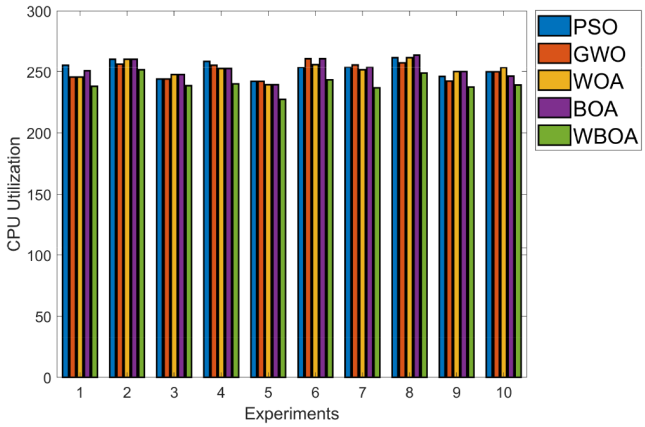

FIGURE 10 Average CPU utilization analysis for the suggested data allocation and reallocation strategy in latency-sensitive cloud object storage systems through altering the experiments.

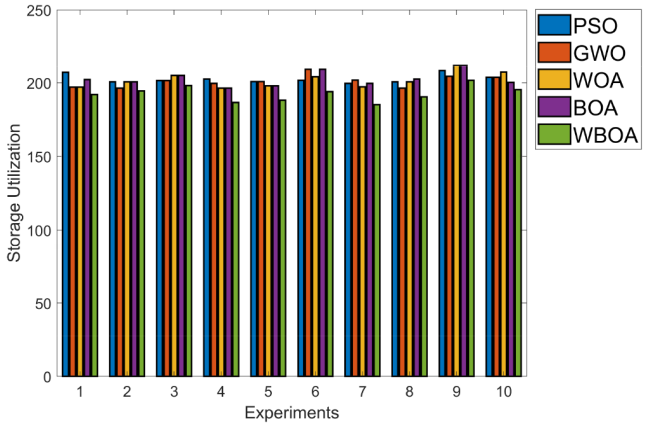

FIGURE 11 Average memory utilization analysis for the suggested data allocation and reallocation strategy in latency-sensitive cloud object storage systems through altering the experiments.

6.4 | CPU Utilization analysis

Figure 10 shows the CPU usage performance for the proposed data allocation and data reallocation. While running various trials, the WBOA shows less CPU utilization than the existing methods. In the fourth experiment, the average CPU utilization of WBOA is 14.81%, 13.37%, 11.53%, and 11.53% surpassing PSO, GWO, WOA, and BOA, respectively. As a result, the designed WBOA is more efficient than traditional approaches for data allocation and reallocation.

6.5 | Memory utilization analysis

Various trials, as shown in Figure 11, confirm the performance of the established data allocation and reallocation method in latency-sensitive cloud object storage systems concerning average memory use. In the 6th experiment, the memory utilization of WBOA is 9.5%, 13.8%, 11.6%, and 13.8% more progressed than PSO, GWO, WOA, and BOA, respectively. As a result, when compared to conventional techniques, the developed data allocation and reallocation approach improves performance in terms of average memory use.

6.6 | Makespan analysis

As shown in Figure 12, the proposed model was studied concerning makespan to indicate the resource use of data allocation and reallocation. From Figure 12A, in the 9th experiment, the makespan of WBOA is 90.49%, 89.55%, 93.84%, and 70.42% higher than PSO, GWO, WOA, and BOA

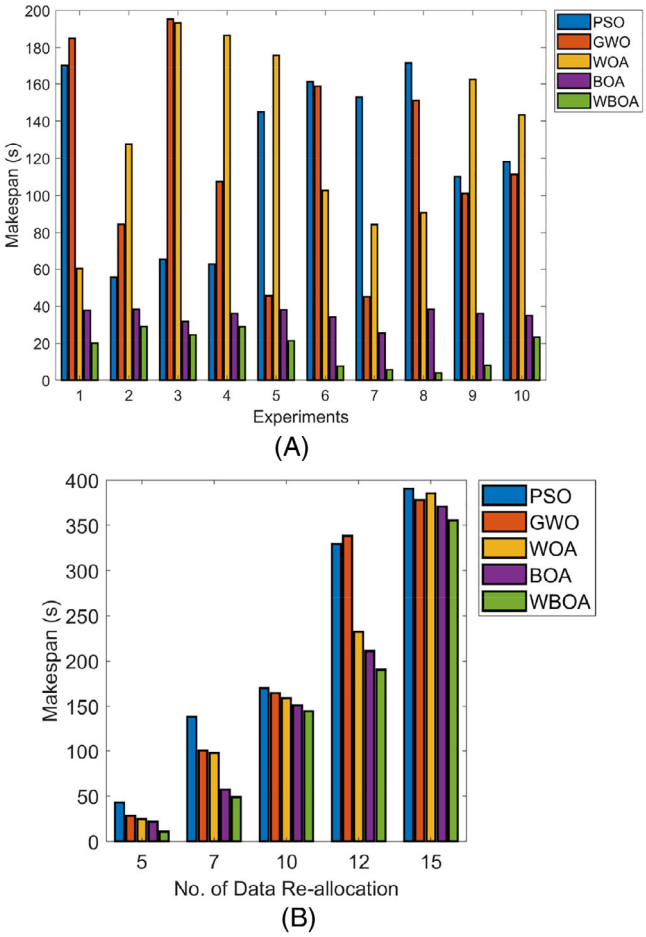

FIGURE 12 Makespan analysis for the suggested data allocation and reallocation strategy in latency-sensitive cloud object storage systems by WBOA through altering (A) experiments and (B) the number of data reallocation.

respectively. Similarly, in Figure 12B, when 10 data are taken for reallocation, the makespan of WBOA is 22.22%, 17.64%, 12.5%, and 6.6% improved than PSO, GWO, WOA, and BOA, respectively. The developed WBOA enables greater resource usage concerning makespan.

6.7 | Evaluation of data allocation

Table 3 shows the evaluation of the recommended data allocation concerning the "number of servers, cost function, and computation time." In experiment 5, the cost function of WBOA is 54.64%, 47.66%, 2.15%, and 66.37% improved than BOA, WOA, GWO, and PSO, respectively. The computation time of WBOA is 9.43%, 1.73%, 1.42%, and 1.62% surpassed that of BOA, WOA, GWO, and PSO, respectively. Table 4 also includes a statistical evaluation of the offered data allocation model, which is necessary owing to the stochastic nature of optimization-oriented algorithms. "Measures like best, worst, mean, median, and standard deviation" are used. The mean is "the average value of the best and worst values, the median is the center point of the best and worst values, and the standard deviation is the degree of variation between every execution." In the 6th experiment, the mean of WBOA is 4.54%, 11.67%, 40.51%, and 22.78% more than BOA, WOA, GWO, and PSO respectively. As a result of the recommended model, the performance of both data allocation and reallocation utilizing the WBOA has improved.

6.8 | Comparative analysis of data reallocation

Tables 5 and 6 depict the comparative and statistical analysis of the data reallocation under various constraints. The superiority of the offered method employing WBOA has been demonstrated in comparison to other traditional methodologies.

TABLE 3 Performance analysis of the suggested data allocation strategy in latency-sensitive cloud object storage systems by WBOA algorithm with distinct techniques for Experiment 1.

			PSO[30]			GWO[31]			WOA[29]		
Description	No.of servers	No. of data files stored	No. of active servers	Cost Function	Computation Time	No.of active servers	Cost Function	Computation Time	No.of active servers	Cost Function	Computation Time
Experiment 1	100	500	98	1181.4	1.6019	100	1106.9	1.5235	100	1339.4	1.3912
Experiment 2	120	600	119	1054.3	3.7681	120	1473.1	3.4328	119	1014.7	3.2788
Experiment 3	140	700	140	1030.4	4.6044	140	372.61	4.5368	139	1424.4	4.348
Experiment 4	160	800	158	827.68	5.6724	159	789.18	5.063	160	1320.7	4.7842
Experiment 5	180	900	179	340.83	6.5805	179	1434.1	5.9866	180	699.52	5.8551
Experiment 6	200	1000	200	1189.7	9.0153	197	2218.9	8.7352	199	1758.5	8.575
Experiment 7	220	1100	218	2161.5	10.223	217	1350.2	10.126	219	2130	10.045
Experiment 8	240	1200	238	2099	10.772	240	2398.2	10.725	238	2624	10.559
Experiment 9	260	1300	257	1337.4	11.86	259	2342.7	11.473	255	1567.8	11.022
Experiment 10	280	1400	277	2655	13.015	277	1916.7	12.83	275	3360.7	12.757

			BOA[26]			Proposed WBOA		
Description	No. of active servers	No. of data files stored	No. of active servers	Cost function	Computation time	No.of active servers	Cost function	Computation time
Experiment 1	100	500	99	1469.2	1.3501	99	1064.5	1.2834
Experiment 2	120	600	119	497.5	3.1861	118	1112	3.1696
Experiment 3	140	700	139	1483.6	4.0257	138	750	3.7869
Experiment 4	160	800	160	1617.3	4.6998	160	866.56	4.6394
Experiment 5	180	900	180	843.57	5.8511	180	1143.6	5.7387
Experiment 6	200	1000	197	1533.5	8.0698	199	1482.5	8.0392
Experiment 7	220	1100	218	1632	10.023	220	2254.2	9.4455
Experiment 8	240	1200	237	3353.2	10.531	238	3087.2	10.37
Experiment 9	260	1300	255	1171.5	10.965	255	1801.7	10.811
Experiment 10	280	1400	279	1557.2	12.055	276	4134.5	12.023

TABLE 4 Statistical analysis of the suggested data allocation strategy in latency-sensitive cloud object storage systems with distinct techniques for Experiment 2.

	PSO[30]			GWO[31]			WOA[29]		
Description	Experiment 1	Experiment 4	Experiment 6	Experiment 1	Experiment 4	Experiment 6	Experiment 1	Experiment 4	Experiment 6
Best	98.046	160.56	387.18	96.27	159.56	200.4	96.27	158.62	235.98
Worst	280.97	550.11	912.83	100.5	356.17	744.53	101.37	338.56	888.94
Mean	145.45	337.05	608.91	98.709	234.88	422.44	99.691	226.31	533.12
Median	101.21	318	628.81	99.334	160.56	388.38	100.05	159.56	372.24
Standard deviation	78.672	151.17	202.32	1.6978	102.44	201.62	2.002	92.33	298.8

	BOA[26]			Proposed WBOA		
Description	Experiment 1	Experiment 4	Experiment 6	Experiment 1	Experiment 4	Experiment 6
Best	97	158	266.9	33,158	64,743	79,818
Worst	254.27	576.9	660	43,119	70,588	85,784
Mean	142.16	430.11	453.2	38,705	67,544	84,176
Median	99.27	510.73	467.68	40,914	67,560	84,758
Standard deviation	68.497	179.78	143.56	4276.8	2572.7	2488.9

TABLE 5 Performance analysis of the introduced data reallocation strategy in latency-sensitive cloud object storage systems with distinct algorithms for Experiment 1.

			PSO[30]		GWO[31]		WOA[29]	
Description	No.of servers	No. of data files stored	Cost Function	Computation Time	Cost Function	Computation Time	CostFunction	Computation Time
Experiment 1	100	500	750.61	4.5271	684.11	4.3237	775.15	4.2808
Experiment 2	120	600	1518.4	5.5984	1341.7	5.0931	1025.3	5.0584
Experiment 3	140	700	1088.3	6.0947	1705.1	5.8154	746.58	5.7698
Experiment 4	160	800	1477	7.2614	1720.7	7.2522	1311.6	6.8127
Experiment 5	180	900	1453	7.6211	925	7.6201	758.26	7.4925
Experiment 6	200	1000	3765.3	8.3323	2507	8.0918	1452.3	8.0782
Experiment 7	220	1100	1897	8.9978	1170.8	8.8874	2249.4	8.7148
Experiment 8	240	1200	1681.4	9.3011	2440.1	9.2571	2844.5	9.217
Experiment 9	260	1300	1926.4	9.7811	2735.1	9.5928	1885.7	9.3552
Experiment 10	280	1400	1053.1	10.293	2483.5	10.24	2368.3	10.215

			BOA[26]		Proposed WBOA	
Description	No.of servers	No. of data files stored	Cost function	Computation time	Cost function	Computation time
Experiment 1	100	500	823.16	4.2359	43,373	4.0972
Experiment 2	120	600	1015.2	4.971	53,468	4.9533
Experiment 3	140	700	1015.4	5.7108	61,497	5.6082
Experiment 4	160	800	1773.5	6.5692	70,988	6.4201
Experiment 5	180	900	927.05	7.3577	77,873	7.3243
Experiment 6	200	1000	1769.3	7.9243	90,235	7.7388
Experiment 7	220	1100	2649	8.5633	96,923	8.443
Experiment 8	240	1200	1575.2	9.0624	1.08 E+05	9.0165
Experiment 9	260	1300	2394.3	9.3259	1.14 E+05	9.3117
Experiment 10	280	1400	2904.2	10.185	1.23 E+05	9.9302

TABLE 6 Statistical analysis of the introduced data reallocation strategy in latency-sensitive cloud object storage systems with distinct algorithms for Experiment 2.

	PSO[30]			GWO[31]			WOA[29]		
Description	Experiment 1	Experiment 4	Experiment 6	Experiment 1	Experiment 4	Experiment 6	Experiment 1	Experiment 4	Experiment 6
Best	97	161.19	200.2	98	156	198.27	99.17	158.22	199.27
Worst	99.322	700.43	268.5	100.5	659.42	497.27	185	549.95	676.27
Mean	97.729	462.1	213.9	98.934	348.13	360.74	117.95	337.54	397.67
Median	97.322	469.08	200.27	98	175.71	354	100.42	295	372.39
Standard deviation	0.97975	201.6	30.52	1.2836	254.77	108.89	37.587	155.83	175.85

	BOA[26]			Proposed WBOA		
Description	Experiment 1	Experiment 4	Experiment 6	Experiment 1	Experiment 4	Experiment 6
Best	99.17	250	199.53	96	156.58	199.2
Worst	195	468.61	602	98.07	713.08	200.27
Mean	136.55	372.39	379.2	97.614	495.16	199.66
Median	99.386	409.47	339.21	98	516	199.27
Standard deviation	51.11	100.98	163.23	0.90276	209.91	0.56245

6.9 | Evaluation of the ANOVA test

Evaluation of the ANOVA Test for the suggested data allocation and reallocation strategy in latency-sensitive cloud object storage systems is shown in Figure 13. Hence, it is revealed that the designed WBOA method attains more advanced results than the other baseline approaches for all experiments.

The estimation of the average convergence result of data allocation and reallocation strategy in latency-sensitive cloud object storage systems is given in Table 7. While taking Table 7, the average convergence results of the designed WBOA method achieve a minimum result. Hence, it is revealed that offered WBOA method attains elevated performance.

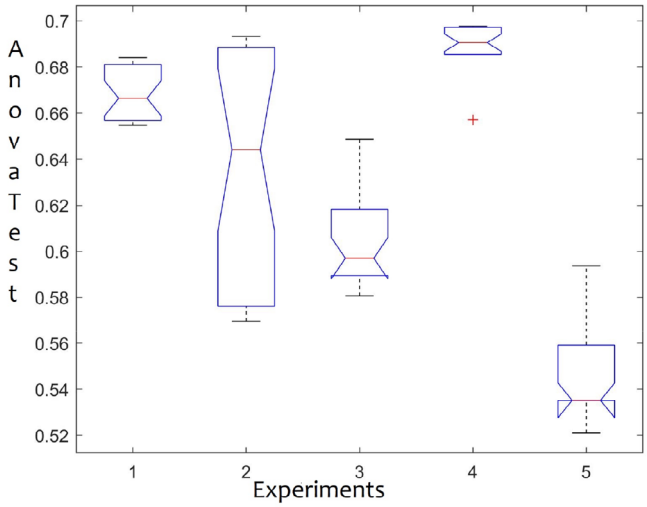

FIGURE 13 Estimation of ANOVA teat for the offered data allocation and reallocation strategy in latency-sensitive cloud object storage systems 6.10 Estimation of average convergence results for all algorithms.

TABLE 7 Estimation of average convergence result of data allocation and reallocation strategy in latency-sensitive cloud object storage systems.

PSO[30]	GWO[31]	WOA[29]	BOA[26]	Proposed WBOA
No. of servers as 100 and no. of files as 500				
171.2301	152.4085	119.4171	79.23072	67.15561
191.3586	144.3719	113.7402	109.4071	109.4071
179.7112	156.7273	105.7225	91.5776	91.5776
141.5588	130.2582	88.72724	71.88592	49.89303
117.9091	98.23006	98.23006	56.30103	34.61474
No. of servers as 160 and no. of files as 800				
391.6943	274.4103	258.4697	258.4697	258.4697
328.9633	244.2105	187.4956	185.8245	185.8245
287.6635	246.0414	246.0414	133.4213	131.8379
216.1076	170.6877	170.6877	92.33795	92.17401
313.2034	238.6977	177.7886	109.9962	73.26089
No. of servers as 200 and no. of files as 1000				
404.6258	326.5156	316.4528	316.4528	316.4528
341.4327	262.322	187.7852	183.4927	183.4927
290.164	246.7949	168.9128	153.162	153.162
199.9156	165.3381	116.3816	93.85004	93.49662
313.8709	283.6343	159.0356	89.85663	46.2982

7 | CONCLUSION

The core aim of this work was to provide a new approach for cloud object storage systems that addressed issues such as latency-sensitive data allocation, and latency-sensitive data re-allocation. The primary contribution here was that effective data allocation, and data re-allocation, were carried out by dispersing the latency of the hybrid meta-heuristic-based cloud object storage system. The main goal was to employ the smallest number of servers possible to fulfill all requests without breaking their latency requirements so that the overall data transfer cost was kept low. As a result, this work has offered a unique hybrid meta-heuristic algorithm called WBOA to solve NP-hard problems. From the experimental evaluation, for data allocation, the convergence analysis of WBOA was 54.64%, 47.66%, 2.15%, and 66.37% improved than BOA, WOA, GWO, and PSO, respectively. similarly, for data reallocation, the cost function of WBOA was 86.59%, 84.73%, 61.13%, and 68.64% more progressed than BOA, WOA, GWO, and PSO, respectively. Thus, the overall analysis showed that the suggested WBOA was better than the other methods for latency-sensitive cloud object storage systems.

DATA AVAILABILITY STATEMENT

No new data were generated or analysed in support of this research.

ORCID

N. Nataraj https://orcid.org/0000-0002-1190-8140

REFERENCES

1. Schulz P, Ong L, Abdullah B, Simsek M, Fettweis G. End-to-end latency distribution in future mobile communication networks. *Smart Antennas*. 2020;1-5.
2. Yoon SK, Youn YS, Son MH, Kim S-D. Harmonized memory system for object-based cloud storage. *Cluster Comput*. 2018;21:15-28.
3. Huang C, Abdelzaher T. Bounded-latency content distribution feasibility and evaluation. *IEEE Trans Comput*. 2005;54(11):1422-1437.
4. Firmin L, Müller S, Rösler KM. The latency distribution of motor evoked potentials in patients with multiple sclerosis. *Clin Neurophysiol*. 2012;123(12):2414-2421.
5. Ni Z, Leodori G, Vial F, et al. Measuring latency distribution of transcallosal fibers using transcranial magnetic stimulation. *Brain Stimul*. 2020;13(5):1453-1460.
6. Ghosh SK, Burns CB, Prager DL, Zhang L, Hui G. On nonparametric estimation of the latent distribution for ordinal data. *Comput Stat Data Anal*. 2018;119:86-98.
7. Grozev N, Buyya R. Regulations and latency-aware load distribution of web applications in multi-clouds. *J Supercomput*. 2016;72:3261-3280.
8. Smith DG, Mewhort DJK. The distribution of latencies constrains theories of decision time: a test of the random-walk model using numeric comparison. *Aust J Psychol*. 1998;50(3):149-156.
9. Celesti A, Galletta A, Fazio M, Villari M. Towards hybrid multi-cloud storage systems: understanding how to perform data transfer. *Big Data Res*. 2019;16:1-17.
10. Leyva-Mayorga I et al. A hybrid method for obtaining the distribution of report latency in wireless sensor networks. *Wireless and Mobile Networking. IEEE*; 2015.
11. Szymaniak M, Presotto D, Pierre G, van Steen M, Amsterdam VU. Practical large-scale latency estimation. *Comput Netw*. 2008;52(7):1343-1364.
12. Mulinti RB, Nagendra M. An Efficient latency aware resource provisioning in cloud assisted mobile edge framework. *Peer-to-Peer Netw Appl*. 2021; 14:1044-1057.
13. Choy S, Wong B, Simon G, Rosenberg C. A hybrid edge-cloud architecture for reducing on-demand gaming latency. *Multimed Syst*. 2014;20:503-519.
14. Sharma P, Zhichen X, Banerjee S, Lee S-J. Estimating network proximity and latency. *ACM SIGCOMM Comput Commun Rev*. 2016;36(3):39-50.
15. Nielsen JJ, Popovski P. Latency analysis of systems with multiple interfaces for ultra-reliable M2M communication. *Wirel Commun*. 2016;1-6.
16. Wang T, Xu K, Song M. A scalable network proximity estimate algorithm for the service provider selection method. *Human Centered Computing*. Springer; Vol 8944: 2015.
17. Chandramouli B, Goldstein J, Barga R, Riedewald M, Santos I. *Accurate Latency Estimation in a Distributed Event Processing System*, *Data Engineering*. Hannover; 2011:255-266.
18. Su Y, Feng D, Hua Y, Shi Z. Uunderstanding the latency distribution of cloud object storage systems. *Journal of Parallel Distrib Comput*. 2019;128:71-83.
19. Kou S, Xia W, Zhang X, Gao Q, Gao X. Self-supervised graph convolutional clustering by preserving latent distribution. *Neurocomputing*. 2021;437:218-226.
20. Arora S, Bala A. An intelligent energy efficient storage system for cloud based big data applications. *Simul Modell Pract Theory*. 2021;108:102260.
21. Wu L, Zhuge Q, Sha EH, Chen X, Cheng L. BOSS: an efficient data distribution strategy for object storage systems with hybrid devices. *IEEE Access*. 2017;5:23979-23993.
22. Shen T, Nagai Y. Non-homogeneous distributed cloud storage system with minimal redundancy in heterogeneous environment. *Measurement*. 2019;144:1-6.
23. Baun C, Cocos HN, Spanou RM. OSSperf – a lightweight solution for the performance evaluation of object-based cloud storage services. *J Cloud Comp*. 2017;6:24.
24. Yin J et al. ASSER: an efficient, reliable, and cost-effective storage scheme for object-based cloud storage systems. *IEEE Transactions on Computers*. 2017;66(8):1326-1340.
25. Mohammed SA. Artificial intelligence-based latency estimation for distributed systems. *Adv Artif Intell*. 2019;609-612.
26. Arora S, Singh S. Butterfly optimization algorithm: a novel approach for global optimization. *Soft Comput*. 2018;23:715-734.
27. Jose D, Kumar PN, Shirley JaA, Ghayathrrie S. *Implementation of Genetic Algorithm framework for Fault-Tolerant System on Chip*. Vol 17. International Information Institute (Tokyo) Information; 2014:3921-3945.

28. Roy D, Dutta M. A systematic review and research perspective on recommender systems. *J Big Data*. 2022;9:59.
29. Mirjalili S, Lewis A. The whale optimization algorithm. *Adv Eng Softw*. 2016;95:51-67.
30. Mahi M, Baykan OK, Kodaz H. A new approach based on particle swarm optimization algorithm for solving data allocation problem. *Appl Soft Comput*. 2018;62:571-578.
31. Sanjay R, Jayabarathi T, Raghunathan T, Ramesh V. Optimal allocation of distributed generation using hybrid Grey wolf optimizer. *IEEE Access*. 2017;1-1:99.

How to cite this article: Nataraj N, Nataraj RV. A novel hybrid meta-heuristic-oriented latency sensitive cloud object storage system. *Concurrency Computat Pract Exper*. 2023;e7672. doi: 10.1002/cpe.7672