electronics

Article

Intelligent Replica Selection in Edge and IoT Environments Using Artificial Neural Networks

Nour Mostafa *, Wael Hosny Fouad Aly, Samer Alabed and Zakwan Al-Arnaout

College of Engineering and Technology, American University of the Middle East, Egaila 15453, Kuwait
* Correspondence: nour.moustafa@aum.edu.kw

Abstract: Cloud, edge and Internet of Things (IoT) technologies have emerged to overcome the challenges involved in sharing computational resources and information services. Within generic cloud systems, two models have been identified as having widespread applicability: computation clouds and data clouds. A data cloud is cloud computing that aims to manage, unify and operate multiple data workloads. Many current applications generate datasets consisting of petabytes (PB) of information. Managing large datasets is a complex issuel; in particular, datasets associated with many applications can be distributed widely in geographical terms, particularly in IoT systems. Edge and IoT systems are facing new challenges with increased complexity, making scalability an important issue that will affect the performance of the system. Data replication services are widely accepted techniques to improve availability and fault tolerance, and to improve the data access time. Current replication services, however, often exhibit an increase in response time, reflecting the problems associated with the ever-increasing size of databases. This paper proposes a prediction model to predict replica locations using the files' access profile, which feeds the neural networks with the access and location behavior (file profile) to minimize the overhead of transferring large volumes of data, which slows down the system and requires careful management. This new model has shown high accuracy and low overheads. The result shows a significant improvement in total task execution time using the proposed model for locating files by 16.34% and 30.45%; in addition, the results show bandwidth improvement by 24.7% and 49.4% compared to the user profile prediction model and replica service model without prediction, respectively. Consequently, the proposed algorithm can improve data access speed, reduce data access latency and decrease bandwidth consumption.

Keywords: cloud; edge; IoT; data replication; clustering; neural networks

Citation: Mostafa, N.; Aly, W.H.F.; Alabed, S.; Al-Arnaout, Z. Intelligent Replica Selection in Edge and IoT Environments Using Artificial Neural Networks. *Electronics* **2022**, *11*, 2531. https://doi.org/10.3390/electronics11162531

Academic Editors: Gianluca Cornetta, Abdellah Touhafi, Antonio Mariscal Jiménez and Juan-Carlos Cano

Received: 5 July 2022
Accepted: 10 August 2022
Published: 13 August 2022

1. Introduction

The largest astronomical observatory generates databases of around 10 terabytes, and such databases are increasing in size annually; such systems require a means of organizing, handling and manipulating such high volumes of data [1,2]. In addition, this type of dataset tends to be accessed by users in different locations who may create local replicas of the data to reduce the latency involved in wide area data transfers and, thereby, improve their application's performance. To support this, model replica management systems, or data replication mechanisms [2,3], have been developed to create, register and manage replicas. By using such systems, data can be accessed from another site (reducing server load), and recovery from system failures can be supported.

Within a cloud environment, files can have replicas stored at many cloud sites on 'storage elements'. If one replica site is down, data can be accessed from another site, and queries can be executed more efficiently since they can access local or nearby copies. However, when several large databases are required, then systems based on this current model are slow to respond to transfer requests [2,4]. In addition, there are high overheads in the form of the increased cost associated with updates as each replica must be updated and there is an (obvious) increase in data redundancy. It is easy to see how further performance

issues will arise with the growth and usage of such very large databases using the current replication strategies. It can be argued that central to cloud, edge and IoT systems is the need to provide efficient access to resources in a scalable manner. As mentioned above, the current approaches do not meet these requirements.

Lately, with today's increase in the number of cloud services, smartphones and other IoT devices with limited specifications that build the core elements of edge computing, many organizations have to deal with limited resources. There are several cloud services that are designed to facilitate and support cloud applications, e.g., replication services. A replication service aims to select a data replica from those available to minimize the application access time. However, current replication services are exhibiting an increase in response time, reflecting the problems associated with the ever-increasing size of databases [5,6]. The proposed File Prediction Replication Model (FPRM) system can access the replica in a minimum response time for given tasks under execution. The performance of the proposed system overcomes similar systems using full dynamic replication.

Both the proposed and current replication systems, Full Replica Management (RM) [1,2,7] and Prediction Model (PM) [8,9], have been implemented using a simulation environment, and it has been shown, for a range of experiments, that the proposed system can serve an application's data requirements in a lesser transfer time. Correspondingly, it can be argued that the efficiency of the proposed system is higher than the existing system based on current replica models. The proposed FPRM is treated as an optimization problem that minimizes the sum of the data access costs and replica maintenance costs. Optimizations are performed by restructuring workflows. The first optimization is to find out whether there are any intermediate results available. The second technique is to cluster similar tasks and map them to the same required files/replicas, saving the overhead of searching and transferring large volumes of data that affect system performance. Mapping is performed for the first task, and then all similar tasks are mapped similarly.

1.1. Motivation

Large data management and task scheduling within the cloud, IoT and edge environments are complex issues. Resources are distributed at different geographical locations and owned by different organizations that have different usage policies, cost, access, availability and load patterns. The owners and consumers of the resources have conflicting goals, objectives, strategies and requirements; in addition, they aim to achieve high throughput, and consumers aim to get a minimum cost within the specified time limits, also referred to as deadlines. These systems are facing new challenges of task execution, and with the increased complexity of IoT and edge systems, it is necessary to deal with these queries using intelligent management of terabyte data transfer over wide area networks to cope with current and future data sizes. The complexity means the size of the database is growing, and user queries could request several files/replicas stored at different sites, which takes a long time to locate, and this delays the task's execution. One of the most vital subjects in IoT and edge computing is data replication and selection, which requires high reliability and efficiency. With the increased number of users, resources and data in such an environment, it is costly to maintain data reliability, performance and availability. A reliable, intelligent and dynamic file/replica selection technique is required. This research was carried out keeping this target in mind, and it resulted in the development of a file's location prediction model.

1.2. Contributions

As the volume of data contributing to cloud, edge and IoT systems grows, so do the problems associated with efficient and effective replica selection and allocation increase in proportion. The work presented in this paper presents a novel approach that aims to go some way toward finding a solution to solve this problem by developing an intelligent dynamic prediction model based on neural networks. The run-time prediction model is meant to generate file location predictions for incoming tasks using historical executions.

It utilizes clustering to separate related tasks from the history and generates predictions using a mean predictor. The proposed model collects and maintains a repository of replica activity data in the form of execution logs stored in standard formats. The prediction model was based on the fact that historical jobs are clustered on the basis of six parameters, which have a minimum coefficient of variation calculated for their replica's location and run times. It presented an opportunity to use these clustered tasks to generate the prediction of a replica's location of an incoming task. When a new task arrives, the prediction for the location of its required files/replicas is made using these logs instead of using the replica catalog approach, which results in decreasing the queuing time and run time, as shown in the results section. Queue time is a very important parameter since it has a strong effect on the task execution time. Every time tasks are submitted, and results are returned, the execution history data are updated, thereby ensuring that an accurate and up-to-date view of replica activity is maintained. CloudSim simulator is used to test and evaluate the performance of the proposed model. The experimental results illustrate that FPRM reaches improved total execution time, bandwidth consumption and algorithm run time compared with the referenced models.

The remainder of the paper is organized as follows: Section 2 presents background reading and related work. To provide a deeper understanding of this area, Section 3 introduces references to the replica model without prediction and the replica model with prediction using a user profile. Section 4 introduces the proposed replica model using a file profile. Section 5 presents the simulation testbed and results. Finally, the paper concludes with Section 6, focusing on the contribution along with future recommendations. The next section discusses the related work and approaches of data replication.

2. Related Work

With the increasing number of users, resources and data, edge has emerged as a promising solution for such complicated systems. Replication has been regarded as one of the major optimization techniques for providing fast data access [10] to such seamless and ubiquitous computing systems; however, studies of replication in edge computing are still premature. The proposed model in [11] introduced a data replication technique to optimize the consumption of resources in edge environments, which reduces the total time for a search request or another procedure, and it shares structures in the device context for analyzing workloads and storing data utilizing mathematical models. The proposed model in [12] improves the high-level architecture performance using a replication method that employs various synchronization approaches for rapid replications.

The replication approaches described in [13,14,16**?**] are known to improve the average processing time using the combined replication method with local and cluster storage compared to the centralized replica management under the traditional edge computing model. Many proposed systems were developed by scholars for various models, proving the truthfulness of replicating data in different storage sites [14**?**]. The authors in [**?**] proposed a comparative study of a multi-replica verification method using data integrity checking of multi copies based on PDP and Merkle hash tree. The results show that IDMPMR-PDP over-comes and is more economical compared to the MuR-DPA method. The Cassandra dynamic snitch mechanism in [16] calculates a score for each node based on the latency and severity information and uses this score for selecting the fastest node for the routing read request. The proposed C3 model in [17] is a replica selection mechanism in which each client calculates a score of individual servers to select the fastest server for each read request and C3. The work of Liu et al. [13] aims to replicate a local copy on the IoT devices and the local cluster rather than the centralized replica management system; the proposed model replicates the data based on data high priority and real-time performance. In [18], the authors proposed a replica selection model based on the response time (DRS-RT); the proposed model calculates the file access tendency periodically, then the DRS-RT returns to the user the node that has the highest service capability that has the required file or replica. Sudalai et al. [19] proposed a multi-dimension parameters

model that quantifies the importance of a replica by using the replica frequency number of requests and the cost of the replica to predict the future usage of the replica in a data grid environment.

Several techniques aim to reduce data processing at the network edge by offering alternative cloud computing options, such as cloudlets, mobile edge computing and fog computing [20]. The findings in [21] also exhibited that data replication is ideal for improving data sharing, worldwide traffic and lowering response times, in addition to the ability to perform on a disconnected server. Saranya et al. [21] proposed using simple and random replication techniques for the Mobile Ad hoc Networks (MANET) to improve the availability of data, which covers the related issues to the MANET, such as the availability of resources, response time, power consumption and consistency management. The results show that the random algorithm can achieve bandwidth consumption better than the simple algorithm. The results have not shown the access response time of the proposed algorithms, which is one of the most important factors in any replication system. Gill and Singh [22] have proposed an algorithm to optimize the cost of replication based on the concept of the knapsack problem; the algorithm is named the Dynamic Cost-aware Re-replication and Re-balancing Strategy (DCR2S). The proposed system uses the "pay as use" model, where the system charges the users for the services they use. It designed to determine the relationship between the number of replicas, availability and the cost of replication. DCR2S has three steps, (1) determining what and when to replicate the file by the theory of temporal locality, (2) check availability requirement and (3) the replacement of the new replica. The aim of this system is to have low replication costs to meet the user's subscription, regardless of the high response time and low consistency rate.

Junfeng et al. [23] proposed a replica selection algorithm based on a genetic algorithm (GASA), aiming for replica selection optimization in cloud storage of massive data. The proposed model maps the key steps of the genetic algorithm and replica selection criteria. Then, a probability equation is used to obtain the optimal solution. The proposed model uses the GASA algorithm as a self-adapting optimal probabilistic algorithm as GASA simulates a biotic population evolutionary mechanism in the natural environment, such as mutation, realization and overlapping, where it ultimately gains an optimal replica. Wakil et al. [24] proposed a hybrid ant colony optimization (ACO) and genetic algorithm (GA) as an optimization strategy for replica selection in the IoT. The ACO is used to produce diversity, while the GA is utilized to perform a comprehensive search over the search space. Nevertheless, the run was longer as the suggested model is based on hybrid character. The authors in [1] suggested using neural networks and fuzzy logic in the upcoming studies, which could take into consideration the performance of some novel meta-heuristic algorithms to solve the replica selection issue in the IoT. In [25], the authors introduced a comprehensive review of enabling computation offloading and task execution in vehicular networks. First, the architectures of the task execution in computational paradigms in vehicular networks were introduced. In addition, the key features between different computing paradigms were investigated to distinguish similarities and dissimilarities in these paradigms.

In [26], an optimization scheme is proposed to construct a reconfigurable radio environmen; the proposed model optimizes beamforming weight vectors by applying uplink–downlink duality and a singular value decomposition. Further, the proposed model investigated the shifter's iterative optimization using Taylor expansion and penalty function methods. The authors in [27] proposed a joint beamforming and power allocation for satellite-terrestrial integrated networks for non-orthogonal multiple access (NOMA)-based satellite terrestrial integrated network (STIN). The proposed model maximizes the sum rate of the STIN by formulating a constrained optimization. In [28], the authors investigated the multicast communication of a satellite and aerial-integrated network (SAIN) with rate-splitting multiple access (RSMA). The aim of the proposed model is to achieve a fast convergence rate and maximize the sum rate of all users under the constraints of the QoS requirements of both ES and IoTDs by exploiting the SCA, the first-order Taylor expansion and penalty function methods. The authors in [29] proposed a multitask learning method

using geometry reasoning for scene depth and semantics. The proposed model performs depth and camera pose estimation using semantic segmentation by reasoning the geometric correspondences between the pixel semantic outputs and the semantic labels at multiscale resolutions.

3. Reference Models

3.1. Reference I: Replica Management System without Prediction

RM is a widely accepted technique in distributed environments that manage data selection and replication at different sites. The RM uses a local replica catalog [30] to process the queries from users and resources and it is responsible for indexing files, which, in turn, store the mapping between the Logical File Name (LFN) and Physical File Name (PFN). When an LFN does not exist in the local replica catalog, which is called (regionalRC), then the Replica Catalog (RC) will contact the Top Replica Catalog (TopregionalRC), which is the root of all replica catalogs for a list of names that maps this LFN either in the local or the remote site. The local or remote replica catalog is concerned with searching for files or replicas within a site and is organized as a tree. The catalog is stored within databases as metadata files; whenever a digital entity (i.e., creation of a replica) is performed by an operation, the replica state information is maintained and stored (i.e., the physical file name and location of the replica) in the metadata catalog. After that, a TopregionalRC is created that acts as a central replica catalog, or the top replica catalog, in a tree of replica catalogs, because the RCs are also attached with each other in the RC tree. For example, a scientific experiment site generates a large volume of data that are stored in a data center. The data center notifies the local RC about a list of available datasets in the center, which, in turn, notifies the TopRegionalRC of the new datasets generated and stored in the data center. When the user submits a task, his/her resources will request a file copy of the dataset; the regionalRC will be checked first. If the dataset is stored locally, the PFN will be sent; if not, the regionalRC will send a request to the TopRegionalRC and return a mapping between LFN and PFN to the regionalRC, which, in turn, sends it to the user's resources, as shown in Figure 1.

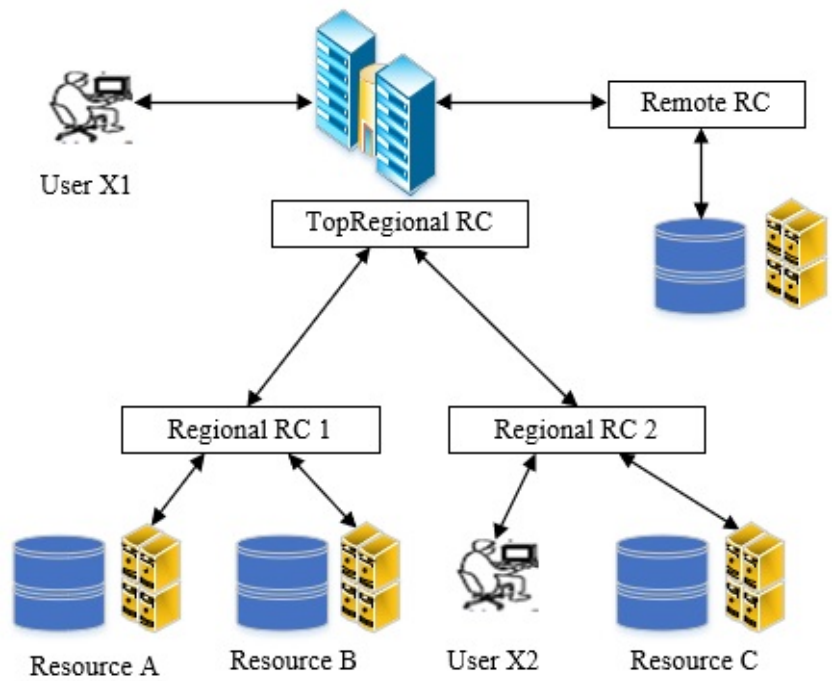

Figure 1. Communication process flow between Top, Regional and Remote RC.

These approaches provide a simple matchmaking approach based on a file's name and take a long time to query a particular file [31–34]. Moreover, if the file's name is not

registered in the replica catalog, the task turnaround time will be doubled each time due to the delay in locating the file from another replica catalog. The replica catalog works well when the system is small. However, current cloud and edge storage systems are expected to scale up, and database sizes are already at petabyte levels. In such environments, it is easy to see that the overheads required to retrieve data are increasing and will continue to grow.

Consequently, cloud and edge storage systems are facing new challenges of task execution, and with the increased complexity of cloud storage systems, it is necessary to deal with these queries using intelligent management of terabyte data transfer over wide area networks to cope with current and future data sizes. The complexity means the size of the database is growing, and user queries could request several files/replicas stored at different sites, which take a long time to locate and this delays tasks execution, as explained below. Consequently, the potentially large number of users/devices and resources in a cloud and edge storage system means that a centralized algorithm may be ineffective. For example, a scheduling algorithm that focuses only on maximizing task utilization, disregarding costs associated with fetching remote data, is unlikely to be efficient [10,13], as the effort required to retrieve relevant information has become significantly greater, especially in large-scale databases.

3.2. Reference II: Replica Management System with Prediction

The proposed PM in [8,9] provides an efficient solution to access remote files that access very large databases in a faster and more efficient way. The model assumes the following:

- There will be multiple users spread over different remote sites.
- These users will submit a number of tasks (jobs).
- These tasks (jobs) can require one or more files.
- The files can be located in local or remote resources.

Existing techniques for accessing files were considered, and the most frequently used were included, e.g., replication techniques. The proposed system developed a prediction model responsible for predicting a file's location for incoming task queries in order to minimize the total overhead of the task turnaround time. The proposed system provides a replica selection facility as a high-level service. As the intention of the work is to provide a series of high-level services, it is clear that a suitable base system had to be selected as the source of the core services. This model will enable the system to predict the location of data required by users. This prediction model will determine a file's location either in the cache, local or remote resources using a neural network that analyzes the user's past history.

It is noted that many users of parallel computers have a tendency to be repeatedly doing the same work and using the same data. This creates an opportunity to develop the proposed model to use task parameters, e.g., user id, file name, file location and resource id, etc. These parameters are actually used to identify the file location of the new task from execution logs. These execution logs can then be used to generate the prediction for the new task. The parameter sets that are used to predict a file's location are stored in the history database after a task is completed. For a new task, this history database is searched, and tasks with a similar parameter set are used to make predictions for the incoming tasks. As the number of historical executions increases, it is expected that the accuracy of predictions will also increase [19]. Based on the above specification, an Artificial Intelligent (AI) technology was chosen to develop the proposed model.

4. FPRM Proposed Model

An AI technology was chosen to develop the proposed model. The next section explains AI technologies and features.

4.1. AI Technologies

AI technologies have a long history of a wide range of technologies that have been developed and used successfully in parallel and distributed systems, e.g., expert systems

and neural networks. Expert systems are computer programs that can perform a task that normally requires the abilities of a skilled human. These tasks are usually decision-making tasks rather than physical activities, i.e., predicting (forecasting) weather conditions. Neural networks are computer programs developed and trained to store, recognize and solve combinatorial optimization problems by associatively retrieving patterns or database entries, e.g., to filter noise from measurement data or to control imprecise problems [35]. As mentioned above, the proposed model is intended to predict file locations, and an NN is suitable for large datasets to determine an accurate result as an NN can work with a very large number of passes within datasets. This is a positive advantage as IoT and edge computing environments deal with huge datasets, and the need for automated processing becomes clear; therefore, accurate results can be achieved [36–38].

4.1.1. The Neural Networks

Neural networks learn from experience, just as the brain does by being exposed to the information to identify patterns in data. There are a set of processing elements in the neural networks called nodes, which are interconnected in a network similar to neurons in the brain and can then be used in the pattern-identification process. This differentiates NN from traditional computing programs that follow instructions in a predetermined order [9,39]. Neural networks generally have at least three layers, referred to as input, hidden and output [36]. The data enters the networks from the input layer, which is the starting point. The hidden layer receives the data from the input layer for processing, which acts as an intermediate unit. Then, a new signal is passed to the output layer. The interconnection weights play a key role in processing information in neural networks. Transferring data from layer to layer is measured by the relative strength expressed by weights. These weights are repeatedly adjusted during the network's learning process, this adjustment process. There are two types of learning algorithms, unsupervised and supervised algorithms, which are explained in the next section [39,40].

There are two phases of the construction of the components of a network, the learning and recall operations. The weights are modified in the learning phase by passing input data into the input layer to receive accurate results. Then, the training data are measured and compared to check how close these data are to the predictions of the networks. The goal is to produce trainable weights using the training set to design and train the network where the desired output matches the network's output [41]. The recall phase involves presenting the desired response at the output layer. There are two basic categories of learning algorithms for neural networks: supervised learning and unsupervised learning. In supervised learning, the neural network calculates the output and compares this to the actual output of the network to create an error signal. In unsupervised learning algorithms, there is no desired output. As the aim of the proposed model is to predict the location of the file/replica, therefore, the supervised learning algorithm was used in the proposed model.

4.1.2. The Neural Network Tool justNN

The neural networks tool justNN [42] is a development and prototyping tool used to allow the training and testing of a perceptron multi-layer network. justNN has an important advantage that allows parameters to be defined using numeric and textual data and provides the output either in textual or numeric form, which fits with the proposed model construction. Furthermore, CloudSim produces a text file that contains information about users and files in the form of textual and numeric data. The text file is sent as XML input data to the input layer, and the network produces similar XML output data, which is used by the CloudSim.

A typical neural network development process follows five steps [36]:

- Data Gathering: acquisition and pre-processing of training data.
- Data Preparation: preparing the data for the neural network to transform data into a form the network can use.
- Network Creation: this involves network architecture, i.e., multi-layer perception.

- Network Training: this involves training the network with the relevant collected data.
- Network Testing: testing the network with unseen data derived from real simulation code.

The proposed model follows this pattern, and each of the steps are explained in the following sections. To reduce the expected delays during accessing files/replicas in IoT and edge environment, a possible approach would be to predict file (data) requirements and pre-load (or pre-access) this data. The purpose of the proposed prediction model is to predict the location of the file/replica. Ultimately, this will reduce the overheads for a complete complicated search approach that would involve accessing the local, top or remote replica catalogs. An important goal of the proposed prediction system is the ability to support a dynamic environment, such as IoT and edge. In other words, the system must be flexible, expandable and sustainable to meet the needs of a rapidly developing domain. In many respects, the process used can be viewed as straightforward. A history data of files' profiles can be determined based on task executions' logs. Consequently, such a history can be used to train an artificial neural network. Subsequently, when a user's task is detected in the queue, the system can predict file requirements and issue the required instructions to pre-load the required file data.

The proposed model is designed to support a large number of requests and very large databases (i.e., databases involving petabytes of storage). Consequently, the model's components must be designed in a manner that supports system enhancement, i.e., the addition of more participants and physical servers. Such a structure is necessary to support scalability, a fundamental requirement for large-scale systems. Furthermore, it can be argued that such a flexible structure renders the system scalable, facilitating data and information sharing. The proposed model will support the dynamic, widely accepted view of the IoT and edge environment in which resources may join and leave the system, allowing the system to be scaled up or down as requested and supporting a dynamic model of interconnection.

4.2. Description of Proposed Algorithm

As mentioned above, while there are existing data access systems based on replication, it has not been shown that they support scalability (i.e., have not been implemented or tested on large-scale systems), and hence they may not be suitable for very large databases. To overcome this weakness, the work in [8,9] has been extended, which creates a profile for each file in the prediction model and feeds the prediction model with these data to predict a file's location instead of using the users' historical data or the RM. The proposed FPRM prediction model for cloud, edge and IoT systems will provide the data required by the user's task in a minimum response time. The proposed model will be used in combination with the replica management system in the case of prediction failure or a new task that requires file(s) that are not trained in the neural networks. Figure 2 shows the interaction between the prediction model and the current model.

Multiple users/devices spread across different remote sites will submit a number of tasks; these tasks require one or more files, and these files are located in local and remote resources. The run-time prediction model is meant to generate run-time predictions for incoming tasks using historical executions. It utilizes clustering to separate related tasks from the history and generates predictions using a mean predictor. This service performs a simple statistical analysis of the data being produced by historical executions and makes predictions about the location of the file/replica for incoming tasks.

Existing replication techniques for accessing files or replicas were considered. The proposed system can access the replica in a minimum response time. Furthermore, the subject of replacement algorithms has been investigated in [43], and this approach is considered a placement algorithm for the proposed model. The objective of the replacement policy is to make the best use of available resources, including disk, memory space and network bandwidth. A typical cloud, edge and IoT architecture in Figure 3 is divided into three layers. A lower layer composed of IoT devices, such as smart vehicles, traffic systems, sensors, personal health care devices, etc., is used to assist in the construction of an upper layer of

high-level edge layer, which consists of micro data centers. The upper layer is the cloud layer, which consists of data centers and third-party cloud storage systems. The proposed system provides a replica selection facility as a high-level service at both the cloud and edge layers. As the intention of the work is to provide a series of high-level services (replica and data consistency service), it is clear that a suitable base system had to be selected as the source of the core services.

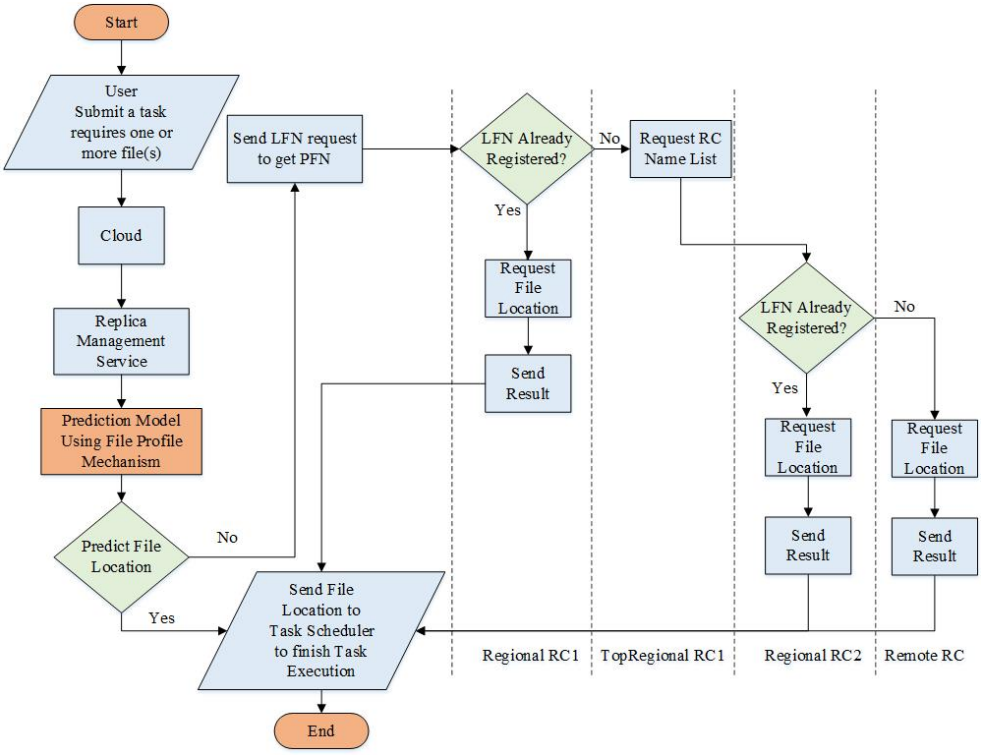

Figure 2. Replica prediction and selection process.

The proposed model creates profiles of replicas using the users' regular replica access activities; these data will be used by the prediction model as training data, which will be used at the final stage to predict the location of the replica resulting in a dynamic system reflecting real-life scenarios. There are six parameters used as training data for the NN as follows:

- Name: the file name.
- Owner Name: the owner name of this file.
- Attribute Size: the size of this object (in bytes). This object size is not the actual file size. Moreover, this size is used for transferring this object over a network.
- Size: the file size (in MBytes).
- Resource ID: the resource ID that stores this file.
- Creation Time: the file creation time (in milliseconds).
- Transaction Time: the last transaction time of this file (in seconds).

As an initial experiment, it was decided to provide a model reflecting local and global communication data access requirements and to provide an experimental framework that could be expanded over time to reflect greater complexity. Given that a reasonably high level of stability was required, it was felt appropriate to select a simulation environment, as this is a well-tried approach in experimental work involving cloud, edge and IoT architectures. Based on the high-level overview of the proposed model described above and its requirements, CloudSim [30] has been chosen as the experimental framework, and the next section describes the system setup. In the next section, the initialization required by the proposed model is outlined.

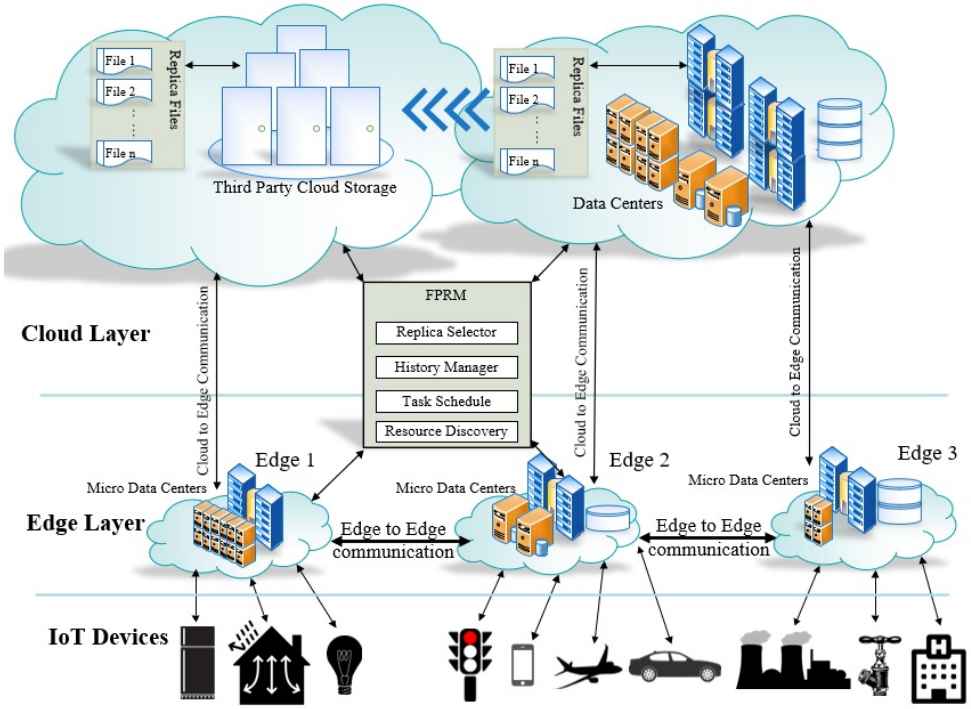

Figure 3. FPRM in Cloud, Edge and IoT Architectures.

The RM and PM were implemented in the simulation environment and tested. The replication algorithm organizes the data into a master file and replica files. The algorithm also permits users' resources to replicate copies of data by moving popular data closer to potential users. When a request for data is generated from a user's task, a replica copy will be accessed instead of accessing the origin copy [8,11]; this is referred to as the replication model. Users/IoT devices have the potential to access a large number or some of the cloud and edge resources and is able to submit tasks to edge resources and expect a response in a reasonable time (minimum time). The proposed FPRM algorithm in this paper is suitable for large-scale data-intensive problems, such as those that arise in physics experiments and medical images that generate petabytes of scientific data. The results show that the proposed approach reduced the total task execution time by 30.45% and 16.34% compared to RM and PM, respectively., in addition to the bandwidth consumption, which shows better improvement in the proposed model.

4.3. Edge-Side Replica Prediction and Selection

Due to the huge amount of data generated in IoT and edge environments and the fact that the storage resources are distributed widely in geographical terms, the result is large data traffic produced at both the IoT layer and the network's edge layer, which leads to an increase in time and resource consumption [44]. Using a replication system in edge computing will overcome the constraints of bandwidth and resources of the modem centralized cloud system [1]. Moreover, the replication technique in IoT has shown performance improvement of storage sites through load balancing [10]. As such, the set of distributed replication nodes must be selected to improve network performance; this is achieved through the replica node discovery and selection process described below. Lately, with today's increase in the number of cloud services, smartphones with limited specifications build the core elements of edge computing; therefore, many organizations have to deal with limited resources. There are several cloud services that are designed to facilitate and support cloud applications, e.g., replication services [2], IoT nodes are becoming both service providers and subscribers simultaneously. Hence, a client of replica nodes can advertise themselves as both service providers and subscribers.

In the proposed model, the task execution history is a very useful tool in predicting many aspects of resource performance, e.g., run time, queuing times and data transfer delays. Thus, for clients to participate in the replication sharing process, a node must advertise itself to a nearby cloudlet [1] while the service subscriber/provider provides its own specs (e.g., processing power, available memory, etc.). Once the list of candidate replication nodes is available, the replica node selection process is initiated. As shown in the following algorithm for predicting the replica location, when a user/IoT device submits a task that requires a file/replica, the prediction model will run first to predict the file/replica's location. If it is found, the location will be sent to the task schedule to execute the file. If the prediction fails to predict the file location, then the task manager will search and get the list of the available resources for a storage node that can provide the replica from the replica selector. Then the Task Manager will send the replica location to the Task Schedule to get the replica and execute the task.

During the process of predictions, the run time of new tasks on a candidate resource is calculated faster in time. This early availability of probable run time facilitates the overall task management. Previous research in the areas of predictions has found that predictions can be made by using certain task parameters, e.g., user id, group id, executable and degree of parallelism [8,9]. These parameters are actually used to identify similar file/replica's location to the new task from execution logs. The parameters that are used to find similar tasks can also be referred to as features. These features form a set called the feature set. After a task is completed, the feature set is stored in the history database along with other execution parameters, i.e., memory used, processors used, total run time and wait time. For a new task, this history database is searched, and files with a similar feature set are used to make predictions of the replica's location for the incoming tasks on certain resources.

The size of the history relates to the maximum number of records for a certain cluster of tasks. Each cluster is based on six parameters, as described in a previous section. If for a certain cluster, the size of the history is very high, i.e., beyond the maximum size (thirty according to the central limit theorem), then the oldest record is deleted, as maintaining a larger history size might result in outdated data that are no longer valid. On the other hand, a smaller history might result in data that are not truly representative. The execution run time estimate is a random process and requires a certain minimum number of trials before a stable value is reached.

The History Manager is responsible for keeping the history data up to date. It is interfaced with the Task Scheduler and history database. History updates are performed by using the history management algorithm in Figure 4. The execution records are managed and controlled by the History Manager. The finished task's logs are stored in a database as soon as they are received; these logs are managed by the history management algorithm. After the new task has just been stored, a query is sent to the history database to check the size of the cluster related to this task, the oldest member of this cluster will be deleted if the size of the cluster is larger than the maximum history limit. Otherwise, no action will be taken if the size of the cluster is equal to or less than the limit. The history management algorithm is responsible for keeping the history logs up to date, and it works separately. Therefore, for every successful execution, the older logs are removed, and the new logs are stored. After the task is completed, the history management algorithm can be executed offline. Consequently, its complexity is *O(1)*. Subsequently, it is not contributing to the overall delay as its speed is not very critical to the overall execution time. On the other hand, the prediction algorithm contributes to the total delay in the execution time.

The degree of similarity is measured among old tasks, and the prediction of a file's location is based only on the most similar tasks. This phenomenon, in mathematical language, is called data clustering. Data clustering forms the basis of the prediction model and is presented in the coming sections, as shown in Figure 5.

Data: workload history
Input: execution log of completed task
Input: task submission description file
Result: history updated
initialization;
store record to the history database;
if size of this cluster is more then the size limit **then**
    delete the oldest record;
end

Figure 4. History management algorithm.

Replica Prediction Algorithm:

Data: user preferences and replicas log history
Input: available resource list
Input: new task submission description file
Result: run time prediction for the replica location
Start
While Taskset <> empty for all required replicas **do**
    select next replica location;
    find cluster using six parameters;
    calculate mean replica location for this cluster;
    calculate standard deviation for the run time for this cluster;
    add mean and standard deviation to this replica location;
    add replica location to the list;
    process request for Task(s) k_i;
    NN predict required replica location for k_i;
    send back predicted replica location to execute k_i;
    update Task Manager log history;
end

Figure 5. Replica location prediction algorithm.

4.4. Clustering

Data clustering is used to identify a structure in unlabeled datasets by organizing data into homogeneous groups in such a way that between groups, the similarity is minimal, and inside a certain group, the similarity is maximal [45,46]. Clustering is useful in many fields, such as data mining, document retrieval, image segmentation and pattern classification. Likewise, the term clustering is used in several research communities that have different terminologies and assumptions for the components of the clustering process and the context in which clustering is used [47]. In the replication management paradigm, clustering is used to identify files of similar characteristics [47–50]. Clustering cannot be applied blindly

on all attributes of workloads. In such cases of clustering, it is possible that attributes do not form a good cluster with each other. For example, the scatter plots of parallel task sizes and run times do not show clustering behavior. Clustering is applicable when workload distributions are Modal; this is normally the case when a workload is coming from the same user. As mentioned above, users tend to do the same work again and again and request the same files; in such situations, it is possible to apply clustering to the workload and the requested data coming from a single user, which is used to update the proposed file profile and, consequently, will be used to predict its future behavior. To assess whether clustering is applicable, it is validated. Simple validation is performed by evaluating the similarity among the members of a certain cluster [49]. Clustering can only be applied when enough historical information is available. For a user submitting new tasks that require a new file, it is possible that enough historical data of the file is not available. In such cases, alternate prediction strategies are required to be used. These alternate strategies are file supplied predictions, random predictions or static predictions. Predictions can also be generated by applying different clustering levels. A clustering level is defined as the number of attributes being used for predictions. The proposed FPRM is based on six attributes, but in the absence of enough data, clustering can be carried out by using fewer attributes. This will decrease the accuracy but, as a first-time penalty, may be acceptable.

Clustering is a multi-step process; the first step is the pattern representation that involves the recognition number of classes in a data sample, the number of available patterns, type and scale of features available to the clustering algorithm. During this process, patterns are reduced by removing less-useful patterns and/or merged to produce the most useful classes of patterns. The next step is pattern proximity, which measures the distance between a pair of patterns. A simple distance measure such as the Euclidean distance can often be used to measure proximity. The grouping or clustering step involves the merging of closer lying groups or further dividing into subgroups if the proximity is higher within a group.

4.4.1. Mathematical Notations for Clustering

Below are the most important terms and notations used for clustering [51].

- A feature vector X is a single data item used by the clustering algorithms. It is a d dimensional vector consisting of d measurements: $X = (x_1, x_2, x_3, \ldots x_d)$. In the case of workloads, these measurements can represent values of different parameters.
- The scalar components x_i of a feature vector X are called individual features or attributes, e.g., file id, resource id and creation time. Dimension d is the length of a feature vector and represents the total number of features making up the feature vector space.
- A pattern set is denoted by $\Psi = X_1, X_2, X, \ldots X_n$. The i_{th} pattern vector of this pattern set Ψ is denoted by $X_i = x_{(i,1)}, x_{(i,2)}, x_{(i,3)}, \ldots x_{(i,d)}$. It can be seen that the pattern set to be clustered can be shown as an $n \times d$ matrix.
- The files can be located in local or remote resources.
- A distance measure is a special metric calculated for a feature space and is used to quantify the similarity of different patterns. It will be explained in detail in the coming sections.

4.4.2. Feature Selection

A pattern can represent different physical objects found in a real-life situation. As has been mentioned earlier, patterns are represented by multidimensional vectors. Each dimension of these vectors represents a feature in a real-life scenario. These features can be either quantitative or qualitative. For example, if file id and resource id are two features, then these can be represented by a vector X (Test22, 225). Where Test22 is the qualitative measurement for the executable and 225 is the quantitative measurement of resource id [51].

It is very important that only the most descriptive and discriminatory features in the input set are selected and used for subsequent cluster analysis. A feature selection

technique identifies a subset of existing features for cluster analysis. It might involve a trial and error process where various subsets of features are selected, and then the output is evaluated using certain validity criteria. An example of features in the context of task submission is the attributes of a task. Examples of these attributes are storage id, file id, submission time, run time, memory required, processors required, time required, file creation time, file owner, etc. Six parameters were selected to be used for clustering after performing a validity criteria test, which is also referred to as a similarity analysis. It was found that similarity was maximal when clustering was performed using all these parameters. The six parameters, as mentioned previously, are name, owner name, attribute size, file size, resource id, creation time and transaction time.

4.4.3. Similarity Measure

Similarity is an extremely important part of the definition of a cluster. A measure of the similarity between two patterns drawn from the same feature space is essential to all clustering procedures. Since many different types of features are used, the similarity measure between them must be chosen very carefully. It is common practice to calculate the dissimilarity between two patterns by using a distance measure defined on the feature space. A very common metric is the *Euclidean Distance (ED)* [51,52]. *ED* is a distance between two points $X(x_1, x_2, x_3, \ldots x_n)$ and $Y(y_1, y_2, y_3, \ldots y_n)$, in Euclidean n-space, and is defined in the Equation (1).

$$ED = \sqrt{\sum_{i=1}^{n} x_i - y_i} \tag{1}$$

ED is commonly used to evaluate the proximity of objects in two or three-dimensional space. The drawback in using *ED* directly is that the largest scaled feature will dominate the results; this can be avoided by using normalization or other weighting techniques. When the average distance is measured with respect to the mean point of n data points, this is called *Standard Deviation (SD)*. In probability and statistics, the *SD* is a measure of the dispersion between a set of values. *SD* is defined below in Equation (2).

$$SD = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2} \tag{2}$$

Where the mean for n data points $(x_1, x_2, x_3, \ldots x_n)$ is shown in Equation (3).

$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3}$$

Mean and *SD* are dimensional quantities. *SD*, such as the *ED*, measures the spread in data and hence can also be used to measure similarity or dissimilarity. The greater the spread means less similarity and vice versa. The *SD* can be converted to a dimensionless metric by dividing it by the mean. This will result in another statistical measure called the *Coefficient of Variation (CV)*. The *CV* is another statistical measure of dispersion in data, and its mathematical representation is shown in Equation (4).

$$CV = \frac{SD}{\overline{X}} \tag{4}$$

The *CV* is useful because the *SD* of data must always be understood in the context of the mean of the data. The *CV* is a dimensionless number that makes comparison easier among the values with different units. The *CV* is also called a normalized measure of dispersion; this makes the *CV* a very useful statistical parameter to measure dispersion among clustered tasks. The resulting set of nodes that are willing to perform replication while satisfying the minimum QoS level requirements will be denoted as set n. While applied to workload data, this technique of clustering separates related tasks that require the same data. Once related tasks are found, the next step is to make replica location

predictions using these tasks. Since all tasks in the selected cluster are related to the new task, it is expected that the new task will also have a replica location similar to the clustered tasks. If the replica location of the tasks in a cluster is the same, then it is very easy to assume that the new task will also require the same replica(s), but it is observed that normally it is not the case. The replica location varies from time to time.

This situation requires a different strategy to determine the replica location. One possible strategy is to calculate the mean of clustered tasks named as a mean predictor. Other possibilities are last observation, median, mode, low pass filter, k-nearest neighbors, weighted average and regression analysis. The mean predictor has been used by different researchers and has proved to be reasonably accurate [53–55]. Different types of predictors: a mean, a linear regression, an inverse regression and a logarithmic regression, were evaluated by these authors; their work also found that the mean is the single best predictor. Regression-based predictors were also considered before choosing a mean predictor. Predictions generated by regression predictors are not stable and also have a higher computational overhead.

The benefit of a mean predictor is that it is easy to calculate and has very little overhead; hence it was decided to use a mean predictor for this model. One of the initial targets of the proposed model is to minimize prediction overhead. With complex prediction strategies, the risk is that the computational overhead would increase with little gain in the prediction accuracy if any. A mean predictor was successful in providing reasonably accurate predictions at low computational overhead. Generated predictions can be utilized directly or can be used to calculate the expected node to provide a replica within a certain level of confidence. Sometimes, in critical applications, the user/IoT device is more interested in the replica's location than in just a prediction. Keeping this in mind, the evaluation of the prediction model was carried out for both predictions using file profile parameters and a replica management system.

The maximum run time can be calculated by assuming that run times within a cluster follow a normal distribution. A normal distribution describes data that are clustered around a mean. Equation (5) describes the maximum run time limit at a certain confidence interval. This confidence interval is dependent on the value of "n". Predictions will be generated by calculating the mean execution time for these clusters of tasks. Clusters will be defined on the basis of the six parameters. The CV value will be used to calculate the maximum run time for a certain confidence interval using Equation (5). Calculations were performed for 90% and 95% confidence intervals. CV has already been used in previous work to gain an understanding of the distribution of task run times. In [56], CV was calculated for the task admission control scheme based on a random task filtering policy. The authors in [57] carried out a more elaborate CV analysis of tasks providing results on the basis of seven parameters. The work in this paper suggested that the value of CV is lower when clustering is performed using six parameters. Comparisons have shown that the value of CV is indeed lower when clustering is carried out using all six parameters at a time, as shown in Equation (5).

$$T_{PRT} = T_{MRT} + n \times CV \times \overline{X} \tag{5}$$

where,

$CV * \overline{X} = StandardDeviation$
T_{PRT} = Prediction Run Time
T_{MRT} = Mean Run Time
CV = Coefficient of Variation
n = Real number

As has been shown above, the CV provides a quantitative measure of dispersion for a certain cluster of tasks. In simple words, if for a certain group of previous executions, the CV is less for the run time of tasks, then predictions made using that cluster of tasks will show lower variation, i.e., higher similarity. In other words, the CV is a measure of the

goodness or quality of predictions. Therefore, we can say effectively that small values of *CV* will result in better predictions; on the other hand, higher values of *CV* for a certain cluster will result in less reliable predictions. It is clear that if clusters of tasks are identified with small values of *CV*, then predictions will show higher accuracy. Also, predictions are most likely to have small variations. Experiments were performed with different workload data, and it was found that the value of *CV* is minimal when clusters are defined using the six parameters mentioned. Figure 4 shows the normal distribution curves drawn showing the probability density function on the *y*-axis and the value of the mean on the *x*-axis for different values of *SD*. All five series drawn in Figure 4 are further explained in Table 1. It can be seen from Table 1 and Figure 6 that for high values of *CV*, the bell curve is wide, showing a higher spread in the data, and for lower values of *CV*, the bell curve is narrow.

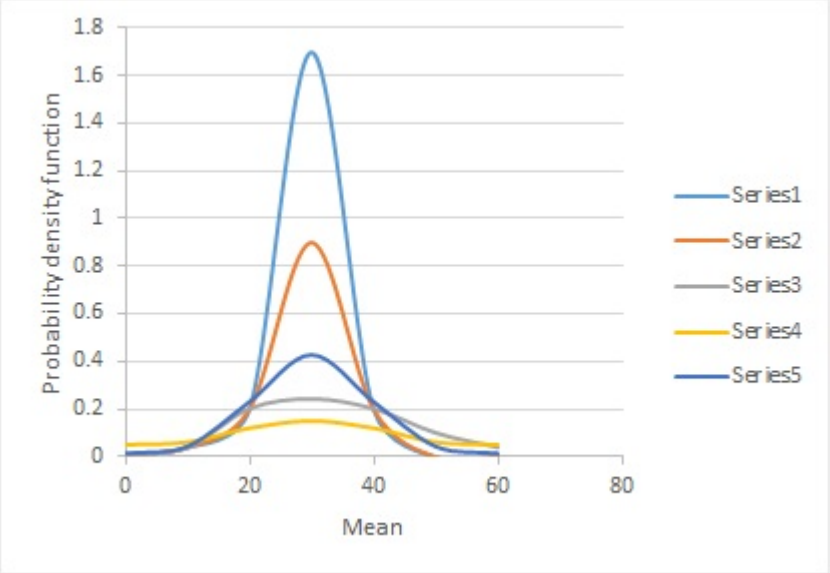

Figure 6. Normal distribution curves drawn for different values of CV.

Table 1. Value of Mean, Standard Deviation and Coefficient of Variation (CV) for different series.

Series	Mean	Standard Deviation	Coefficient of Variation (CV)
Series1	24	2	0.08
Series2	24	4	0.16
Series3	24	8	0.32
Series4	24	16	0.64
Series5	24	32	1.28

It suggests that a lower value of *CV* will result in a high probability of results lying within close vicinity to each other, which confirms the above assumption that the lower values of *CV* will result in better predictions since there is little spread in the data. The data presented in Table 1 and Figure 6 were generated by using Equation (6).

$$Data = (Random.nextGaussian() \times SD) + Mean \quad (6)$$

where $Random.nextGaussian()$ generates normally distributed random variables with mean 0.0 and *SD* 1.0 [58]. Therefore, in short, for every new task that requires certain files, a set of previously completed tasks (clusters) will be identified that belong to the same cluster as the new task.

5. Simulation Testbed and Results

5.1. Simulation Testbed

The modeled testbed contains 30 storage resources spread across distributed locations connected via high-capacity network links; the number of resources is expanded up to fifty resources as explained in the system workload in a large-scale environment. The first set of experiments started with 30 storage resources as a base test bed. All distributed resources were simulated as clusters, and each cluster is associated with processing elements (PEs) or single CPU nodes with a batch task management system. The PEs capabilities are defined in terms of Million Instructions Per Second (MIPS). The total disk capacity available at each site will form the storage at the resources. The network between the resources was modeled as a set of routers and links. The databases were defined as several files distributed across the resources. A number of files were defined with an average file size up to terabytes. A total of 100 users/IoT devices will be simulated in the first experiment on the first test bench; after that, the number will be increased up to 500 users, where each resource is assigned a certain number of users. A user will submit task(s) that requires one or more files to be executed (predetermined files). The next section provides a high-level overview of the relationship between users, resources and files. When a new replica is created for the first time, then a file profile will be created. File profiles identify the file name and the user's local system (speed, storage volume, networks used and their speed, transaction time, etc.). This profile will also retain a record of services/resources used by this client. Each database/file needs a description of file type, size, storage media, network connection, speed, location, etc. The file interface presents an environment in which users will be able to access standard file maintenance operations, such as creating, deleting, copying and editing files, in addition to performing file/database queries. Each user/IoT device has an agent that acts for the user and watches for changes in the system, e.g., (new data resources, new services and updates the file profile accordingly); the cloud information services and the replica service plays this important role.

The below factors were taken into consideration to create a realistic simulation environment:

- The storage system: The storage system has been implemented to simulate the behavior of typical hardware storage. A simple interface that can be used to simulate storage and the retrieval of any amount of data. Accessing files in a SAN at run-time incurs additional delays for task unit execution; this is due to the additional latency that is incurred in transferring the data files through the data center's internal network.
- Cloudlet: The Cloudlet (cloud task) is represented in CloudSim as a package that holds all the execution details and information of the task (i.e., the size of input and output files, the task owner id and task length expressed in Millions Instruction (MI)). The time required to transfer input and output files between user/IoT and remote resources, then return the results to the owner, is the most important factor that helps to determine the execution time.
- Cloud Resource: The cloud resource has been simulated as a resource with properties as explained below:
  - PEs (Processing Elements) have been implemented that objects with a MIPS (Million Instructions Per Second) rating, which represents the CPU speed. The PEs were assembled together to create a machine.
  - Objects of the machine were grouped to form a cloud resource.
  - CloudSim PEList: The CloudSim PEList maintained a list of PEs that make up a machine.

As mentioned above, the simulation environment setup consists of 100 users (i.e., service subscribers), in which each user submits up to 10 tasks; each task requests 1 to 5 files, and each file is in the range of 2.5–20 GB. We assumed the presence of 30 storage sites. A detailed summary of the network configurations is outlined in Table 2.

Table 2. Simulation Environment Configurations.

Experiments Parameters	Values/Ranges
Number of users	100–500
Number of tasks per user	2–10
Total tasks	1000–10,000
Number of files accessed per task	5
Total files	2500
Size of single file	2.5–20 GB
Total size of files	50,000 GB
Number of sites	30
Available storage of each site	100 GB–1 TB
Task delay	2500 ms
Bandwidth	100–1000 MB

In present and future distributed applications, the number of users and resources continues to increase. Consequently, the number of tasks and file requests will keep increasing, and the network traffic will be very heavy. Therefore, in the following section, the performance of the proposed model, PM and RM will be compared when the number of users, tasks and network traffic is increased substantially.

5.2. Simulation Results

This section provides a concise description of the experimental results, their interpretation, as well as the experimental conclusions that can be drawn. In an ideal world, the simulation configuration should be close to reality. In a typical real-world cloud environment, tasks are submitted to the resource broker by the users, and the resource broker then finds the best site to run the task. Typically, the tasks under execution will require one or more files, as mentioned in the previous sections. It is the responsibility of the replica manager to locate the required files for these tasks. The current replication system PM, RM and the proposed FPRM have been implemented by expanding the number of users, resources and files to provide greater complexity. The performance of the current and proposed system was evaluated in five different scenarios by varying the number of tasks submitted by the user/IoT device. The system's performance was evaluated in each scenario for 1000 up to 10,000 tasks. The same number of users, resources, files and tasks were implemented for both the current and proposed systems. Figure 7 shows the sample row results for the FPRM, PM and RM models, respectively.

The results of the simulation show that the Task Turnaround Time (TTT) for the proposed FPRM system is less than the TTT of PM and RM by 16.34% and 30.45%, respectively, as an average for all scenarios, as shown in Table 3 and Figure 7. The results show better performance than the PM due to more parameters and information related to the file/replica used in the prediction model instead of the user profile.

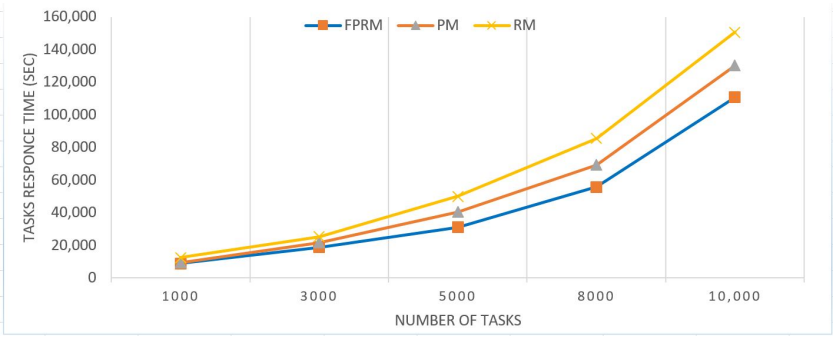

Figure 7. Number of tasks versus response time for , FPRM, PM and RM.

Table 3. Simulation results of TTT in different scenarios.

# of Tasks	FPRM/sec	PM/sec	RM/sec
1000	8900	9500	12,540
3000	18,865	21,608	25,243
5000	30,988	40,512	50,020
8000	55,796	69,214	85,600
10,000	110,796	130,214	150,600

The network bandwidth affects the task execution time, in addition, to file transfer time as it defines the exchanging of data between resources (both downlink and uplink). Therefore, bandwidth is an important factor, and it was tested in different scenarios by varying the bandwidth from 150 to 1000 MB. The result shows an average improvement of the proposed FPRM by 24.7% and 49.4% compared to the PM and RM, respectively; the proposed model outperforms these models in all scenarios, especially with narrow bandwidth; even with larger bandwidth, the proposed model still provided significant performance, as shown below in Figure 8.

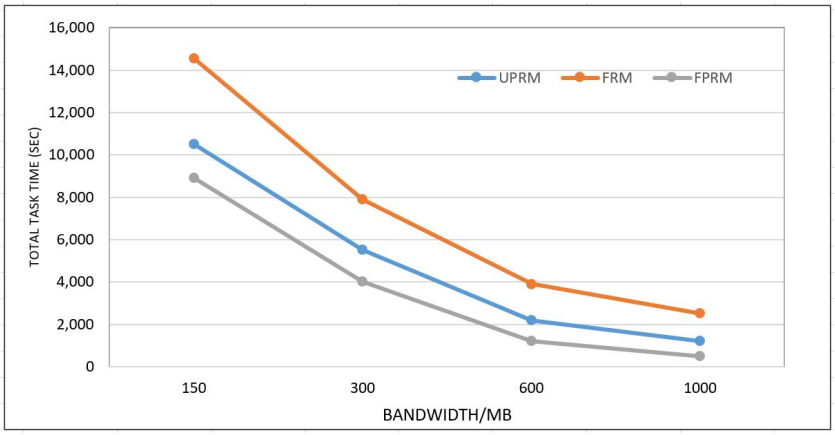

Figure 8. Bandwidth consumption for FPRM, PM and RM.

The proposed prediction model will provide run-time predictions, which provide estimates of the processing time. Tasks are executed in four stages, which start with the task submission, waiting in the batch queue (waiting for required files/replicas), run time and end with the result's retrieval. The task will spend some time in the queue until all required files/replicas are ready for execution. Given that there are delays during the start of each stage of execution, the resource management system must also spend some time preparing tasks for execution. An example of one such delay is prediction time (algorithm running time) of the replica manager to find the location of the required replicas. Queuing time is a very important parameter because it has a strong effect on the total turnaround time, as shown in Table 3. To evaluate the proposed algorithm run time, the below experiment was carried out. The results presented in Table 4 show that the proposed algorithm outperforms the PRM and RM models, which affects the TTT.

Table 4. Simulation results of the proposed algorithm's run time in different scenarios.

# of Tasks	FPRM/Milsec	PM/Milsec	RM/Milsec
1000	33,000	59,000	98,000
3000	105,000	189,000	312,000
5000	190,000	375,000	965,000
8000	318,000	796,000	1,628,000
10,000	420,000	940,000	2,230,000

6. Conclusions and Future Work

This paper proposed a prediction model responsible for predicting the location of files to overcome the shortcoming of the increasing number of users and devices in edge and IoT systems, which, in turn, increases the number of tasks. The proposed approach uses a neural networks tool to predict a file's location accurately based on the characteristics taken from CloudSim using a data clustering technique. If the complete task query is fulfilled by the prediction model, then no further processing search will take place, and the result is handed over immediately to the replica management service. If the location of files is not available in the prediction model, the remaining query will then be handed over to the complete search approach. The result shows a decrease in the TTT using the FPRM by 16.34% and 30.45%. In addition, the proposed model outperforms the PM and the RM in bandwidth consumption by 24.7% and 49.4%. In future work, more experimental evaluations will be carried out to evaluate the impact of the file's size on the response time, in addition to the number of replicated files on the storage space utilization.

Author Contributions: Math modeling, N.M. and W.H.F.A.; methodology, N.M. and W.H.F.A.; simulation, N.M. and Z.A.-A.; investigation, N.M. and Z.A.-A.; data acquisition, N.M. and Z.A.-A.; writing the introduction, N.M. and S.A.; formal analysis, N.M. and S.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shao, Z.L.; Huang, C.; Li, H. Replica selection and placement techniques on the IoT and edge computing: A deep study. *Wirel. Netw.* **2021**, *27*, 5039–5055. [CrossRef]
2. Qin, J.; Liang, S.; Song, Y.; Zong, P. Study on Replica Strategy of Big Data Storage based on Cloud Environment. In Proceedings of the 2020 15th International Conference on Computer Science & Education (ICCSE), Delft, The Netherlands, 18–22 August 2020; pp. 642–645.
3. Ali, M.; Bilal, K.; Khan, S.U.; Veeravalli, B.; Li, K.; Zomaya, A.Y. DROPS: Division and replication of data in cloud for optimal performance and security. *IEEE Trans. Cloud Comput.* **2015**, *6*, 303–315. [CrossRef]
4. Mostafa, N. Cooperative Fog Communications using A Multi-Level Load Balancing. In Proceedings of the 2019 Fourth International Conference on Fog and Mobile Edge Computing (FMEC), Rome, Italy, 10–13 June 2019; pp. 45–51. [CrossRef]
5. Sadiku, M.N.; Musa, S.M.; Momoh, O.D. Cloud computing: Opportunities and challenges. *IEEE Potentials* **2014**, *33*, 34–36. [CrossRef]
6. Li, C.; Tang, J.; Luo, Y. Scalable replica selection based on node service capability for improving data access performance in edge computing environment. *J. Supercomput.* **2019**, *75*, 7209–7243. [CrossRef]
7. Yang, C.; Huang, Q.; Li, Z.; Liu, K.; Hu, F. Big Data and cloud computing: Innovation opportunities and challenges. *Int. J. Digit. Earth* **2017**, *10*, 13–53. [CrossRef]
8. Mostafa, N.; Al Ridhawi, I.; Hamza, A. An intelligent dynamic replica selection model within grid systems. In Proceedings of the 2015 IEEE 8th GCC Conference & Exhibition, Muscat, Oman, 1–4 February 2015; pp. 1–6.
9. Al Ridhawi, I.; Mostafa, N.; Masri, W. Location-aware data replication in cloud computing systems. In Proceedings of the 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Abu Dhabi, United Arab Emirates, 19–21 October 2015; pp. 20–27.
10. Sun, S.; Yao, W.; Qiao, B.; Zong, M.; He, X.; Li, X. RRSD: A file replication method for ensuring data reliability and reducing storage consumption in a dynamic Cloud-P2P environment. *Future Gener. Comput. Syst.* **2019**, *100*, 844–858. [CrossRef]
11. Kumar, K.A.; Quamar, A.; Deshpande, A.; Khuller, S. SWORD: Workload-aware data placement and replica selection for cloud data management systems. *VLDB J.* **2014**, *23*, 845–870. [CrossRef]
12. Li, Z.; Cai, W.; Turner, S.J. Un-identical federate replication structure for improving performance of HLA-based simulations. *Simul. Model. Pract. Theory* **2014**, *48*, 112–128. [CrossRef]
13. Liu, R.; Feng, S.; Sun, S.; Liu, M. Edge node data replica management method for distribution Internet of Things. In Proceedings of the 2020 4th International Conference on HVDC (HVDC), Xi'an, China, 6–9 November 2020; pp. 830–832.
14. Wei, J.; Yi, M.; Song, L. Efficient Integrity Verification of Replicated Data in Cloud Computing System. *Comput. Secur.* **2016**, *65*. [CrossRef]
15. oi: 10.1002/cpe.3573 Zhang, Y.; Ni, J.; Tao, X.; Wang, Y.; Yu, Y. Provable multiple replication data possession with full dynamics for secure cloud storage. *Concurr. Comput. Pract. Exp.* **2016**, *28*, 1161–1173. [CrossRef]

16. Lakshman, A.; Malik, P. Cassandra: A decentralized structured storage system. *ACM SIGOPS Oper. Syst. Rev.* **2010**, *44*, 35–40. [CrossRef]
17. Suresh, P.L.; Canini, M.; Schmid, S.; Feldmann, A. C3: Cutting Tail Latency in Cloud Data Stores via Adaptive Replica Selection. In Proceedings of the 12th USENIX Symposium on Networked Systems Design and Implementation, Oakland, CA USA, 4–6 May 2015.
18. Li, C.; Zhang, Y.; Luo, Y. Adaptive replica creation and selection strategies for latency-aware application in collaborative edge-cloud system. *Comput. J.* **2020**, *63*, 1338–1354. [CrossRef]
19. Muthu, T.S.; Pandiaraj, S. Data Grid Optimization using Replica Replacement. In Proceedings of the 2019 6th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 13–15 March 2019; pp. 410–414.
20. Hassan, N.; Gillani, S.; Ahmed, E.; Yaqoob, I.; Imran, M. The role of edge computing in internet of things. *IEEE Commun. Mag.* **2018**, *56*, 110–115. [CrossRef]
21. Saranya, N.; Geetha, K.; Rajan, C. Data replication in mobile edge computing systems to reduce latency in internet of things. *Wirel. Pers. Commun.* **2020**, *112*, 2643–2662. [CrossRef]
22. Gill, N.K.; Singh, S. Dynamic cost-aware re-replication and rebalancing strategy in cloud system. In Proceedings of the 3rd International Conference on Frontiers of Intelligent Computing: Theory and Applications (FICTA), Bhubaneswar, India, 14–15 November, 2014; Springer: Berlin/Heidelberg, Germany, 2015; pp. 39–47.
23. Junfeng, T.; Weiping, L. Pheromone-based genetic algorithm adaptive selection algorithm in cloud storage. *Int. J. Grid Distrib. Comput.* **2016**, *9*, 269–278. [CrossRef]
24. Wakil, K.; Nazif, H.; Panahi, S.; Abnoosian, K.; Sheikhi, S. Method for replica selection in the Internet of Things using a hybrid optimisation algorithm. *IET Commun.* **2019**, *13*, 2820–2826. [CrossRef]
25. Waheed, A.; Shah, M.A.; Mohsin, S.M.; Khan, A.; Maple, C.; Aslam, S.; Shamshirband, S. A Comprehensive Review of Computing Paradigms, Enabling Computation Offloading and Task Execution in Vehicular Networks. *IEEE Access* **2022**, *10*, 3580–3600. [CrossRef]
26. Lin, Z.; Niu, H.; An, K.; Wang, Y.; Zheng, G.; Chatzinotas, S.; Hu, Y. Refracting RIS Aided Hybrid Satellite-Terrestrial Relay Networks: Joint Beamforming Design and Optimization. *IEEE Trans. Aerosp. Electron. Syst.* **2022**, *58*, 3717–3724. [CrossRef]
27. Lin, Z.; Lin, M.; Wang, J.B.; de Cola, T.; Wang, J. Joint Beamforming and Power Allocation for Satellite-Terrestrial Integrated Networks With Non-Orthogonal Multiple Access. *IEEE J. Sel. Top. Signal Process.* **2019**, *13*, 657–670. [CrossRef]
28. Lin, Z.; Lin, M.; de Cola, T.; Wang, J.B.; Zhu, W.P.; Cheng, J. Supporting IoT With Rate-Splitting Multiple Access in Satellite and Aerial-Integrated Networks. *IEEE Internet Things J.* **2021**, *8*, 11123–11134. [CrossRef]
29. Zhang, J.; Su, Q.; Tang, B.; Wang, C.; Li, Y. DPSNet: Multitask Learning Using Geometry Reasoning for Scene Depth and Semantics. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, 1–12. [CrossRef] [PubMed]
30. Calheiros, R.N.; Ranjan, R.; De Rose, C.A.; Buyya, R. Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services. *arXiv* **2009**, arXiv:0903.2525.
31. Mansouri, N. Network and data location aware approach for simultaneous job scheduling and data replication in large-scale data grid environments. *Front. Comput. Sci.* **2014**, *8*, 391–408. [CrossRef]
32. Spillner, J.; Gkikopoulos, P.; Buzachis, A.; Villari, M. Rule-Based Resource Matchmaking for Composite Application Deployments across IoT-Fog-Cloud Continuums. In Proceedings of the 2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC), Leicester, UK, 7–10 December 2020; pp. 336–341. [CrossRef]
33. Rajalakshmi, A.; Vijayakumar, D.; Srinivasagan, K.G. An improved dynamic data replica selection and placement in cloud. In Proceedings of the 2014 International Conference on Recent Trends in Information Technology, Chennai, India, 10–12 April 2014; pp. 1–6. [CrossRef]
34. Kapgate, D. Efficient Service Broker Algorithm for Data Center Selection in Cloud Computing. *Int. J. Comput. Sci. Mob. Comput.* **2014**, *3*, 355–365.
35. Aslam, S.; Herodotou, H.; Mohsin, S.M.; Javaid, N.; Ashraf, N.; Aslam, S. A survey on deep learning methods for power load and renewable energy forecasting in smart microgrids. *Renew. Sustain. Energy Rev.* **2021**, *144*, 110992. [CrossRef]
36. Jones, M. *Artificial Intelligence: A Systems Approach*; Computer science Series; Infinity Science Press: Hingham, MA, USA, 2008.
37. Yarali, A. Artificial Intelligence, 5G, and IoT. In *Intelligent Connectivity: AI, IoT, and 5G*; John Wiley & Sons, Inc.: Hoboken, NJ, USA, 2022; pp. 251–268. [CrossRef]
38. Mostafa, N.; Ridhawi, I.A.; Aloqaily, M. Fog resource selection using historical executions. In Proceedings of the 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC), Barcelona, Spain, 23–26 April 2018; pp. 272–276. [CrossRef]
39. Long, Y.; Rong, J. Research on Model of Seismic Anomaly Data Mining Based on Neural Network. In Proceedings of the 2021 IEEE 4th International Conference on Information Systems and Computer Aided Education (ICISCAE), Dalian, China, 24–26 September 2021; pp. 518–522. [CrossRef]
40. Bui, T.D.; Nguyen, D.K.; Ngo, T.D. Supervising an Unsupervised Neural Network. In Proceedings of the 2009 First Asian Conference on Intelligent Information and Database Systems, Dong Hoi, Vietnam, 1–3 April 2009; pp. 307–312. [CrossRef]
41. Shanthi, D.; Sahoo, G.; Saravanan, S. Designing an Artificial Neural Network Model for the Prediction of Thromboembolic Stroke. *Int. Journals Biom. Bioinform.* **2004**, *3*, 10–18.

42. Application: JustNN Help—justnn.com. Available online: http://www.justnn.com/application/JustNN.htm (accessed on 21 July 2022).
43. Mostafa, N. A dynamic approach for consistency service in cloud and fog environment. In Proceedings of the 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 20–23 April 2020; pp. 28–33.
44. Chen, X.; Tang, S.; Lu, Z.; Wu, J.; Duan, Y.; Huang, S.C.; Tang, Q. iDiSC: A new approach to IoT-data-intensive service components deployment in edge-cloud-hybrid system. *IEEE Access* **2019**, *7*, 59172–59184. [CrossRef]
45. Huang, D.; Wang, C.D.; Peng, H.; Lai, J.; Kwoh, C.K. Enhanced ensemble clustering via fast propagation of cluster-wise similarities. *IEEE Trans. Syst. Man, Cybern. Syst.* **2018**, *51*, 508–520. [CrossRef]
46. Wong, A.K.; Li, G.C. Simultaneous pattern and data clustering for pattern cluster analysis. *IEEE Trans. Knowl. Data Eng.* **2008**, *20*, 911–923. [CrossRef]
47. Tian, Y.; Zheng, R.; Liang, Z.; Li, S.; Wu, F.X.; Li, M. A data-driven clustering recommendation method for single-cell RNA-sequencing data. *Tsinghua Sci. Technol.* **2021**, *26*, 772–789. [CrossRef]
48. Tang, R.; Li, P. Index optimization replication algorithm by using the soft subspace clustering method. In Proceedings of the 2014 IEEE 7th Joint International Information Technology and Artificial Intelligence Conference, Chongqing, China, 20–21 December 2014; pp. 414–418.
49. Gkatzikis, L.; Sourlas, V.; Fischione, C.; Koutsopoulos, I.; Dán, G. Clustered content replication for hierarchical content delivery networks. In Proceedings of the 2015 IEEE International Conference on Communications (ICC), London, UK, 8–12 June 2015; pp. 5872–5877.
50. Nguyen, D.N.; Tran, X.H.; Nguyen, H.S. A cluster-based file replication scheme for DHT-based file backup systems. In Proceedings of the 2016 International Conference on Advanced Technologies for Communications (ATC), Hanoi, Vietnam, 12–14 October 2016; pp. 204–209.
51. Armano, G.; Javarone, M.A. Clustering datasets by complex networks analysis. *Complex Adapt. Syst. Model.* **2013**, *1*, 1–10. [CrossRef]
52. Faizah, N.; Fabrianto, L.; Prasetyo, R. Unbalanced data clustering with K-means and euclidean distance algorithm approach case study population and refugee data. *J. Physics Conf. Ser. Iop Publ.* **2020**, *1477*, 022005. [CrossRef]
53. Lv, M.; Wang, L.; Hou, Y.; Gao, Q.; Hou, R. Mean shift tracker with grey prediction for visual object tracking. *Can. J. Electr. Comput. Eng.* **2018**, *41*, 172–178. [CrossRef]
54. Bishnu, P.S.; Bhattacherjee, V. Software fault prediction using quad tree-based k-means clustering algorithm. *IEEE Trans. Knowl. Data Eng.* **2011**, *24*, 1146–1150. [CrossRef]
55. Li, G.; Chang, W.; Yang, H. A novel combined prediction model for monthly mean precipitation with error correction strategy. *IEEE Access* **2020**, *8*, 141432–141445. [CrossRef]
56. Khojasteh, H.; Mišić, J.; Mišić, V.B. Task filtering as a task admission control policy in cloud server pools. In Proceedings of the 2015 International Wireless Communications and Mobile Computing Conference (IWCMC), Dubrovnik, Croatia, 24–28 August 2015; pp. 727–732.
57. Wang, K.; Fang, F.; Da Costa, D.B.; Ding, Z. Sub-channel scheduling, task assignment, and power allocation for OMA-based and NOMA-based MEC systems. *IEEE Trans. Commun.* **2020**, *69*, 2692–2708. [CrossRef]
58. Oracle. Java 2 Platform SE. 2022. Available online: https://docs.oracle.com/javase/1.4.2/docs/api/index.html (accessed on 3 July 2022).