A Cuckoo Search Based Heuristic for Replicating IOT Data in Cloud Edge System

P. Sankara Rao, R. Usha Rani

Abstract*: The rapid development in information technology has rendered an increase in the data volume at a speed which is surprising. In recent times, cloud computing and the Internet of Things (IoT) have become the hottest among the topics in the industry of information technology. There are many advantages to Cloud computing such as scalability, low price, and large scale and the primary technique of the IoTs like the Radio-Frequency Identification (RFID) have been applied to a large scale. In the recent times, the users of cloud storage have been increasing to a great extent and the reason behind this was the cloud storage system bringing down the issues in maintenance and also has a low amount of storage when compared to other methods. This system provides a high degree of reliability and availability where redundancy is introduced to the systems. In the replicated systems, objects get to be copied many times and every copy resides in a different location found in distributed computing. So, replication of data has been posing some threat to the cloud storage for users and also for the providers since it has been a major challenge providing efficient storage of data. So, the work has been analysing different strategies of replication of data and have pointed out several issues that are affected by this. For the purpose of this work, replication of data has been presented by employing the Cuckoo Search (CS) and the Greedy Search. The research is proceeding in a direction to reduce the replications without any adverse effect on the reliability and the availability of data.

Index Terms*: Cloud Computing, Cloud Storage, Cuckoo Search (CS) Algorithm, Data Replication, Greedy Search and Internet of Things (IoT).

I. INTRODUCTION

The Internet of Things (IoT) is a technology connecting machines and their tools with one another through wireless technology or the Internet such as the ZigBee, the Bluetooth Low Energy (BLE), the Radio-Frequency Identification (RFID) and the Wi-Fi. These things connected to the IoT can outnumber the population of the world in the year 2011. Now, there are 9 billion things that have been connected and the number may exceed 24 billion by the end of 2020. There is plenty of data generated owing to a rapid increase to the number of things [1].

The IoTs have been presenting some widened applications and opportunities that include the smart grids for improving the reliability and the efficiency of the power supplies. There is intelligent transportation to this for optimization of traffic management and the reduction of traffic accidents, the clogged routes and the emissions of carbon dioxide. Environmental monitoring refers to the overseeing of the sources of drinking water and the urban atmosphere supervises transmission of wastes that are dangerous. They also handle e-health that accelerates and further co-ordinates medical information management, drug provision, patient care, and hospital wards. But, there are several other challenges faced by the development of things-related applications like the scalability of the end-user, storage of data, energy efficiency, variable geospatial deployment, and heterogeneous resource-constrained Things. Thus, cloud computing and Big Data play crucial roles in the storage of data and the terms of analysis [2].

Cloud computing has created another new way of designing, testing, developing, testing, running, deploying and finally maintaining the applications found on the Internet. The developer will have to take proper care of the running of operation systems, load balancing, networks, firewalls, routers, and their storage. The developer will also have to consider scalability or the manner in which the will be able to scale the users distributed geographically. Cloud computing has applied a new utility model for producing and further consuming the resources of computing wherein the cloud can abstract all computing resources that include storage and services. The user (the developer or the consumer) will be able to access the services over the Internet where cloud users pay once when they need their services. The cloud also scales to support service requests [3]. Finally, cloud computing handles the management of the micro-lifecycle of the applications and permits the managers to completely focus on the development and monitoring of the applications. The platform of cloud computing has been designed to contain various services in order to develop, test, run, deploy and maintain applications of the cloud. Some examples are the Microsoft Windows Azure Platform, the Google App Engine, and the Amazon Web Services.

Since the IoT has only a limited amount of capabilities in terms of processing power or storage there may be some consequential issues such as security, reliability, performance, privacy and so on for which integration of the IoT with the cloud is beneficial. The Cloud also benefits from this since it is able to extend its limits to more dynamic or distributed ways to deliver several services. The traits of the IoT and the cloud from various schemes that inspire the paradigm of the Cloud IoT: The IoT is pervasive (there are things placed everywhere) and are real-world things with limited capabilities of computation that have a limited or no storage.

The cloud is also ubiquitous (there are resources that are usable from everywhere), and these are virtual resources that are unlimited in terms of their capabilities of computation

Revised Manuscript Received on December 5, 2019.

***Dr. P. Sankara Rao,** Assistant Professor, Department of CSE, GIT, GITAM (deemed to be University), Visakhapatnam, AndhraPradesh, India, mail id: dr.sankararaop@gmail.com

Dr. R. Usha Rani, Associate Professor, Department of CSE, CVR College of Engineering, Ibrahimpatnum, Hyderabad mail id: teaching.usha@gmail.com

where the cloud has unlimited capabilities of storage using the Internet for delivery of service that means managing big data [4].

The cloud acts as an intermediate layer that falls between the applications and the things concealing the complexities and functionalities for processing. This framework further affects the application development in the future and also delivers some new challenges that need to be addressed. The primary advantages in the adoption of the paradigm of the Cloud IoT are [5].

Storage Resources: The IoT is by differentiation of large sums of sources of information (i.e., the things) creating plenty of semi-structured or sometimes non-structured that has three different attributes in Big Data: the velocity (frequency of data generation), volume, collection, visualizing, processing, achieving, sharing and also searching data of large volumes. This data is also treated in a homogeneous manner using the API standard and this is protected by means of applying security of top-level and is also visually accessed from anywhere.

Computational Resources: The IoT has not permitted any on-site data processing with limited capabilities of processing that is not permitted. The scalability was very challenging and there is an on-demand paradigm permitting processing capacities that are boundless in the cloud. Most of these needs of processing were satisfied and empowered analysis had complexities that were extraordinary. The predictive algorithms and the decisions that are data-driven are integrated into the IoT having a low cost and providing an increased revenue with a reduced risk. There are other tasks that may be performed in real-time processing for applying scalability that was sensor-centric in order to handle tasks that are huge and for executing offloading for the purpose of power saving [6].

Communication resources: one primary need of the IoT was to make the IP-enabled devices to communicate using dedicated hardware that has support for communication that is quite expensive. The cloud provides a cheap and effective solution for connecting, tracking and finally managing anything that can be done from anywhere. The high-speed network enables both monitoring and controlling of the things that are remote, coordinate them, communicate with them and provide real-time access.

New capabilities. The IoT has been characterized using a high heterogeneity of the devices, protocols, and technologies. Thus, the scalability, reliability, availability, scalability, efficiency and finally security which is challenging to obtain. Integration with the Cloud can solve most problems providing some more features like the ease-of-access, the ease-of-use and reduced costs of deployment [7].

New paradigms. Adopting the paradigm of CloudIoT can enable some more new scenarios that are based on an extension of the cloud by:

- The Sensing as a Service (SaaS), provides ubiquitous access to data;
- The Sensing and Actuation as a Service (SAaaS) enabling the logic of automatic control to that of the Cloud;
- The Sensor Event as a Service (SEaaS), to dispatch the messaging services that are triggered by the sensor events;
- The Sensor as a Service (SenaaS), that enables the ubiquitous remote sensor management;
- The DataBase as a Service (DBaaS), that enables the ubiquitous database management;
- The Data as a Service (DaaS), provides ubiquitous access to all data;
- The Ethernet as a Service (EaaS), provides ubiquitous layer-2 connectivity to all remote devices;
- The Identity and Policy Management as a Service (IPMaaS), that enables ubiquitous access to functionalities of identity management and policies;
- Video Surveillance as a Service (VSaaS), that provides ubiquitous access to all videos that implement a complex analysis.

Cloud storage is an extremely powerful IaaS model storing the data of customers. These systems will have to meet many essential needs such as high reliability, consistency of data, replication, performance, and availability for maintenance of data of customers. In the cloud storage systems, the files have been split into multiple blocks stored in various nodes across a distributed network. The unreliable connectivity of the network, the erratic node failure, and its limited bandwidth can result in challenges to data sharing efficiency. In case the data node storing the block files results in a failure, the whole file can become unavailable. For applications that are intensive in terms of data, there are scientific applications that have to be available. There may be a large-scale system of cloud storage using replication of data for ensuring a high level of availability, dependability and data consistency. Replication is a strategy used commonly where the same data has been stored in various storage devices. The chances of data being available can increase if there are multiple copies of such data stored in various nodes [8].

Replication of data is a technique employed in the distributed systems for meeting challenges of availability and improving the performance of data access. Replication of data can increase the reliability and availability and also brings down waiting time and consumption of cloud bandwidth. It also increases tolerance to faults and improves scalability. At the time of replication of data, data file copies get created in different nodes and in case of failure of one node, there may be a replica of the same data available in a different node for processing this request thus giving an uninterrupted service [9].

Data replication has two different categories such as static replication, as well as dynamic replication. The data is replicated statistically on the nodes that are randomly chosen for a certain fixed number of instances. The method also provides a faster response with high efficiency and availability. The strategies can be easily implemented but not used often as it may not be able to adapt to the changes in the user request, bandwidth or capacity of storage.

Replication of dynamic data [10] will be able to dynamically create and also delete the replicas that are based on the changes in the patter of user access, the capability of storage or bandwidth. The data available in the storage does not follow any pattern of common access. There are some data accessed frequently and some least accessed.

There is a policy of dynamic replication that is necessary for this scenario in which the replicas for data has been decided on the basis of the popularity of such data. There are several dynamic approaches to reduce utilization of storage space and also the cost by means of maintaining replica for data items. The replication of dynamic data and their policies have to make certain important decisions as to when the data has to be replicated, how many such replicas are created and where the new replicas are can be placed.

Replication [11] is a very important subject of investigation in the environment of the cloud. Based on the actual number of replicas with their location which is a Non-Deterministic Polynomial (NP)-hard problem. Thus, there are several heuristic algorithms that were proposed. For the purpose of this work, the heuristic algorithm that is based on the CS in cloud computing was employed. The rest of the investigation has been organized thus. Section 2 explains all the related work in the literature. The methods employed are explained in Section 3. The results of the experiment are discussed in Section 4 and the conclusion has been duly made in Section 5.

II. RELATED WORK

Most of the existing works focus on the offloading of tasks that are based on the premise that there are sufficient resources with the edge servers and this ignores the constraints of a budget. In case we fail to consider this, the offloading schemes that are in existence now may result in the user to overspend which may be unacceptable to the user. He et al., [12] had made an investigation of the problem of task offloading in the edge-cloud computing that aims at bringing down the duration of tasks generated by users having a constrained budget. Aside from this, the edge services were equipped having limited storage and computation resources. More specifically, this problem formulates the NP-hard problem. For the purpose of solving this, the work had proposed a new and heuristic strategy. The results of the simulation proved that the scheme was able to improve the success ration and bring down the duration of the task when compared to the greedy and offloading schemes.

Liu et al., [13] had observed that the real-world sensing data can exhibit a strong temporal and spatial correlation with advanced methods of inference employed for efficient recovery of missing data. The work further provided an approach of online task dispatching that was near-optimal to the services of crowdsourcing that was given by a new mobile edge cloud that aims at minimizing the total cost that is incurred on the devices guaranteeing the quality of service. Aside from this, the users of strategic devices having private cost information and the work proposed a pricing policy that was truthful. There were some extensive simulations that were based on the real datasets showing that the approach was able to outperform the other schemes thus producing a quality of service that was high having a lower budget.

The Virtual Process Control Functions (VPFs) are executed in the edge clouds found in 5G mobile networks and in the wireless backhaul to improve efficiency. The basic challenge was to make sure the placement of the VPFs that are resilient to the components and their failures and the cyber-attacks aside from being efficient. Zhao and Dán [14] had addressed the challenge by means of considering the costs of VPF placement that have been incurred by means of reserving the resources of Mobile Edge Computing (MEC) that execute the instances of the VPF and also communication of data. This also formulates the problem of VPF placement as its problem of Integer Programming (IP). The work also proposed a solution that was based on the decomposition of generalized Benders and linear relaxation of the sub-problems to effectively bring down the integer variables to the MEC codes.

For a large scale infrastructure of the IoT, there is some optimum VM allocation to physical hosts that result in a reduction of consumption of energy of the data centres. Furthermore, it may also be able to prevent pollution of the environment and show improvement in efficiency. A comprehensive investigation has shown that employing these types of virtualization where an ideal combination of such methods applying strategies that are evolutionary has been quite promising in developing newer algorithms. The novel algorithms developed were able to improve the efficiency of energy in cloud computing. Farhadian et al., [15] had proposed another optimum VM allocation to the physical hosts that employ an Imperialistic Competitive (IC) and the Genetic Algorithms (GA). To this end, there was a crossover operation from the GA that was mixed with the IC. There was a considerable improvement in the consumption of energy that was achieved by means of this method proposed.

The IoT system has various applications that have requirements of quality services and constraints of resources. The infrastructure of the cloud will provide the resources with various capabilities of processing and memory spaces. For the purpose of achieving an optimal level of scheduling, there is a diversity in both the applications along with the resources that were considered. Further, it will also have the need to avoid conditions in which some resources get overloaded and some of them are underutilized. The Particle Swarm Optimization (PSO) is very simple and is also fast in its technique of responding. Krishnapriya and Joby [16] are taking a reduced response time of application as the need for quality. In this, the scheduling technique known as the load balanced PSO scheduling had been proposed to allocate the tasks for the suitable machines aside from ensuring a low application response time and load balance.

Najjar-Ghabel et al., [17] had further proposed a new and reliable algorithm of spanning tree construction known as the Reliable Spanning Tree construction in the IoT (RST-IoT). This algorithm makes use of an algorithm called the Artificial Bee Colony (ABC) for generating proper trees. For the purpose of this method, the distances of hop count in the devices from that of the base station, the devices and their residual energy with the probabilities of mobility taken to have measured the trees and their appropriateness. Furthermore, the algorithm can generate many trees as opposed to a single one. The trees have been arranged in accordance with preferences and have been used for the purpose of data gathering. Every tree was employed for this data gathering on splitting the earlier one. The results of the simulation proved that the RST-IoT had improved the dependability of that of data gathering in the emergency applications when compared

to the earlier approaches.

Basu et al., [18] had employed an intelligent or a cognitive model which was bio-inspired to identify task scheduling for the IoT applications in a cloud environment of a heterogeneous multiprocessor. The natural choice of genes and the traits of foraging have proved that the species that are the fittest can survive. For such a case, there is a fit schedule that was considered to be one that was efficient and also follows a task that orders in the environment of the multiprocessor. The hybrid algorithm for the GA and the Ant Colony Optimization (ACO) (GAACO) was used for the choice of an ideal combination of tasks for every stage. A unique combination of the GA and the ACO that was used had ensured a suitable convergence along with optimality at the time the GAACO that was developed. Scheduling that makes use of the GAACO has not been pre-emptive and has been assumed that a single task may be assigned only to one processor. While being tested on the different sizes of the task graphs along with different processors, the GAACO is proven to be competent compared to the conventional ACO and the GA in an environment of a heterogeneous multiprocessor.

Luo and Ren [19] had made an analysis of the cloud computing application and the IoTs in the field of medicine. It further managed to bring about a combination of two different types of medical monitoring and its managing. This architecture in the Remote Monitoring Cloud Platform of Healthcare Information (RMCPHI) had been initially established and the RMCPHI architecture had been analysed. lastly, there was an efficient algorithm of PSO-Simulated Annealing (SA) (PSOSA) that was proposed for both medical monitoring and managing in cloud computing. The results of the simulation proved that the scheme proposed was able to improve efficiency by about 50%.

Elhoseny et al., [20] had proposed another new model for optimizing the selection of the VM in the applications of the cloud-IoT health services in order to manage a large amount of data in an integrated industry 4.0. The industry 4.0 applications need to process and further analyse Big Data from various sources like sensor data without any human intervention. The model proposed aimed at enhancing the performance of healthcare systems by means of reducing the request and time of execution of the stakeholders. It also optimized the necessary Big Data storage of the patients thus providing the real-time data and its mechanism of retrieval for such applications. The architecture belonging to the hybrid cloud-IoT contains four components: the devices of the stakeholders, the requests of the stakeholders (the tasks), the cloud broker and the network administrator. For the purpose of optimizing the selection of the VM, there are three well-known optimizers (the GA, the PSO and the Parallel PSO (PPSO) have been used for building this model proposed. For the purpose of calculating the time taken for the execution of the requests of the stakeholders, the fitness function proposed was a composition of three critical criteria which were the utilization of CP, the waiting time and the turn-around time.

The technique of Load-balancing is a primary issue in a distributed system of computing. As there have been certain large-scale resources and plenty of user demands in the cloud computing techniques and the problems of load balancing, it may be one of the primary reasons owing to which several researchers addressed this to be an NP-hard problem. There was a proposal of certain heuristic algorithms like the Firefly Algorithm (FA) and the Imperialist Competitive Algorithm (ICA) for solving the problems mentioned by Kashikolaei et al., [21]. Even though the ICA and the FA were able to get a result that was approximately satisfying in the process of solving a cloud computing load balancing to obtain better results in order to make improvements to the speed of planning, stability, load balancing, the CPU time and the make span. The research has been motivated by a proposal of an intelligent algorithm that was a combination of the ICA and the FA to attain the needed results. The FA and its local search ability were able to reinforce the ICA. The results obtained proved to show drastic levels of improvements.

III. METHODOLOGY

The large-scale applications of the IoT will have to process and also manage several datasets found across the data centres distributed geographically. The applications are provisioned in many data centres for exploiting the geographically distributed independent infrastructure of Information Technology (IT). The replication of cloud computing is the approach the creates several copies of similar data which stores them in different sites. The cloud data also bring down the waiting time and improved availability of data. it also increased the consumption of bandwidth. For the purpose of this work, replication of data with the CS algorithm and the Greedy Search were discussed.

A. Problem Formulation

This had modelled a sensor network to be the graph G = (V; E). It also proved that there are two nodes which are v_i and v_j that had been connected directly to one another (through a single hop) and if there was a data transfer that was direct among them without an intermediate node. Every node in a sensor network may either be a sensor, an actuator, a gateway or a router. There are some direct connections between pairs of nodes that are modelled through an edge which was $e_i \in E$. There is an edge weight w_{ij} for every edge that models the data transfer latency among two different nodes which made up the edge [22].

The Mini-Clouds were placed in certain specific points found in the sensor network. every mini-cloud is configured for interacting in a particular set of nodes that have resulted in a division of sensor network so that every node transmits data to a single mini-cloud. It was further assumed that the total mini-clouds installed within the network were limited by N owing to the considerations of cost. Further, if li is the latency of transmission of data from the node v_i to that of the mini-cloud to which it has been connected. So, the problem of optimization was to bring down the latency and its maximum value expressed as below:

$$\min \max l_i$$
$$s.t.\ x_i \in \{0,1\}\ \forall i \in V$$
$$\sum_{i \in V} x_i \le \mathrm{N} \qquad (1)$$
$$\sum_{i \in V} y_{ik} * x_i = 1\ \ \forall i \in V$$

The N mini-clouds and the V nodes upon which the mini-cloud has been placed to ensure maximum latency from the node to the mini-cloud which is minimized. x_i is the placement of these mini-clouds on the chosen network nodes. The constraint of inequality has depicted the limit to this. y_{ik} in an equality constraint proves that the node v_i will send the data to a mini-cloud which is positioned at a node v_k, and so y_{ik} is 1, or else is zero, $y_{ik} \in E$. This was noted to be the formulation of the problem and it does not make any of these assumptions as regards the capacity of mini-cloud storage.

For an approximation algorithm, the sensor networks are configured and the data from sensor data is sent to the gateway through routers. Irrespective of the topology of the sensor network, it may be assumed that a gateway collection will have to be connected to a set of the N mini-clouds.

For each of these gateways, G_i the connected mini-Cloud C_j, represents latency as $l(G_i, C_j) = \max(L_{G_i}) + l'_{ij}$, wherein L_{G_i} is the maximum latency of sensor nodes that are connected to a gateway G_i, and l'_{ij} this is the latency of it directly connecting the G_i to C_j. This represents latency values through matrix M (G; C). This problem tries to find a suitable mapping between the gateways and the mini-clouds to ensure the maximum value of l (G_i; C_j) has been minimized.

One key problem of research in this scenario was to upload data from sensor gateways to replicate date in mini-clouds that cater to availability and the requirements of disaster recovery. This needs data to be replicated optimally among mini-clouds within minimal time. There are several parameters such as the bandwidth of network and data available in storage for every mini-cloud and the data items for every mini-cloud and the data that needs to be replicated. For this, an approach was proposed with an optimal solution that was intractable using various heuristics.

B. Data Replication using Greedy Search Algorithm

In the case of the Greedy Algorithm, every time the data items were taken in a sequence which was deemed best in accordance with certain criterion. The GA will arrive at a new solution by making some choices that look the best [23]. This algorithm will begin with an empty set that adds items to a set in a sequence until such time a set represents a new solution to the problem. Given below are components according to this iteration:

- The process of selection will select the subsequent item to be added to a set. This will be performed in accordance with the Greedy criterion satisfying a locally optimal consideration.
- The feasibility check will find out a new feasible set by checking the possibility of being able to complete the set by giving a solution.
- The solution check will find out if the new set has a solution to the instance.

Considering all possible JOBS, the work gives a greedy heuristic with the assumption below: a data item is set only once from its gateway to the mini-cloud. As the uplinks are not reliable or robust among the mini-clouds that are made. These include the links between mini-clouds, the data item replication which was from its initial destination of the mini-cloud to the mini-cloud both greedily and opportunistically. But the sequence wherein data items are sent to the mini-cloud is replicated among the mini-clouds that can be a possible solution based on the transfers. There have been a total of eleven sequences as depicted below: this has been described formally and has encoded a new and greedy strategy keeping all the networks and their links busy until there is a data item in the source end of a network link not with a factor of replication duly satisfied [24]. This greedy heuristic algorithm is as below:

Initialization;
Let job J be composed of data item D_j, *source* $Source_j$, *destination* $Dest_j$ *and link* L_j *reflecting a transfer event*;
Let S be the set of all possible JOBs at any point in time. For D data items, M min i-Clouds and L links / gateways, S will have $D \times M \times L$ *JOBs initially.*
repeat
Order S based on ordering heuristic; *foreach* $JOB_j \in S$ *do*
if $Dest_j$ *has no space for* D_j *then*
Delete JOB_j;
end
if L_j *is not busy then*
Schedule JOB_j;
Delete all JOB_k *from S such that* $D_k = D_j$ *and* $Source_k = Source_j$;
Mark L_j *busy*;
end
end
In parallel, when any JOB_j *completes Mark* L_j *free*;
delete JOB_j;
If (D_j *needs to be further replicated*) *Add new* JOB_k *to S, such that* $Source_k = Dest_j$, $Dest_k \neq Source_k$ *and* $Dest_k \neq$ *gateway*;
until $S \neq \phi$;

As mentioned earlier, it customizes the GA with the orderings below along with the JOBS that have been scheduled.

1) The arrival of data – the earliest is first.
2) The smallest remaining bandwidth link is first.
3) the largest remaining bandwidth link is first.

4) the smallest available space at the destination first – the tightest destination first.

5) the largest available space at the destination first – the loosest destination first.

6) the data item with the smallest data size is first.

7) the data item with the largest data size is first.

8) the smallest ratio of (data available size or bandwidth) is first.

9) the largest ratio (the data size or bandwidth available) is first.

10) the smallest value of (the bandwidth available*data size) is first.

11) the largest value of (the bandwidth available*data size) is first.

It has been evaluated in all orderings in the simulation. It is to be noted that the heuristic is greedy in the manner below:

- The JOBs were scheduled once the links on which there is a transfer that has to happen will now be free. So, in case the gateway or the mini-cloud has more number of link ups, there may be more than a JOB in progress.
- Once this JOB is completed, there is another new set of these JOBs spawned for a new data item set and their free links until such time the needs of a specification replication factor for data items which is satisfied.
- An ordering of a set of JOBs was done based on the chosen orderings in a single simulation.

The loop that is the outermost will execute this until there are no more JOBs left. The JOBs have been added to S and at the time the algorithm begins and also when the JOB is complete and the data item is yet to be replicated a certain number of times. Once the JOB is complete, all the other JOBs which are for a similar data item and found in the same source will be removed. So, the S will grow and also shrink as its algorithm continues to proceed and also gets empty at the time the data items were replicated sufficiently. So, the algorithm always terminates.

C. Proposed Data Replication using Cuckoo Search (CS) Algorithm

For the purpose of this work, the strategy of replication has been based on the data items ordered in a sequence to be uploaded from the gateways to the mini-cloud. This will have to assign an order of transfer of the various data items until such time all replicas have been stored on the mini-cloud. It has been assumed that the store is of a forward type of transfer and so for any such transfer to begin, a data item that is copied completely will have to be present.

The CS algorithm was based on the behaviour of an obligate brood parasitic species of cuckoo that was combined with the behaviour of Levy flight of certain types of fruit flies and birds. For the purpose of implementing a CS heuristic that was based on ordering, there was a need for an abstraction to simulate the transfer or the copying among that of the mini-clouds from their gateways. This was known as the abstraction of a JOB. It further modelled every JOB to be a 4-tuple that consisted of: a source identifier, a destination identifier, the value of the bandwidth for the link between the identifier of the source and the destination and finally the data identifier. The JOB here will represent a data item that is transferred, either in the form of a gateway to the mini-cloud or from one mini-cloud to that of another (for purposes of replication). Such a transfer will take place only on the following conditions being met: (1) the JOB's data identifier has to be available at source that is identified by the source identifier of the JOB, (2) data item may not be present in the destination which is identified by the JOB's destination identifier and (3) there is no transfer which takes place between that of the destination in this moment. In other words, every JOB will simulate a process of transfer in this program for a certain number of data items in a pair of both a source and simulation of a process within the program for certain data items in a pair that can be the gateways or the mini-clouds and the transfer destination is always the mini-cloud.

The Cuckoo Breeding Behaviour: the cuckoos are very beautiful birds with aggressive strategies for reproduction which is more interesting to all. The Cuckoos lay eggs in the communal nests that had been chosen by the levy flight [25]. They tend to choose a nest in which the host bird lays its eggs. In order to increase the chances of hatching their eggs, the cuckoos remove the eggs of the host birds. Some cuckoos also imitate the colours and the patterns of the host species. The cuckoos bring down the probability of eggs getting abandoned in order to increase their chances of productivity. Thus, at the time the host bird identifies the strange egg, it may throw it away or leave the nest and go elsewhere to build another one. The cuckoo generally chooses the nest that has a few eggs in it.

The Lévy Flights: the behaviour of the animals in searching for food is quasi-random. In recent times, there has been a lot of investigation that has shown the flight behaviour of animals and the insects that have been demonstrating the typical traits of Levy flights. Normally, the path of foraging of animals is in the form of a random walk and the subsequent move is based on the present state or location and the probability of transition to its next location. There has been a lot of investigation made which shows the flight behaviour of animals and the insects will demonstrate the traits of Levy flights. Later, this behaviour was applied to optimal search and optimization where the preliminary results have been showing a promising level of capability.

The CS algorithm: the CSA is yet another new approach which models the behaviour of cuckoos. For describing the simplicity of the new CSA, given below are the idealized rules [26]:

- Every cuckoo lays only one egg and dumps it in a nest chosen randomly.
- The nests that are the best, having eggs of high quality (solutions) are carried out to the subsequent generations.
- The hosts that are available in terms of nests are fixed as n and the host will be able to identify the alien egg in a probability Pa [0, 1].

- A host bird may throw away the egg or may abandon its nest to build a new nest in a different location.

For purposes of simplicity, the final assumption is approximated using a probability of n nest pa. Here, x denotes the vector and the entries $x_i^{(t+1)}$ indicate the subsequent generation of the cuckoo from x_i^t of ith one computed as (2 and 3):

$$X_i^{(t+1)} = X_i^{(t)} + \alpha \oplus Levy(u) \quad (2)$$

And

$$Levy\ u = t^{-\lambda},\ 1 < \lambda \le 3 \quad (3)$$

Wherein α > 0 denotes the size of the step which is dependent based on the scale of this problem, ⊕ denotes the multiplications made entry-wise.

The cuckoo's consecutive steps are from the process of a random walk that follows the power law and step-length distribution that has a heavy tail. In reality, if the egg of the cuckoo is similar to the eggs of the host bird, the chances of the cuckoo's egg being discovered are less likely. So, it may be a good idea to do a random walk in a manner that is biased using some step sizes that are random.

The Fitness Value: the quality or fitness value depicts the fitness of the solution. For a problem of maximization, the solution and its fitness may be proportional to the objective function value. For purposes of simplicity, if every egg in the nest is a solution, the egg of the cuckoo is a new solution. This is for using a set of better solutions (cuckoos) to replace the set of not-good solutions in these nests.

On the basis of these rules, the primary steps in the [27] may be summarized to be the pseudo code as shown below:

Data : Given a set of gateways with data items, a set of min i - Clouds with a known capacity and a set of communication links among them
Result : Time taken to transfer all data items from the gateways and to replicate all data items
begin
Objective function $f(x),\ x = (x1, ..., xd)^T$
Generate initial population of
n host nests $xi(i = 1, 2, ..., n)$
while $(t < MaxGeneration)$ *or* $(stop\ criterion)$
Get a cuckoo randomly by Levy flights
evaluate its quality / fitness F_i
Choose a nest among n (say, j) *randomly*
if $(F_i > F_j)$,
replace j by the new solution;
end
A fraction (P_a) *of worse nests is abandoned and new ones are built;*
Keep the best solutions $(or\ nests\ with\ quality\ solutions)$;
Rank the solutions and find the current best
end while
Postprocess results and visualization
end

IV. RESULTS AND DISCUSSION

Experiments were conducted with data First in First Out (FIFO) scenario and destination with maximum available space and bandwidth. Experiments are carried out using 10000 to 80000 number of data items.

The greedy search and CS methods are used. Table 1 shows the parameters used. The fixed data size of 32kb and 1 Mbps bandwidth, fixed data size of 32kb and 8 Mbps bandwidth, fixed data size of 64kb and 1 Mbps bandwidth and fixed data size of 64kb and 8 Mbps bandwidth as shows in tables and figures.

Table 1 Parameters Used

Number of gateways	25
Number of edges	15
Bandwidth	1 Mbps to 8 Mbps
Data size	32kb to 64kb

Table 1 Fixed Data Size of 32kb and 1 Mbps Bandwidth

Fixed data size of 32kb and 1 Mbps Bandwidth. Number of data items	Time in microsecond - Greedy Search	Time in microsecond - Cuckoo Search
50000	10325	9644
10000	22949	21537
20000	47627	45181
40000	106417	97937
80000	211755	195734

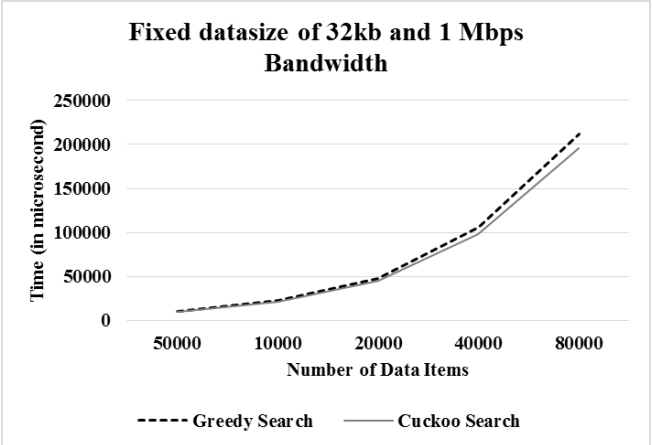

Fig 1 Fixed Data Size of 32kb and 1 Mbps Bandwidth

From the Fig 1, it can be observed that the CS has lower fixed data size of 32kb and 1 Mbps bandwidth by 6.82% for 50000 number of data items, by 6.35% for 10000 number of data items, by 5.27% for 20000 number of data items, by 8.29% for 40000 number of data items and by 7.86% for 80000 number of data items when compared with greedy search method respectively.

Table 2 Fixed Data Size of 32kb and 8 Mbps Bandwidth

Fixed data size of 32kb and 8 Mbps Bandwidth. Number of data items	Time in microsecond - Greedy Search	Time in microsecond - Cuckoo Search
50000	4133	4188
10000	9527	8882
20000	20348	19787
40000	45993	41092
80000	91509	80260

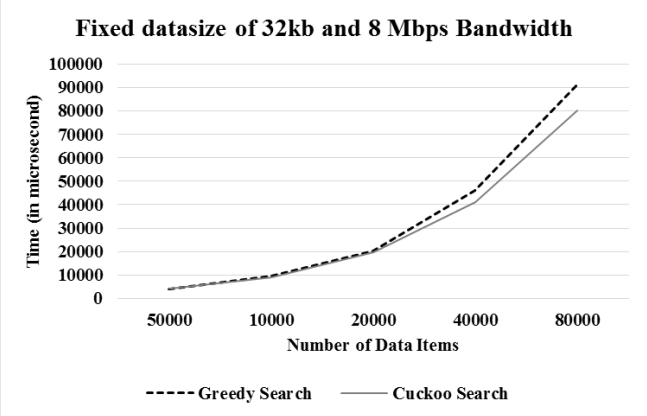

Fig 2 Fixed Data Size of 32kb and 8 Mbps Bandwidth

From the Fig 2, it can be observed that the CS has lower fixed data size of 32kb and 8 Mbps bandwidth by 1.32% for 50000 number of data items, by 7.01% for 10000 number of data items, by 2.79% for 20000 number of data items, by 11.25% for 40000 number of data items and by 13.09% for 80000 number of data items when compared with greedy search method respectively.

Table 3 Fixed Data Size of 64kb and 1 Mbps Bandwidth

Fixed data size of 64kb and 1 Mbps Bandwidth. Number of data items	Time in microsecond - Greedy Search	Time in microsecond - Cuckoo Search
50000	17864	16467
10000	36240	33511
20000	69017	63868
40000	139571	128924
80000	321339	298550

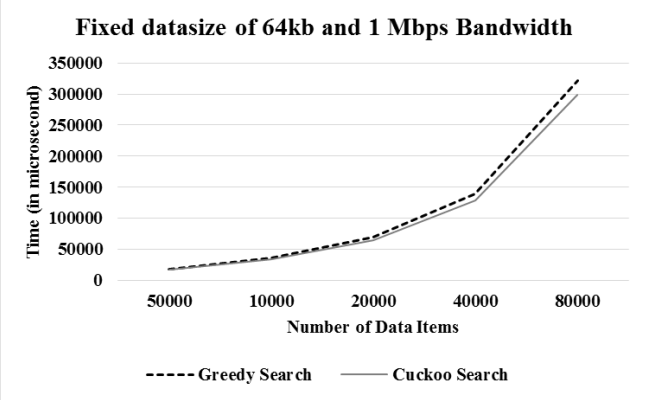

Fig 3 Fixed Data Size of 64kb and 1 Mbps Bandwidth

From the Fig 3, it can be observed that the CS has lower fixed data size of 64kb and 1 Mbps bandwidth by 8.14% for 50000 number of data items, by 7.82% for 10000 number of data items, by 7.75% for 20000 number of data items, by 7.93% for 40000 number of data items and by 7.35% for 80000 number of data items when compared with greedy search method respectively.

Table 4 Fixed Data Size of 64kb and 8 Mbps Bandwidth

Fixed data size of 64kb and 8 Mbps Bandwidth. Number of data items	Time in microsecond - Greedy Search	Time in microsecond - Cuckoo Search
50000	7214	7165
10000	15049	14539
20000	27685	27064
40000	59028	51772
80000	134259	128713

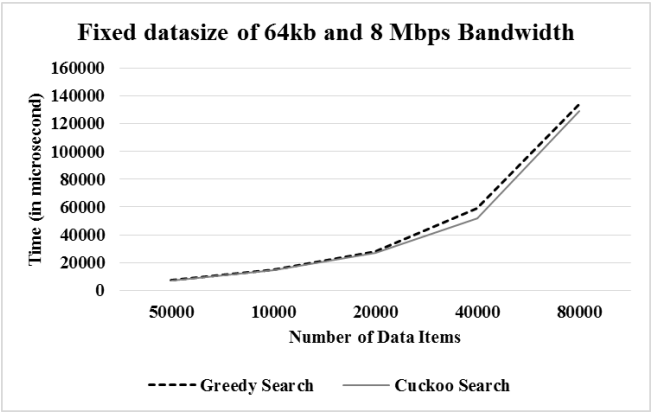

Fig 4 Fixed Data Size of 64kb and 1 Mbps Bandwidth

From the Fig 4, it can be observed that the CS has lower fixed data size of 64kb and 8 Mbps bandwidth by 0.68% for 50000 number of data items, by 3.45% for 10000 number of data items, by 2.27% for 20000 number of data items, by 13.09% for 40000 number of data items and by 4.22% for 80000 number of data items when compared with greedy search method respectively.

V. CONCLUSION

For this work, there is a very crucial problem in the integration of cloud computing and the IoT. Optimally, this denotes the mean time taken for replication of data that may be considerable in terms of the size of the generated data in the IoT and it has characterised this problem based on parameters of the capacity of storage, size of data, the bandwidth of link and latency. As there is a general problem that is intractable, this work presents the Greedy and the CS heuristic that consists of many orderings. The GA is effective in solving issues with optimal substructures which means the local optimum will be able to determine its global optimum. The CS is an algorithm based on swarm intelligence that is inspired by the cuckoos and their behaviour especially in their obligate brood parasitism of the species that lays eggs in

the nests belonging to the host birds. The CS also has several advantages owing to efficiency and simplicity in problem-solving that includes non-linear problems of optimization. The results have proved that the CS has a lower size of fixed data of about 32kb and 1 Mbps bandwidth by about 6.82% for the 50000 number of the data items, by about 6.35% for the 10000 number of the data items, by about 5.27% for the 20000 number of the data items, by about 8.29% for the 40000 number of the data items and finally by about 7.86% for the 80000 number of the data items on being compared to the Greedy Search method. In the same way, the CS also has a fixed data size that was lower of about 32kb and a bandwidth of 8 Mbps, a fixed data size of about 64kb and 1 Mbps bandwidth along with a fixed data size of about 64kb and a bandwidth of 8 Mbps compared to the Greedy Search method and its various data items

REFERENCES

1. Kumrai, T., Ota, K., Dong, M., Kishigami, J., & Sung, D. K. (2017). Multiobjective optimization in cloud brokering systems for connected Internet of Things. IEEE Internet of things journal, 4(2), 404-413.
2. Aazam, M., & Huh, E. N. (2014, August). Fog computing and smart gateway based communication for cloud of things. In 2014 International Conference on Future Internet of Things and Cloud (pp. 464-470). IEEE.
3. Zhou, J., Leppanen, T., Harjula, E., Ylianttila, M., Ojala, T., Yu, C., ... & Yang, L. T. (2013, June). Cloudthings: A common architecture for integrating the internet of things with cloud computing. In Proceedings of the 2013 IEEE 17th international conference on computer supported cooperative work in design (CSCWD) (pp. 651-657). IEEE.
4. Shanthan, B. J., Kumar, A. D. V., Govindrajan, E., & Arockian, L. (2017). Scheduling for Internet of Things Applications on Cloud: A Review. Imperial Journal of Interdisciplinary Research, 3(1), 1649-1653.
5. Botta, A., De Donato, W., Persico, V., & Pescapé, A. (2016). Integration of cloud computing and internet of things: a survey. Future generation computer systems, 56, 684-700.
6. Babu, S. M., Lakshmi, A. J., & Rao, B. T. (2015, April). A study on cloud based Internet of Things: CloudIoT. In 2015 global conference on communication technologies (GCCT) (pp. 60-65). IEEE.
7. Botta, A., De Donato, W., Persico, V., & Pescapé, A. (2014, August). On the integration of cloud computing and internet of things. In 2014 International Conference on Future Internet of Things and Cloud (pp. 23-30). IEEE.
8. Selvi, M. S. A. E., & Anbuselvi, R. (2015, March). An Analysis of Data Replication Issues and Strategies on Cloud Storage System. In International Journal of Engineering Research & Technology (IJERT), NCICN-2015 Conference Proceedings, pp18-21.
9. Milani, B. A., & Navimipour, N. J. (2016). A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions. Journal of Network and Computer Applications, 64, 229-238.
10. Milani, B. A., & Navimipour, N. J. (2017). A systematic literature review of the data replication techniques in the cloud environments. Big Data Research, 10, 1-7.
11. Ebadi, Y., & Jafari Navimipour, N. (2019). An energy-aware method for data replication in the cloud environments using a Tabu search and particle swarm optimization algorithm. Concurrency and Computation: Practice and Experience, 31(1), e4757.
12. He, L., Xu, H., Wang, H., Huang, L., & Ma, J. (2018, November). Task Offloading in Edge-Clouds with Budget Constraint. In International Conference on Algorithms and Architectures for Parallel Processing (pp. 311-326). Springer, Cham.
13. Liu, T., Zhu, Y., Yang, Y., Ye, F., & Yu, J. (2018). Online task dispatching and pricing for quality-of-service-aware sensing data collection for mobile edge clouds. CCF Transactions on Networking, 1-15.
14. Zhao, P., & Dán, G. (2018). A Benders decomposition approach for resilient placement of virtual process control functions in mobile edge clouds. IEEE Transactions on Network and Service Management, 15(4), 1460-1472.
15. Farhadian, F., Kashani, M. M. R., Rezazadeh, J., Farahbakhsh, R., & Sandrasegaran, K. (2019). An efficient IoT cloud energy consumption based on genetic algorithm. Digital Communications and Networks.
16. Krishnapriya, S., & Joby, P. P. (2015). QoS Aware Resource Scheduling in Internet of Things-Cloud Environment. International Journal of Scientific & Engineering Research, 6(4), 294-297.
17. Najjar-Ghabel, S., Yousefi, S., & Farzinvash, L. (2018). Reliable data gathering in the Internet of Things using artificial bee colony. Turkish Journal of Electrical Engineering & Computer Sciences, 26(4), 1710-1723.
18. Basu, S., Karuppiah, M., Selvakumar, K., Li, K. C., Islam, S. H., Hassan, M. M., & Bhuiyan, M. Z. A. (2018). An intelligent/cognitive model of task scheduling for IoT applications in cloud computing environment. Future Generation Computer Systems, 88, 254-261.
19. Luo, S., & Ren, B. (2016). The monitoring and managing application of cloud computing based on Internet of Things. Computer methods and programs in biomedicine, 130, 154-161.
20. Elhoseny, M., Abdelaziz, A., Salama, A. S., Riad, A. M., Muhammad, K., & Sangaiah, A. K. (2018). A hybrid model of internet of things and cloud computing to manage big data in health services applications. Future generation computer systems, 86, 1383-1394.
21. Kashikolaei, S. M. G., Hosseinabadi, A. A. R., Saemi, B., Shareh, M. B., Sangaiah, A. K., & Bian, G. B. (2019). An enhancement of task scheduling in cloud computing based on imperialist competitive algorithm and firefly algorithm. The Journal of Supercomputing, 1-28.
22. Narendra, N. C., Koorapati, K., & Ujja, V. (2015, November). Towards cloud-based decentralized storage for internet of things data. In 2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM) (pp. 160-168). IEEE.
23. Baghshahi, S. S., Jabbehdari, S., & Adabi, S. (2014). Virtual machines migration based on greedy algorithm in cloud computing. International Journal of Computer Applications, 96(12), 32-35.
24. Kumar, A., Narendra, N. C., & Bellur, U. (2016, June). Uploading and replicating internet of things (iot) data on distributed cloud storage. In 2016 IEEE 9th International Conference on Cloud Computing (CLOUD) (pp. 670-677). IEEE.
25. Navimipour, N. J., & Milani, F. S. (2015). Task scheduling in the cloud computing based on the cuckoo search algorithm. International Journal of Modeling and Optimization, 5(1), 44-47.
26. Polap, D., Wozniak, M., Napoli, C., & Tramontana, E. (2015). Real-time cloud-based game management system via cuckoo search algorithm. International Journal of Electronics and Telecommunications, 61(4), 333-338.
27. Bulatovic, R. R., Đordevic, S. R., & Đordevic, V. S. (2013). Cuckoo search algorithm: a metaheuristic approach to solving the problem of optimum synthesis of a six-bar double dwell linkage. Mechanism and Machine Theory, 61, 1-13.