Journal of Signal Processing Systems (2019) 91:1191–1204
https://doi.org/10.1007/s11265-019-1444-5

A Latency-Aware Multiple Data Replicas Placement Strategy for Fog Computing

Tiansheng Huang[1] · Weiwei Lin[1] · Yin Li[2] · LiGang He[3] · ShaoLiang Peng[4,5]

Received: 1 January 2019 / Revised: 21 January 2019 / Accepted: 30 January 2019 / Published online: 22 February 2019

Abstract
With the rapid increase of the number of IoT devices, transmitting big amount of data from these devices to data centers which are far away will cause problems like high latency or network congestions. Fog Computing provides a better solution for Fog-enabled latency sensitive data services to place data on Fog nodes which are closer to the data generators. However, recent studies only focus on the data placement problem of placing one single data replica to the proper Fog node. Under the situation that there are several data consumers whose topology positions are different subscribing the same data, one single data replica cannot meet the latency requirement of all the consumers. Hence, we build a multi-replica data placement model iFogStorM for Fog Computing to formulate the problem of how many data replicas need to be placed on Fog nodes and how to optimize the data placement. Furthermore, we propose a greedy algorithm based data replica placement strategy, MultiCopyStorage, to reduce the overall latency. MultiCopyStorage uses a pruning method to filter the inferior solutions calculates the overall latency and chooses the solution with the minimum overall latency as the final solution. We conducted experiments on iFogSim, a toolkit for modeling and simulation of Fog Computing, evaluated the proposed strategy with the CloudStorage strategy, Closest Node strategy, iFogStor strategy, and two kinds of heuristic strategy, iFogStorZ, and iFogStorG. The experiment result demonstrates that MultiCopyStorage strategy reduces the overall latency by 6% and 10% compared to iFogStor and iFogStorG strategy respectively. Meanwhile, execution time of the MultiCopyStorage is less than the heuristic strategy, iFogStorG and iFogStorZ, which proves that the proposed strategy can support real-time scheduling.

Keywords Data placement · Fog computing · Greedy algorithm · Internet of things · Multiple data replicas

1 Introduction

In recent years, there is a rapid development of Internet of Things (IoT). It is predicted that the number of IoT equipment will reach over 75 billion by 2025 [1]. Such a large number of connected equipment will generate a massive amount of data. To deal with the placement problem of data, the traditional way is to transmit all of them to data centers for storage and processing [2]. However, data centers in the cloud often fail to process and store the data generated by a massive number (billions) of distributed IoT devices [3] and could cause the problem of high

✉ Weiwei Lin
linww@scut.edu.cn

✉ Yin Li
liyin@gz.iscas.ac.cn

Tiansheng Huang
873966702@qq.com

LiGang He
liganghe@dcs.warwick.ac.uk

ShaoLiang Peng
pengshaoliang@nudt.edu.cn

1 School of Computer Science and Engineering, South China University of Technology, Guangzhou, China

2 Institute of Software Application Technology, Guangzhou & Chinese Academy of Sciences, Guangzhou 511458, China

3 Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK

4 College of Computer Science and Electronic Engineering & National Supercomputing Centre in Changsha, Hunan University, Changsha 410082, China

5 School of Computer Science, National University of Defense Technology, Changsha 410073, China

latency, network congestion, poor quality of Service, etc. [3]. To deal with the problem mentioned above, Cisco proposes the concept of Fog Computing [4], which engages the Fog nodes for data storage and processing. Fog nodes are the type of network equipment which is close to the users, e.g. routers, switches, set top boxes, proxy servers, Base Stations(BS), etc. [5]. Nowadays, more and more applications (e.g. intelligent surveillance, smart cities, wireless sensor networks, etc) require to store, process and obtain the data with extremely low latency and the introduction of Fog Computing could achieve this requirement. With the proper utilization of the computing and storage capacity of Fog nodes, the overall data transmission latency will drop down significantly because typically Fog nodes are closer to the IoT equipment than data centers. Hence, how to properly place the massive amount of data generated by IoT equipment on fog nodes which have different storage capacity in order to minimize the overall transmission latency for Fog-enabled data services is the main research topic in this work.

In the area of Cloud Computing, [6, 7] have proposed some data placement strategies. However, those strategies both tend to place data replicas on the Cloud. Compared with data centers, Fog nodes are more widely distributed in network topology, larger in number and more limited in storage capacity. As a result, the strategies proposed in [6, 7] cannot be applied directly in the field of Fog Computing. To the best of our knowledge, only in [8, 9], some data placement strategies in the field of Fog Computing were proposed. However, those strategies only focus on the data placement problem of placing one single data replica to the proper Fog node. Under the situation that there are several data consumers whose topology positions are different subscribing the same data, one single data replica cannot meet the latency requirement of all the consumers. Hence, on the basis of [8, 9], we improve the data placement model to support multiple data replicas placement and the overall latency is significantly reduced compared to the single data replica strategies. The multi-replica model proposed in this paper is a MILP model which has been proved to be NP-hard for solving. It will take an unbearable amount of time to solve the model by using mathematical programming optimizer like CPLEX [10]. In order to solve the model efficiently, we design a few heuristic rules and propose a latency-aware strategy to dramatically reduce the searching space for solving the target model. We have conducted the extensive experiments. The results show that the solving time of the model is significantly reduced and the scheduling decision can be made at real time.

In summary, this paper makes the following contributions:

1. Under the multi-replica storage architecture for Fog Computing, we propose a latency-aware model, iFogStorM, which can support the multiple data replicas placement.
2. We analyze the complexity of iFogStorM and further develop a heuristic data placement strategy, MultiCopyStorage, to solve the model efficiently even when the problem scale is large.
3. We implement and integrate our latency-aware strategy into iFogSim [2], and evaluate the performance of MultiCopyStorage with extensive tests. The results show that compared to the existing strategies, our strategy can achieve much better performance in terms of reducing data transmission latency and execution time.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 formulates the multiple data replicas placement problem and constructs the latency-aware model, iFogStorM. Section 4 proposes the heuristic algorithm to solve the model. After evaluating the performance of our strategy in Section 5, we summarize our conclusions and points out the future work in Section 6.

2 Related Work

Many research efforts were done to propose some module placement strategies. In [11], Mahmud et al. propose a latency-aware heuristic module placement strategy. According to the subscription relationship between modules and the tolerable delay, the modules should be placed to the North end (near the cloud) as far as possible without violating the maximum tolerable delay. Furthermore, they propose a module forwarding strategy in order to optimize number of computationally active Fog nodes and reduce energy consumption. In [12], Taneja et al. abstract the subscription relationship between modules into a directed acyclic graph (DAG), and propose a placement strategy to maximize the utilization of computing resources of fog nodes. In [13], a method based on fuzzy theory [14] for fractional evaluation of modules and fog nodes is proposed, and then a matching model between modules and fog nodes is established. Finally, the model is solved by SCIP [15], an integer programming solver. In [16], Skarlat et al. propose the concept of fog colonies, which abstracts multiple fog nodes into a cluster, and then propose a module placement model considering module response time, CPU, memory and storage resources required by the module. On the premise of meeting the corresponding constraints, the number of modules placed in fog nodes was maximized. Sun et al. [17] also combine multiple fog nodes into clusters and proposes a two-step scheduling scheme: 1) resource scheduling among various fog clusters. 2) resource scheduling among fog nodes in the same fog cluster. For step 2, a multi-objective placement model is proposed to meet the minimum completion time of the module, and two objectives are proposed: 1) minimize the service latency of the application and 2) maximize the stability of the application. And a NSGA-II [18] based strategies applied to the multi-objective optimization of the model.

However, due to the heterogeneity of fog nodes, the computing capacity and storage space of fog nodes are often asymmetric, and the research mentioned above does not consider the storage of data. Considering the case that data is stored in a fog node with storage space and then forwarded to the consumption service (module) which subscribes to the data, we formulate the data placement problem.

In order to reduce the overall latency of data transmission, many solutions have been proposed recently. In [19], Wang et al. propose a new architecture for computation and storage offloading which can reduce the communication cost and latency. In [8], Naas et al. propose a data placement model of data production, storage and consumption. The data can be produced by sensors or modules, and the produced data is sent to a storage node (host) in the network topology for storage and then forwarded to consumption services (modules) which subscribe to the data. To achieve the model proposed in [8], CPLEX MILP [10] is used for accurate solution, and a heuristic graph partitioning-based strategy, iFogStorZ, is proposed to reduce the complexity of the problem. iFogStorZ defines Regional Point of Presences (RPOPs) as points of partitioning but it does not consider the data flow between different parts after partitioning. In [9], based on the model designed in [8], a new graph partitioning method, iFogStorG, is proposed, which balances the complexity of various parts after graph partitioning and minimizes the data flow across parts. This strategy improves the accuracy of model solution to a certain extent. However, after graph partitioning, all parts are still using CPLEX MILP [10] for problem solving. Under large-scale scheduling, the solving speed is not ideal for real-time scheduling. In addition, both iFogStorZ and iFogStorG are based on the model proposed in [8]. The data produced by the data producer is stored on only one single storage node (host). When the corresponding number of data consumers is large and geographically distributed, a single replica cannot meet the latency requirements of all consumers.

In addition, many scholars have proposed some data placement strategies for multiple replicas in distributed environments such as big data computing. In [20], based on the network topology and the storage load of each node, a multiple replicas data placement strategy is proposed on the Hadoop platform, which ensures load balancing and data transmission performance. In [21], Wu et al. propose a multiple data replicas placement strategy under the tree network, which significantly reduced the cost of the entire tree network. In [22], Lizhen et al. propose a genetic algorithm based data replica placement strategy for scientific applications in Clouds. The experiment result shows that when the number of replicas changes from 1 to 2, the data transfer time decreases by nearly 50%. In [23], Rajaretnam et al. propose a dynamic multi-copy placement strategy for data grid. According to the request frequency of data, the number and the locations of data replicas are dynamically adjusted. However, to the best of our knowledge, no other research has proposed multiple data replica placement strategies in the Fog Computing environment.

Therefore, we propose the multiple data replicas placement model, iFogStorM, in view of the deficiency of the model above. A data producer can send data to multiple Fog nodes for storage at the same time, and consumers who subscribe to the data can choose any one of the storage nodes to get the data. In order to achieve the model, we propose a greedy algorithm based strategy, MultiCopyStorage, which can obtain the nearly optimal solution under the speed requirement of real-time scheduling.

3 System Modeling

As shown in Fig. 1, we assume that our system architecture is composed of a certain number of Fog nodes (consist of Region Point of Presences (RPOPs), Local Point of Presences (LPOPs) and Gateways), data centers, IoT equipment such as sensors and a set of services. A Region Point of Presence (RPOP) covers a geographical zone in ISP infrastructures and a Local Point of Presence (LPOP) is assigned to a Remote Point of Presence (RPOP) in the hierarchical system. Services can be deployed on any of the data centers, Fog nodes or IoT equipment. Generally, the service on IoT equipment will generate the raw data, which will be processed by the services on Fog nodes. Services can be divided into two types, data producers, denoted by $DP = \{dp_1, \dots, dp_i, \dots, dp_n\}$, and data consumers, denoted by $DC = \{dc_1, \dots, dc_k, \dots, dc_q\}$ (note that a service can be both producer and consumer). Every Fog node which has storage capacity can be the data hosts, denoted by $DH = \{dh_1, \dots, dh_j, \dots, dh_m\}$. All the data produced by data producers must be stored on at least one of the data hosts and it takes a certain amount of time to transmit those data to the data hosts (in single replica placement model proposed in [8], the produced data is only stored on one data host). The data produced by data producer dp_i is denoted by $data_i$.Note that if the services are the producers but not deployed on sensors or other IoT equipment which can produce the data themselves, they must have the consumption relation with other services (i.e., these services

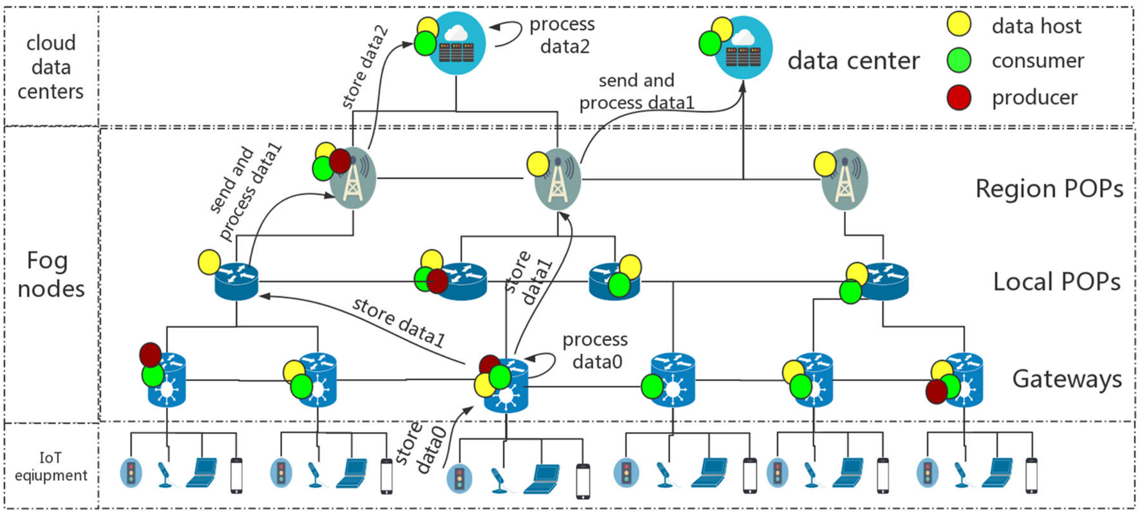


Figure 1 Architecture of multi-replica Fog storage system.

are consumer services too and consume the data transmitted from other services). Each consumer has a subscription relationship with the data generated by the producer, which is denoted by pc_{ik}. A simplified multi-replica data flow example is shown in Fig. 1. Note that every service deployed on IoT equipment is a producer (we do not mark them so in the figure for the sake of clarity).

Under the above architecture, the model named iFogStorM is proposed, aiming to solve the multi-replica data placement problem (see the main notation in Table 1). Before presenting the model in detail, the following constraint conditions are first given.

Table 1 Notation Dictionary.

Notation	Description
n	The number of data producers
m	The number of data hosts
q	The number of data consumers
$data_i$	The data which is produced by dp_i
dp_i	The data producer
dh_j	The data host
dc_k	The data consumer
c_{ijk}	dc_k gets $data_i$ from dh_j
$stor_j$	The storage capacity of dh_j
cp	The number of consumers that subscribes the same data
b	Minimum data exchange granularity
lr_{jk}	Time required for dc_k to get b from dh_j
lw_{ji}	Time required for dp_i to transmit b to dh_j
tr_{ijk}	Time required for dc_k to get $data_i$ from dh_j
tw_{ji}	Time required for dp_i to transmit $data_i$ to dh_j
x_{ij}	A replica of $data_i$ places on dh_j
s_i	The size of $data_i$
pc_{ik}	dc_k subscribes $data_i$

1) c_{ijk} denotes that the data consumption relation, i.e., if dc_k obtains (consumes) $data_i$ from dh_j then c_{ijk} is set to 1 else c_{ijk} is set to 0. There is at least one replica of $data_i$ being placed on the host nodes:

$$\sum_j^m \sum_k^q c_{ijk} \geq 1 \quad \forall i \tag{1}$$

2) x_{ij} denotes that a replica of $data_i$ placed on dh_j.In the case where there exists a consumer obtaining $data_i$ from dh_j, there must be a replica of data$_i$ placed on dh_j:

$$x_{ij} = \begin{cases} 1, & \sum_k^q c_{ijk} \geq 1 \\ 0, & \sum_k^q c_{ijk} = 0 \end{cases} \tag{2}$$

And (2) is equal to (3), (4), (5)

$$x_{ij} \geq c_{ijk} \quad \forall i, \forall j, \forall k \tag{3}$$

$$x_{ij} \leq \sum_k^q c_{ijk} \quad \forall i, \forall j \tag{4}$$

$$x_{ij} \epsilon \{0, 1\} \quad \forall i, \forall j \tag{5}$$

3) $stor_j$ denotes the storage capacity of dh_j and s_i denotes the size of $data_i$.The total storage usage of all the data replicas placed on dh_j must be smaller than the storage capacity:

$$\sum_i^n s_i{}^* x_{ij} \leq stor_j \forall j \tag{6}$$

4) pc_{ik} denotes the subscription relationship between dc_k and $data_i$.When $pc_{ik} = 1$, which means that there is a

subscription relationship between dc_k and $data_i$, there must be at least one replicas of $data_i$ placing in the fog nodes and dc_k choose one of the fog nodes which has replica of $data_i$ to obtain the data .Thus, in this case, one of the c_{ijk} when $j \in m$ is equal to 1 .When $pc_{ik} = 0$, dc_k needn't to obtain $data_i$, so in this case, all of the c_{ijk} when $j \in m$ is set to 0.

$$\sum_j^m c_{ijk} = pc_{ik} \quad \forall i, \forall k \tag{7}$$

5) lr_{jk} denotes the time required for dc_k to obtain b from dh_j and lw_{ji} denotes the time required for dp_i to transmit b to dh_j. tr_{ijk} denotes the time required for dc_k to obtain $data_i$ from dh_j and ts_{ji} denotes the time required for dp_i to transmit $data_i$ to dh_j.The relationships between them are:

$$tr_{ijk} = \frac{1}{b} \cdot s_i \cdot lr_{jk} \tag{8}$$

$$ts_{ji} = \frac{1}{b} \cdot s_i \cdot lw_{ji} \tag{9}$$

We define the overall latency as the sum of the required time of storing data on the data host and the time of transferring them to the data consumers. Note that the more replicas are placed on fog nodes, more required time is needed to store data while probably (not definitely) the less time of transferring them to the consumers is needed because the superfluous replicas which are not data providers of any data consumers would not be beneficial to decrease the time of data consumers to achieve the data they need. Since there is a negative correlation between the data transmitting time from data producers to data hosts and that from data hosts to data consumers, there is a trade-off between them and that's the reason we set overall latency as the scheduling objective. Moreover, the number of replicas of a piece of data is affected by the number of consumers that subscribe the specified data (which is denoted by cp) and in general, the bigger cp is, the more replicas would be needed to achieve our goal of minimizing the overall latency. The required time is calculated according to the minimum latency between nodes as shown in (8) and (9). The problem can be modeled as follows:

Table 2 Experimental Parameters.

S_I	96 bytes
S_O	960 bytes
N_s	15
t_p	1000 ms
N_p	10
T_{sim}	600 s

Table 3 Network Latency.

Network link	latency(ms)
IoT-GW	10
GW-LPOP	50
LPOP-RPOP	5
RPOP-DC	100
RPOP-RPOP	5
DC-DC	100

$$Minimize \sum_i^n \sum_j^m \left(\sum_k^q \left(c_{ijk} \cdot tr_{ijk} \right) + x_{ij} \cdot ts_{ji} \right) \tag{10}$$

$$s.t. \quad \sum_j^m \sum_k^q c_{ijk} \geq 1 \quad \forall i \tag{10a}$$

$$x_{ij} \geq c_{ijk} \quad \forall i, \forall j, \forall k \tag{10b}$$

$$x_{ij} \leq \sum_k^q c_{ijk} \quad \forall i, \forall j \tag{10c}$$

$$\sum_i^n s_i{}^* x_{ij} \leq stor_j \forall j \tag{10d}$$

$$\sum_j^m c_{ijk} = pc_{ik} \quad \forall i, \forall k \tag{10e}$$

$$c_{ijk} \epsilon \{0, 1\} \quad \forall i, \forall j, \forall k \tag{10f}$$

$$x_{ij} \epsilon \{0, 1\} \quad \forall i, \forall j \tag{10g}$$

4 MultiCopyStorage: A Greedy Data Placement Strategy

4.1 Overview of the Greedy Data Placement Strategy

Similar to the data allocation (or resource management) model proposed in [24–26], iFogStorM can't be solved in polynomial time and it is a MILP model. Thus solving the formulated model is NP-hard. It will take an unbearable amount of time to solve the model by using the mathematical programming optimizer such as CPLEX [10]. In order to solve the model efficiently,

Table 4 Data Host Storage Capacity.

Data host	Storage capacity
GW	1GB
LPOP	100GB
RPOP	1 TB
DC	1 PB

in this paper we design a few heuristic rules and propose a latency-aware strategy to dramatically reduce the searching space for solving the target model. In particular, the greedy strategy MultiCopyStorage, which is based on iFogStorM model, is proposed in this section. MultiCopyStorage is a greedy strategy for solving the problem of data placement in the Fog Computing area, which can greatly improve the solving speed while maintaining the quality of the solution when the problem scale is large.

In the iFogStorM model, after determining the placement of all the replicas, it is also necessary to determine which host the consumer obtains the data from. In the MultiCopyStorage strategy, a heuristic rule is added: all consumers get the corresponding data from the host that stores the subscribed data replica with the lowest latency, so as to reduce the scope of searching for good solutions. It is a greedy algorithm in essence. In turn, every piece of data is placed in the local optimal placement solution with the lowest overall latency. After obtaining all the local placement solutions of every piece of data, the final solution is obtained. The procedure of finding the local optimal solution for every piece of data is a recursive procedure. Starting with the case of one replica, the recursion is carried out level by level until the latency cannot be reduced even if the number of replicas increases. Moreover, a few heuristic pruning strategies are used to control the number of recursion levels.

In summary, the following heuristic rules are applied when solving the model:

1. The consumers select the host which holds the replica of data that they subscribe to and offers the least latency as the node to obtain the subscribed data from.
2. When the number of replicas is higher than the number of consumers for that data, it deemed impossible to have the optimal solution with the minimum overall latency because of the extra write latency of the replicas that not being used by any of the consumers.
3. In every recursion level, only the nodes that reduce the overall latency after adding to the temporary solution are considered as the candidate nodes for the next recursion level.

4.2 The Algorithm Details

In this subsection, we present the greedy algorithm, MultiCopyStorage, and discuss it in detail.

MultiCopyStorage is shown in Algorithm 1. In the algorithm, for each data that needs to be placed, its candidate nodes are initialized to be all host nodes (line 4), and then the recursive procedure (line 6) is performed in order. After each recursive solution, the idle capacity of the host where the data is placed is updated (lines 7-9), and then the placement result of this data is added to the final result (line 10).

The recursive process uses the RecursiveSolve() function to solve the problem. When the number of recursive layers (which indicates the number of replicas) becomes more than the number of consumers subscribed to each data, adding more replicas of the data will only increase the write latency and will not reduce the read latency. Therefore, returning the current placement and latency cost (lines 1-3) is a better choice. In lines 7-17, we try to choose among candidate hosts the best node to place a new replica of the data on. To achieve this, we calculate the latency of different placement situations (line 10), add the host to the set of candidate hosts for next recursive level (line 12), and record the placement solution with the minimum latency (line 13-14). If there is at least one placement solution which offers lower latency than the currently best solution, the placement solution with the minimum latency is recursively passed to the next level (lines 18-21). The best placement solution and its minimum latency at the current level are updated after the recursion is completed (lines 22-25).

In order to clarify our algorithm, we illustrate the RecursiveSolve() procedure which is the core-part of the algorithm in detail. At first, the candidates are initialized to all the data hosts in the system architecture. After initialization, we iterate all the candidates and calculate the score of temporary solutions when adding the corresponding candidate to the data hosts of the to-be scheduled data. Then we record the solution whose score is the smallest and add the solutions which are lower than the minimum score of the previous recursion level to the next candidates set. Note that adding a replica would cause the increase of transmitting time from data producers to data hosts (let's call it write latency), so there must be several consumers which originally get data from another replica in the previous recursion level changing their original data provider. They decide to obtain data from the newly added replica because it is closer. And the increase of write latency can be offset by reducing read latency for several consumers (time for transmitting from data hosts to data consumers). In other words, the new added replica "steal" the consumers from other replicas. And for those replicas fail to "steal" enough consumers from other replicas in the current recursion level, they can't be able to offset the write latency in the next recursion level too. Thus, some hosts which have a possibility to help obtaining a lower score are put into the next candidates set. And those not in next candidates set would not be considered in the next recursion level. After the iteration of candidates set, we greedily take the temporary solution with minimum score as currently best solution and go into the deeper recursion level in order to find out that if the score could be lower when the number of replicas is bigger.

Algorithm 1 MultiCopyStorage

Input: Set of data needed to be sent, V; Set of consumers of $data_i$, $cons_i$; Number of consumers of each data, $ConsNum$; Max value of numeric, $MaxValue$; Data Size of $data_i$, s_i; Read Lentency of consumer k to host j, lr_{jk}; Write Lentency of data i to host j ,lw_{ji}; Base size of packet ,b;

Output: Placement of data, $result$

```
1: function MultiCopyStorage()
2:     $result \leftarrow \emptyset$
3:     $choose \leftarrow \emptyset$
4:     $hosts \leftarrow allPossibleHosts$
5:     for $data_i \in V$ do
6:         $solution_i$,$score_i$=recursiveSolve($data_i$,$cons_i$,$hosts$,$choose$,$MaxValue$,0)
7:         for select Host $h \in solution$ do
8:             update free Capacity of $h$
9:         end for
10:        add $solution_i$ to $result$
11:    end for
12:    return $result$
13: end function
```

```
1: function recursiveSolve($data_i$,$cons_i$,$candidates$,$choose$,$score$,$depth$)
2:     if $depth == ConsNum$ then
3:         return $choose$,$score$
4:     end if
5:     $nextCandidates \leftarrow \emptyset$
6:     $minScore = MaxValue$;
7:     $minHost = -1$;
8:     for host $h \in candidates$ do
9:         if host $h$ has free Capacity to place $data_i$ then
10:            $tempChoose \leftarrow choose \cup h$
11:            $tempScore = calculateScore(data_i, tempChoose, cons_i)$
12:            if $tempScore < score$ then
13:                add $h$ to $nextCandidates$
14:                if $tempScore < minScore$ then
15:                    $minScore = tempScore$
16:                    $minHost = h$
17:                end if
18:            end if
19:        end if
20:    end for
21:    if $minHost \neq -1$ then
22:        $nextChoose \leftarrow choose \cup minHost$
23:        $retChoose$,$retScore$=recursiveSolve($data_i$, $cons_i$, $nextCandidates$,
24: $nextChoose$, $minScore$,$depth + 1$)
25:        if $retScore < score$ then
26:            $score = retScore$
27:            $choose = retChoose$
28:        end if
29:    end if
30:    return $choose, score$
31: end function
```

```
function CALCULATESCORE($i$, $choose$, $cons_i$)
    $score = 0$
    $packetNum = s_i/b$
    for $host_j \in choose$ do
        $score = score + packetNum * lw_{ji}$
    end for
    for consumer $k \in cons_i$ do
        $min = MaxValue;$
        for host $j \in choose$ do
            if $lr_{jk} < min$ then
                $min = lr_{jk}$
            end if
        end for
        $score = score + packetNum * min$
    end for
    return $score$
end function
```

The latency is calculated by the function CalculateScore(). It calculates the write latency in lines 4-6 and the read latency in lines 7-15. When calculating the read latency, the algorithm traverses the consumers, calculates the read latency from the consumer to the nearest host, where the data replica is placed and the corresponding time is added to the overall latency.

Assuming that *cp* denotes the number of consumers that subscribe to the same data, and *n*, *m* and *q* respectively represent the number of producers, storage nodes and consumers. The time complexity of the function, CalculateScore(), which calculates the overall latency of every solution is $cp \cdot m$. The function RecurceiveSolve() calls calculateScore() m times in each recursive level. The max number of recursive levels is *cp*. Therefore, the time complexity of this method is $cp^2 \cdot m^2$.Finally, the main function, MultiCopyStorage (), must call RecurceiveSolve () for each data. The total number of calls is *n*. Thus, the time complexity of the algorithm is $n \cdot cp^2 \cdot m^2$.

With the exhaustive search method, the time complexity is non-polynomial. Assuming that in the extreme case, the number of replicas of every piece of data is from 1 to *m* and those replicas can be placed on every Fog node. Then, there are $n \cdot \sum_l^m C_m^l$ different placement solutions, and $n \cdot \sum_l^m C_m^l$ is approximately equal to $n \cdot (2^m - 1)$. As a result, the exhaustive search method would not be an efficient method and the time it takes would exponentially increase with the growth of *m*.

5 Simulation and Performance Evaluation

5.1 Experimental Settings

In order to verify the validity of MultiCopyStorage strategy, the extended iFogSim [2] is used for the simulation experiments. iFogSim is a simulation platform for simulating fog computing environment, which is extended in [27] by Naas et al. to support the implementation of data placement strategies in the fog computing environment. The fog nodes consist of gateways(GW), Local PoPs(LPOP) and Regional PoP(RPOP), forming the simulated network topology together with data centers and sensors.

1000 GWs, 50 LPOP, 10 RPOP and 5 data centers were used in the simulation environments. The packet size produced by the services deployed on IoT equipment and by other services are set to S_I and S_O, respectively. Each GW is connected with N_s sensors, which sends a packet every t_p milliseconds. Consumers that consume data from sensors send a packet after receiving and processing N_p packets while other consumers send one packet once they have finished processing one packet from their data providers.

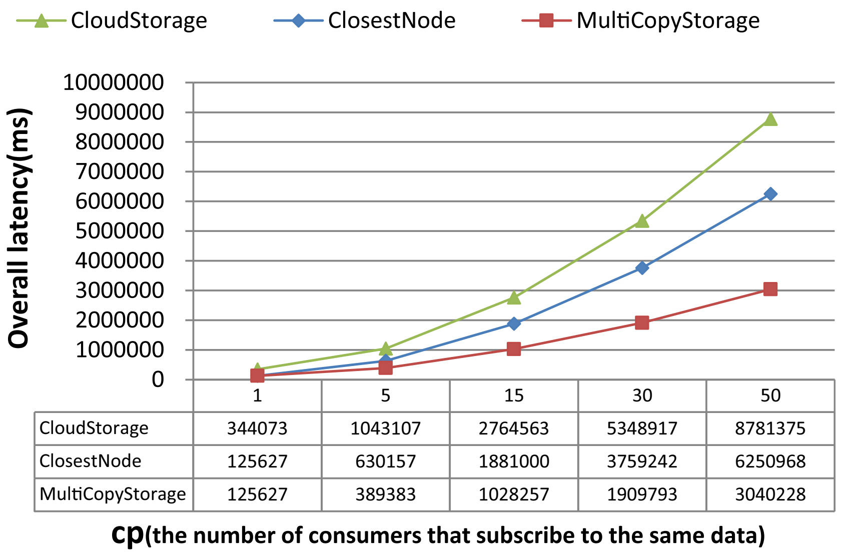
Table 2 shows the default values of the experimental parameters.

Table 3 shows the network latency between different Fog nodes while Table 4 shows the capacity of Fog nodes. The evaluation experiments are conducted on a 16-core CPU and 32GB memory machine.

Two types of different workload are used in the evaluation:

1. Zoning workload: the consumers are situated in the same geographical zone as the producers which produce the data they subscribe to. A zone is a covered by at least one RPOP.
2. Distributed workload: every service, wherever they are located, may be the consumer of the data.

The simulation time is set to be T_{sim}. The number of consumers that subscribe to the same data is *cp*. In order to evaluate the overall latency, we conduct the experiments



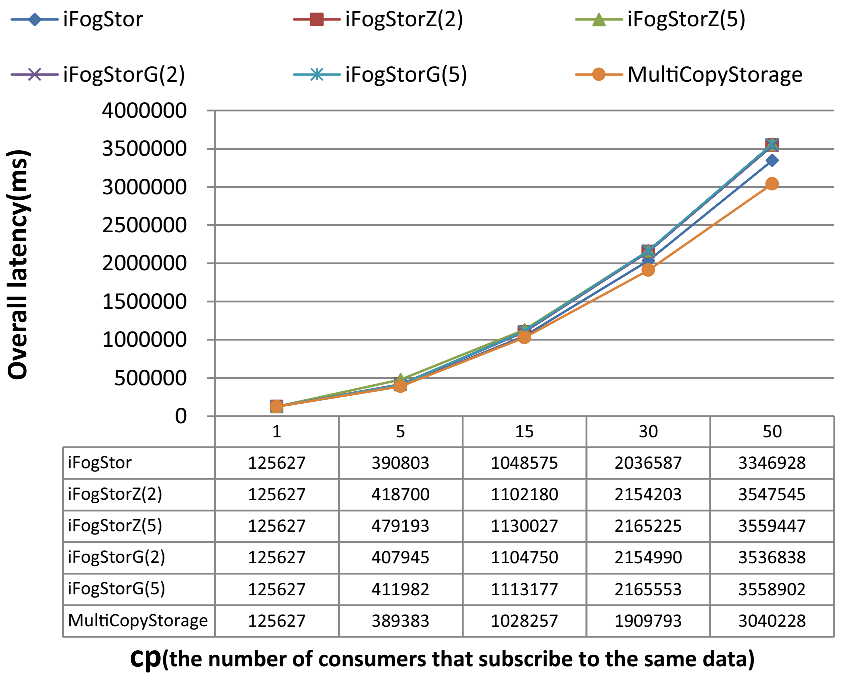
	1	5	15	30	50
CloudStorage	344073	1043107	2764563	5348917	8781375
ClosestNode	125627	630157	1881000	3759242	6250968
MultiCopyStorage	125627	389383	1028257	1909793	3040228

Figure 2 Overall latency of Different strategies on Distributed workload.

to compare the MultiCopyStorage strategy with CloudStorage [8], ClosestNode [8], iFogStor [8], iFogStorZ [8] and iFogStorG [9] when *cp* = 1,5,15,30,50. The CloudStorage strategy [8] is used to store all data replicas on the cloud computing center while ClosestNode [8] to place all data replicas on the nearest storage node. In order to test the running speed of the strategy, we test the running time of iFogStor [8],iFogStorZ [8] and iFogStorG [8] under *cp* = 5, 30, 50 and GWs = [500,3000] and obtain the average value. We have also utilized other optimization toolkits (e.g. Lingo [28]) to test the running speed of iFogStor. We figure out that the running speed of Lingo is almost the same as Cplex [10] to solve the MILP model and they can both precisely obtain the optimal solution. Since Cplex provides a usable java interface while Lingo doesn't, in the latter experiments, we only utilize Cplex as the optimizer for iFogStor [8],iFogStorZ [8] and iFogStorG [8].

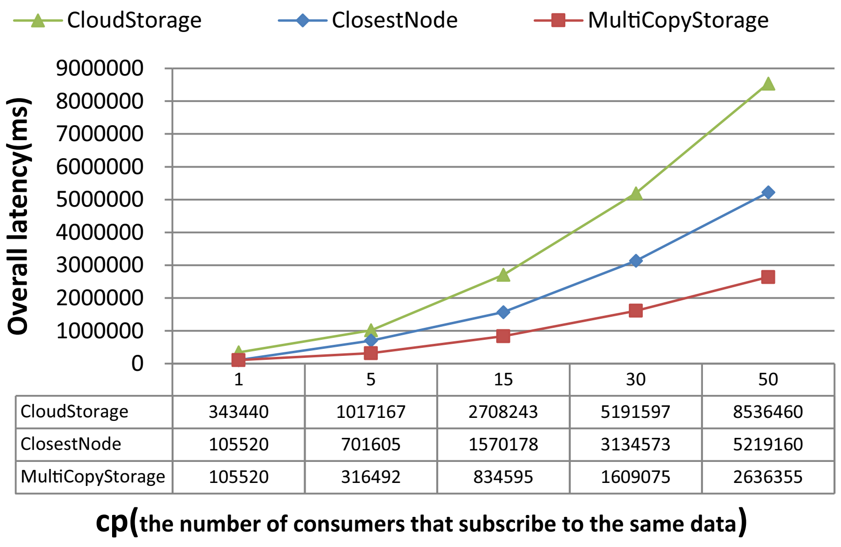
5.2 Experimental Results

In the experiments regarding the overall latency, the latency of data transmission is added to the current value of overall latency after each experiment run is completed. The solving time is obtained by taking the difference between the start time



	1	5	15	30	50
iFogStor	125627	390803	1048575	2036587	3346928
iFogStorZ(2)	125627	418700	1102180	2154203	3547545
iFogStorZ(5)	125627	479193	1130027	2165225	3559447
iFogStorG(2)	125627	407945	1104750	2154990	3536838
iFogStorG(5)	125627	411982	1113177	2165553	3558902
MultiCopyStorage	125627	389383	1028257	1909793	3040228

Figure 3 Overall latency of Different strategies on Distributed workload.



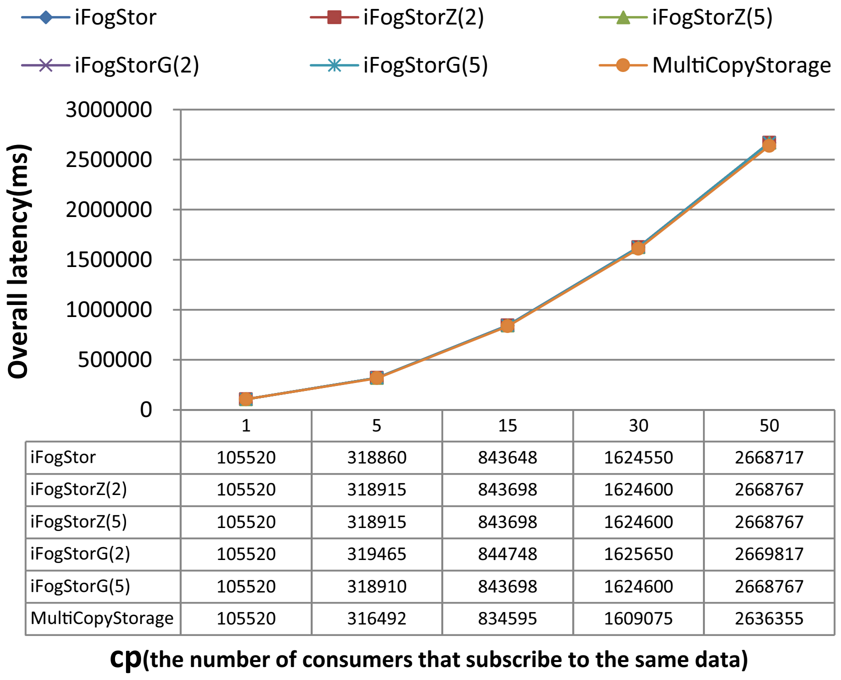
	1	5	15	30	50
CloudStorage	343440	1017167	2708243	5191597	8536460
ClosestNode	105520	701605	1570178	3134573	5219160
MultiCopyStorage	105520	316492	834595	1609075	2636355

Figure 4 Overall latency of Different strategies on Zoning workload.

and the end time of each experimental run. The experiment results are shown in Figs. 2, 3, 4, 5.

1) Overall latency: Fig. 2 shows that the latency achieved by MultiCopyStorage with distributed workload is 63.7% lower on average than that by the CloudStorage strategy, and 36.8% lower than the ClosestNode strategy. Figure 4 shows that the latency of the MultiCopyStorage strategy with zoning workload is lower by 69.1% and 40.0% than that of CloudStorage and ClosestNode, respectively. It can be seen that compared with CloudStorage, both ClosestNode and MultiCopyStorage reduce the overall latency significantly. The reason for this is because in CloudStorage, the data need to be transferred back and forth between the producers and data centers. Since the producers and the consumers often have a non-negligible number of network hops to the data centers, it is a better decision to place the data on the Fog nodes, which are closer to producers and consumers.

Figure 3 shows the overall latency of different strategies on distributed workload. When *cp* = 15, 30 and 50, the latency of MultiCopyStorage drops by 1.94%,6.23% and 9.16%, respectively compared with iFogStor and drops by 7.63%, 11.81% and 14.57%, respectively, compared with iFogStorG(5). One can observe that comparing with the single replica based models



	1	5	15	30	50
iFogStor	105520	318860	843648	1624550	2668717
iFogStorZ(2)	105520	318915	843698	1624600	2668767
iFogStorZ(5)	105520	318915	843698	1624600	2668767
iFogStorG(2)	105520	319465	844748	1625650	2669817
iFogStorG(5)	105520	318910	843698	1624600	2668767
MultiCopyStorage	105520	316492	834595	1609075	2636355

Figure 5 Overall latency of Different strategies on Zoning workload.

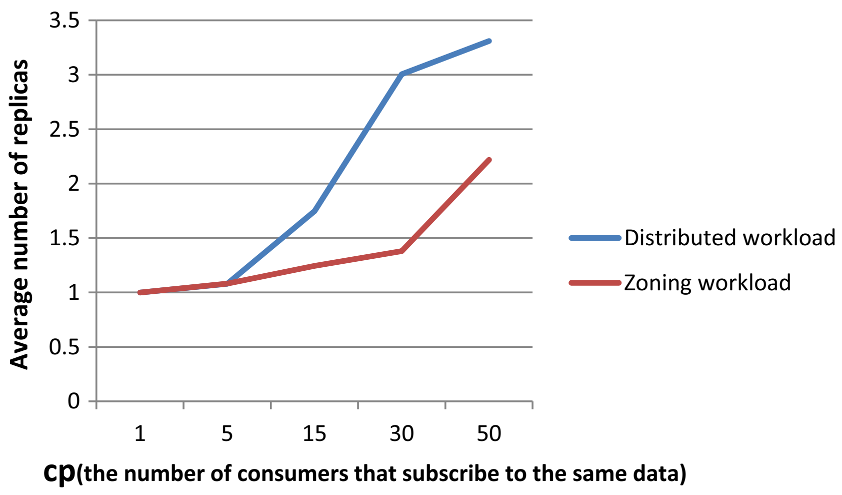


Figure 6 Average number of replicas of MultiCopyStorage when GWs = 1000.

(e.g. iFogStor, iFogStorZ, iFogStorG), the overall latency of our strategy with distributed workload decreases as *cp* increases. This is due to the fact that MultiCopyStorage places replicas of data on multiple hosts. When the number of consumers that subscribe to the same data is large, placing multiple replicas on different hosts increases the corresponding writing latency slightly. However, the consumers can read the data with much lower reading latency and consequently reduce the overall latency compared to the single replica strategy. With Zoning workload, as shown in Fig. 5, the benefit of MultiCopyStorage declines slightly. When *cp* = 15, 30 and 50, the overall latency reduces by 1.07%, 1.14% and 1.21%, respectively, comparing with iFogStor. We can observe that the latency of MultiCopyStorage decreases slightly compared to other strategies. This is because when the consumers are confined to a particular area, a single replica is enough to minimize the overall latency effectively. When the consumers are distributed, placing multiple replicas across the network topology manifests more prominent impact.

2) Average replicas number: Fig. 6 shows the average number of replicas of each piece of data achieved by MultiCopyStorage. It can be observed that the number of replicas climbs with the increase of *cp*. Generally, the number of replicas is bigger on Distributed workload than on Zoning workload since the consumers of each data are distributed on distributed workload and the more replicas are required to lessen the transmit time of extracting the data. Moreover, one cannot discuss a replication strategy without considering the overhead of replicating the data since more replicas in the system accounts for greater overhead in terms of additional network traffic and data storage cost. Nevertheless, the average number of replicas shown in Fig. 6 is acceptable since it is indirectly constrained by our objective function. With the constraint of replicas number, the overhead could be controlled.

3) Solving time: As shown in Fig. 7, compared with MultiCopyStorage, the solving time of iFogStorZ(5),iFogStorG(5) and iFogStorG(10) strategy increases significantly with the increase of GWs. Compared with iFogStorZ(5),iFogStorG(5) and iFogStorG(10), the solving time of MultiCopyStorage decreases by 83.69%,85.75% and 68.18%, respectively, when GW = 3000 and *cp* = 5. Meanwhile, Fig. 9 illustrates that when *cp* = 50 and GW = 3000, the solving

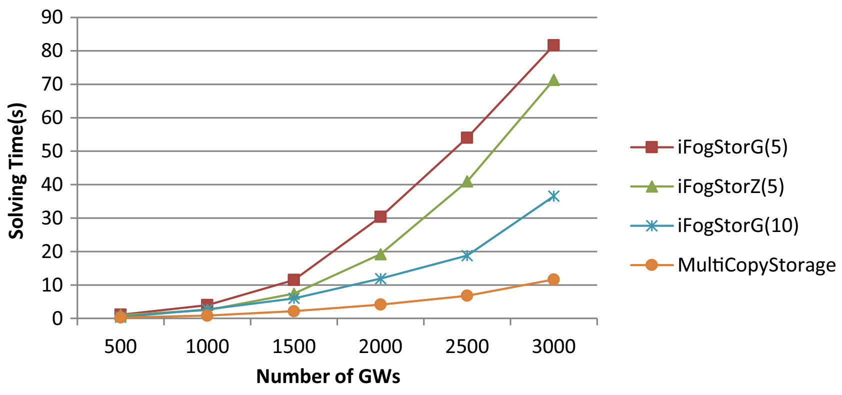


Figure 7 Solving time of different strategies when *cp*=5.

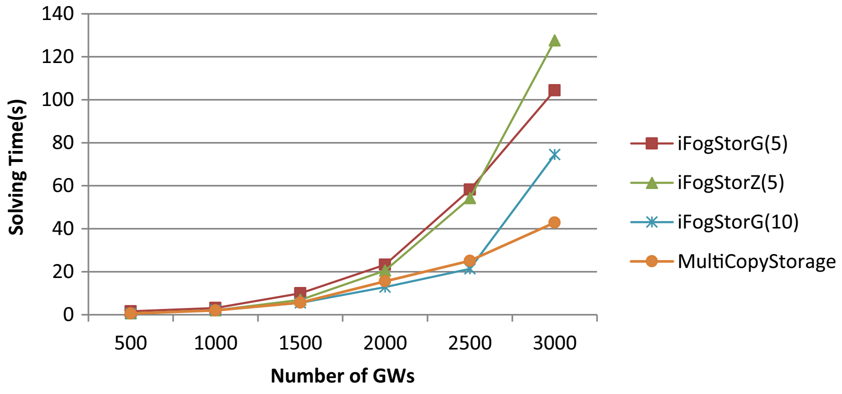


Figure 8 Solving time of different strategies when cp=30.

time of MultiCopyStorage is lower than iFogStorZ(5), iFogStorG(5) and iFogStorG(10) by 39.01%, 50.69% and 39.38% respectively. As can be seen from Figs. 7, 8 and 9, with the increase of cp, it takes MultiCopyStorage more time to solve the problem.

In conclusion, although iFogStor, based on a linear model, can always get the optimal solution of the single replica placement problem, when cp is large and the consumers are widely distributed the single replica strategy fails to enable the distributed consumers to obtain the data with a low latency. In contrast, MultiCopyStorage can reduce the overall latency to some extent. As cp increases, the reduction in overall latency becomes more prominent, although more time is needed to solve the problem. In addition, although the graph partitioning strategies e.g. iFogStorZ and iFogStorG reduce the complexity of the linear model to a certain extent, each partition still uses the CPLEX [10] to solve the problem, which is very time-consuming compared with the MultiCopyStorage strategy. It can be seen from the experiment results that MultiCopyStorage is superior to other single replica strategies in both solution quality and efficiency. This is because of the fact that the MultiCopyStorage strategy is based on the multiple data replicas placement model, which considers the drawback of single replica model. Furthermore, some heuristic rules are applied when solving the problem, which greatly reduces the solving time.

6 Conclusions and Future Work

In this paper, iFogStorM, a multiple data replicas placement model for Fog Computing is proposed to reduce the overall latency of data storage. Finding the optimal solution of this model is a NP-hard problem. Therefore, based on iFogStorM, we propose a greedy strategy called MultiCopyStorage, which applies a number of heuristic rules and greatly reduces the scope of searching for good solutions. In order to verify the proposed strategy, we extended iFogSim [2] to implement the real-time simulation of the multiple data replicas placement strategy, and conducted the experiments to compare our strategy with the latest single replica strategies. The experiment results show that MultiCopyStorage reduces the overall latency by 6% on average compared with FogStorage [8], and by 10% compared with the heuristic strategy iFogStorG [9]. Because of its high efficiency, MultiCopyStorage can be operated at a high rate and offers good data storage solutions at real time in the fog computing environment.

As for the data placement problem in Fog Computing, this paper only considers the resource capacity as the limiting factor. Basically, the more replicas involve greater overhead in terms of additional network traffic, data storage cost, CPU and memory consumption due to transmission. In our study, the number of replicas is indirectly constrained by minimizing the overall latency but with the larger cp (the number of

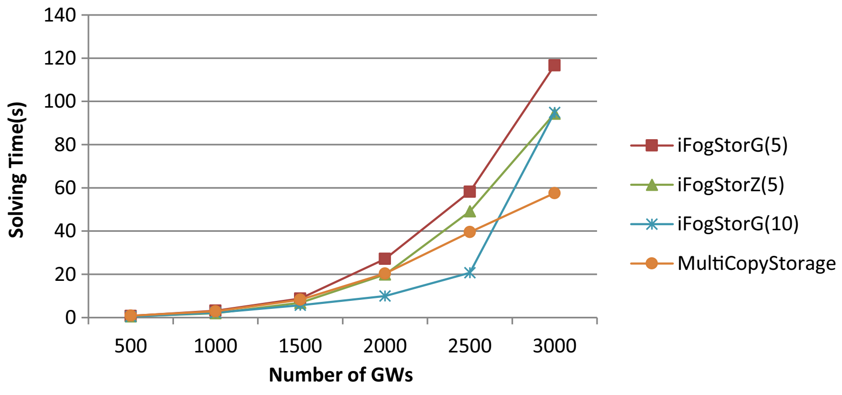


Figure 9 Solving time of different strategies when cp=50.

consumers that subscribe to the same data), the number of replicas increases and the overhead would climb too. Hence, there should be a trade-off between the performance in reducing the overall latency and the number of replicas. We will further investigate this problem in future work. In addition, the fixed latency is used for every network link when calculating the overall latency, which does not consider the latency changes due to network congestion. In addition, we do not consider the stability of fog nodes. Thus, a trust evaluation mechanism [29] may be introduced to prevent the cases like device faults, network congestion. Furthermore, since cyber threats are growing up with the development of cloud technology, some Fog-based storage security scheme should be proposed to ensure the integrity, availability and confidentiality of the data preserving in fog nodes [30, 31]. In the future, we will incorporate these factors and further improve the existing model.

Acknowledgments This work is supported by National Natural Science Foundation of China (Grant Nos. 61772205, 61872084), Science and Technology Planning Project of Guangdong Province (Grant Nos. 2017B010126002, 2017A010101008, 2017A010101014, 2017B090901061, 2016A010101018 and 2018KJYZ009), Guangzhou Science and Technology Projects (Grant Nos. 201802010010, 201807010052 and 201610010092), Nansha Science and Technology Projects (Grant No. 2017GJ001), Special Funds for the Development of Industry and Information of Guangdong Province (big data demonstrated applications) in 2017, and the young teachers training of Guangdong police officer college(2018QNGG06).

References

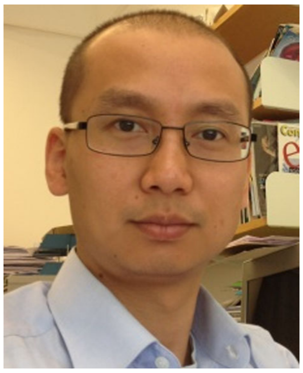
1. Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions). https://www.statista.com/statistics/471264/iot-number-ofconnected-devices-worldwide/.
2. Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K., & Buyya, R. (2017). iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience, 47*(9), 1275–1296.
3. Wang, T., Bhuiyan, M. Z. A., Wang, G., Rahman, M. A., Wu, J., & Cao, J. (2018). Big data reduction for a smart city's critical infrastructural health monitoring. *IEEE Communications Magazine, 56*(3), 128–133.
4. Bonomi, F., Milito, R., Zhu, J., & Addepalli, S. (2012). Fog computing and its role in the internet of things. In Proceedings of the first edition of the MCC workshop on Mobile cloud computing (pp. 13-16). ACM.
5. Mahmud R., Kotagiri R., & Buyya R. (2018). Fog computing: A taxonomy, survey and future directions. In *Internet of everything* (pp. 103–130). Singapore: Springer.
6. Fan, Y., Chen, J., Wang, L., & Cao, Z. (2016). Energy-efficient and latency-aware data placement for geo-distributed cloud data centers. In *International Conference on Communications and Networking in China* (pp. 465–474). Cham: Springer.
7. Zheng, P., Cui, L. Z., Wang, H. Y., & Xu, M. (2010). A data placement strategy for data-intensive applications in cloud. *Jisuanji Xuebao (Chinese Journal of Computers), 33*(8), 1472–1480.
8. Naas, M. I., Parvedy, P. R., Boukhobza, J., & Lemarchand, L. (2017). iFogStor: an IoT data placement strategy for fog infrastructure. In Fog and Edge Computing (ICFEC), 2017 IEEE 1st International Conference on (pp. 97-104). IEEE.
9. Naas, M. I., Lemarchand, L., Boukhobza, J., & Raipin, P. (2018). A graph partitioning-based heuristic for runtime iot data placement strategies in a fog infrastructure. In SAC 2018: Symposium on Applied Computing.
10. CPLEX, I. I. (2009). V12. 1: User's manual for CPLEX. *International Business Machines Corporation, 46*(53), 157.
11. Mahmud, R., Ramamohanarao, K., & Buyya, R. (2018). Latency-aware application module management for Fog computing environments. *ACM Transactions on Internet Technology, 19*(1). https://doi.org/10.1145/3186592.
12. Taneja, M., & Davy, A. (2017). Resource aware placement of IoT application modules in Fog-Cloud Computing Paradigm. In Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on (pp. 1222-1228). IEEE.
13. Mahmud, R., Srirama, S. N., Ramamohanarao, K., & Buyya, R. (2018). Quality of Experience (QoE)-aware placement of applications in Fog computing environments. *Journal of Parallel and Distributed Computing*. https://doi.org/10.1016/j.jpdc.2018.03.004.
14. Mamdani, E. H., & Assilian, S. (1975). An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies, 7*(1), 1–13.
15. Achterberg, T. (2009). SCIP: solving constraint integer programs[J]. *Mathematical Programming Computation, 1*(1), 1–41.
16. Skarlat, O., Nardelli, M., Schulte, S., Borkowski, M., & Leitner, P. (2017). Optimized IoT service placement in the fog. *Service Oriented Computing and Applications, 11*(4), 427–443.
17. Sun, Y., Lin, F., & Xu, H. (2018). Multi-objective optimization of resource scheduling in Fog computing using an improved NSGA-II. *Wireless Personal Communications, 102*, 1369–1385.
18. Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation, 6*(2), 182–197.
19. Wang, T., Zhou, J., Liu, A., Bhuiyan, M. Z. A., Wang, G., & Jia, W. (2018). Fog-based computing and storage offloading for data synchronization in IoT. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2018.2875915.
20. Wei-Wei, L. (2012). An improved data placement strategy for Hadoop. *Journal of South China University of Technology (Natural Science Edition), 1*, 028.
21. Wu, J., Chen, L., Wang, X., Jiang, G., Lam, S. K., & Srikanthan, T. (2017). Algorithms for replica placement and update in tree network. *The Computer Journal, 61*(2), 273–287.
22. Lizhen, L. C., Zhang, J., Yue, L., Shi, Y., Li, H., & Yuan, D. A genetic algorithm based data replica placement strategy for scientific applications in clouds. *IEEE Transactions on Services Computing, 11*(4), 727–739.
23. Rajaretnam, K., Rajkumar, M., & Venkatesan, R. (2016). Rplb: A replica placement algorithm in data grid with load balancing. *International Arab Journal of Information Technology (IAJIT), 13*(6).
24. Gai, K., Qiu, M., & Zhao, H. (2016). Cost-aware multimedia data allocation for heterogeneous memory using genetic algorithm in cloud computing. *IEEE Transactions on Cloud Computing*. https://doi.org/10.1109/TCC.2016.2594172.
25. Qiu, M., Chen, Z., Ming, Z., Qin, X., & Niu, J. (2017). Energy-aware data allocation with hybrid memory for mobile cloud systems. *IEEE Systems Journal, 11*(2), 813–822.
26. Qiu, M., & Sha, E. H. M. (2009). Cost minimization while satisfying hard/soft timing constraints for heterogeneous embedded systems. *ACM Transactions on Design Automation of Electronic Systems (TODAES), 14*(2), 25.

27. Naas, M. I., Boukhobza, J., Parvedy, P. R., & Lemarchand, L. (2018). An Extension to iFogSim to Enable the Design of Data Placement Strategies. In Fog and Edge Computing (ICFEC), 2018 IEEE 2nd International Conference on (pp. 1-8). IEEE.
28. Schrage, L. E. (2006). *Optimization modeling with LINGO*. Chicago: Lindo System.
29. Wang, T., Zhang, G., Liu, A., Bhuiyan, M. Z. A., & Jin, Q. (2018). A secure IoT service architecture with an efficient balance dynamics based on cloud and edge computing. *IEEE Internet of Things Journal*. https://doi.org/10.1109/JIOT.2018.2870288.
30. Wang, T., Zhou, J., Huang, M., Bhuiyan, M. Z. A., Liu, A., Xu, W., & Xie, M. (2018). Fog-based storage technology to fight with cyber threat. *Future Generation Computer Systems, 83*, 208–218.
31. Hu, F., Qiu, M., Li, J., Grant, T., Taylor, D., McCaleb, S., et al. (2011). A review on cloud computing: Design challenges in architecture and security. *Journal of Computing and Information Technology, 19*(1), 25–55.

Tiansheng Huang is currently pursuing the B.S. degree with the Department of Computer Science and Engineering in South China University of Technology, China. His research interests include cloud computing and fog computing.

Weiwei Lin received his Ph.D. degree in Computer Application from South China University of Technology, China, in 2007. He is currently a Professor with the Department of Computer Science and Engineering, South China University of Technology. He has published more than 80 papers in refereed journals and conference proceedings. His research interests include distributed system, cloud computing and big data.

Yin Li is associate director of Institute of Software Application Technology, Guangzhou & Chinese Academy of Sciences, Guangzhou, China.



Ligang He received the Bachelors and Master degrees from the Huazhong University of Science and Technology, China, in 1998 and 2001 respectively. He received the PhD degree in Computer Science from the University of Warwick, UK, in 2006. He was also a Post-doctoral researcher at the University of Cambridge, UK. In 2006, he joined the Department of Computer Science at the University of Warwick as an Assistant Professor, and then became an Associate Professor. He has published more than 100 papers in journals (such as IEEE TPDS, ACM Computing Surveys, JPDC, JCSS) and conferences (such as IPDPS, ICPP,ICWS, VLDB). His areas of interest are parallel and distributed computing, high performance Computing and Big Data Analysis.

Shaoliang Peng works in College of Computer Science and Electronic Engineering, in Changsha (HNU, Changsha, China),and National University of Defense Technology (NUDT, Changsha,China), and is an adjunct professor of BGI. He was a visiting scholar at CS Department of City University of Hong Kong (CityU) from 2007 to 2008 and at BGI Hong Kong from 2013 to 2014. His research interests are high performance computing, bioinformatics, big data, AI, and biology simulation. He has participated in many keystone projects in China such as Tianhe supercomputers. He has published 5 books and over 100 papers in Science, Nature Communications, Cell AJHG, NAR, Bioinformatics, ACM/IEEE Transactions, and SCIENCE CHINA. He is Executive Editor and Associate Editors of several international journals (International Journal of Biological Sciences, Interdisciplinary Sciences: Computational Life Sciences, IJCSE, IJHPCN, and IJES). Moreover, he is the PI of several key projects including 973, 863 and National Natural Science Foundation of China (NSFC).