Received February 23, 2021, accepted April 2, 2021, date of publication April 6, 2021, date of current version April 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3071325

SLA-Aware Multi-Criteria Data Placement in Cloud Storage Systems

MOHAMMAD MAGHSOUDLOO[1], AREZOO RAHDARI[1], AND NAVID KHOSHAVI[2,3]
[1]Department of Computer Engineering, Golestan University, Gorgan 39361-79142, Iran
[2]Department of Computer Science, Florida Polytechnic University, Lakeland, FL 33805, USA
[3]Department of Electrical and Computer Engineering, Florida Polytechnic University, Lakeland, FL 33805, USA

Corresponding author: Mohammad Maghsoudloo (mo.maghsoudloo@gu.ac.ir)

ABSTRACT This paper proposes a multi-criteria data placement mechanism for typical cloud storage systems. The primary goal of this study is to place the clients' files in the storage cluster by taking into account the principles of service level agreements and the levels of user support services. For this purpose, a multi-criteria scoring model is defined based on a trade-off between three criteria that can be crucial from the clients' viewpoint: data access latency, data privacy, and computational performance. The second goal of the proposed mechanism is to improve the fairness of the storage allocation mechanism for various kinds of files compared to the conventional mechanisms. In this case, the cloud service provider gives each data storage node a chance to become the host of an already received file using a probabilistic ranking model. The experimental results demonstrate that the proposed data placement mechanism increases the robustness of cloud storage architecture under different file, network, and storage configurations compared to the existing data placement mechanisms.

INDEX TERMS Cloud storage systems, Hadoop distributed file system, data placement, service level agreement, data storage nodes.

I. INTRODUCTION

The emerging adoption of cloud computing in different aspects of information technology such as financial services, social networks, e-health, media, and entertainment drives the growing demand for cloud storage systems [1]–[3]. A cloud storage system provides a means of permanent storage of a set of data in the form of files in which clients can access files via a lightweight user agent [1]–[3]. The Cloud storage market is projected to reach a total market size of $92.488 billion by 2022, with a compound annual growth rate of 53% from 2011 to 2016 [3].

Even though cloud storage provides many intrinsic benefits, some concerns, such as the lack of knowledge and control on the physical locations of data, still prevent many customers from migrating to the cloud [4]. Adherence to the terms and principles promised by cloud providers is the key point to attract and retain current and potential customers [4]–[6]. Nowadays, several methods are provided for customers to verify contractual obligations and detect malicious or accidental misbehavior on the part of the service provider [7]–[11]. Therefore, there is a clear need for service level agreements (SLA) that allows cloud storage users to specify requirements and concerns about their files [7].

The geographic region of storing data is one of the most influential parameters affecting various customer service objectives, including responsiveness, availability, and privacy [12]. For example, a cloud storage provider (CSP) may require its customer-serving data centers located within the continental of its customers to improve load time and responsiveness [13]. Moreover, many privacy laws, such as those in Canada, Australia, and the EU, require citizens' data remain stored within a political border or other countries with comparable protections [14]. Therefore, the data placement mechanism, deployed by a CSP, plays a vital role in enhancing or degrading the quality of cloud storage and computing services [12]–[14].

The Hadoop cluster usage is widely spread as a CSP in different business and academic spheres [15]. Hadoop is a popular implementation of Google's MapReduce contributed by Cloudera, Facebook, Microsoft, Yahoo, Citrix, IBM, and many more companies [16], [17]. The core of Hadoop consists of a storage part, known as Hadoop Distributed File System (HDFS), and a processing part called

The associate editor coordinating the review of this manuscript and approving it for publication was Weizhi Meng.

MapReduce [16], [17]. The default file placement strategy of HDFS assumes that each node has the same computation capacity and capability in a homogeneous environment. This strategy has its potential benefits in a homogeneous environment; however, it might not be suitable in a heterogeneous environment. The heterogeneity of state-of-the-art cloud environments raises such problems as optimizing data placement and distribution across the cluster's computing and storage resources [18].

The main goal of most previous data placement techniques is to make the heterogeneity of data storage nodes transparent to the clients. This goal was achieved by picking the best choices for storing the clients' files based on a predetermined criterion, user concerns, and specifications of data storage nodes [19]–[23]. For instance, there are some data placement strategies to locate files in the storage cluster so that the response time of HDFS clients can be improved [19], [20], [37]. Moreover, different innovative ideas have been proposed to distribute file blocks among data storages to enhance cloud computation time [21], [22], [36]. Furthermore, some data placement techniques have concentrated on moderating the risk of unauthorized access to a file in the storages [23]. However, the increased heterogeneity of large-scale cloud storage and diversity of clients causes that the previous data placement mechanisms face two serious problems:

1) The single-criterion approaches for selecting the available data storage nodes cannot simultaneously resolve more than one type of clients' concerns.
2) Selecting the best choices among data storage nodes for each uploaded file can be unfairly favored the files received earlier than the other ones.

This paper proposes an SLA-aware probabilistic multi-criteria data placement to address the above limitations based on two key innovations:

1) In the proposed data placement mechanism, the decision of placing a file in a cluster is taken concerning the heartbeat of data storage nodes, the client concerns, and a trade-off among three criteria that can affect the quality of service in terms of different design parameters.
2) Contrary to the traditional greedy strategy, the proposed data placement mechanism gives each data storage node a chance to become the host of an already received file. The chance parameter is estimated based on the gain of adding files to each data storage node with respect to the levels of user support services.

Three metrics influencing data access latency, data privacy, and computing performance have been defined to assess data placement mechanisms: File Access Latency, File Breach Probability, and File Correlation Factor. The File Access Latency is the time interval between issuing a client's file access request and accomplishing the request by the CSP. This metric can be dynamically measured, considering distance-bounding protocols and estimations. The File Breach Probability is the probability of the intentional or unintentional release of private files to an untrusted environment. This metric can estimate the risk of compromising data storage nodes for leaking some specific files within the cloud storage system. The File Correlation Factor is the sum of network delays between the host of a file and the host(s) of its correlated one(s). If some files are always used together by many jobs, they are supposed to be correlated. Collocating the related data blocks of a file or some files on the same or adjacent data storage nodes can reduce the network overhead and is beneficial for MapReduce's performance.

The main contribution of this work is to propose a Gain estimation model for scoring data storage nodes that are available for hosting an uploaded file. In the estimation model, the mentioned three parameters of the quality of cloud storage services have been quantized and aligned. Moreover, three weighting factors have been defined to tune the importance of each parameter in the estimation model based on the SLA and concerns of clients. Furthermore, the drawbacks of greedy strategies to pick the best available choices have been removed in the structure of the proposed data placement technique. This issue has been achieved by taking advantage of a probabilistic selection mechanism that also reflects the level of user support services.

The CloudSim [24] framework has been enhanced to reflect the user's demands in terms of different expected services to implement design decisions. Furthermore, six different scenarios have been conducted to study the performance of each data placement mechanism. An individual data storage node or file specification has been concentrated as the target parameter in each scenario. The results reveal that the proposed probabilistic data placement mechanism improves the File Access Latency, File Breach Probability, and File Correlation Factor of the Hadoop by about 16.6%, 15.5%, and 15.2%. It obtains more stable results with lower deviation under different files, networks, and storage conditions than the other data placement mechanisms. This issue indicates that the chance of proceeding towards the best global solution is often higher for the proposed probabilistic mechanism compared to the conventional greedy single-objective mechanisms.

The rest of the paper is organized as follows: Section 2 describes the base Hadoop distributed file system, the problem, and related work structures. The features of the proposed data placement mechanism are introduced in Section 3. Section 4 shows the result analysis. Finally, Section 5 concludes the paper.

II. BACKGROUND

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models [15]. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage [15]. The Hadoop Distributed File System (HDFS) and the MapReduce framework are two essential Hadoop components [16]. The HDFS is a scalable, fault-tolerant, distributed storage system that works closely with various concurrent data access applications [25], [26]. An HDFS cluster consists of a single NameNode that manages the file system namespace; and some DataNodes that contain storages attached to the clusters [26]. The DataNodes perform object creation, deletion, and replication under the management of NameNode [26], [27]. The NameNode executes file system namespace operations like opening, closing, and renaming files. It also partitions the files that are written in HDFS into many same-sized blocks. It then allocates these blocks to different DataNodes [26]. The default file/data placement strategy of Hadoop assumes that each node has the same capabilities such as processing power, proximity to clients, security robustness, availability, and cost in a homogeneous environment [19]. Hadoop balances the load by distributing the blocks to each node randomly [19]. Such a data placement strategy can achieve load balance, which is highly efficient in a homogeneous cluster [18]. Nevertheless, in real-world applications, clusters are often worked in a heterogeneous environment [18]. Thus, there is a considerable difference in the consequences of storing data on different DataNodes. In this case, using the Hadoop strategy leads to a reduction of overall performance and clients' satisfaction [19]. Therefore, the need for purposive data placement mechanisms has been felt to enhance the performance of Hadoop in terms of different impressive factors [19]–[22].

Different ideas have been made to distribute file blocks among DataNodes to reduce job execution time or data transfer time [21], [22], [36]. Under this category, data placement mechanisms try to place files based on the locality principle (less proximity to the corresponding processing node), data characteristics, and interdependency [29]. Identifying the related files and placing them in the same DataNode or adjacent DataNodes can reduce network overhead and the query span during job completion [29], [36]. Some existing data placement strategies focus on enhancing the HDFS metrics respecting the SLA and the client's concerns [19]–[23]. Under this category, there are some data placement strategies to locate files in the DataNode cluster so that the response time of HDFS clients has been improved [19], [20], [37]. For example, there is a technique in the literature that tries to revise the basic pipelined HDFS write scheme to a parallel scheme via placing replicas regarding the proximity to the corresponding client [37]. Furthermore, some data placement techniques concentrate on preserving data privacy [23]. Using these techniques, the risk of unauthorized access to a file in a DataNode is estimated through data leakage models. Based on the mentioned model and user's concerns, data distribution among DataNodes can be managed through an estimation method designed to evaluate the security situation of data distribution in the cloud storage systems [23].

Generally, the main goal of most previous data placement techniques is to make the heterogeneity of DataNodes transparent to the clients. This goal is achievable via picking the best choices for storing the clients' files based on an objective (i.e., performance and data privacy), user concerns (i.e., policies and standards of SLA), and specifications of DataNodes (i.e., processing power and proximity). For instance, to locate the computing-intensive files in a cluster, nodes with high processing power can be the best choice. In contrast, the network-closest node can be the best host for storing the heavily accessed files.

Despite all efforts, the high diversity of client's needs and increasing heterogeneity of DataNodes faces conventional data placement techniques to two serious problems: 1) The single-objective mechanisms for scoring and ranking the available data storage nodes cannot simultaneously resolve more than one type of clients' concerns. For example, a performance-oriented data placement mechanism cannot meet the client's demands with privacy concerns. 2) Selecting the best storage in the case of receiving a client's placement request is supposed to be an appropriate storage allocation solution. However, this greedy assignment of the best resources can be unfairly proper in favor of the files uploaded earlier than the other ones.

III. THE PROPOSED SLA-AWARE MULTI-CRITERIA FILE PLACEMENT IN HDFS

To meet the expectations of HDFS's users, cloud service providers must make numerous resource management decisions that satisfy different objectives to meet the SLAs [6]. DataNode allocation (or File Placement) is the process of reserving a portion of DataNode memory for storing a specified file. In this paper, this decision is concerned with mathematical optimization problems involving more than one criterion to be considered simultaneously. The decision needs to be taken in the presence of trade-offs between three metrics: data access latency, data privacy, and computing performance. Reliability has not been considered in the trade-off list because this parameter is so important that it cannot be traded-off against the other criteria. Reliability has always been a significant concern in cloud storage systems [25]. Providing reliable services is essential for maintaining customer confidence and satisfaction and preventing revenue losses. Storage reliability is intrinsic to cloud storage architecture. In this case, the client is not usually willing to take any risk. This parameter has been taken into account in the design of cloud storage regardless of the files' contents, and most of the current cloud storage providers have some specific policies to satisfy this concern [25].

Fig. 1 shows three identities that play vital roles in the mentioned decision-making. The client machine sends their files' specifications along with their corresponding SLA fields to the NameNode. Three weighting factors (α, β, and γ) have been intended to indicate the importance of data access latency, data privacy, and computing performance from the client's viewpoint. On the client-side of the proposed structure, the cloud storage users adjust the values of three weighting factors as the SLA fields considering the type and specification of their files without having further information about the parameter settings of storage nodes. For example,

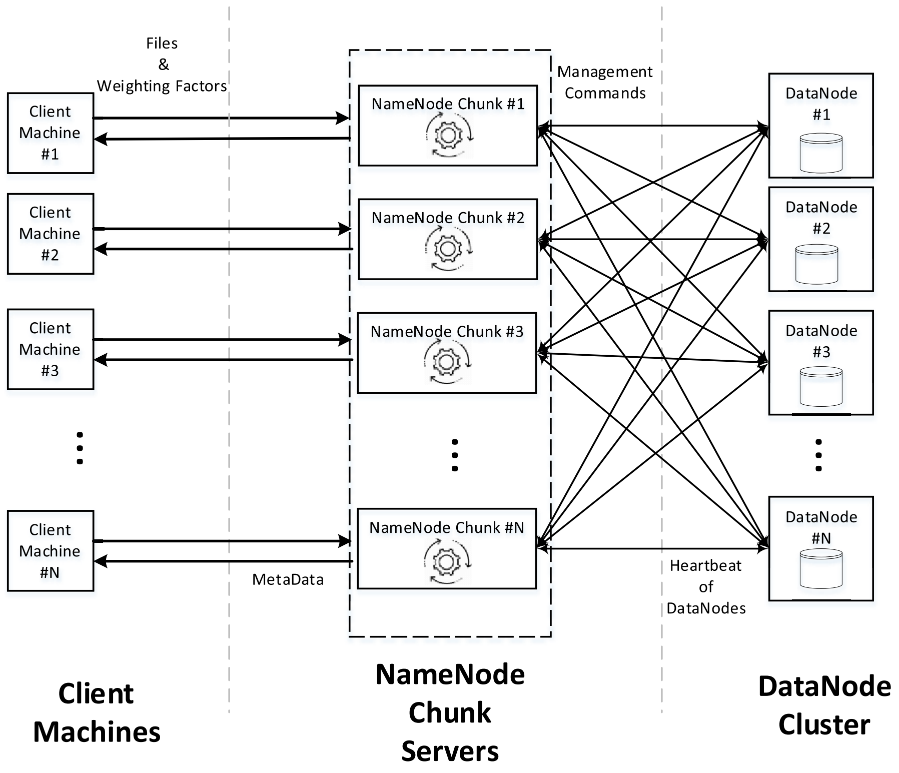

FIGURE 1. **Different roles of identities and flows of data and meta-data in the context of the proposed data placement mechanism.**

in the case of uploading a file that will be frequently accessed by a client machine, the file access latency could be the main concern of its owner. Thus, the client notices the NameNode about this issue by setting a greater value for α during the file uploading process. For another example, suppose a file contains confidential personal information of a user. In that case, the client is primarily concerned with maintaining the file's confidentiality and sets a greater value for β. Moreover, if a file contains a number of tasks' operands exploited during the execution of some performance-intensive jobs, then the value of γ should be supposed greater than the others for that file. Consequently, a client can tune the values of three weighting factors with respect to the file content and application. NameNode that can be federated [30], centralized [15], or elastic [31] is responsible for making the file placement decision. Chunk servers store the file metadata in their namespace directories and analyze the received data to locate and relocate the files in the HDFS cluster. It periodically receives a heartbeat (consisting of memory usage, stored file information, network status, etc.) and a block report from each of the DataNodes. Based on the given information, NameNode makes the decision, and place/replace the files in the cluster via mapper and balancer procedures.

In the structure of the proposed technique, the selection of DataNodes from the available list during mapping and the selection of DataNodes from the source list during balancing are carried out concerning the heartbeat of DataNodes and the client concerns. During mapping, the number of DataNodes are selected from the list of available DataNodes that is functioning correctly. Moreover, when the Hadoop cluster runs for a while, the data load balancing will be broken since nodes are added and deleted dynamically, and data load balancing is required by the newly added nodes [32]. The basic load balancing scheme of HDFS automatically moves data from one DataNode to another if the free space on a DataNode falls below a certain threshold. According to the average rate of all the DataNodes' storage space, the nodes are divided into four categories: over-average, above-average, below-average, and under-average utilized DataNodes [32]. Next, data is moved from the over-average and above-average utilized DataNodes to the below-average and under-average utilized DataNodes to balance the free space of all DataNodes [32].

Fig. 2 illustrates the design factors estimations via an example containing two clients, four DataNodes, and a sample file placement scenario. Each client owns three files of the same size that are already stored on the DataNodes. The network distances between DataNodes, and between clients and DataNodes have been mentioned in terms of a sample time unit (millisecond or ms). *Client 1* decides to store another file (*File 4*) on the HDFS. The design factors estimations and decision-making process for locating the new file in the DataNode cluster are explained in the following. Moreover, Table 1 introduces the parameters that have been exploited to estimate the mentioned design factors.

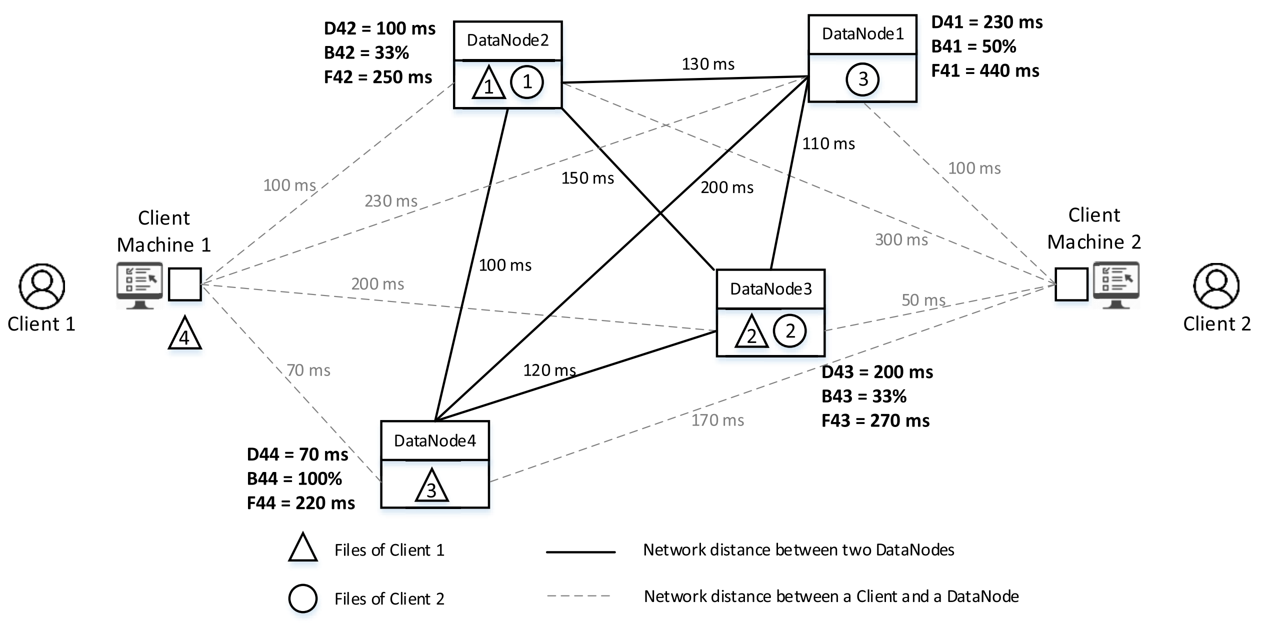

FIGURE 2. A sample topology including two clients and four DataNodes with different specifications and the raw values of D_{ij}, B_{ij}, and F_{ij}.

TABLE 1. Descriptions of notations used in the equations.

Notation	Description
n	Number of files stored on all DataNodes, containing original and replica files.
m	Number of DataNodes in the Hadoop file system.
S_j	Number of files stored on the DataNode j.
X_{ij}	$X_{ij} = 1$ iff file i is stored in DataNode j, otherwise $X_{ij} = 0$.
C_{ij}	$C_{ij} = 0$ iff geolocation of file/data node i is at odds with geolocation of file/datanode j, otherwise $C_{ij} = 1$.
R_{ij}	$R_{ij} = 1$ iff file i and file j are correlated, otherwise $R_{ij} = 0$.
B_{ij}	Breach probability of file i stored on the DataNode j.
P_{ij}	Confidential preservation probability of file i stored on the DataNode j.
D_{ij}	Access latency of file i stored on the DataNode j.
F_{ij}	Correlation factor of file i stored on the DataNode j.
$NL(x,y)$	Network latency between two geolocations (x and y).
α	Weighting factor of File Access Latency.
β	Weighting factor of File Breach Probability.
γ	Weighting factor of File Correlation Factor.

File Access Latency (D_{ij}) is the time interval between issuing a file access (file i) request by a client and accomplishing the request by a DataNode (*DataNode j*) through returning the file or committing updates. This factor can be computed by:

$$D_{ij} = NL\ (Owner\ of\ file\ i.\ DataNode\ j)\ . \qquad (1)$$

In this estimation, *NL (Network Latency(* is the term used to indicate any kind of delay that happens in data communication over the simulated network. This term can be dynamically measured with the help of distance-bounding protocols and the approximate geolocation of different ranges of IP addresses [14]. *T*he impact of the internal components of the cloud storage system on file access delay has been ignored. In the Hadoop architecture, HDFS can play an essential role in enhancing or degrading the File Access Latency. Storing files (especially heavily accessed ones) in the network-closest locations to the owners can be an effective solution for reducing File Access Latency. Regarding Fig. 2, all four DataNodes are supposed to have enough memory space for storing *File 4*. Concerning all options, storing *File 4* on *DataNode 4* leads to the minimum value of File Access Latency ($D_{44} = 70$ ms).

File Breach Probability (*Bij*) is defined as the probability of the intentional or unintentional release of private files to an untrusted environment. *Bij* is the probability of unauthorized transmission of file *i* from DataNode *j* to an external destination or recipient. Unauthorized transmission can occur for various purposes, such as analyzing users' data for advertising and financial gains, stealing private information, or even hide malicious data. In the HDFS system, the NameNode is supposed to be fully trusted because it is the core node in the network, usually well-protected. However, given that a cluster typically includes a large number of self-managed DataNodes, leading to a higher chance that some of them may be compromised intentionally or accidentally. A practical solution is to prevent the files from being compromised via breach-aware file placement in the DataNode cluster. Therefore, one of the main concentration of this work is to estimate the probability of compromising a DataNode for leaking some specific files within the HDFS system. The compromised DataNodes could transfer or copy users' data to any other nodes that may reside outside the legal regions specified by the users. Consequently, two scenarios about the causes of a data breach can be concluded:

1) A DataNode intentionally exposes data to be leaked: if the geolocation of the DataNode *j*, as the host of file *i*, is not trustable for HDFS client (owner of file *i*), ensuring the integrity of out-sourced data becomes a complicated issue. In this case, the mentioned DataNode cannot be a trustable choice for storing the client's sensitive files (based on the SLA provisions). Consequently, the breach probability of the file *i*, stored on DataNode *j*, is supposed 100%.
2) A DataNode accidentally exposes data to be leaked: even if the geolocations of the client and the DataNode have been not listed as the locations with conflict of interests, the stored data could also be leaked due to accidental reasons or weak file protections. The DataNode that is mostly accessed by the clients located in untrusted countries has a higher probability of data breach compared to the DataNode accessed by the clients resided in the trusted countries [23]. In other words, the probability of compromising a DataNode that is mostly occupied by untrusted users is higher than a DataNode containing files of trusted users. In addition to the geolocation of a DataNode, some characteristics of users who have shared the DataNode space can also be an important factor in the strength of a DataNode against attacks.

Regarding the above scenarios, B_{ij} can be estimated by:

$$B_{ij} = 1 - P_{ij}. \tag{2}$$

where P_{ij} is the probability of preserving the confidentiality of file *i* while being stored on DataNode *j*, and can be calculated by:

$$P_{ij} = C_{ij} \cdot \left(\frac{\sum_{k=1}^{S_j} C_{ik}}{S_j} \right). \tag{3}$$

Regarding the two mentioned scenarios in which, if the country of the owner of the file *i* conflicts with the country of DataNode *j*, NameNode cannot ensure the confidentiality of the file. In this case, C_{ij} is equal to 0, P_{ij} will become 0, and B_{ij} will in turn become 1. Otherwise, the probability of preserving the confidentiality of file *i* depends on the proportion of DataNode's space, allocated to the trusted users, to the overall memory space of the DataNode. Suppose that *Client 1* and *Client 2* in Fig. 2 reside in two countries with a conflict of interest. In the case of storing *File 4* on *DataNode 1*, the breach probability (B_{41}) will become 50% because the storage space of *DataNode 1* has been just equally shared between *Client 1* and *Client 2*. The values of B_{42} and B_{43} will be about 33%, while two-thirds of the *DataNodes* 2 and 3 have been filled with the files of trusted users. Finally, assume that the geolocation of *DataNode 4* conflicts with the country of *Client 1*. In this case, the breach probability of *File 4* on *DataNode 4* (B_{44}) will be the maximum value (100%).

Files Correlation Factor for file *i* (F_{ij}) is the sum of network delays between the host of file *i* (DataNode *j*) and the host(s) of its correlated file(s). The correlated files are necessary for accomplishing a specified job in a DataNode cluster. One job may require many data for execution, and many jobs may also access one file. If some files are always used together by many jobs, they are correlated to each other [29]. In the default data placement policy of Hadoop, some data characteristics like interdependency have not been considered. If correlated files have been stored on different far DataNodes, the performance of their corresponded application could be degraded [29]. Colocating the related data blocks of a file or some files on the same set of DataNode(s) or adjacent DataNodes can reduce the network overhead and is beneficial for MapReduce performance in terms of computation time. Based on the above definitions and discussion, F_{ij} can be obtained from:

$$F_{ij} = \sum_{k=1}^{n} R_{ik} \cdot NL\,(x_1 . x_2) \cdot \tag{4}$$

where x1 and x2 refer to the geolocations of DataNode j, and the DataNode that contains file k. Each element of the above arithmetic series can take two possible values. If two files (files *i* and *k*) are not correlated, the element's value will be equal to 0. Otherwise, the element will be equal to the network latency between the hosts of files *i* and *k*. With regards to Fig. 2, suppose that all files of *Client 1* are correlated to each other. Thus, the minimum files correlation factor for the set of *Client 1*'s files will be achieved after adding *File 4* to *DataNode 4*. The F_{44} is obtained from the sum of *NL(DataNode4, DataNode4)*, *NL(DataNode4, DataNode3)*, and *NL(DataNode4, DataNode2)* which is equal to *220 ms*.

Finally, a multi-criteria cost function is intended to score DataNodes concerning the three mentioned parameters simultaneously. In this function, users' concerns about the requirements of file storage and the conditions of its preservation are also considered with the help of three weighting factors: α, β, and γ. These three factors indicate the

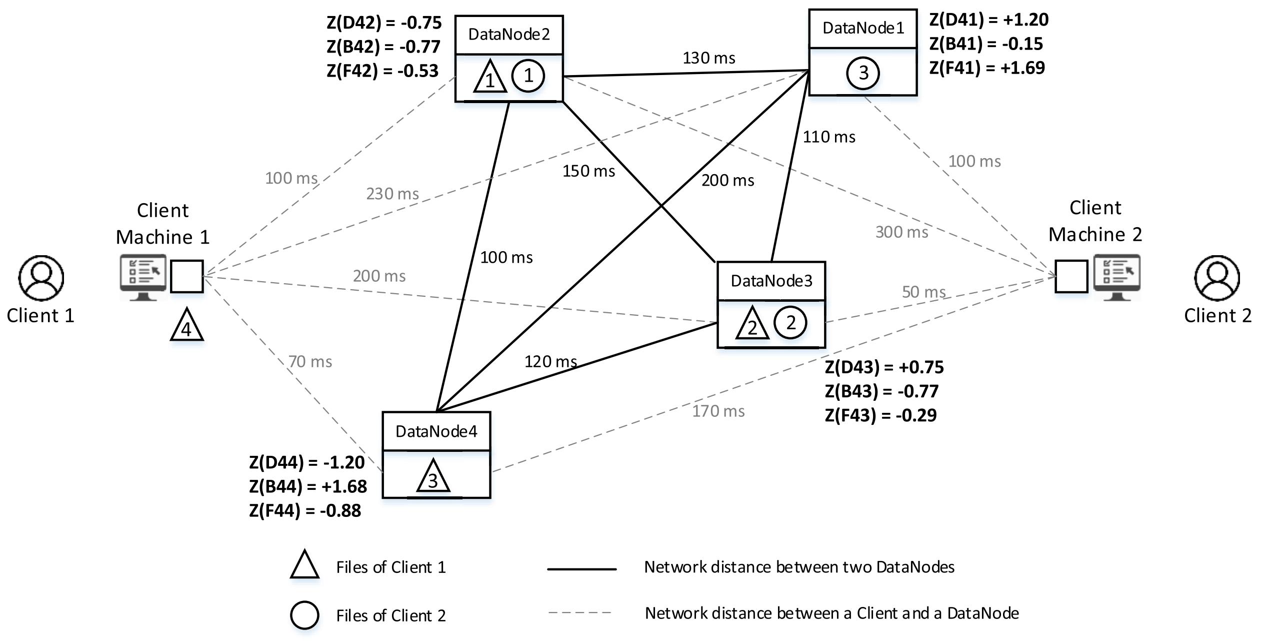

FIGURE 3. A sample topology including two clients and four DataNodes with different specifications and the Z-Score of D_{ij}, B_{ij}, and F_{ij}.

importance of D_{ij}, B_{ij}, and F_{ij} from the users' point of view, respectively. Regarding the above information, the cost of selecting DataNode j for storing file i can be computed by:

$$Cost\ (i.j) = \alpha \times Z_{Score}\left(D_{ij}\right) + \beta \times Z_{Score}\left(B_{ij}\right) + \gamma \times Z_{Score}\left(F_{ij}\right). \quad (5)$$

While D_{ij}, B_{ij}, and F_{ij} are not the same type parameters, using their raw values in Eq. (5) may lead to unfair and inaccurate estimation. Therefore, the Z-Scores (Standard Score) of the mentioned parameters have been exploited instead of their raw values. The Z-Score indicates how many standard deviations an element is from the mean [33]. If a Z-Score is equal to 0, it is on the mean. A positive Z-Score indicates the raw value is higher than the mean average, and a negative Z-Score reveals the raw value is below the mean average [33]. the Z-Score can be calculated from:

$$Z_{Score}\left(x_i\right) = \frac{x_i - \bar{x}}{\sqrt{\frac{\sum_{r=1}^{m}\left(x_r - \bar{x}\right)^2}{m}}}. \quad (6)$$

where x, $\bar{x}$, and m are the raw value of a parameter, the mean of raw values, and the number of all DataNodes, respectively. Fig. 3 shows the Z-Score of different parameters for each DataNode based on the raw values indicated in Fig. 2.

The next step is to select some DataNodes from the ranked list for storing the original and replica versions of a file. The policy, used by the conventional data placement techniques, is based on selecting the top-ranked data storage nodes for each client's request. This greedy assignment of top-scored DataNodes can be unfairly proper in favor of the files, which have been uploaded earlier than the other ones. As the name suggests, the greedy strategy always makes the choice that seems to be the best at that moment. Thus, it makes a locally optimal choice hoping that this choice will lead to a globally optimal solution. While the values of weighting factors related to the upcoming files are not predictable, choosing the best possible options has the potential of starvation during placement of the next resource-intensive files. The greedy strategy never goes back to reconsider the earlier choices. Suppose a situation where the storage space of the most well-protected DataNodes has been assigned to the non-sensitive files. If one of the clients intends to send a privacy-sensitive file to the HDFS, NameNode has no choice but to select the out of favor DataNodes concerning the data breach incidents. Therefore, the file placement issue in cloud storage systems cannot be categorized in the class of problems that can be effectively solved through greedy strategies.

In the SLA-aware file placement technique, NameNode assigns a probability to each DataNode in the event of receiving a file placement/replacement request. The probability indicates the chance of choosing a DataNode as the host of the mentioned file. The chance of adding a file to a DataNode is estimated with regards to their corresponding cost function values. Contrary to the greedy strategy, all available DataNodes can be selected as the host of a file, based on their estimated probabilities. Therefore, DataNode with the lowest cost has a higher chance of being picked up as the host.

At this point, the service provider can influence the chance estimation routine. Based on the revenue model of a cloud service provider, the users can be classified into different categories and levels, such as premium, freemium, and free users. The cloud service provider usually offers all users essential services but holds a select set of key premium features only for its paid subscribers. The random strategy for

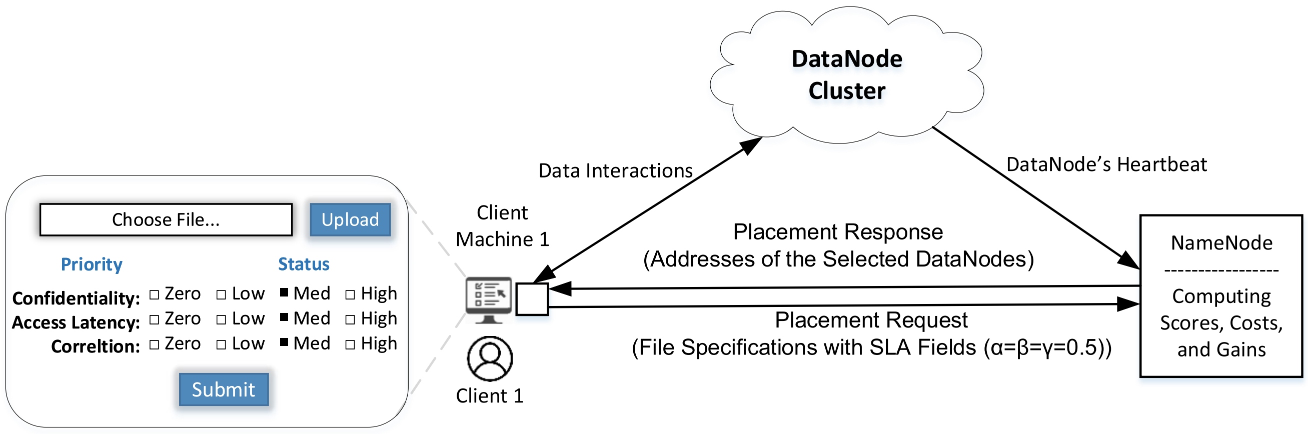

FIGURE 4. The interactions among client, client machine, NameNode and DataNodes in the structure of the proposed technique.

data placement can achieve load balance with low cost and efforts and is totally beneficial for the service provider. However, in real-world applications, clusters are often worked in a heterogeneous environment. Thus, there is a considerable difference in the consequences of storing data on different DataNodes. In this case, using the random strategy leads to a reduction of clients' satisfaction.

On the other hand, objective-based mechanisms can take advantage of this variety to satisfy some clients' concerns about different parameters. In the first step of the proposed technique, the DataNodes are ranked based on their specification and SLA fields. Some DataNodes should be selected from the ranked list to store a file and its replicas in the next step. The constant c determines how much the specifications of the nominated DataNodes can be fitted with the concerns of a client about a specified file. Thus, the cloud service provider set the proper value of constant c respecting the user support level and contractual agreements. In order to increase the quality of services delivered to the paid subscribers, smaller values for constant c leads to the selection of the optimal or near-optimal options for storing clients' files. However, larger values for constant *c* made the placement process more like a random assignment strategy rather than a purposive assignment. Therefore, the benefits of the cloud service provider take precedence over the concerns of the users. The higher user levels are received a more rational and precise adjustment of file placement efforts to user requirements. An intermediate parameter is defined as the gain of adding file *i* to DataNode *j* to estimate the chance of selecting a DataNode as the host of a specified file. This parameter is directly proportional to the chance parameter:

$$Gain\,(i.j) = -Cost\,(i.j) + c. \tag{7}$$

The value of constant *c* in Eq. 7 determines the precision of adjustment of file placement efforts to user requirements. Assigning smaller values to variable *c* leads to the increased chance of selecting top-ranked DataNodes in the list of available DataNodes. On the other hand, setting variable *c* to larger values results in approximately equal probabilities of selection for all DataNodes. As mentioned above, the proposed technique selects DataNodes for storing different files based on a probability mass function. A probability mass function is specified in terms of an underlying sample space, which is the set of all possible observed random phenomenon outcomes. It is a mathematical function that provides the probabilities of different possible options (selecting each available DataNodes as the host of a specified file) based on their gain values. Thus, the chance of choosing DataNode *j* for storing file *i* can be calculated as:

$$Chance\,(i.j) = \frac{Gain(i.j)}{\sum_{k=1}^{m} Gain(i.k)}. \tag{8}$$

Fig. 4 shows the interactions among Client, Client Machine, and NameNode to set the proper values for the weighting factors. We refer to the client-side of the cloud storage application as the "Client Machine". Some custom client settings have been deployed to the user interface of the application to get and address the clients' concerns. The clients can prioritize their files via checking the radio buttons in terms of different parameters. For example, four priority levels are considered for each parameter: Zero, Low, Med, and High. The client machine can assign the static or dynamic values to weighting factors of a file, respecting the selected priority levels for the corresponding file or all outsourced files of the client. Suppose the importance of each file is regarded independently. In that case, four priority levels can represent the static values of 0.00, 0.25, 0.50, and 1.00 for weighting factors. Afterward, the client machine submits the form containing the numerical values of weighting factors to the NameNode. Thus, the NameNode can estimate the gain of using different DataNodes as the host of the file. Although the client and the client machine play a major role in determining the numerical values of weighting factors, the NameNode (or CSP) is the identity that makes the final decision about how much the submitted values can affect the placement solution. NameNode utilizes the constant *c* as a tool for tuning

TABLE 2. Cost and chance of adding File 4 to each DataNode in terms of different values for weigthing factors.

Scenario	Priority			DataNode 1				DataNode 2				DataNode 3				DataNode 4			
					Chance (%)				Chance (%)				Chance (%)				Chance (%)		
	Access Latency	Confidentiality	Correlation	Cost	Service Level 1	Service Level 2	Service Level 3	Cost	Service Level 1	Service Level 2	Service Level 3	Cost	Service Level 1	Service Level 2	Service Level 3	Cost	Service Level 1	Service Level 2	Service Level 3
1	Med	Med	Med	1.4	7.8	16.4	20.7	-1	37.7	31.4	28.2	-0.1	26.9	26.0	25.5	-0.2	27.5	26.3	25.6
2	High	Low	Low	1.6	5.17	15.1	20.0	-1.1	38.4	31.7	28.4	0.49	18.9	22.0	23.5	-1	37.5	31.2	28.1
3	Low	High	Low	0.6	17.8	21.4	23.2	-1.1	38.6	31.8	28.4	-0.7	33.1	29.1	27	1.2	10.5	17.8	21.4
4	Low	Low	High	2	0.5	12.8	18.9	-0.9	36.3	30.7	27.8	-0.3	28.7	26.9	25.9	-0.8	34.5	29.7	27.4
5	High	High	Low	1.5	6.5	15.8	20.4	-1.6	45.6	35.3	30.1	-0.1	26.1	25.5	25.3	0.3	21.8	23.4	24.2
6	High	Low	High	2.9	0.0	7.1	16.1	-1.5	43.3	34.2	29.6	0.3	21.7	23.3	24.2	-1.7	45.7	35.3	30.2
7	Low	High	High	1.8	1.9	13.5	19.2	-1.5	43.5	34.2	29.6	-0.9	35.9	30.4	27.7	0.5	18.7	21.9	23.4
Average				**1.7**	**4.1**	**14.6**	**19.8**	**-1.2**	**40.5**	**32.8**	**28.9**	**-0.2**	**27.3**	**26.2**	**25.6**	**-0.2**	**28.0**	**26.5**	**25.8**

the impact of weighting factors on the placement decision. It determines the value of constant c for different user service levels based on the types of user accounts and the heartbeats of DataNodes. In order to increase the quality of services delivered to the paid subscribers, smaller values for constant c lead to the selection of the optimal or near-optimal options for storing clients' files. The higher user levels are received a more rational and precise adjustment of file placement efforts to user requirements. However, larger values for constant c made the placement process more like a random assignment strategy rather than a purposive assignment. Therefore, the benefits of the cloud service provider take precedence over the concerns of the users. The random placement strategy can achieve load balance and decrease the number of costly data block movements among DataNodes in favor of the CSP. Determining the exact value of constant c for each service level by the NameNode involves taking into account the policies and the business strategy of the CSP for serving different users under different characteristics of DataNodes. Following the example discussed in the manuscript, Table 2 is the color scale table that shows the cost and chance of adding File 4 to each DataNode. Seven scenarios have been taken into account to assess the effects of selecting different priority levels on the file placement decisions in terms of three user service levels. Among these three levels, the users classified in "Service Level 1" experience the highest quality of services, and the users of "Service Level 3" receive the cheapest ones. In this example, the value of constant c in the Gain function is statically set to 2, 4, and 8 for Service Levels 1, 2, and 3. The value of a cell in the table, related to the cost of DataNodes, changes its appearance based on its corresponding value. In this color scale, shades of red/blue are applied to the higher/lower cost. In a general view, selecting the DataNode 2/DataNode 1 as the host of File 4 implies the lowest/highest cost compared to the other DataNodes. As shown by the table, the probability of selecting DataNode 2 as the best possible option is about 40.5% in service level 1, and the chance of selecting DataNode 1 as the worst choice of placement is about 4.1% in this level. In service levels 2 and 3, the chances of choosing the best and worst options of placement are closer than service level 1. Consequently, the impacts of users' concerns on the probabilities of choosing a specified DataNode as the host of File 4 depends on the user service levels and the value of constant c.

Finally, Fig. 5 shows the steps and sequence of functions and calculations used in the context of the proposed data placement mechanism.

IV. EVALUATION RESULTS

The CloudSim [24] framework is considered as the base simulation environment to implement design decisions. The CloudSim is an open-source framework for modeling and simulation of cloud-based infrastructures and services [24]. The cloud-based simulation tools, such as CloudSim, is a suitable alternative solution for the real evaluation environments, where access to the infrastructure incurs payments in real currency [34], [35]. Simulation-based approaches offer significant benefits. It allows cloud customers to test their services in a repeatable and controllable environment, free of cost, and tune the performance bottlenecks before deploying on the

```
- For all received file with the index of i from 1 to n with specified values for α, β, and γ
    i.   Determining the value of c based on the level of user support services by the CSP
    ii.  For all available DataNodes with the index of j from 1 to m, calculating
            • $D_{ij}$, $B_{ij}$, and $F_{ij}$
    iii. For all available DataNodes with the index of j from 1 to m, calculating
            • $Z_{score}(D_{ij})$, $Z_{score}(B_{ij})$, and $Z_{score}(F_{ij})$
            • Cost(i,j)
            • Gain(i,j)
            • Chance(i,j)
- Selecting three DataNodes for storing the file i from the ranked list based on the Chance(i,j)
```

FIGURE 5. The algorithm and steps of proposed data placement mechanism.

real clouds [24]. The CloudSim is a general and extensible simulation framework that enables seamless modeling, simulation, and experimentation of emerging cloud-based infrastructures and application services [34], [25]. This simulator is currently the most sophisticated discrete event simulator for Clouds [24]. Due to the mentioned features and benefits, the CloudSim has been chosen for building the simulation environment on top of it.

Fig. 6 shows the architecture of the enhanced CloudSim used as the evaluation environment. Regarding Fig. 6, the CloudSim consists of three abstraction layers:

1) User code: This layer enables CloudSim users to define general simulation configurations such as cloud scenarios, requirements/policies, data sets, and brokers. A new broker (HDFS Broker) has been designed to act as the client machines of HDFS's users. It also works as an interface for CloudSim users to specify the number of clients, files, and corresponding specifications, such as network delay among different geolocations, privacy conflict between nations/countries, and different aspects of HDFS services as the parameters of SLA.
2) CloudSim: The second layer consists of several sub-layers that model the key elements of simulated cloud-based structures. The User Interface Structure is the top sub-layer, responsible for adjusting and adapting system modules and resources considering the user's demands. The details of network topology, message delays, Virtual Machine (VM) provisioning, Processing Element (PE) or Physical Machine (PM) instantiations, storage architecture, memory allocation, file specifications are some of the system requirements that should be arranged concerning user's demands. In the enhanced version of CloudSim, a new class (Client) has been designed and developed to reflect the user's demands in terms of different expected services. Moreover, some important attributes of client machines that can affect the performance of internal mechanisms of cloud storage systems have been considered in this class. Furthermore, the file and file attributes classes have been revised concerning the new attributes, defined in the client class, and some further file specifications that have effects upon user influence-based file/data distribution. Furthermore, a new attribute has been defined and considered in the file attributes class of the CloudSim to reflect the frequency of file access. The middle sub-layer plays the role of resource manager in allocating cloudlets (jobs) among VMs and distributing files among storage nodes. JobTracker and NameNode are two Hadoop entities that handle job allocation and file distribution based on pre-defined mechanisms and policies. While Cloudlet Execution and VM Management classes can cover the duties of Hadoop JobTracker, however, there is a lack of CloudSim software module to act as the NameNode for HDFS-based file distribution. Therefore, a new class (NameNode) has been designed and developed to implement the base Hadoop file distribution mechanism and the proposed design decisions in the context of CloudSim. The Cloudlet attribute that defines the path for the list of files

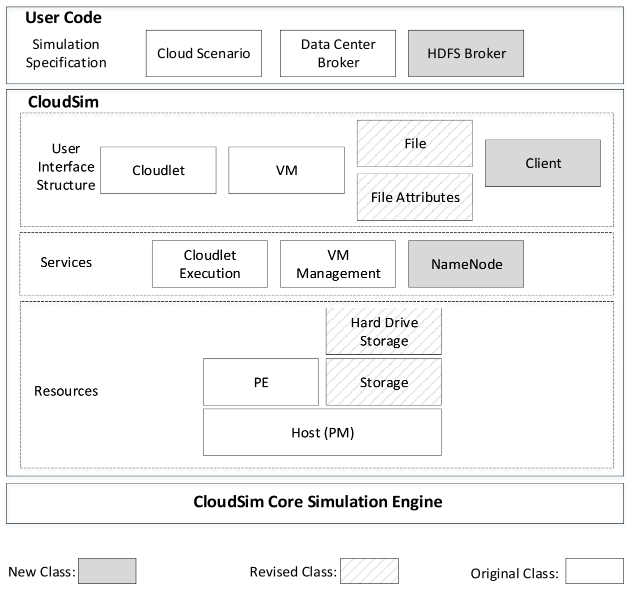

FIGURE 6. CloudSim-based simulation environment.

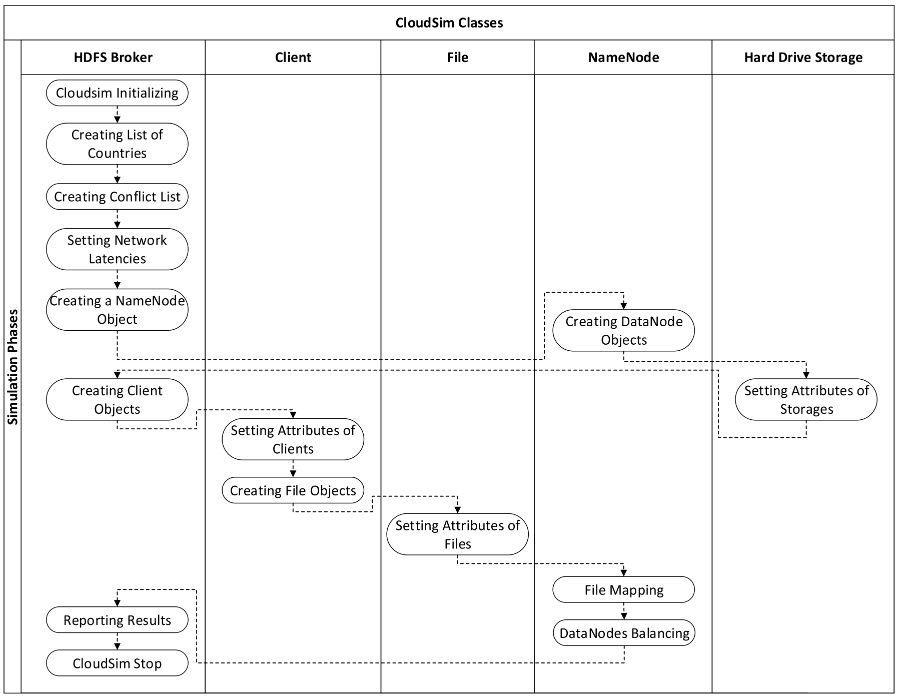

FIGURE 7. **The simulation work-flow and revised classes of the enhanced version of CloudSim.**

required by the cloudlet during execution (requiredFiles) has been revised to adjust the number of file accesses based on the new file attributes. We have tried to simulate the actions and interactions of TestDFSIO benchmark via creating the workload trace in SWF format. The TestDFSIO benchmark is a read and write test for HDFS. It is helpful for tasks such as stress testing HDFS, to discover performance bottlenecks in your network, to shake out the hardware, OS, and Hadoop setup of your cluster machines (particularly the NameNode and the DataNodes). The bottom sub-layer simulates the physical and virtual resources such as Hadoop DataNodes employing enhanced attributes and methods of Storage and Hard Drive Storage classes. The values assigned to the attributes of the Hard-drivestorage class of the CloudSim are those of a Maxtor DiamonMax 10 ATA harddisk (latency = 4.17 ms, avg seek time = 9 ms, max transfer rate = 133 MB/sec).

3) Core Simulation Engine: The bottommost layer of the CloudSim architecture handles the interaction between CloudSim entities and components such as PMs, Hosts, VMs, and brokers. It also controls the system events via queuing and scheduling. Moreover, this layer enables the CloudSim to interact with the system software on which CloudSim is running.

In Fig. 7, more details about the new and revised classes in the enhanced version of CloudSim in terms of attributes and software methods are demonstrated. Moreover, Fig. 7 clarifies the simulation workflow consists of constructors, interfaces, and parameters of the estimation functions.

Three parameters have been defined to compare the design decisions: average File Access Latency, average File Breach Probability, and average File Correlation Factor. These parameters can represent and measure CSP services' quality in terms of file read/write performance, file privacy, and computation performance. With regards to the description of notations, listed in Table 1, the average File Access Latency is the mean of D_{ij} for all files in the cluster:

$$Avg\ File\ Access\ Latency = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}\cdot D_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}}. \quad (9)$$

Moreover, the average File Breach Probability is the mean of B_{ij} for all files in the cluster:

$$Avg\ File\ Breach\ Probability = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}\cdot B_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{m} X_{ij}}. \quad (10)$$

Finally, the average File Correlation Factor is the mean of F_{ij} for each possible combination of correlated files:

$$Avg\ Files\ Correlation\ Factor = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} F_{ij}}{\sum_{i=1}^{n}\sum_{j=1}^{n} X_{ij}}. \quad (11)$$

The following DataNode scoring and data placement mechanisms have been implemented and applied on the Hadoop architecture:

1) The base HDFS data placement
2) The proposed greedy data placement (GDP)

TABLE 3. The details of seven simulated scenarios.

Scenarios	File Specifications							DataNode Specifications			Geolocation Specifications		
	Number	Size	α	β	γ	Correlation Intensity (%)	Correlation group size	Number	Size	Utilization (%)	Number	Alliance size	Delays
1	100~100000	14MB-14GB	0.33	0.33	0.33	20%	10	20	300GB	70%	4	2	(300ms,800ms)
2	2800~11400	140MB	0.33	0.33	0.33	20%	10	20	300GB	20%~80%	4	2	(300ms,800ms)
3	10000	140MB	0.33	0.33	0.33	20%	10	20	300GB	70%	4	1~4	(300ms,800ms)
4	10000	140MB	0.33	0.33	0.33	20%~80%	10	20	300GB	70%	4	2	(300ms,800ms)
5	10000	140MB	0.33	0.33	0.33	20%	10~10000	20	300GB	70%	4	2	(300ms,800ms)
6	10000	140MB	0.33	0.33	0.33	20%	10	20	300GB	70%	4	2	(300ms,800ms) ~ (200ms,2000ms)
7	10000	140MB	0~1	0~1	0~1	20%	10	20	300GB	70%	4	2	(300ms,800ms)

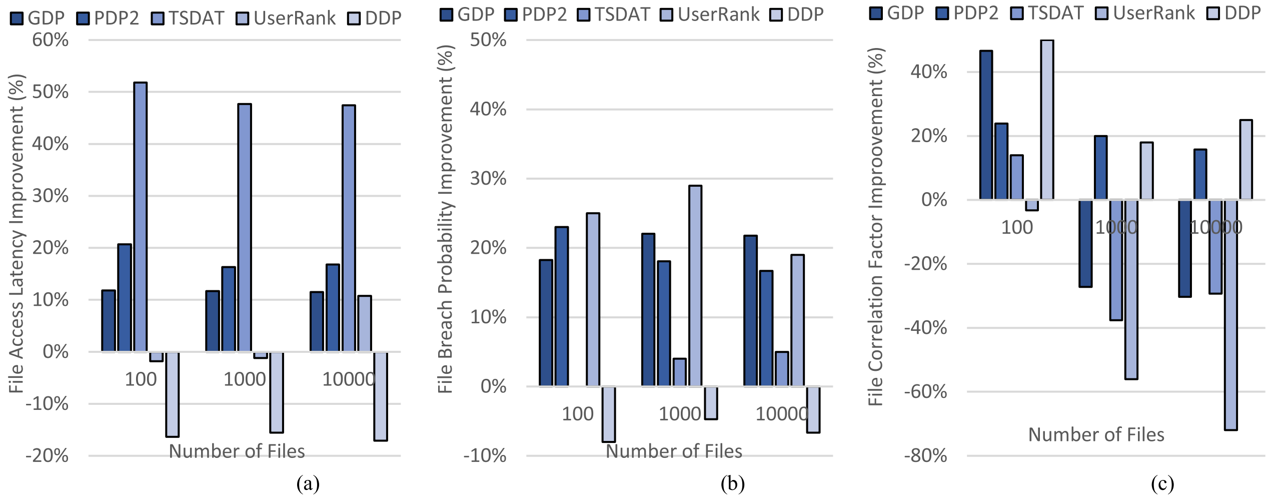

FIGURE 8. Comparison of performance of data placement mechanisms in terms of file access latency (a), file breach probability (b), and file correlation factor (c) in the case of different number of files (the results are normalized to the results of base HDFS).

3) The proposed probabilistic data placement with c = 2 (PDP2)
4) The TSDAT data placement technique [37]
5) The UserRank data placement technique [23]
6) The DDP data placement technique [36]

Furthermore, six different scenarios have been intended to study the performance of each data placement mechanism. Table 3 summarizes the details of simulated scenarios in terms of HDFS cluster and file specifications. In each situation, a certain DataNode or file specification has been concentrated as the target parameter. A target parameter is varied during different simulations classified into a scenario while the other parameters are kept unchanged. The specifications have been categorized into three classifications: file specifications, DataNode specifications, and geolocation specifications. File specifications consist of the number and size of files, the weighting factors, the file correlation intensity, and the file correlation group size. For reliability reasons, two replicas are made of each file by all of the data placement mechanism. DataNode specifications consist of the number of DataNodes in the cluster, the size of each DataNode, and the data load utilization (storage space usage) of each DataNode. Finally, geolocation specifications consist of the number of geolocations in which clients and DataNodes can reside, the size of the alliance, and the intervals of network delays among geolocations.

Fig. 8 shows the results of an experiment intended to analyze the performance of different data placement mechanisms for different file sizes: 100 files with the size of 14 GB, 1000 files with the size of 1.4 GB, and 10000 files with the size of 140 MB. The number and size of files are adjusted so that the utilization of the DataNode cluster is kept at 70% in all conditions of scenario 1. This adjustment has

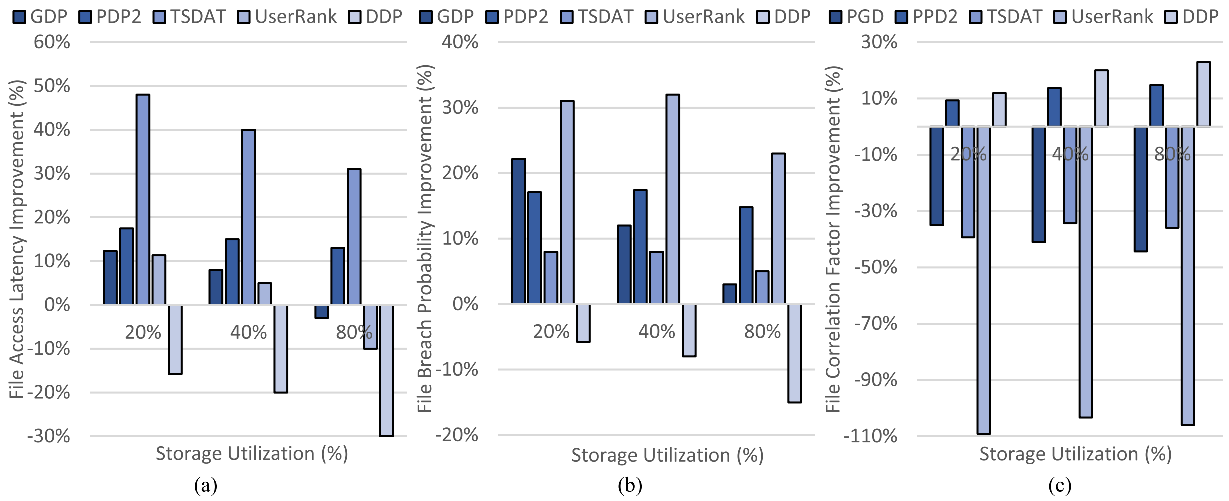

FIGURE 9. Comparison of performance of data placement mechanisms in terms of file access latency (a), file breach probability (b), and file correlation factor (c) in the case of different storage utilization values (the results are normalized to the results of base HDFS).

been considered to eliminate the effects of different DataNode utilization on the performance of the design decisions. Regarding all charts, the results of PDP2 follow practically the same pattern, and it is often in the top two ranked design decisions respecting different cases (despite TSDAT, UserRank, and DDP). This issue reveals that the chance of proceeding towards the best comprehensive solution is often higher for the probabilistic mechanism. However, the greedy strategy always makes the choice that seems to be the best at that moment. The fluctuation of the GDP's results shows that it can more easily get stuck at locally optimal solutions compared to the PDP2.

Fig. 9 shows the results of an experiment intended to analyze the performance of different data placement mechanisms in the case of different storage utilizations: 20%, 40%, and 80%. The number of files (with the fixed-size of 140MB) has been changed from 2800 to 11400 to vary the storage utilization. Increasing the complexity of mapping and balancing processes by the PDP2 raised some concerns about this technique's effectiveness in the highly utilized DataNode cluster. Contrary to the mentioned concerns, the experimental results show that the File Access Latency and File Breach Probability have not been significantly degraded when the storage utilization increases. In most cases, the File Correlation Factor is also improved by the PDP2. Increasing the DataNode utilization causes an expanded design space solutions with an increased number of relative extrema. Therefore, the probability of getting stuck in local optimum solutions by the GDP is subsequently increased. Furthermore, the distance among the results related to solutions obtained by the single objective data placement mechanisms (TSDAT, UserRank, and DDP) increases while the storage utilization grows. This issue has been observed due to the intensified divergence among the acquired placement solutions in the expanded design space.

Fig. 10 shows the results of an experiment intended to analyze the performance of different data placement mechanisms under the network containing four alliance sizes: one, two, and four geolocations. The alliance members are mutually interested and have a conflict of interests with the geolocations belonging to other partnerships. The inconsistency in the GDP results and the predictable pattern of the PDP2 results are two common aspects of the charts in Fig. 10. The inconsistency in the GDP results has been observed due to divergence and convergence of local minimum solution (achievable by the GDP) and global minimum solution. However, the PDP2 leads to consistent moderate improvements in all three design parameters. Since all three design parameters are of equal importance to all users ($\alpha = \beta = \gamma = 0.33$) in the scenario, the results completely support probabilistic data placement mechanisms' objectives. In the experiments, the nearby geolocations are in the same alliance. Therefore, the TSDAT and UserRank are aligned in different parameters due to the coincidence of geographical proximity and division of trusted regions. Regarding Fig. 10 (b), the File Breach Probability of the HDFS cluster under the management of different design decisions is 0% in the third case. When the four geolocations have no conflicts, the probability of preserving confidentiality is supposed 100% by the proposed data breach prediction model. The second issue about the results of related work is the partial conflict between TSDAT and DDP results. Using TSDAT causes the enhanced score of solutions in which the data locality is improved concerning the geolocations of files and their owners. However, using DDP leads to enhanced data locality with respect to the geolocations of correlated files. Storing the files in the network-closest DataNodes to the geolocation of its owner and storing the files in the network-closest DataNodes to its correlated set of files are two goals that are not always aligned.

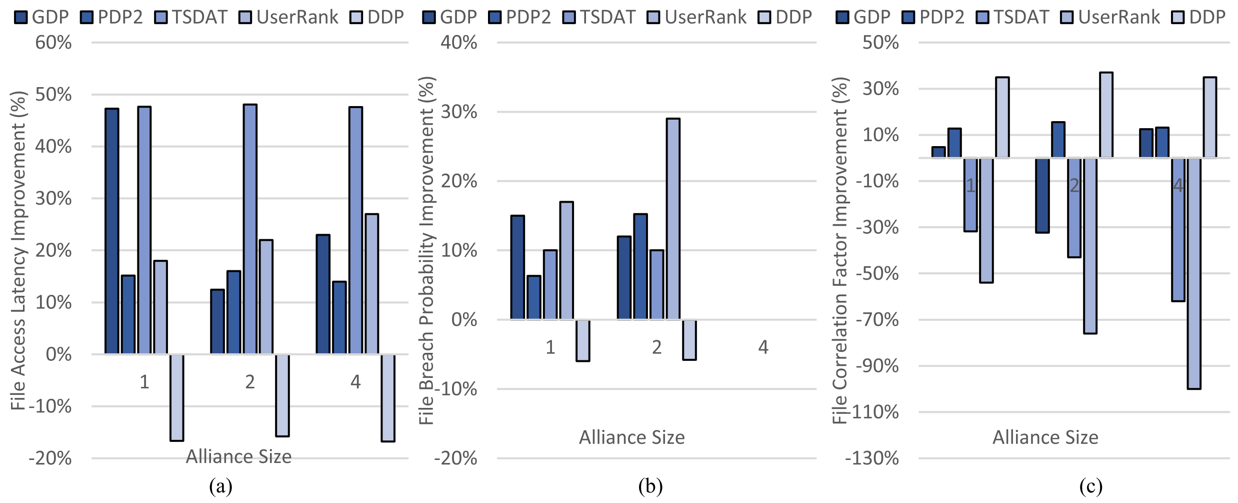

FIGURE 10. **Comparison of performance of data placement mechanisms in terms of file access latency (a), file breach probability (b), and file correlation factor (c) in the case of different alliance sizes (the results are normalized to the results of base HDFS).**

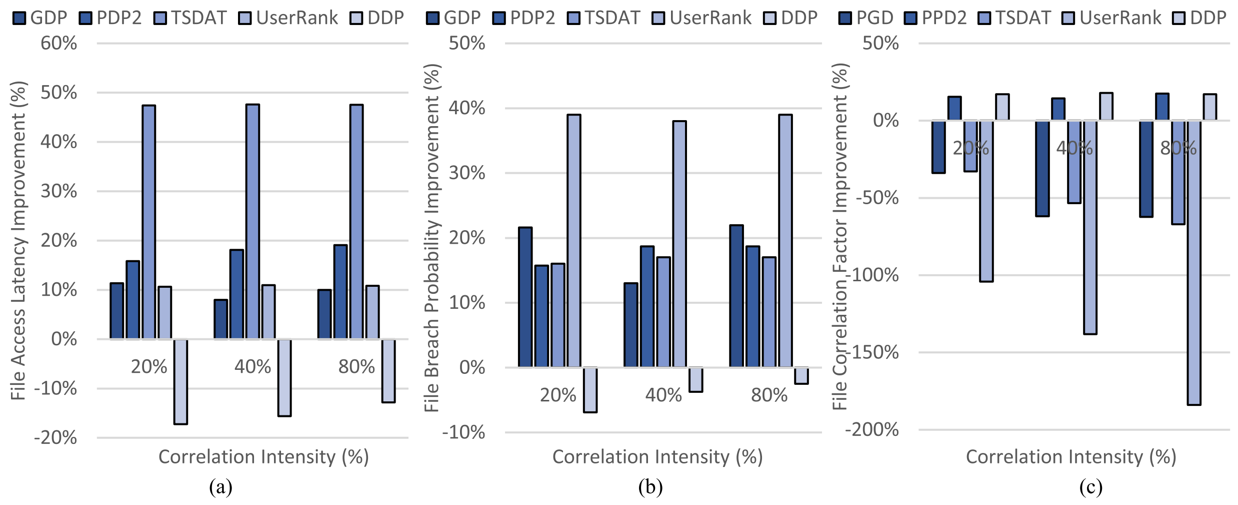

FIGURE 11. **Comparison of performance of data placement mechanisms in terms of file access latency (a), file breach probability (b), and file correlation factor (c) in the case of different values for correlation intensity (the results are normalized to the results of base HDFS).**

Fig. 11 and 12 show the results of two experiments intended to analyze different data placement mechanisms' performance under different file correlation conditions. The correlation among files can be adjusted by tuning two factors: the correlation intensity and the correlation group size. The file correlation intensity indicates the degree of correlation among the out-sourced files. For example, the correlation intensity of 20% shows that 20% of all uploaded files are mutually correlated. The files that belong to a certain correlation group are necessary for accomplishing a specified job in a cluster. The size of correlation groups in a cluster may also vary depending on the type of their related jobs. Regarding Fig. 11 and 12, the effects of increasing the size of correlation groups and correlation intensity on File Access Latency, File Breach Probability, and File Correlation Factor follow the almost same pattern. As clarified by Fig. 11, increasing the intensity of correlation among out-sourced files from 20% to 80% cannot degrade PDP2's performance of different design parameters. Increasing the correlation intensity among out-sourced files leads to increased deviation among the results of different possible solutions in terms of File Correlation Factor. In this case, even a small difference in the storage location of a file can make significant differences in the File Correlation Factor of almost similar data placement solutions. This issue causes an increased number of relative extrema in the design space of solutions' results. Regarding Fig. 11 (c), the distance among the results related to solutions obtained by the TSDAT/UserRank and the DDP increases while the correlation intensity grows.

Fig. 13 shows the results of an experiment intended to analyze different data placement mechanisms' performance under three different ranges for the delays

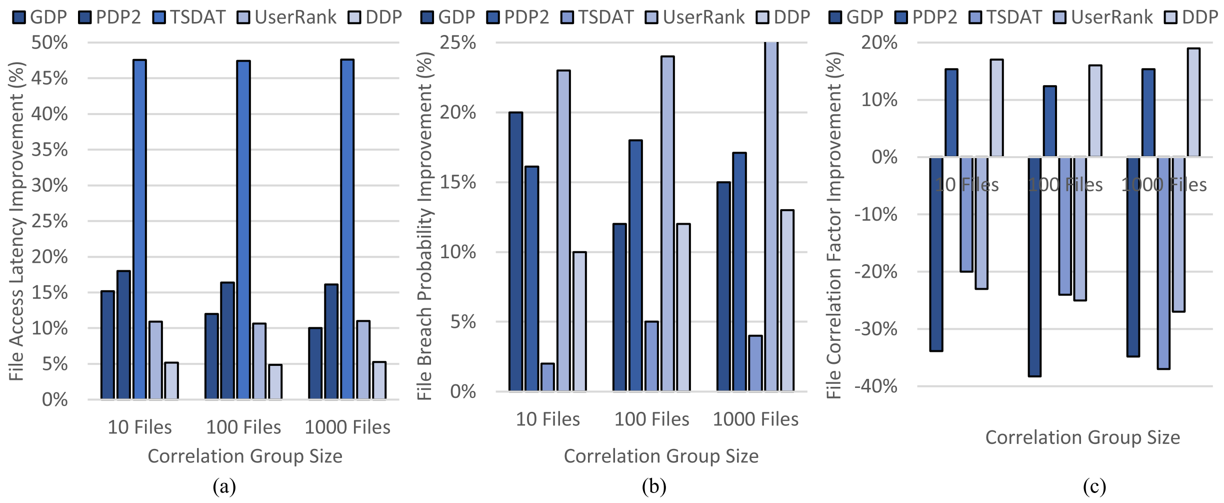

FIGURE 12. Comparison of performance of data placement mechanisms in terms of file access latency (a), file breach probability (b), and file correlation factor (c) in the case of different sizes for correlation group (the results are normalized to the results of base HDFS).

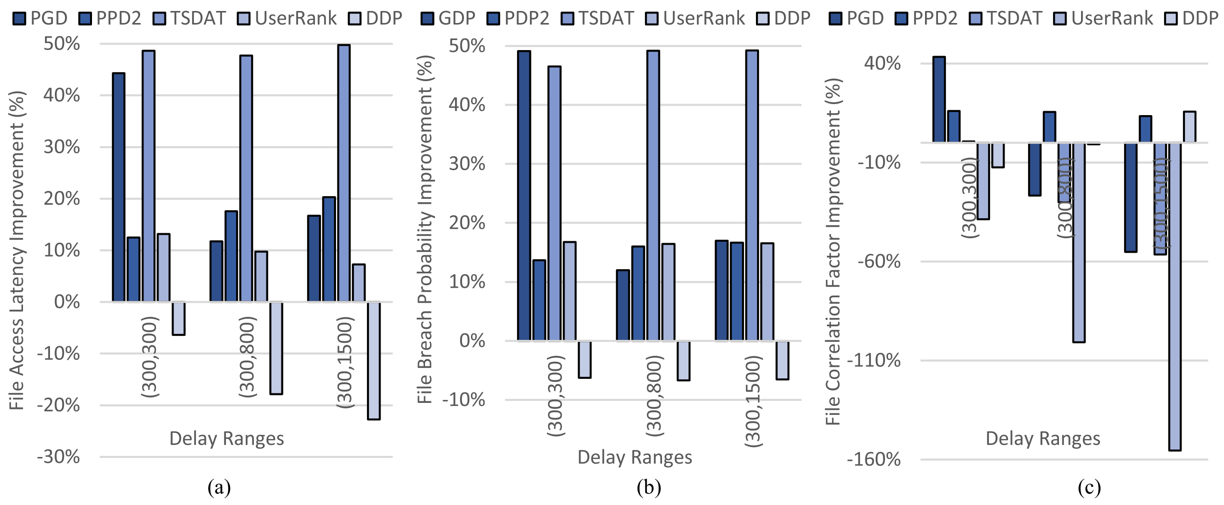

FIGURE 13. Comparison of performance of data placement mechanisms in terms of file access latency (a), file breach probability (b), and file correlation factor (c) in the case of four different ranges of delay between geolocations (the results are normalized to the results of base HDFS).

between geolocations. The network delay interval among geolocations was varied from (300ms, 800ms) to (300ms,1500ms) during the simulation. In the first case, the delay between all geolocations is equal to 300 ms. However, in the second, third, and fourth cases, the geolocations' delays are different and varied within tight and wide ranges. As shown by Fig. 13 (a) to (c), the PDP2 improves the design parameters in almost all cases. The random assignment strategy used by the base HDFS cannot hide the variety of delays between the geolocations. At the same time, different data placement solutions imply different significant values for File Access Latency. On the other hand, objective-based mechanisms can take advantage of this variety to satisfy some clients' concerns about access latency. The File Correlation Factor is the parameter that is most affected by the variation of geolocations' delay. Even so, PDP2 is the only objective-based mechanism that can maintain its enhancement in all cases. Similar to the previous scenario, the conflict between TSDAT/UserRank and DDP results is observed due to the data locality contexts exploited in their designs' structure.

Tail latency is the small percentage of access latency to a system, out of all accesses to the requests it serves, that takes the longest compared to the bulk of its access latencies. The data placement strategies can affect the file access latency by influencing network latency between client machines and DataNodes, and the response time of a DataNode. The network proximity of a client machine and the DataNodes (containing the corresponding client files) adjusted by the data placement strategies can affect the network latency

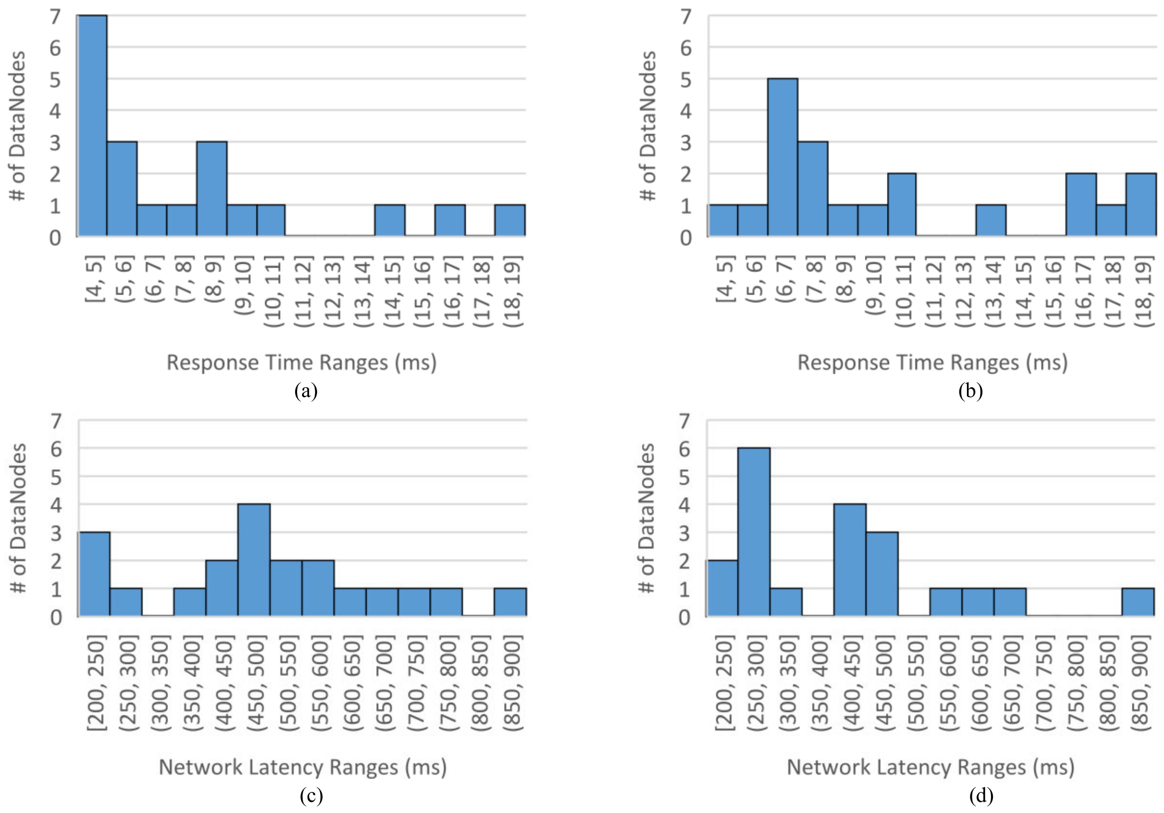

FIGURE 14. Comparison of the effects of placement strategies on the distribution of DataNode access latency in terms of response time and network latency (Hadoop: (a), and (c); PDP2: (b), and (d)).

parameter. Moreover, the complexity of data placement strategies and the I/O bandwidth usage of disk storage are two parameters that influence the response time of a DataNode.

Fig. 14 shows the distribution of access latency of 20 simulated DataNodes in terms of response time, and network latency under the management of the base Hadoop data placement strategy and the PDP2. As illustrated by Fig. 14 (a) and (b), computing complexity imposed by the PDP2 leads to a higher number of DataNodes with response times of more than 10 ms, compared to the Hadoop data placement strategy. In the case of using Hadoop, the response time of 85% of the DataNodes is less than 10 ms, while this portion is about 70% for PDP2. Hadoop's random strategy is the most straightforward data placement mechanism with less computational effort than the proposed probabilistic data placement mechanism. It achieves load balance and needs less-frequent rebalancing compared to PDP2. However, PDP2 takes advantage of the purposive ranking model to improve data locality and improved network latency compared to Hadoop. Fig. 14 (c) and (d) show that network latency for accessing 80% of DataNodes is less than 500 ms for the PDP2, while this portion is about 55% for the Hadoop.

In order to study the effects of setting different values of weighting factors for different files, another scenario has been defined (Scenario 7 in Table 3). In this scenario, the values of weighting factors for each of 10000 files were assigned randomly, and the PDP2 was selected to place the files in the cluster. Fig. 15 summarizes this scenario so that the results of files whose weighting factors in the same range were considered together. For example, the averages of File Access Latency improvement (FAC), File Breach Probability Improvement (FBP), and File Correlation Factor Improvement (FCF) for the files whose α are in the same range have been calculated and shown in Fig. 15 (a). Moreover, Fig. 15 (b) and (c) have been intended to show the same condition for β and γ, respectively. As indicated by the charts, the weighting factors can still tune the importance of design parameters in the placement ranking model, even after setting different weighting factors for each file. However, there is no linear relationship between the improvement of the FAC and FBP and increasing the values of α and β. The FAC and FBP are aligned under different conditions due to the coincidence of geographical proximity and division of trusted regions in the simulation. The top-ranked DataNodes in terms of FAC also have the best characteristics concerning FBP. Therefore, the choices for simultaneous improvement of both FAC and FBP have been limited especially for the close values of α and β. Moreover, there is a partial conflict between FAC and FCF due to different data locality concepts that could have positive influences on these parameters. Storing the files in the network-closest DataNodes to its owner's geolocation and storing the files in the network-closest DataNodes to its

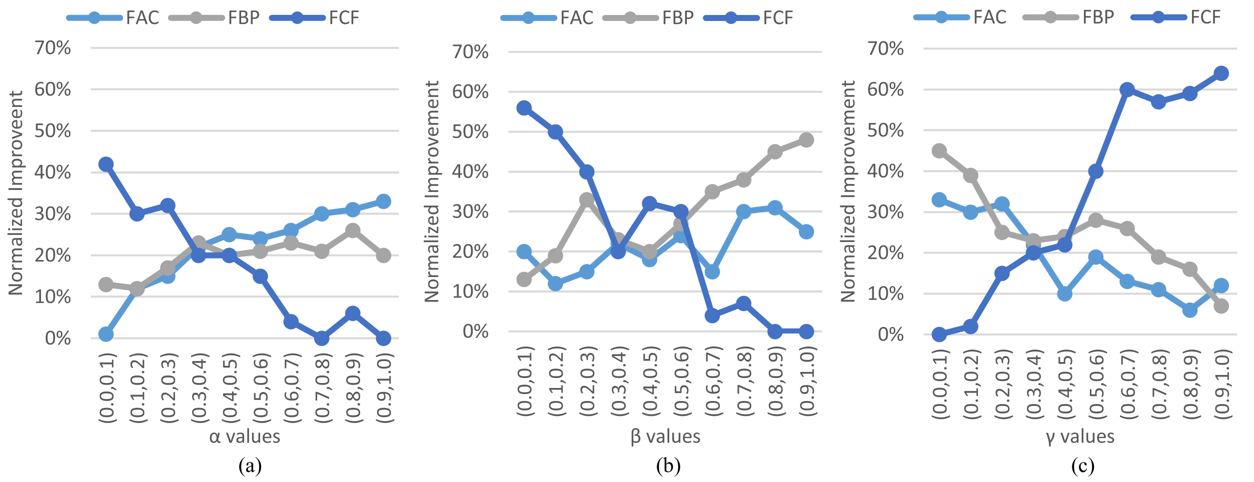

FIGURE 15. **Effects of setting random values of weighting factors for each file on the design parameters. (a): the average of FAC, FBC, and FCF for the file in terms of different ranges of α, (b): the average of FAC, FBP, and FCF for the file in terms of different ranges of β, (c): the average of FAC, FBC, and FCF for the file in terms of different ranges of γ (the results are normalized to the results of base Hadoop).**

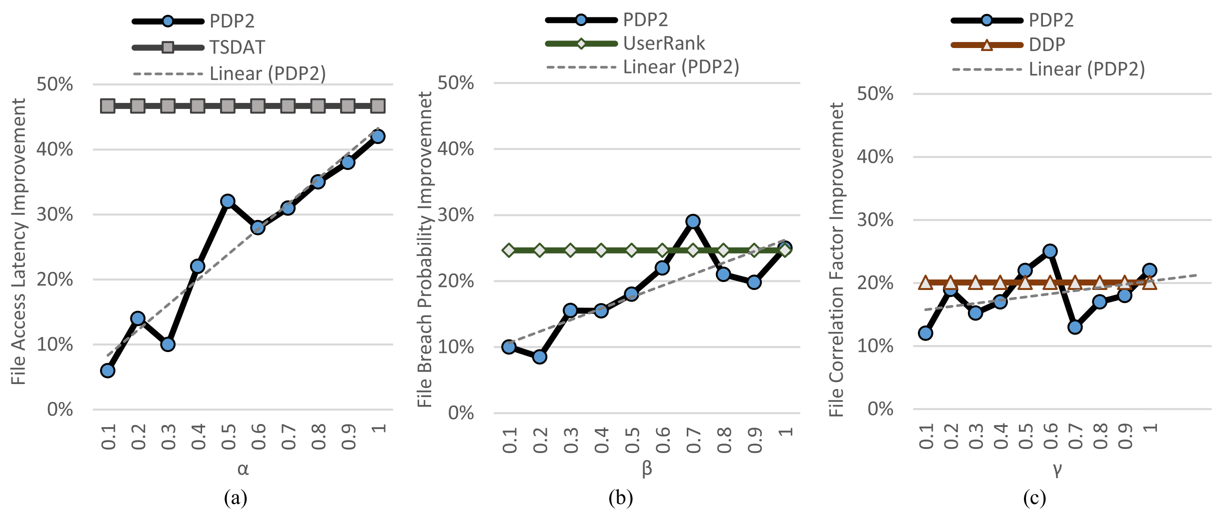

FIGURE 16. **Comparison the results of PDP2 with (a) TSDAT in terms of different values of α, (b) UserRank in terms of different values of β, (c) DDP in terms of different values of γ.**

correlated set of files are two data locality concepts that are not always aligned.

Table 4 is the color scale table that summarizes the results of all scenarios and cases, respecting the average values of the results related to each mechanism. The value of a cell changes its appearance based on its corresponding value. In this color scale, shades of blue/red are applied to the higher/lower values. On average, the PDP2 improves the File Access Latency, File Breach Probability, and File Correlation Factor by about 16.6%, 15.5%, and 15.2%, with lower standard deviations than the other design decisions. The PDP2 results tend to be very close to the mean value. It obtains more stable results under different file, network, and storage conditions compared to the other data placement mechanisms. Regarding Table 4, the standard deviation of PDP2's results is about 1.2%, 4.3%, and 2.5% for File Access Latency, file breach probability, and File Correlation Factor. In contrast, the results of UserRank, TSDAT, and DDP are spread out over the larger range of values.

The results of PDP2 were acquired under a circumstance in which all three design parameters are of equal importance to all users ($\alpha = \beta = \gamma = 0.33$). Also, the experiments were extended to assess the performance of the PDP2 in the cases where only one/two parameter(s) is/are a severe concern(s) of the users. Fig. 16 shows the comparison of the results related to PDP2 and the other mechanisms in terms of different values for weighting factors. Regarding Fig. 16 (b) and (c), by increasing the values of β and γ individually, the PDP2 can resolve the users' concerns about a single criterion as well as single-objective placement mechanisms. Fig. 16 (a) shows that the TSDAT leads to better File Access Latency for all cases. However, it should be emphasized that the PDP2 can

TABLE 4. The summary of the experimentl results related to all implemented scenarios.

Mechanisms	Design Parameters	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5	Scenario 6	**Average**	**STDEV**
GDP	FAC[1]	11.65%	5.76%	27.58%	9.79%	12.40%	24.25%	**15.24%**	**8.65%**
	FBP[2]	20.70%	12.38%	9.00%	18.86%	15.67%	26.04%	**17.11%**	**6.10%**
	FCF[3]	-3.64%	-40.13%	-5.06%	-52.70%	-35.67%	-12.80%	**-25.00%**	**20.56%**
PDP2	FAC[1]	17.95%	15.15%	15.04%	17.68%	16.84%	16.78%	**16.57%**	**1.23%**
	FBP[2]	19.24%	16.43%	7.19%	17.69%	17.07%	15.46%	**15.52%**	**4.27%**
	FCF[3]	19.89%	12.63%	13.82%	15.74%	14.37%	15.00%	**15.24%**	**2.51%**
TSDAT	FAC[1]	48.99%	39.67%	47.76%	47.50%	47.54%	48.70%	**46.70%**	**3.50%**
	FBP[2]	3.00%	7.00%	6.67%	16.67%	3.67%	48.33%	**14.22%**	**17.41%**
	FCF[3]	-17.62%	-36.51%	-45.59%	-51.06%	-27.00%	-28.59%	**-34.39%**	**12.47%**
UserRank	FAC[1]	2.56%	2.12%	22.33%	10.81%	10.86%	10.06%	**9.79%**	**7.36%**
	FBP[2]	24.33%	28.67%	15.33%	38.67%	24.33%	16.59%	**24.65%**	**8.54%**
	FCF[3]	-43.75%	-106.14%	-76.67%	-142.07%	-25.00%	-98.29%	**-81.99%**	**42.89%**
DDP	FAC[1]	-16.34%	-21.93%	-16.42%	-15.25%	5.10%	-15.66%	**-13.42%**	**9.39%**
	FBP[2]	-6.47%	-9.60%	-3.93%	-4.39%	11.67%	-6.50%	**-3.20%**	**7.56%**
	FCF[3]	31.00%	18.33%	35.67%	17.33%	17.33%	0.85%	**20.09%**	**12.25%**

1: File Access Latency Improvement
2: File Breach Probability Improvement
3: File Correlation Factor Improvement

TABLE 5. The effects of centralized and distributed architectures on the design parameters (The results are normalized to the results of base HDFS).

		PDP2			TSDAT			UserRank			DDP		
		FAC (%)	FBP (%)	FCF (%)	FAC (%)	FBP (%)	FCF (%)	FAC (%)	FBP (%)	FCF (%)	FAC (%)	FBP (%)	FCF (%)
Centralized Master	AVG[1]	16.57	15.52	15.24	46.70	14.22	-34.39	09.79	24.65	-81.99	-13.42	-03.20	20.09
	DEV[2]	01.23	04.27	02.51	03.50	17.41	12.47	07.36	08.54	42.89	09.39	07.56	12.25
Distributed Master	AVG[1]	18.03	15.11	15.92	54.52	14.22	-35.97	12.15	26.41	-80.88	-10.31	-03.95	24.65
	DEV[2]	02.42	05.23	02.49	03.46	17.24	12.34	07.28	08.45	42.46	09.30	07.48	12.13

1: Average
2: Standard deviation

moderate the side effects of the single-criterion mechanisms on the other criteria (even for the higher values of α). The probabilistic selection and ranking models cause an increased chance of selecting comprehensive data placement solutions. In this case, the maximum degradations of File Breach Probability and File Correlation Factor for PDP2 are 12.5% and 8.8% less than the TSDAT, respectively.

The HDFS, GFS, and S3 are three major cloud storage systems. The HDFS is a collection of open-source software utilities, while the GFS and S3 are unique to Google and Amazon. The GFS and S3 have been exclusively used by their owners to organize and manipulate huge files and to serve as a model for file systems for organizations with similar needs. Some details about these two systems remain a mystery to anyone outside of the companies. However, despite this veil of secrecy, some knowledge about the GFS and S3's structures and operations have been made public. Generally, two typical architectures have been designed and utilized by the mentioned cloud storage systems: single centralized master and multiple distributed masters. Previous experiments were conducted upon the simulation infrastructure established based on the single centralized master. The experiments were repeated in the simulation environment, which has been set up considering the characteristics of multiple distributed masters. In order to scale the name service horizontally, cloud storage systems use multiple independent masters. The masters are federated and independent and do not require coordination with each other. The slaves are used as common storage for blocks by all the masters. Each slave registers with all the masters in the cluster. Slaves send periodic heartbeats and block reports. They also handle commands from the masters. Table 5 compares the average and standard deviation of the FAC, FBP, and FCF extracted from the simulations of six scenarios in the cases of using the centralized master and the distributed masters architectures. According to Table 5, a slight improvement can be observed in terms of FAC and FCF due to the scalable architecture of the distributed model. The distributed model can more effectively manage overload DataNodes, especially in the case of high disk utilization. The improvement of FAC and FCF can be recognized for all the design decisions. At the same time, there have not been any considerable changes in FBP. The geolocation of DataNodes is the key factor of determining the probability of unauthorized access to the files of a DataNode and cannot be affected by the implementation model of the master node.

V. CONCLUSION

This paper proposes an SLA-aware multi-criteria data placement to remove the limitations of the earlier data placement mechanisms in cloud storage systems. The main goal of most previous techniques is to make the heterogeneity of storage nodes transparent to the clients. This goal was achieved by picking the best choices for storing the clients' files based on a predetermined criterion, user concerns, and the heartbeats of data storage nodes. However, to serve several different clients with diverse needs, the single-criterion approaches for scoring the available data storage nodes cannot simultaneously resolve more than one type of clients' concerns. Moreover, the greedy assignment of top-scored resources can be unfairly proper in favor of the files, which have been uploaded earlier than the other ones. In the proposed placement mechanism, the decision to place a file in a cluster is taken concerning a trade-off between three criteria that can be crucial from the clients' viewpoint, such as file access delay, computing performance, and data privacy. The proposed cloud storage provider gives each data storage node a chance to become the host of an already received file. The chance parameter is estimated with regard to the gain of adding the mentioned file to each data storage node. In the case of selecting a set of storage nodes in the cluster, the proposed cloud storage provider can also consider levels of user support services. To implement and compare design decisions, the CloudSim framework has been revised to reflect the user's demands in terms of different expected services, attributes of client machines that can affect the performance of internal mechanisms of cloud storage systems, and some new file specifications that have effects upon user influence-based file/data distribution. Furthermore, six different scenarios have been intended to study the performance of each data placement mechanism under various conditions of the file specification, cloud storage architecture, and network topologies. The experimental results show about 16.6% data access latency, 15.5% data privacy, and 15.2% computing performance improvements with high stability by the proposed mechanism compared to the conventional data placement mechanisms under different files, networks, and storage conditions.

REFERENCES

[1] H. Cai, B. Xu, L. Jiang, and A. V. Vasilakos, "IoT-based big data storage systems in cloud computing: Perspectives and challenges," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 75–78, Oct. 2016.

[2] T. Mahmood, S. P. Narayanan, S. Rao, and T. N. Vijaykumar, "Karma: Cost-effective geo-replicated cloud storage with dynamic enforcement of causal consistency," *IEEE Trans. Cloud Comput.*, vol. 1, no. 1, pp. 18–28, Mar. 2018.

[3] A. Mittal, V. Jain, and T. Ahuja, "Google file system and Hadoop distributed file system: An analogy," *Int. J. Innov. Advancement Comput. Sci.*, vol. 4, no. 1, pp. 29–43, 2015.

[4] A. Albeshri, C. Boyd, and J. G. Nieto, "GeoProof: Proofs of geographic location for cloud computing environment," in *Proc. 32nd Int. Conf. Distrib. Comput. Syst. Workshops*, Macau, China, Jun. 2012, pp. 506–514.

[5] K. Benson, R. Dowsley, and H. Shacham, "Do you know where your cloud files are?" in *Proc. 3rd ACM workshop Cloud Comput. Secur. workshop (CCSW)*, New York, NY, USA, 2011, pp. 73–82.

[6] C. Liao, A. Squicciarini, and D. Lin, "LAST-HDFS: Location-aware storage technique for Hadoop distributed file system," in *Proc. IEEE 9th Int. Conf. Cloud Comput. (CLOUD)*, San Francisco, CA, USA, Jun. 2016, pp. 662–669.

[7] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.

[8] E. Katz-Bassett, J. P. John, A. Krishnamurthy, D. Wetherall, T. Anderson, and Y. Chawathe, "Towards IP geolocation using delay and topology measurements," in *Proc. 6th ACM SIGCOMM Internet Meas. (IMC)*, Brazil, Brasilia, 2006, pp. 71–84.

[9] B. Gueye, A. Ziviani, M. Crovella, and S. Fdida, "Constraint-based geolocation of Internet hosts," *IEEE/ACM Trans. Netw.*, vol. 14, no. 6, pp. 1219–1232, Dec. 2006.

[10] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 331–346, Feb. 2019.

[11] A. Razaque and S. S. Rizvi, "Privacy preserving model: A new scheme for auditing cloud stakeholders," *J. Cloud Comput.*, vol. 6, no. 1, pp. 1–17, Dec. 2017.

[12] N. Paladi and A. Michalas, "One of our hosts in another country: Challenges of data geolocation in cloud storage," in *Proc. 4th Int. Conf. Wireless Commun., Veh. Technol., Inf. Theory Aerosp. Electron. Syst. (VITAE)*, Denmark, Copenhagen, May 2014, pp. 1–6.

[13] V. Ubarhande, A.-M. Popescu, and H. Gonzalez-Velez, "Novel data-distribution technique for Hadoop in heterogeneous cloud environments," in *Proc. 9th Int. Conf. Complex, Intell., Softw. Intensive Syst.*, Brazil, Brasilia, Jul. 2015, pp. 217–224.

[14] M. Gondree and Z. A. Peterson, "Geolocation of data in the cloud," *Proc. ACM Conf. Data Appl. Secur. Privacy*, New York, NY, USA, 2013, pp. 25–36.

[15] (2012). *Apache Hadoop version 1.X.Y*. Accessed: Mar. 2, 2019. [Online]. Available: https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

[16] J. Ju, J. Wu, J. Fu, Z. Lin, and J. Zhang, "A Survey on Cloud Storage," *J. Comput.*, vol. 6, no. 8, pp. 1764–1771, 2011.

[17] (2015). *Apache Hadoop version 2.X.Y*. Accessed: Mar. 2, 2019. [Online]. Available: https://hadoop.apache.org/docs/r2.7.2/

[18] S. V. Ambade and P. R. Deshpande, "Heterogeneity-based files placement in Hadoop cluster," in *Proc. Int. Conf. Comput. Intell. Commun. Netw. (CICN)*, Jabalpur, India, Dec. 2015, pp. 876–880.

[19] P. Neha, P. Narendra, M. I. Hasan, S. Parth, and P. Mayur, "Improving HDFS write performance using efficient replica placement," in *Proc. 5th Int. Conf. Confluence Next Gener. Inf. Technol.*, Noida, India, Sep. 2014, pp. 36–39.

[20] S. Vipulkumar, "A survey on data placement in heterogeneous cloud environment for big data," *Int. J. Eng. Develop. Res.*, vol. 4, no. 4, pp. 583–588, 2016.

[21] S. Dolev, P. Florissi, E. Gudes, S. Sharma, and I. Singer, "A survey on geographically distributed big-data processing using MapReduce," *IEEE Trans. Big Data*, vol. 5, no. 1, pp. 60–80, Mar. 2019.
[22] J.-X. Wu, C.-S. Zhang, B. Zhang, and P. Wang, "A new data-grouping-aware dynamic data placement method that take into account jobs execute frequency for Hadoop," *Microprocessors Microsyst.*, vol. 47, pp. 161–169, Nov. 2016.
[23] C. Guo, Q. Shen, Y. Yang, and Z. Wu, "User rank: A user influence-based data distribution optimization method for privacy protection in cloud storage system," in *Proc. IEEE 39th Annu. Comput. Softw. Appl. Conf.*, Taiwan, Taipei, Jul. 2015, pp. 104–109.
[24] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.
[25] D. Hu, D. Chen, S. Lou, and S. Pei, "Research on reliability of Hadoop distributed file system," *Int. J. Multimedia Ubiquitous Eng.*, vol. 10, no. 11, pp. 315–326, Nov. 2015.
[26] (2017). *HDFS High Availability Using the Quorum Journal Manager*. Accessed: Feb. 20, 2020. [Online]. Available: https://hadoop.apache.org/docs/r2.7.7/hadoop-project-dist/hadoop-hdfs/HDFSHighAvailabilityWithQJM.html
[27] M. Li, Y. Ma, and M. Chen, "The dynamic replication mechanism of HDFS hot file based on cloud storage," *Int. J. Secur. Appl.*, vol. 9, no. 8, pp. 439–448, Aug. 2015.
[28] (2017). *Apache Hadoop Version 3.X.Y.* Accessed: Jan. 10, 2020. [Online]. Available: https://hadoop.apache.org/docs/r3.0.0
[29] B. G. Babu and M. Kumar, "Dynamic colocation algorithm for Hadoop," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Delhi, India, Sep. 2014, pp. 2643–2647.
[30] (2018). *HDFS Federation*. [Online]. Available: https://hadoop.apache.org/docs/r2.7.7/hadoop-project-dist/hadoop-hdfs/Federation.html
[31] M. Maghsoudloo and N. Khoshavi, "Elastic HDFS: Interconnected distributed architecture for availability–scalability enhancement of large-scale cloud storages," *J. Supercomput.*, vol. 76, no. 1, pp. 174–203, 2019.
[32] K. Liu, G. Xu, and J. Yuan, "An improved Hadoop data load balancing algorithm," *J. Netw.*, vol. 8, no. 12, pp. 2816–2822, Dec. 2013.
[33] (2020). *Standard Score*. Accessed: Mar. 5, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Standard_score
[34] S. K. Garg and R. Buyya, "NetworkCloudSim: Modelling parallel applications in cloud simulations," in *Proc. 4th IEEE Int. Conf. Utility Cloud Comput.*, Toronto, ON, Canada, Dec. 2011, pp. 105–113.
[35] S. Sturm, "Storage CloudSim: A simulation environment for cloud object storage infrastructures," in *Proc. Int. Conf. Cloud Comput. Services Sci.*, Madrid, Spain, 2014, pp. 186–192.
[36] C.-W. Lee, K.-Y. Hsieh, S.-Y. Hsieh, and H.-C. Hsiao, "A dynamic data placement strategy for Hadoop in heterogeneous environments," *Big Data Res.*, vol. 1, pp. 14–22, Aug. 2014.
[37] S. Yang, P. Wieder, M. Aziz, R. Yahyapour, and X. Fu, "Latency-sensitive data allocation for cloud storage," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, Portugal, Lisbon, May 2017, pp. 1–9.

MOHAMMAD MAGHSOUDLOO received the B.Sc., M.Sc., and Ph.D. degrees from the Department of Computer Engineering, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2009, 2012, and 2016, respectively. He has been an Assistant Professor with the Department of Computer Engineering and Information Technology, Golestan University, since 2017. In 2018, he established the Cloud of Things (CoT) Research Center at Golestan University and has been chairing the research center since then. His research interests include dependability evaluation, fault-tolerant computing, dependable computer architecture, high-performance computing, cloud computing/storage architectures, and the Internet of things. He is currently a member of the IEEE Computer Society and the Computer Society of Iran (CSI).

AREZOO RAHDARI received the B.Sc. degree from the Department of Computer Engineering, University of Sistan and Balouchestan, Zahedan, Iran, in 2018, and the M.Sc. degree from the Department of Computer Engineering and Information Technology, Golestan University, Gorgan, Iran, in 2021. Her research interests include fault-tolerant computing, dependable computer architecture, high-performance computing, cloud computing/storage architectures, and the Internet of things. She is currently a member of the Computer Society of Iran (CSI).

NAVID KHOSHAVI received the M.Sc. degree from the Department of Computer Engineering, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2009, and the Ph.D. degree in computer engineering from the Department of Computer Engineering, University of Central Florida, FL, USA, in 2017. He has been an Assistant Professor with Department of Computer Science, Florida Polytechnic University, since 2017. His research interests include dependability evaluation using fault injection techniques, fault-tolerant hardware architectures and domain-specific architecture for machine learning applications, high-performance computing, and cloud computing and storage. He has received multiple awards recognizing his research initiative and academic success. He is currently a member of the IEEE Computer Society.