Computing and Informatics, Vol. 42, 2023, 311–339, doi: 10.31577/cai_2023_2_311

A PROPOSED SCHEDULING ALGORITHM FOR IOT APPLICATIONS IN A MERGED ENVIRONMENT OF EDGE, FOG, AND CLOUD

Xuan Thi Tran

University of Information and Communication Technology
Thai Nguyen University
Z115, Quyet Thang, Thai Nguyen, Vietnam
e-mail: ttxuan@ictu.edu.vn

Abstract. With the rapid increase of Internet of Things (IoT) devices and applications, the ordinary cloud computing paradigm soon becomes outdated. Fog computing paradigm extends services provided by a cloud to the edge of network in order to satisfy requirements of IoT applications such as low latency, locality awareness, low network traffic, mobility support, and so forth. Task scheduling in a Cloud-Fog environment plays a great role to assure diverse computational demands are met. However, the quest for an optimal solution for task scheduling in the such environment is exceedingly hard due to diversity of IoT applications, heterogeneity of computational resources, and multiple criteria. This study approaches the task scheduling problem with aims at improving service quality and load balancing in a merged computing system of Edge-Fog-Cloud. We propose a Multi-Objective Scheduling Algorithm (MOSA) that takes into account the job characteristics and utilization of different computational resources. The proposed solution is evaluated in comparison to other existing policies named LB, WRR, and MPSO. Numerical results show that the proposed algorithm improves the average response time while maintaining load balancing in comparison to three existing policies. Obtained results with the use of real workloads validate the outcomes.

Keywords: Cloud-fog computing, job service demand, load balance, job scheduling

Mathematics Subject Classification 2010: 68-M20

1 INTRODUCTION

With the rapid increase of Internet of Things (IoT) and its applications, the ordinary cloud computing paradigm faces a challenge to satisfy applications that require frequent data access, low latency, real-time interaction, high-speed communication, and so forth. Edge computing was introduced to support task computation on source devices of generated data. Generally, computational capacity is limited on edge nodes. Fog computing appears as an effective complement of Cloud center to cope with those issues by extending cloud services to the edge network [1, 2, 3, 4, 5].

Task scheduling in such distributed environment plays a great role to assure that diverse computational demands are met. However, the quest for an efficient and effective solution in merged computing environments is exceedingly hard due to diversity of IoT applications, heterogeneity of computational resources, and multiple criteria [3, 6, 7, 8]. In the literature, task scheduling has been studied with various desired factors of effective processing [9, 10, 11, 12], load balancing [13, 14] and/or power efficiency [15, 16, 17].

Energy cost contributes a significant factor to overall operation cost of large-scale computational systems. The use of dynamic power management (DPM) techniques has been crucial to achieve energy efficiency as addressed in [15, 16, 18] and references therein. Switching off technique (that is, switching idle servers off and only turning them back on when they are needed) has been addressed as an effective method of power/energy consumption management of Cloud systems [13, 15, 16, 17, 19, 20].The application of switching off technique in a merged, heterogeneous computing environment of Edge, Fog, Cloud has been studied in [13]. Authors proposed the use of load thresholds of computing resources in a subsystem to determine the number of active servers in the subsystem.

Due to the diversity of applications and the resource heterogeneity, we argue that characteristics of both user jobs and resources play a role in the system cost and performance. Therefore, scheduling policy should take into account job characteristics and resource capacity in order to improve the performance. In addition, we follow the resource utilization based approach given in [13] for load balance and energy efficiency. The contributions of this work are highlighted in what follows.

- This study argues that job service demand and resource processing capacity play a role for efficient computation;
- Load threshold based policy helps to avoid load stress at power-sensitive machines;
- Numerical results (both theoretical and traced workloads) address that our proposed algorithm improves both performance and load balance of a Cloud-Fog computing system, in comparison with three other existing algorithms named Weighted Round Robin, Load-Balance, and Modified Particle Swarm Optimization (MPSO) based heuristic.

- The energy cost of Cloud center is reduced with Load-Balance and our proposed algorithm at low load, while our proposal yields lower energy cost of Raspberry Pi cluster than Load-Balance does.

The paper is organized as follows. Section 2 gives a review on the literature works. Section 3 describes the system model and the proposed scheduling solution. Section 4 presents obtained numerical results and discussions. Finally, Section 5 concludes the paper.

2 RELATED WORK

This section is a brief review on the related works of task scheduling in Cloud-Fog computing environments. In [9], authors proposed a batch-mode task scheduling algorithm based on the relationship between a fog node set and a task set. Compared with existing batch-mode scheduling algorithms (MCT, MET, MIN-MIN), the proposal yields a shorter total completion time of tasks. Rafique et al. [15] focused on balancing task execution time and energy consumption at Fog layer computing resources; the proposed NBIHA algorithm resulted in resource utilization, average response time, and energy consumption but an increase in task execution time. A time-cost based scheduling algorithm was introduced in [11], wherein authors applied a set of modifications of Genetic Algorithm in their proposal. They showed that the time-cost based algorithm achieves a better trade-off between time and cost execution.

Xiang et al. [21] proposed a solution for mode selection and resource allocation with the aim to maximize the energy efficiency of the fog-RAN system. The proposed algorithm that is based on particle swarm optimization leads to energy savings – delay trade-off. Oueis et al. [22] introduced three variants of the algorithm that clusters small cells into computational clusters to process the users' requests, they and indicated that the power-centric solution results in a low energy consumption per user. In [23], a workload allocation policy was proposed to solve the trade-off problem between job execution delay and energy consumption.

Agarwal et al. [24] proposed an algorithm that allows efficiently distributing the workload over the fog and the cloud domains according to the available resources. Simulation results showed that the proposed algorithm is more efficient when compared to other existing strategies. Huedo et al. [25] focused on processing latency-critical application in Edge Cloud computing and proposed a platform model of Edge computing. Workload redistribution in the fog stratum was studied in [14]. The proposed framework focused on balancing between communication load and computation latency, taking into account the task redundancy and mobility.

In [26], authors considered a hierarchical architecture of cloud and fog and proposed a task scheduling policy, wherein real-time tasks are to be processed in the fog layer, while computational-intensive tasks are to be executed by cloud servers. Their study covered both time and fog energy consumption. However, the results

showed only the fitness value of the proposed algorithm, which does not represent comprehensively the system performance.

In scalable computing systems, load balancing refers to efficient use of computational resources for task execution. D. Tychalas et al. [13] proposed a load balancing solution wherein all available computing resources have been utilized. Authors showed that their proposed scheduling algorithm can improve the resource utilization and reduce energy costs at the cloud center in low load in comparison to a weighted round robin policy. Recently, artificial intelligence (AI) has attracted a great attention in the research of the task scheduling and resource management problem as shown in [27, 28, 29].

In this study, we partly take the load based scheduling approach in [13] to achieve load balance. Moreover, we also investigate the impact of job characteristics in terms of job service demand and task size model on the system performance.

3 SYSTEM MODELING AND PROPOSAL

3.1 Cloud Fog Computing Overview

Stand-alone cloud centers have become outdated for extremely heterogeneous IoT applications, of which a portion requires high-speed communication and quick response time. Edge computing paradigm represents the use of IoT devices for data storage and computation to avoid data transfer and retrieve near real-time processing. Normally edge devices have limited storage and computation capacity. Fog computing was born to enhance cloud services nearby IoT devices. Fog computing is not a replacement but a complement of cloud by extending cloud services to the edge of the network. To give a comprehensive look on the hierarchical, distributed environment of Cloud-Fog, we can consider a multi-tiered Cloud-Fog architecture (shown in Figure 1), including the following layers:

Edge layer: In the edge, end-user smart devices connect to the (IoT) gateways which provide various services of computation, local storage, data routing, security, and so forth.

Fog layer: Lying in the middle of the architecture, Fog layer represents a bridge between user devices in edge networks and cloud center. In fog layer, data and resource management are decided by the fog broker. Due to the increase of IoT applications and data, a solution of locating a small-scale version of cloud data centers geographically nearby users (so-called Cloudlet) becomes feasible to improve job response.

Cloud layer: The cloud represents the most available storage and computing capacity to provide big data storage, real-time and batch processing for generated data from IoT devices.

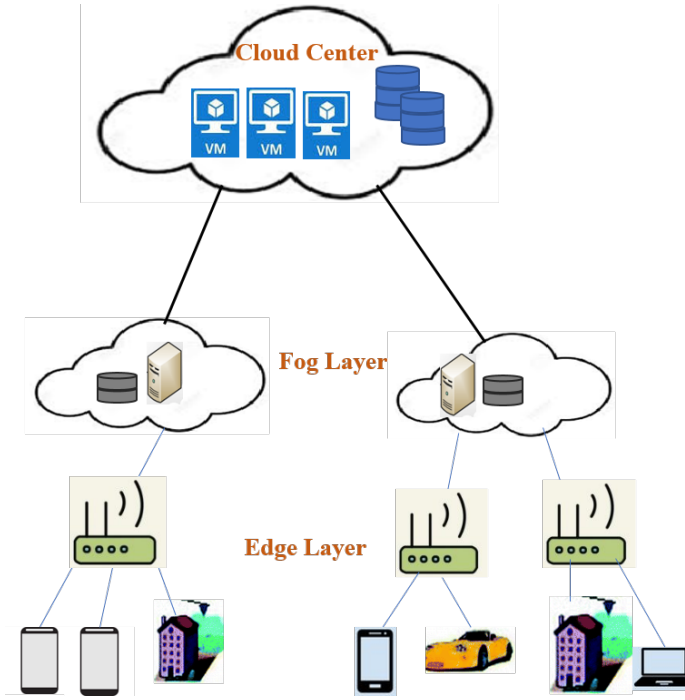


Figure 1. The Cloud-Fog architecture model

3.2 System Modeling

In the big picture of an IoT-enable Cloud computing environment, any device composed of processing capacity and storage in the network can be referred as a fog node. Therefore, there is a wide range of fog node types from end-user smart devices, low-performance gateways, powerful cloudlet servers, to virtual or physical machines at cloud center.

The considered computing system makes use of all available computational resources from four subsystems:

1. end-user smart devices,
2. low-performance network gateways using Raspberry Pi devices,
3. powerful cloudlet servers, and
4. VM pool at cloud center (as shown in Figure 2).

Let $M(i)$ $(i = 1, 2, 3, 4)$ be the number of nodes in subsystem i. We assume that nodes of subsystem i $(i = 1, 2, 3, 4)$ are homogeneous and have a service rate μ_i. Load Dispatcher is responsible to distribute incoming workloads to computational resources.

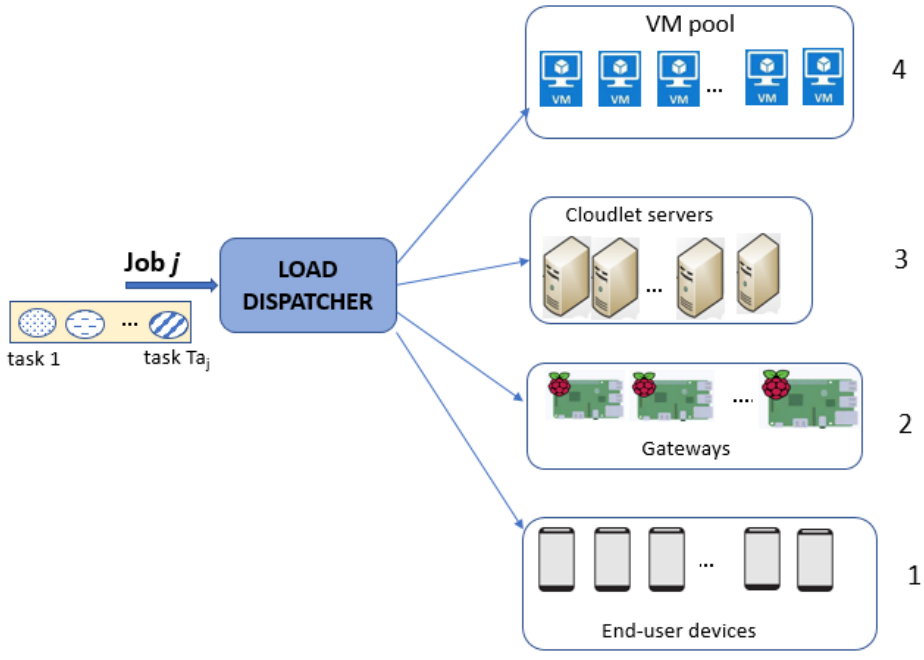


Figure 2. The considered Cloud-Fog computing system model

We consider Bag-of-Tasks (BoT) application model as the input workload. A BoT job consists of a set of independent tasks (i.e., tasks do not require communication with each other during their execution and can be executed in an arbitrary order) [30, 31]. This application model represents a such wide range of practices as data mining, heavily searches, computer imaging sweeping parameters, bioinformatics, and fractal calculations which occur in cloud computing environments. Thanks to their simplicity, BoT applications are appropriate to run over widely distributed, large-scale computing systems. As a result, efficient scheduling for the BoT applications has received a great attention of researchers [31, 13, 32, 33, 34]. We assume that incoming jobs have the following characteristics:

- a job task can be executed on any fog node;
- a job task is non-preemptible (i.e., task is uninterruptible while being processed);
- jobs are compute-intensive and have service time demand known by the scheduler;
- jobs are independent of each other.

3.3 Scheduling Problem and the Proposal of MOSA

Task scheduling problem in a single-machine system or homogeneous computing cluster simply refers to dispatching tasks in an appropriate order of execution. On the other hand, scheduling problem in a heterogeneous system composed of various computing machine types pays more attention to resource allocation for jobs/tasks. Task allocation refers to the selection of a computing machine to which a task is routed. Allocating an appropriate resource has a major impact on both user's satisfaction of services and operation cost paid by a service provider.

This study focuses on the resource allocation policy. If not stated otherwise, we assume that jobs arriving the system are to be served with First Come – First Served (FCFS) policy. Each task of a job is routed to node based on a resource allocation policy applied by the scheduler. A job task in the system will be served immediately if there is an available computing node. If all nodes are busy, a task will be routed to a node with the shortest queue and wait in the selected node's queue.

In [13], authors proposed a Load-Balance allocation policy that uses subsystem loads as thresholds for deciding resource selection. Their goal was to reduce the operation cost of cloud and Raspberry Pi cluster by keeping a low number of active servers in those subsystems so that idle servers can be switched off. Their proposal was compared with the best-effort Round-Robin policy. It is worth to note that switching off technique is inefficient to be applied when Round-Robin policy is used. The reason is Round-Robin selects computing servers with roughly equal probability, which causes the idle period of a server not significantly long enough to power it off.

In this study, we take the same approach of Load-Balance given in [13] to balance loads of all subsystems (cloud center, cloudlet, poor-resource devices, and edge devices). To enhance switching off technique for a system, the resource allocation policy attempts to reduce the number of servers needed for task processing (i.e., the more free servers are available and can be switched off, the lower the energy consumption of the system). Since switching on a sleeping server takes time, it may add more cost of execution delay and energy consumption (if the sleep period is shorter than switching on delay). To avoid the added cost, we only switch off free servers when the subsystem load meets the minimum threshold (θ_3) and switch on a server if the load reaches the maximum threshold (θ_4).

In addition, Cloud-Fog computing is a highly heterogeneous environment. Hence, user jobs' characteristics and resource heterogeneity should be taken into account to decide an appropriate task-resource mapping.

We propose a Multi-Objective scheduling algorithm (MOSA) taking into account job service time demands, resource processing capacity, and the loads of subsystems to improve quality of service as well as system operation cost with the following rules:

- End-user devices and poor-resource Raspberry Pis are preferred if their loads are less than the lower load threshold θ_1;

- Utilization of Cloudlet subsystem should be kept under the upper load threshold θ_2;
- Tasks with acute service time demand (that is, less than the service demand threshold $\beta(t)$) should be executed in resources closer to them (end-devices or Cloudlet);
- Cloud center is chosen if the above rules are not satisfied.

Let a job be identified by (j, Ta_j, sd_j), where j is job identification, Ta_j is the number of tasks, and sd_j is the service time demand of job j. To estimate the service demand of job, we use a threshold called statistical mean service demand in what follows. Let N(t) and $\beta(t)$ denote the number of historical incoming jobs and the average service time demand of those at the considered time t. The statistical mean service demand is calculated as:

$$\beta(t) = \frac{1}{N(t)} \sum_{j=1}^{N(t)} sd_j. \tag{1}$$

Algorithm 1 presents pseudo-code of the proposed resource allocation solution. Algorithm 2 describes the switching off policy for cloud and RaspberryPi subsystems.

4 RESULTS

4.1 Experimental Design

The evaluation is conducted using a simulation software written in C. We make a long-term run for each simulation that stops after five million completions. We also apply the statistical module [35] developed by Politecnico di Torino to collect and evaluate statistics during simulation. The results are obtained with the confidence level of 95 % and the accuracy (i.e., the ratio of the half-width of the confidence interval to the mean of collected observations) of 0.05.

The proposed MOSA is evaluated in comparson to other existing scheduling policies, Load-Balance (LB) and Weighted Round Robin (WRR) given in [13] and MPSO heuristic [36]. The system constructed for simulation runs is composed of 64 end-user devices, 64 powerful Cloudlet servers, 32 Raspberry Pis, and 128 virtual machines (VMs) located at cloud center.

We assume that the inter-arrive time and the execution time of jobs are exponentially distributed with means of $1/\lambda$ and $1/\mu$, respectively. An end-user device, a Raspberry Pi, a Cloudlet node, and a VM are assumed to have ability of executing tasks at service rate $\mu_1 = 2.0$ (tasks/s), $\mu_2 = 0.5$ (tasks/s), $\mu_3 = 1.0$ (tasks/s), $\mu_4 = 1.0$ (tasks/s), respectively.

We consider that the number of tasks of BoT jobs follows a uniform distribution in range of $[1, 8]$. Being frequently observed in practice, Power-of-two and Square job models [37] are also used as input workloads for evaluation. Workload models and their parameters are given in Table 1.

Algorithm 1 Pseudo-code of resource allocation in MOSA

for each new arriving job j **do**
  CALCULATE Load[End-Devices], Load[Cloudlet], Load[RasberryPi], Load[Cloud]
  if Load[End-Devices] $\leq \theta_1$ OR ($sd_j \leq \beta(t)$ AND Load[End-Devices] $\leq \theta_2$) **then**
    chosen_subsystem $\leftarrow$ End-Devices
  else if Load[Cloudlet] $\leq \theta_2$ **then**
    chosen_subsystem $\leftarrow$ Cloudlet
  else if Load[RaspberryPi] $\leq \theta_1$ OR ($sd_j \leq \beta(t)$ AND Load[RaspberryPi] $\leq \theta_2$) **then**
    chosen_subsystem $\leftarrow$ RaspberryPi
  else
    chosen_subsystem $\leftarrow$ Cloud
  end if
  GOTO ALLOCATION
end for
ALLOCATION: {Shortest queue based task scheduling}
for each task in task set Ta_j of job j **do**
  if found *free_server* in *chosen_subsystem* **then**
    ROUTE task to *free_server*
    GOTO ALGORITHM 2
  else
    Calculate server.queue in *chosen_subsystem*
    ROUTE task to the server with the shortest queue
  end if
end for

Algorithm 2 Pseudo-code of switching-off policy

if *chosen_subsystem* is Cloud or RaspberryPi **then**
  $s \leftarrow$ *chosen_subsystem*
  CALCULATE Load[s]
  if Load[s] $\leq \theta_3$ AND *number_active_servers*[s] > 1 **then**
    Switch off a free server in the subsystem
    Decrement *number_active_servers*[s]
  else if Load[s] $\geq \theta_4$ AND *number_active_servers*[s] is less than total number of servers in the system s **then**
    Switch on a sleep server in the subsystem s
    Increment *number_active_servers*[s]
  end if
end if

Workload Model	Description	Task Size	Average Task Size
Uniform	Task size (i.e., the number of tasks) is an integer that follows the uniform distribution within the range of $[1, 8]$	$[1, 2, \ldots, 8]$	4.5
Power-of-two	Task size is an integer that is calculated by 2^k, $k = 1, 2, 3$	$[2, 4, 8]$	≈ 4.67
Square	Task size is an integer that is calculated by k^2, $k = 1, 2, 3$	$[1, 3, 9]$	≈ 4.67

Table 1. Workload models and their parameters

The average service rate of the entire system is calculated as:

$$\mu = \left(\sum_{i=1}^{4} M(i) \times \mu_i \right) / T_{avg}. \tag{2}$$

Therefore the considered system has the average service rate of $(64 \times 2.0 + 64 \times 1.0 + 32 \times 0.5 + 128 \times 01.0)/4.5 = 336/4.5 \approx 74.67(jobs/s)$. We run the simulations with various arrival rates of 16.8 (jobs/s), 22.68 (jobs/s), 28.56 (jobs/s), 34.44 (jobs/s), and 40.32 (jobs/s).

We choose the load thresholds that $\theta_1 = 0.3$, $\theta_2 = 0.7$ for allocation decision and $\theta_3 = 0.5$, $\theta_4 = 0.7$ for switching off policy.

System performance metrics are as follows.

- Average response time of job (RT): The response time of a job is defined as the time period between job arrival and its departure. Let N be the total number of completed jobs during simulation time and rt_j be the response time of job j. The average response time is calculated as:

$$RT = \frac{1}{N} \sum_{j=1}^{N} rt_j. \tag{3}$$

- Average waiting time of task (WT): Let wt_t be the wait time in queue of task t before its execution and T_N be the total number of tasks of N completed jobs.

$$WT = \frac{1}{T_N} \sum_{t=1}^{T_N} wt_t. \tag{4}$$

- Average service time of tasks (ST): Let st_t be the service time of task t. The average service time of T_N tasks (the total number of tasks of N completed jobs)

is calculated as:

$$ST = \frac{1}{T_N} \sum_{t=1}^{T_N} st_t. \tag{5}$$

- Resource utilization ($U(i)$): defined as the average percentage of time that each server of subsystem i is in the busy state over the simulation time and calculated as

$$U(i) = \frac{(\sum_{m=1}^{M(i)} busy\_time_m)/M(i)}{simulation\_time} * 100\,\%. \tag{6}$$

- Average busy servers: the average number of servers processing tasks during the simulation time.

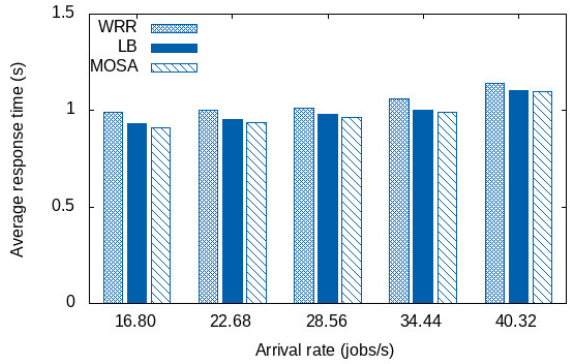
To estimate the effectiveness of switching off technique, metrics of number of busy servers and the resource utilization are used. The energy cost is directly proportional to the busy time of servers and the server power consumption. Therefore, utilization of a subsystem that addresses the percentage of time that a server is in busy state can be interpreted as the cost of subsystem. Moreover, the number busy servers can determine the energy cost of the total system. The system notations are listed in Table 2.

Notation	Description
$M(i)$	Number of servers in Subsystem i
μ_i	Service rate of a server in Subsystem i
μ	System service rate
λ	System arrival rate
sd_j	service demand of job j
Ta_j	Number of tasks of job j
T_{avg}	Average task number per job
$\beta(t)$	Service demand threshold at time t
θ_1	Lower Load threshold
θ_2	Upper Load threshold
θ_3	Minimum Load threshold for switching off
θ_4	Maximum Load threshold for swithcing on
RT	Average response time per job
WT	Average waiting time per task
ST	Average service time per task
$U(i)$	Average resource utilization of subsystem i

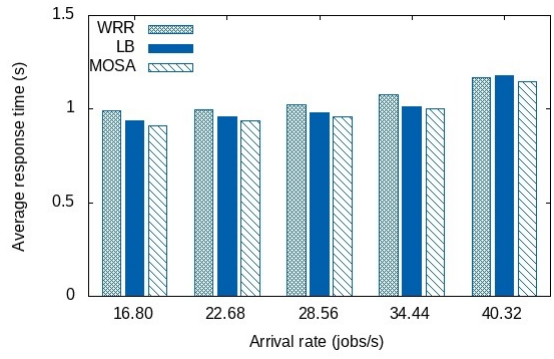
Table 2. System Notations

4.2 Numerical Results with Theoretical Loads

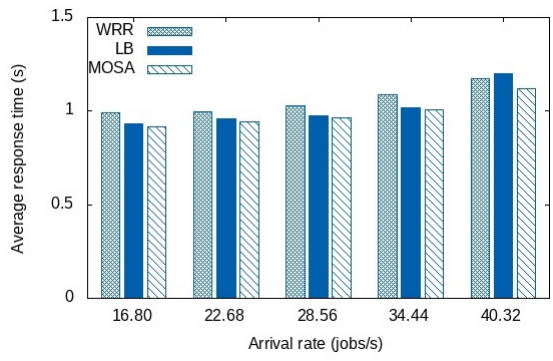
Figure 3 plots the average response time per job where three types of workload model are used. Figure 3 a) (with a uniform distribution workload model) shows that



a) Uniform task size model

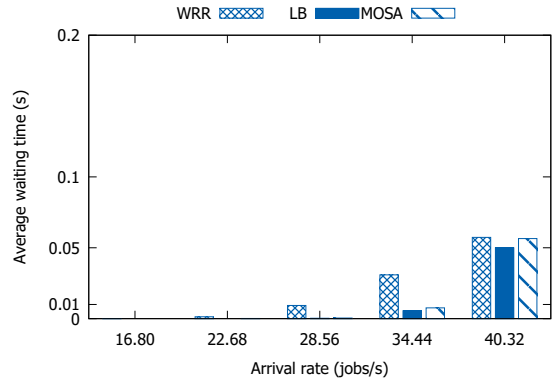


b) Power-of-two task size model

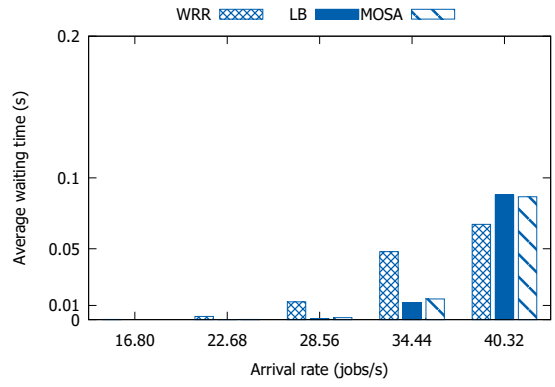


c) Square task size model

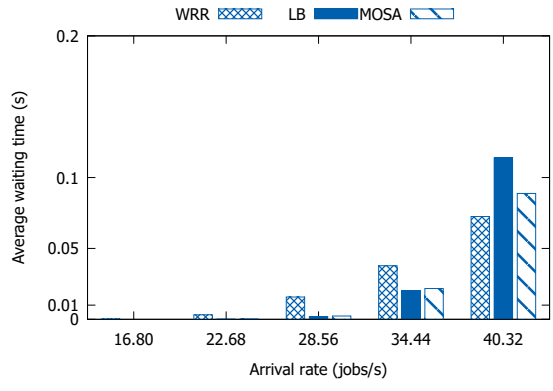
Figure 3. Average Response time (s) (a) Uniform model, b) Power-of-two model, c) Square model)



a) Uniform task size model



b) Power-of-two task size model



c) Square task size model

Figure 4. Average Waiting time of tasks (s) (a) Uniform model, b) Power-of-two model, c) Square model)

the proposed MOSA outperforms the other policies, regardless the used workload model. Particularly, MOSA improves the performance by 10 % and ≈ 3 % compared to Weighted Round Robin (WRR) and Load-Balance (LB) at low load, respectively. When the intensity is high, MOSA performs 6 % better than WRR and as well as the LB policy. Figures 3 b) and 3 c) (wherein Power-of-two and Square workload size models are used) point out that LB outperforms WRR at low load, but causes a slight increase in the response time when the load intensity is high. For the instance of Square workload size model, the average response time is increased by 3 % with the use of LB algorithm but decreased by approximately 6 % with MOSA, in compared to that achieved by WRR policy. In summary, the proposed MOSA attains better performance regardless the workload model and intensity.

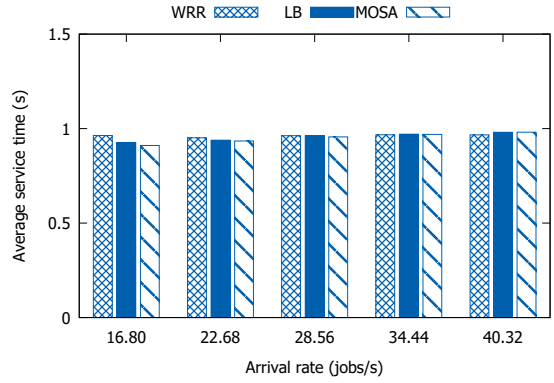
The average waiting time and average service time of tasks are plotted in Figures 4 and 5, respectively. It can be observed that at low to medium workload intensity, tasks have to wait for shorter time with LB and MOSA in compared to WRR. At high load, the waiting time with LB and MOSA is higher. It can be explained that tasks need to wait for servers to be switched on. Figure 5 shows that MOSA results in a slightly faster service at low load and there is no difference among algorithms at high load.

Figure 6 presents the resource utilization of subsystems where three scheduling policies are applied. We can observe that when the workload intensity increases, Weighted Round Robin (WRR) policy puts load stress on end-user devices (Figure 6 a)) while letting powerful servers in Cloudlet and Cloud center under-utilized (see Figures 6 b) and 6 d)). That can degrade the performance of end-user devices and cause a discomfort to users. Load-Balance (LB) policy leads to high utilization of low-performance Raspberry Pis when load is high (see Figure 6 c)). With the MOSA policy, loads of end-user devices and of Raspberries Pi are kept under the desired thresholds, as well as a balanced utilization between Cloudlet and Cloud center is achieved. It can be interpreted that the proposed MOSA results in better resource utilization compared to Weighted Round Robin (WRR) and Load-Balance policy.

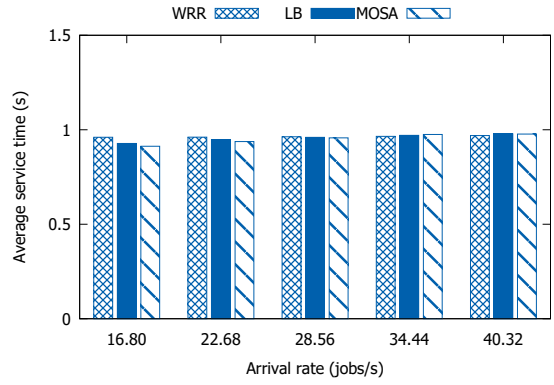
The resource utilization also depicts the energy cost of subsystems. In Figure 6 c), the portion of busy time of Raspberry Pi machines where MOSA is applied is slightly higher compared to LB at low load but less than LB when the intensity increases. Figure 6 d) shows that our proposed MOSA yields 2 % reduction in busy time of cloud servers at low load in comparison to LB. There is no such difference between these two algorithms at high load. It means that MOSA is able to maintain energy cost reduction as LB did. It is worth noting that WRR is not to be compared since switching off is not effective as aforementioned in Section 3.3.

Obtained results of subsystem utilization with the presence of Power-of-two and Square workload model are given in Figures 7 and 8. We observe stable outcomes and system behavior regardless the types of workloads.

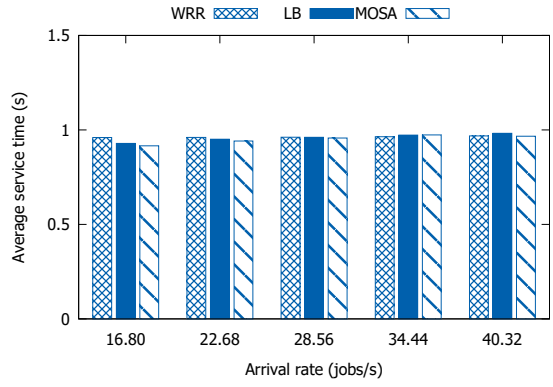
Figure 9 depicts the average number of busy servers over the run time while three job models are applied as input workload alternatively. It shows that the proposed MOSA and LB use less servers for task processing than WRR at low



a) Uniform task size model



b) Power-of-two task size model



c) Square task size model

Figure 5. Average Service time of tasks (s) (a) Uniform model, b) Power-of-two model, c) Square model)

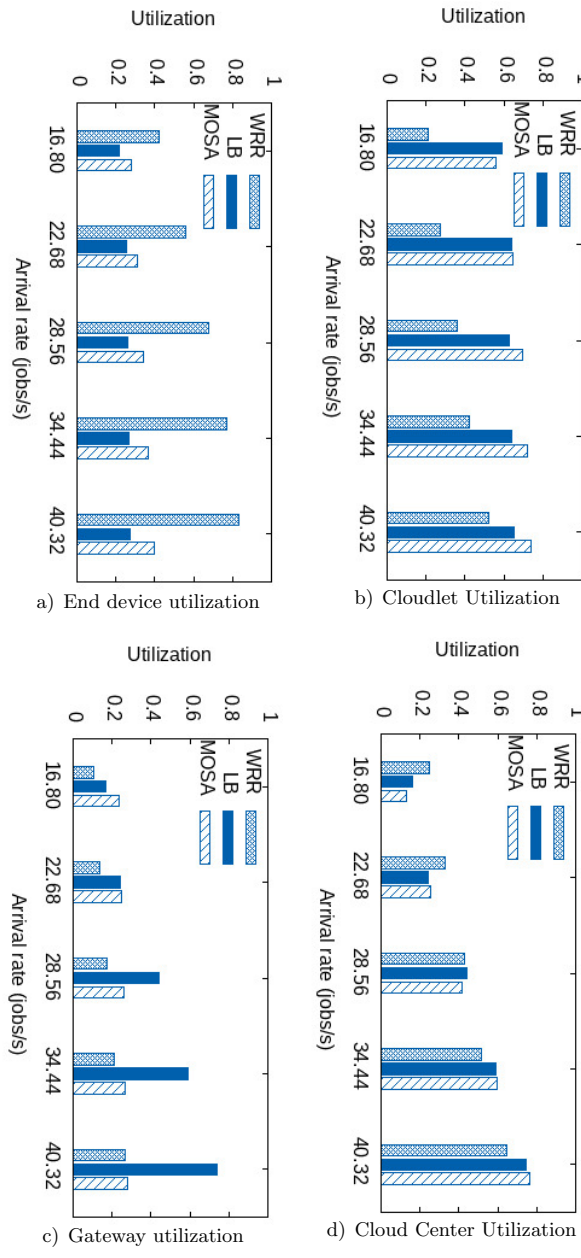


Figure 6. Utilization of subsystems with Uniform workload model (a) End-device, b) Cloudlet, c) Raspberries, d) Cloud center)

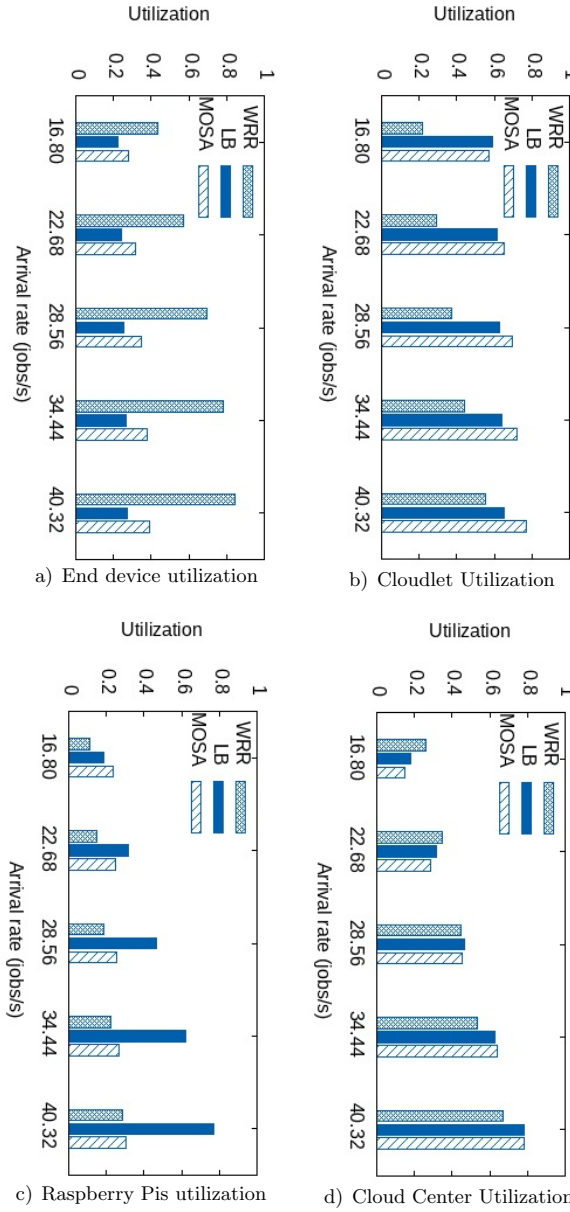


Figure 7. Utilization of subsystems with Power-of-two workload model (a) End-device, b) Cloudlet, c) Raspberries, d) Cloud center)

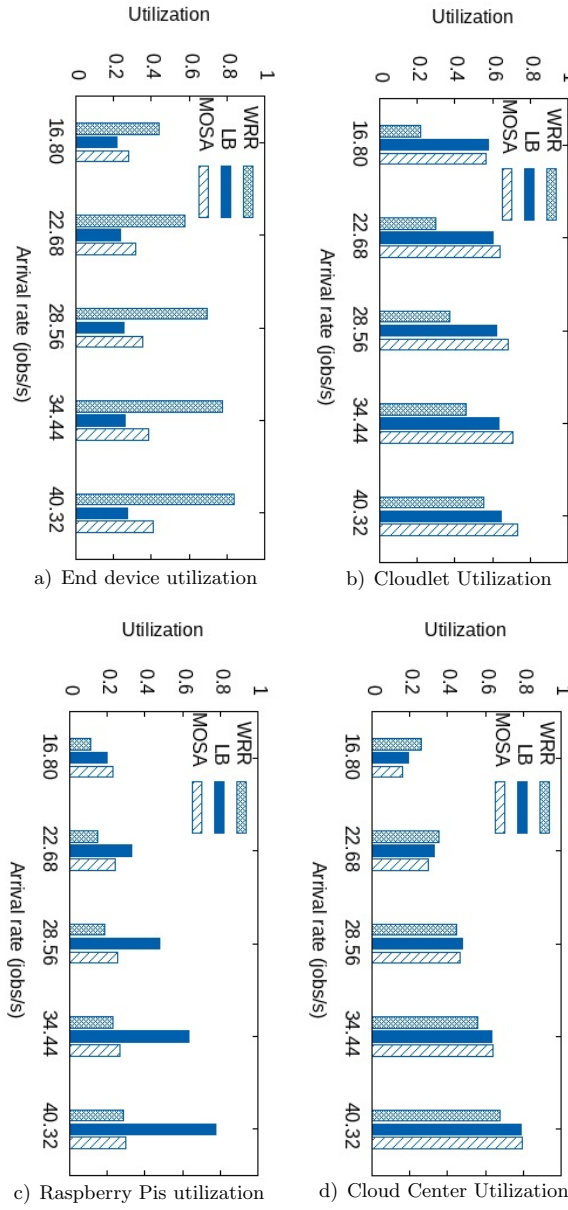
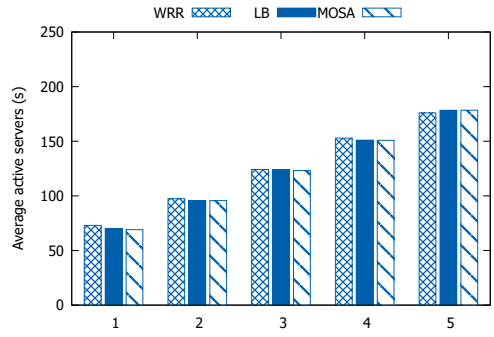
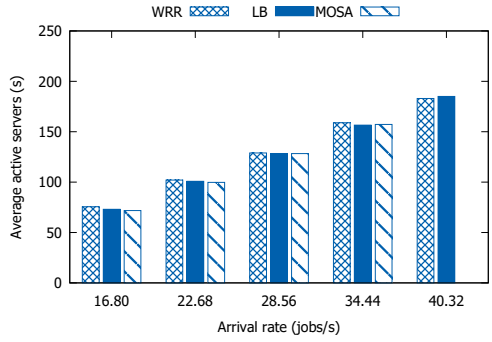


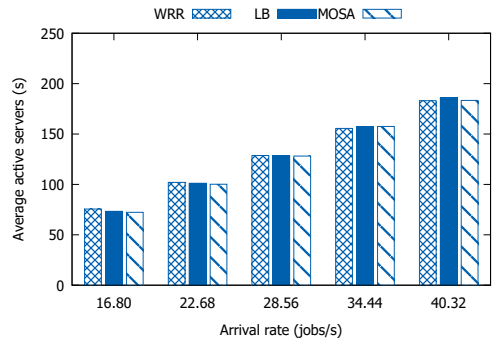
Figure 8. Utilization of subsystems with Square workload model (a) End-device, b) Cloudlet, c) Raspberries, d) Cloud center)



a) Uniform task size model



b) Power-of-two task size model



c) Square task size model

Figure 9. Average active servers (s) (a) Uniform model, b) Power-of-two model, c) Square model)

load and there is no difference observed at high workload intensity. That means the overall energy cost of the Cloud-Fog system can be reduced with MOSA and LB at low load. Similar behaviors are obtained with different workload models.

4.3 Comparison Between the Proposed MOSA and MPSO

This section presents a comparison between the proposed algorithm and the MPSO-based scheduling heuristic. The MPSO (Modified Particle Swarm Optimization) was studied in [36] with the aim to balance the job makespan and total operation cost of the system. The principle of MPSO based heuristic in the comparison can be drawn as follows:

1. Calculate the fitness values of particles as a function of execution time and the utilization of nodes;
2. Find the personal best position for a task in each subsystem according to fitness values;
3. Update the global best position of which the lowest fitness value is obtained.

Figures 10, 11 and 12 plot the job average response time, task service time, and task waiting time, respectively. It can be observed in Figure 10 that the MOSA policy outperforms the others (WRR, LB and MPSO) at low load. At high workload intensity, there is only an obscure difference in the performance with LB, MPSO and MOSA whilst WRR performs worst. Figure 11 indicates that MPSO provides the fastest execution service among the solutions. However, MPSO also causes the longest waiting time of tasks for the service as shown in Figure 12. Therefore, we observe a performance degradation with MPSO, showed in Figure 10.

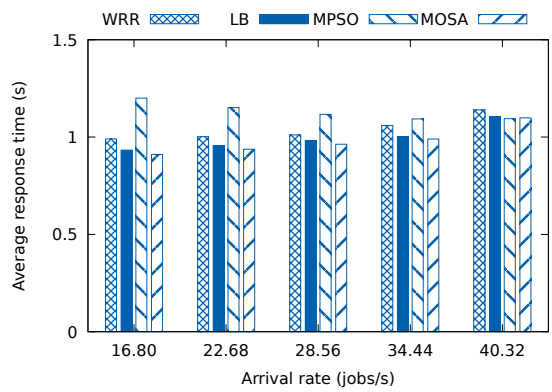


Figure 10. Average response time per job (uniform model)

The average active servers versus load intensity with the application of different algorithms are illustrated in Figure 13. It shows that MPSO leads to lowest number

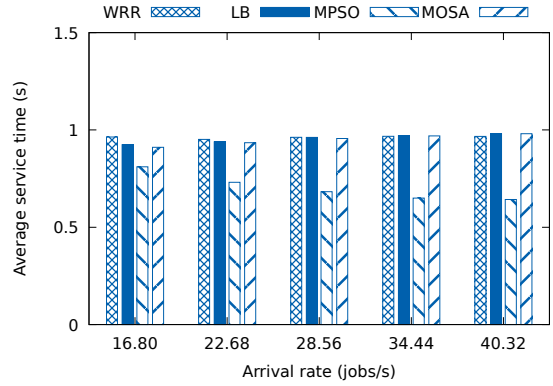


Figure 11. Average service time per job (uniform model)

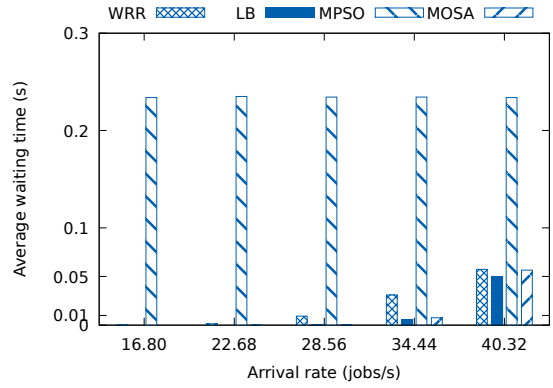


Figure 12. Average waiting time per job (uniform model)

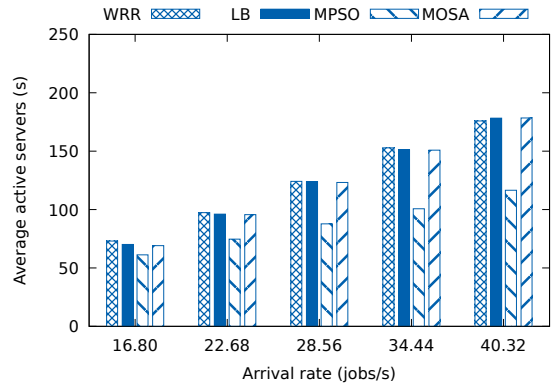
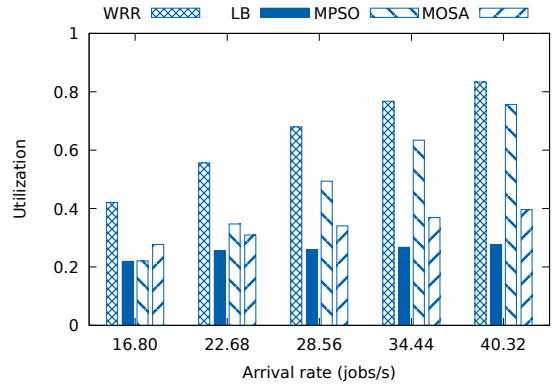


Figure 13. Average active servers (uniform model)

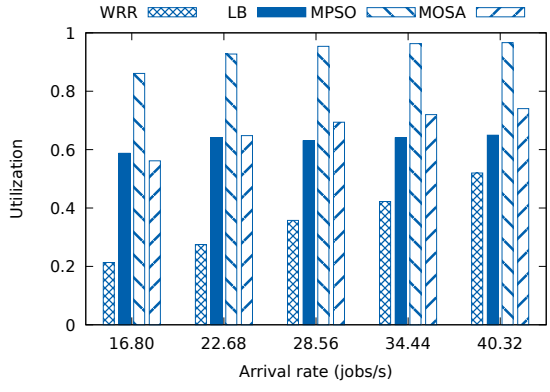
of busy server among solutions, which may result in lower operation energy cost of the system.

Figure 14 depicts the utilization ratio of the four subsystems. It is shown that MPSO causes an intense amount of workloads processed in end devices and cloudlet severs which are closer to users. On the other hand, cloud servers and network gateways are mostly underutilized with the MPSO algorithm. This phenomenon can be explained by the powerful capacity of local processing. A heavy execution load on end devices may cause service discomfort of users, while other resources are not utilized properly.

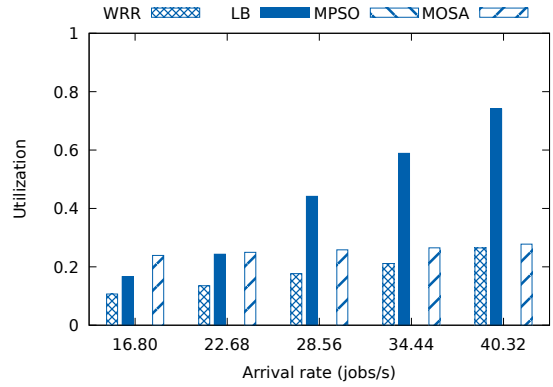
In summary, MPSO yields a balance for execution time and total operation cost from operator's perspective. However, the overall service quality defined by the job response time and a balanced resource utilization which consider user's experience is achieved by the proposed solution MOSA.



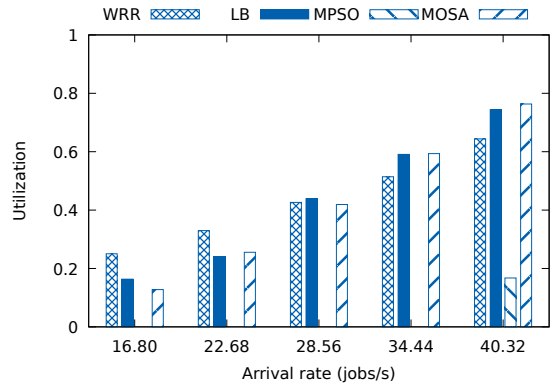
a) End device utilization



b) Cloudlet Utilization



c) Raspberry Pis utilization



d) Cloud Center Utilization

Figure 14. Utilization of subsystems with Square workload model (a) End-device, b) Cloudlet, c) Raspberry Pi, d) Cloud center)

4.4 Experimental Results with Traced Workloads

In order to examine the proposed algorithm thoroughly, we also experiment the simulation with various real workloads which were captured from practical executions. Table 3 presents the statistical metrics of traced workloads in terms of mean and variance. It is observed that these do not follow exponential distribution.

Figure 15 depicts the average response time of job when three traced workloads are used for comparing the proposed MOSA with three other scheduling algorithms. It shows that with INCC workload trace the proposed MOSA outperforms WRR and LB with a reduction of over 15 % in the average response time, while resulting in roughly the same performance to MPSO. With other workload traces, the

Workload	Number of Samples	Inter-Arrival Time		Service Time	
		Mean	Var	Mean	Var
INCC trace	1 037 093	15.236	102 062.096	546.591	1 789 574.190
NVS trace	2 188 683	62.861	671 414.041	404.909	446 969.583
UPR trace	1 352 714	129.436	473 524.089	850.539	1 509 552.918

Table 3. Traced workloads

proposed MOSA outperforms other solutions with a reduction of 30 %–40 % in average response time. There is no clear difference observed between WRR, LB and MPSO.

Figure 16 shows that the utilization of subsystems with the use of INCC workload. The outputs verify that the proposal yields better resource utilization by maintaining the resource load under desired thresholds. It can also be observed that WRR and LB policies send a larger percentage of workload to remote cloud center and low-performance gateways (Raspberry Pi). That means the idle period of servers in those subsystems become shorter which lead to the higher energy cost in comparison to our solution. On the other hand, the MPSO overloads end user devices that causes dissatisfaction to users.

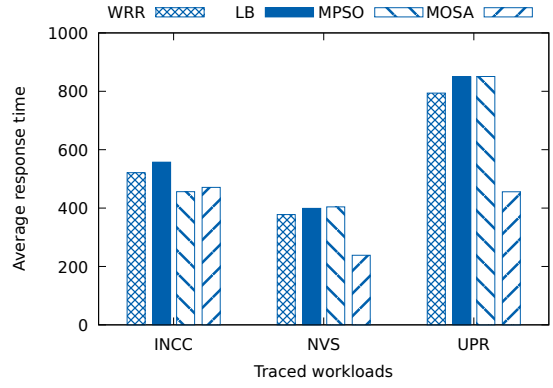


Figure 15. Average response time vs. traced workloads

5 CONCLUSIONS

Fog computing has become significant to develop 5G/6G network and allow more IoT applications to connect to the system. As a result, a merged computing environment for IoT applications is spread from edge node to remote cloud center. In this study, we investigate the impact of characteristics of jobs and resources on the system performance of a merged computing environment of Edge, Fog and Cloud. With the aim at improving system performance and energy efficiency of this

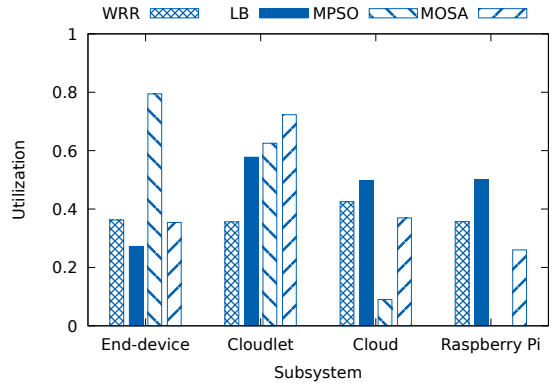


Figure 16. Utilization of subsystems (INCC workload)

such heterogeneous environment, we propose a multi-objective scheduling algorithm-MOSA using various characteristics of user workloads and computational resources. The proposed algorithm MOSA uses job service demand and resource processing capacity to find an appropriate task-resource mapping. Numerical results show that MOSA leads to shorter average response time per job than existing WRR, LB, and MPSO policies.

For load balancing and energy efficiency, the proposed MOSA uses load threshold based approach. Obtained results indicate that MOSA yields a slight improvement in load balancing than LB in some scenarios and similar outcomes in other cases. Compared with MPSO that balances execution time and total system cost, MOSA achieves similar or better performance while yielding better resource utilization and guaranteeing user's service experience. Experimental outputs with the use of real workloads validate the foregoing conclusions wherein MOSA yields a reduction of 15 %–40 % in the average response time per job and balances resource utilization among subsystems.

The proposed MOSA is based on job service demand as well as hard thresholds of resource utilization to make decision, while different service level agreements (SLAs) also depend on other proprieties of jobs and computing resources. Therefore, we must investigate more complex scenarios wherein other characteristics such as types of jobs, physical location of job request should be taken into account to achieve various SLA requirements in future work.

REFERENCES

[1] Bonomi, F.—Milito, R.—Zhu, J.—Addepalli, S.: Fog Computing and Its Role in the Internet of Things. Proceedings of the First Edition of the MCC Workshop

on Mobile Cloud Computing, Association for Computing Machinery, New York, NY, USA, MCC '12, 2012, pp. 13–16, doi: 10.1145/2342509.2342513.

[2] Sarkar, S.—Misra, S.: Theoretical Modelling of Fog Computing: A Green Computing Paradigm to Support IoT Applications. IET Networks, Vol. 5, 2016, No. 2, pp. 23–29, doi: 10.1049/iet-net.2015.0034.

[3] Atlam, H. F.—Walters, R. J.—Wills, G. B.: Fog Computing and the Internet of Things: A Review. Big Data and Cognitive Computing, Vol. 2, 2018, No. 2, doi: 10.3390/bdcc2020010.

[4] Yi, S.—Li, C.—Li, Q.: A Survey of Fog Computing: Concepts, Applications and Issues. Proceedings of the 2015 Workshop on Mobile Big Data, Association for Computing Machinery, New York, NY, USA, Mobidata '15, 2015, pp. 37–42, doi: 10.1145/2757384.2757397.

[5] Dastjerdi, A.—Gupta, H.—Calheiros, R.—Ghosh, S.—Buyya, R.: Chapter 4 – Fog Computing: Principles, Architectures, And applications. In: Buyya, R., Vahid Dastjerdi, A. (Eds.): Internet of Things. Morgan Kaufmann, 2016, pp. 61–75, doi: 10.1016/B978-0-12-805395-9.00004-6.

[6] Das, R.—Inuwa, M. M.: A Review on Fog Computing: Issues, Characteristics, Challenges, and Potential Applications. Telematics and Informatics Reports, Vol. 10, 2023, Art. No. 100049, doi: 10.1016/j.teler.2023.100049.

[7] Margariti, S. V.—Dimakopoulos, V. V.—Tsoumanis, G.: Modeling and Simulation Tools for Fog Computing – A Comprehensive Survey from a Cost Perspective. Future Internet, Vol. 12, 2020, No. 5, doi: 10.3390/fi12050089.

[8] Shakarami, A.—Ghobaei-Arani, M.—Masdari, M.—Hosseinzadeh, M.: A Survey on the Computation Offloading Approaches in Mobile Edge/Cloud Computing Environment: A Stochastic-Based Perspective. Journal of Grid Computing, Vol. 18, 2020, pp. 639–671, doi: 10.1007/s10723-020-09530-2.

[9] Liu, L.—Qi, D.—Zhou, N.—Wu, Y.—Lin, F.: A Task Scheduling Algorithm Based on Classification Mining in Fog Computing Environment. Wirel. Commun. Mob. Comput., Vol. 2018, 2018, doi: 10.1155/2018/2102348.

[10] Deng, R.—Lu, R.—Lai, C.—Luan, T. H.—Liang, H.: Optimal Workload Allocation in Fog-Cloud Computing Toward Balanced Delay and Power Consumption. IEEE Internet of Things Journal, Vol. 3, 2016, No. 6, pp. 1171–1181, doi: 10.1109/JIOT.2016.2565516.

[11] Nguyen, B. M.—Thi Thanh Binh, H.—The Anh, T.—Bao Son, D.: Evolutionary Algorithms to Optimize Task Scheduling Problem for the IoT Based Bag-of-Tasks Application in Cloud-Fog Computing Environment. Applied Sciences, Vol. 9, 2019, No. 9, doi: 10.3390/app9091730.

[12] Aladwani, T.: Scheduling IoT Healthcare Tasks in Fog Computing Based on Their Importance. Procedia Computer Science, Vol. 163, 2019, pp. 560–569, doi: 10.1016/j.procs.2019.12.138 (16^{th} Learning and Technology Conference 2019 Artificial Intelligence and Machine Learning: Embedding the Intelligence).

[13] Tychalas, D.—Karatza, H.: A Scheduling Algorithm for a Fog Computing System with Bag-of-Tasks Jobs: Simulation and Performance Evaluation. Simulation Modelling Practice and Theory, Vol. 98, 2020, Art. No. 101982, doi:

10.1016/j.simpat.2019.101982.

[14] Li, S.—Maddah-Ali, M. A.—Avestimehr, A. S.: Coding for Distributed Fog Computing. IEEE Communications Magazine, Vol. 55, 2017, No. 4, pp. 34–40, doi: 10.1109/MCOM.2017.1600894.

[15] Rafique, H.—Shah, M. A.—Islam, S. U.—Maqsood, T.—Khan, S.—Maple, C.: A Novel Bio-Inspired Hybrid Algorithm (NBIHA) for Efficient Resource Management in Fog Computing. IEEE Access, Vol. 7, 2019, pp. 115760–115773, doi: 10.1109/ACCESS.2019.2924958.

[16] Nan, Y.—Li, W.—Bao, W.—Delicato, F. C.—Pires, P. F.—Dou, Y.—Zomaya, A. Y.: Adaptive Energy-Aware Computation Offloading for Cloud of Things Systems. IEEE Access, Vol. 5, 2017, pp. 23947–23957, doi: 10.1109/ACCESS.2017.2766165.

[17] Kabirzadeh, S.—Rahbari, D.—Nickray, M.: A Hyper Heuristic Algorithm for Scheduling of Fog Networks. 2017 21st Conference of Open Innovations Association (FRUCT), 2017, pp. 148–155, doi: 10.23919/FRUCT.2017.8250177.

[18] Katal, A.—Dahiya, S.—Choudhury, T.: Energy Efficiency in Cloud Computing Data Centers: A Survey on Software Technologies. Cluster Computing, 2022, doi: 10.1007/s10586-022-03713-0.

[19] Wadhwa, H.—Aron, R.: TRAM: Technique for Resource Allocation and Management in Fog Computing Environment. Journal of Supercomputing, Vol. 78, 2022, pp. 667–690, doi: 10.1007/s11227-021-03885-3.

[20] Qiu, Y.—Jiang, C.—Wang, Y.—Ou, D.—Li, Y.—Wan, J.: Energy Aware Virtual Machine Scheduling in Data Centers. Energies, Vol. 12, 2019, No. 4, doi: 10.3390/en12040646.

[21] Xiang, H.—Peng, M.—Sun, Y.—Yan, S.: Mode Selection and Resource Allocation in Sliced Fog Radio Access Networks: A Reinforcement Learning Approach. IEEE Transactions on Vehicular Technology, Vol. 69, 2020, No. 4, pp. 4271–4284, doi: 10.1109/TVT.2020.2972999.

[22] Oueis, J.—Strinati, E. C.—Sardellitti, S.—Barbarossa, S.: Small Cell Clustering for Efficient Distributed Fog Computing: A Multi-User Case. 2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall), 2015, pp. 1–5, doi: 10.1109/VTCFall.2015.7391144.

[23] Abbasi M., Mohammadi P. E., K. M.: Workload Allocation in IoT-Fog-Cloud Architecture Using a Multi-Objective Genetic Algorithm. Journal of Grid Computing, Vol. 18, 2020, pp. 43–56, doi: 10.1007/s10723-020-09507-1.

[24] S. Agarwal, S. Yadav, A. K. Y.: An Efficient Architecture and Algorithm for Resource Provisioning in Fog Computing. International Journal of Information Engineering and Electronic Business (IJIEEB), Vol. 8, 2016, No. 1, pp. 48–61, doi: 10.5815/ijieeb.2016.01.06.

[25] Huedo, E.—Montero, R.—Moreno-Vozmediano, R.: Opportunistic Deployment of Distributed Edge Clouds for Latency-Critical Applications. Journal of Grid Computing, Vol. 19, 2021, No. 2, doi: 10.1007/s10723-021-09545-3.

[26] Movahedi, Z.—Defude, B.—Hosseininia, A. m.: An Efficient Population-Based Multi-Objective Task Scheduling Approach in Fog Computing Systems. J. Cloud

Comput., Vol. 10, 2021, No. 1, doi: 10.1186/s13677-021-00264-4.

[27] Hassanat, A. B. A.: Furthest-Pair-Based Decision Trees: Experimental Results on Big Data Classification. Information, Vol. 9, 2018, No. 11, doi: 10.3390/info9110284.

[28] Xu, F.—Yang, F.—Zhao, C.—Wu, S.: Deep Reinforcement Learning Based Joint Edge Resource Management in Maritime Network. China Communications, Vol. 17, 2020, No. 5, pp. 211–222, doi: 10.23919/JCC.2020.05.016.

[29] Zhang, W.—Yang, D.—Peng, H.—Wu, W.—Quan, W.—Zhang, H.—Shen, X.: Deep Reinforcement Learning Based Resource Management for DNN Inference in Industrial IoT. IEEE Transactions on Vehicular Technology, Vol. 70, 2021, No. 8, pp. 7605–7618, doi: 10.1109/TVT.2021.3068255.

[30] Tran, N. M.—Wolters, L.: Towards a Profound Analysis of Bags-of-Tasks in Parallel Systems and Their Performance Impact. Association for Computing Machinery, New York, NY, USA, 2011, doi: 10.1145/1996130.1996148.

[31] Iosup, A.—Sonmez, O.—Anoep, S.—Epema, D.: The Performance of Bags-of-Tasks in Large-Scale Distributed Systems. Association for Computing Machinery, New York, NY, USA, 2008, doi: 10.1145/1383422.1383435.

[32] Moschakis, I. A.—Karatza, H. D.: A Meta-Heuristic Optimization Approach to the Scheduling of Bag-of-Tasks Applications on Heterogeneous Clouds with Multi-Level Arrivals and Critical Jobs. Simulation Modelling Practice and Theory, Vol. 57, 2015, pp. 1–25, doi: 10.1016/j.simpat.2015.04.009.

[33] Oprescu, A. M.—Kielmann, T.: Bag-of-Tasks Scheduling Under Budget Constraints. Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science, IEEE Computer Society, USA, CLOUDCOM '10, 2010, pp. 351–359, doi: 10.1109/CloudCom.2010.32.

[34] Weng, C.—Lu, X.: Heuristic Scheduling for Bag-of-Tasks Applications in Combination with QoS in the Computational Grid. Future Generation Computer Systems, Vol. 21, 2005, No. 2, pp. 271–280, doi: 10.1016/j.future.2003.10.004 (Advanced Grid Technologies).

[35] di Torino, P.: A Statistical Module. `https://www.telematica.polito.it/oldsite/class/statistics.ps.gz` [accessed 2020-10-15].

[36] Izakian, H.—Tork Ladani, B.—Zamanifar, K.—Abraham, A.: A Novel Particle Swarm Optimization Approach for Grid Job Scheduling. In: Prasad, S. K., Routray, S., Khurana, R., Sahni, S. (Eds.): Information Systems, Technology and Management. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 100–109.

[37] Aida, K.: Effect of Job Size Characteristics on Job Scheduling Performance. In: Feitelson, D. G., Rudolph, L. (Eds.): Job Scheduling Strategies for Parallel Processing. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000, pp. 1–17, doi: 10.1007/3-540-39997-6_1.

Xuan Thi Tran works as Researcher and Lecturer at the University of Information and Communication Technology, Thai Nguyen University, Vietnam. She earned her M.Sc. and Ph.D. degree at the Budapest University of Technology and Economics, Hungary in 2014 and 2020, respectively. Her research focuses on performance evaluation, distributed computing, job scheduling, and big data processing. Her research interests are cloud computing, 5G/6G mobile network, and machine learning.