BILINGUAL PUBLISHING CO.
Pioneer of Global Academics Since 1984

Journal of Electronic & Information Systems
https://ojs.bilpublishing.com/index.php/jeis

ARTICLE

Computation Offloading and Scheduling in Edge-Fog Cloud Computing

Dadmehr Rahbari* **Mohsen Nickray**

Department of Computer Engineering and Information Technology, University of Qom, Qom, Iran

ARTICLE INFO

Article history
Received: 16 August 2019
Accepted: 27 September 2019
Published Online: 18 October 2019

Keywords:
Cloud computing
Edge computing
Fog computing
Offloading
Scheduling

ABSTRACT

Resource allocation and task scheduling in the Cloud environment faces many challenges, such as time delay, energy consumption, and security. Also, executing computation tasks of mobile applications on mobile devices (MDs) requires a lot of resources, so they can offload to the Cloud. But Cloud is far from MDs and has challenges as high delay and power consumption. Edge computing with processing near the Internet of Things (IoT) devices have been able to reduce the delay to some extent, but the problem is distancing itself from the Cloud. The fog computing (FC), with the placement of sensors and Cloud, increase the speed and reduce the energy consumption. Thus, FC is suitable for IoT applications. In this article, we review the resource allocation and task scheduling methods in Cloud, Edge and Fog environments, such as traditional, heuristic, and meta-heuristics. We also categorize the researches related to task offloading in Mobile Cloud Computing (MCC), Mobile Edge Computing (MEC), and Mobile Fog Computing (MFC). Our categorization criteria include the issue, proposed strategy, objectives, framework, and test environment.

1. Introduction

In recent years, wireless sensor networks (WSNs) have been extensively developed independently or partially from another system. WSNs collect data in healthcare, vehicles, smart home, and more. These networks as the infrastructure for the IoT require real-time processing and decision-making. Data transmission from end-sensor nodes to the cloud by passing several mid-sensor nodes, routers, and gateways have high total network power consumption and delay [1]. In many sensitive cases such as medical care and transportation systems, high delays in IoT applications can lead to a patient's death or cause an accident. Edge computing with processing near the IoT devices have been able to reduce the network congestion and delay to some extent, but the problem is distancing itself from the Cloud [2,3].

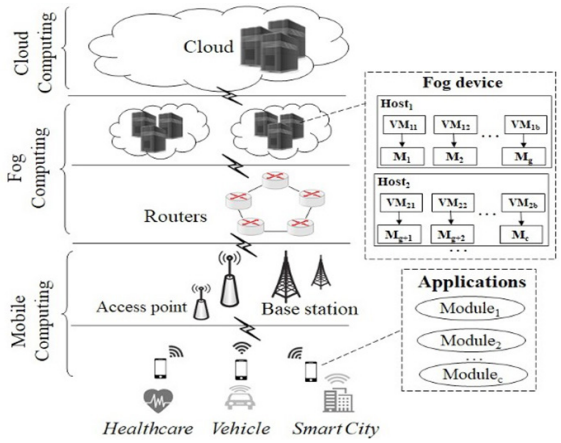


Figure 1. Architecture of Edge-Fog Cloud computing

**Corresponding Author:*
Dadmehr Rahbari,
Department of Computer Engineering and Information Technology, University of Qom, Qom, Iran;
Email: d.rahbari@stu.qom.ac.ir

DOI: https://doi.org/10.30564/jeisr.v1i1.1135

As a relatively new architecture, FC sits between the cloud and the sensors, so data aggregation, processing, and storage can be done near the sensors, as well as data sent to the cloud data center only if necessary [4]. Processing operations are performed by the fog nodes at the edge of the network, where the sensors are, so the network traffic decreases and the transport speed increases. Since cloud computing reduces data transfer to the cloud, it consumes less energy than the cloud [5,6].

According to Figure 1, FC has a hierarchical structure. The sensors are at the lowest level of this architecture, collecting data at specified intervals and delivering it to the fog layer at the mid-level [7,8]. Fog nodes are responsible for measuring, processing, and sending data to the cloud. At the highest level, the cloud performs heavy storage and processing operations.

Applications in this network can be run by multiple modules by processors. Modules in each FD have different tasks depending on the application type [9]. As a small data center, FDs implement modules with their resources. An appropriate way of allocating CPUs to modules is to increase the resource efficiency of the fog nodes [10]. At FC, scheduling resources and modules is a challenge. Resource allocation can be performed according to a number of QoS parameters [11]. The performance of scheduling algorithms is evaluated using several parameters, such as power consumption, waiting time, execution time, task completion time as well as some security criteria [12,13].

FC is a great architecture for IoT applications such as smart home, wearables, health care and vehicles [14, 15]. For example, in a treatment clinic, one or more FDs may be used to monitor the activity of the elderly or special patients. In transportation systems, FD is also used to track and control cars. FC has distributed architecture and cloud has centralized architecture. The main advantage of FC is that it can deliver services provided in cloud data centers on the edge of the network near the end sensors [6].

Scheduling issues are classified into several types, including resource allocation, load balancing, and offloading. These categories are implemented in various architectures such as cloud, edge, and fog. Scheduling can be monitored using a variety of parameters. There are also different ways to solve these problems, each with its own advantages and disadvantages. This research examines, classifies and analyzes these challenges in various applications including the well-known IoT framework. In fact, this paper categorizes the different methods of computation offloading and scheduling in Cloud, Edge, FC. Also, another classification is presented for offloading mobile computing. The main objectives of this paper are to:

(1) Provide a comprehensive review of the literature in the scheduling and offloading issues.

(2) Categorize scheduling algorithms in a variety of centralized and distributed computing.

(3) We summarize the research findings, conclude the paper, and suggest some research subjects in scheduling scope.

The rest of this article is organized in the following sections. In Section 2, the past works of scheduling methods are provided. The mobile computing is explained in Section 3 and its offloading methods in MCC, MEC, and MFC are presented in Section 4. Section 5 include an analysis and comparison of offloading and scheduling methods. Section 6 presents the summary and conclusions of this work.

2. Scheduling

Scheduling is in many areas. One of the meanings is to plan entry and exit. These include the arrival and departure schedules of ships on the docks [16-20], trucks in transit [21-23], industrial equipment [24], and supply chains [25]. The second meaning that is most often considered in networks is the allocation of resources to input tasks. Here the second meaning is of interest. Topics such as load balancing, load prediction, reliability and fault tolerance in offloading, resource provisioning, software-defined networking, network function virtualization, and scheduling with fog architecture are considered to be very appropriate developments that are at the beginning of the road today and have a great deal of research.

Scheduling is responsible for optimizing CPU usage and allocating resources appropriately to applications. A scheduler, considering the possible sets of executable tasks, decides in which order and where they will be executed. Scheduling goals include cost, interest, maximization of the number of executable tasks, use of VM or their migration, energy consumption, error tolerance, reliability and security [26]. Optimization strategies include heuristic [27], meta-heuristic [28,29] and other methods. Resource models include different VM deployment patterns, single or multiple providers, medium data-sharing model, data transfer, cost, static and dynamic types, resource sharing, single VM pricing model and Delay is the supply of VM [28,30].

2.1 Concepts of Resource Management

Customers can request multiple services at the same time. There are various algorithms for allocating resources to input tasks [13,4]. Resource management has three main functions: provisioning, scheduling, and monitoring [31].

Resources provisioning: The term resource provision-

 DOI: https://doi.org/10.30564/jeisr.v1i1.1135

ing was used in the grid computing framework. Providing suitable resources to works depends on the QoS parameters. The consumer of the service communicates with the agent of providing resources RPA and sends the application (workflow). The RPA finds the right resources and selects the best one based on customer needs. After submitting the workflow to the RPA Information Center section, access operations are performed according to customer request. The process of selecting the source is the best source for the workflow according to the requirements of the QoS [31].

Resource Scheduling: Challenges in providing resources include dispersion, uncertainty, and heterogeneity. Resource scheduling includes two functions as the allocation of resources and maps them. The purpose of resource allocation is to allocate appropriate resources for tasks in the correct order so that tasks can use resources effectively. Resource mapping is the process of mapping tasks to suitable resources based on the quality of service and determined by the user in accordance with the SLA agreement. Task scheduling is the allocation of VMs to tasks [31], that is shown in Figure 2. In this figure, the meaning of the physical machine in the cloud is the host of data center, on the edge means the edge device and, in the fog, the fog device.

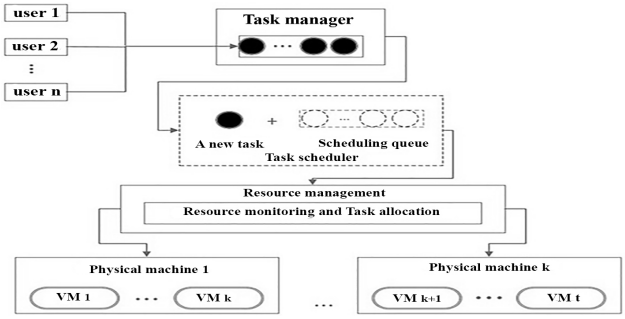


Figure 2. Resource Scheduling

Resource monitoring: Monitoring and controlling resource efficiency can improve system performance. Therefore, a global supervisor is needed to examine how resources are allocated. Supervisory criteria include CPU usage, memory and storage space. The supervisor expects tasks to be executed with minimal cost and time without SLA violation [31].

2.2 Scheduling Objectives

The scheduling process assigns tasks within workflows to appropriate resources according to specific scheduling criteria. Scheduling parameters are effective in the success of the workflow scheduling problem. Scheduling objectives are classified into two groups based on the service approach: service provider and consumer services [32].

Consumer Service:

(1) Makespan: This criterion is equal to the time all tasks are completed. The makespan can be considered as the length of time the user sends the job until it completes the work and is the results generated.

(2) Budget: This is equal to the financial constraint on the use of resources. To run the total workflow can be used several VM from different types. The total cost of execution is equal to the sum of all types of VMs used in the implementation, which should be less than the user-defined budget.

(3) Deadline: Critical applications need to be completed within a certain time period. Scheduling is defined under the time limitation for the applications to be completed before the deadline.

(4) Security: In distributed computing such as Fog, resources are varied and vast, so maintaining security is an important issue. Data protection and privacy in the haze environment are more complex than traditional systems because of the nature of the distribution.

(5) Cost: This parameter includes computing costs, data transmission costs, and storage costs [32].

Service Provider:

(1) Load balancing: VMs are the most important resource in the computing environment. On scheduling, you can assign more than one task to a VM to run tasks simultaneously, which results in load imbalances on VMs. Load balancing between resources improves resource efficiency and thus improves the overall performance of the scheduling process.

(2) Consuming resources: Increasing the use of resources for a helpful service provider.

To obtain the maximum benefit is allocated limited resources to the user, are fully used resources.

(3) Energy efficiency: The use of processors and the use of resources directly affects the energy consumed by a task. When the processor is not used properly, the energy consumption will be high because is not effectively used the idle time [32].

2.3 Traditional Methods

The authors in [33] devised a dynamic programming (DP) method for allocating resources to runtime constrained input tasks. In this way, each provider offers several different VMs and global services for data sharing. One of the methods of resource allocation is the fastest time-out algorithm that has increased cost. In [34], researchers analyzed the Multi-Objective Optimization (MOO) method for business infrastructure services. They used the Pareto Front as a de-

DOI: https://doi.org/10.30564/jeisr.v1i1.1135

cision-making method for trading optimal solutions. They cut the cost of the timetable by half, but increased the rate by 5%. In another paper, resource allocation is performed considering the fault tolerance according to the proposed method for point-and-time VMs for executing user requests. Prices have started roughly and increased during execution to get closer to user demand. The results show that due to the use of spot VM [35] the scheduling performance is low. n [36], a method of allocating resources based on hardware defects in instant time is presented. This method has three steps in term of elastic resource provisioning in the clouds. First, backward shifting of overlapping tasks with VM migration. Second, increasing the resource scale to increase the capability of VM operations or builds due to synchronization with subsequent input work. Third, shrinking of the processing capacity of an idle VM. The results of this work caused to optimized resource utilization than other baseline algorithms.

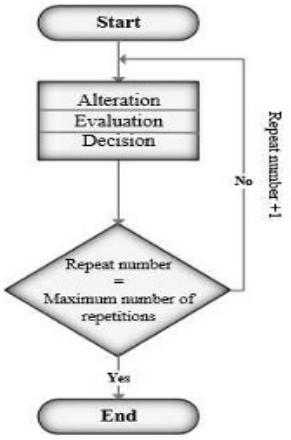
In [37], the authors investigated QoS parameters using three resource allocation algorithms in the Fog architecture, namely: concurrent priority, first-input-first-output, and delay priority. In the concurrent method, input tasks are assigned to them regardless of the capacity of the resources. In the first-input-first-output method, tasks are executed the same way they were entered. If the data center is unable to execute the request, then the task is queued. In priority delay-based methods, the input tasks are executed based on the least time delay. The paper uses the iFogsim simulator [1] with two applications of brain signal tracking and object tracking in video images. The results show that the concurrent method has more delay than the first-input-first-output and delay priority method. In tracking brain signals, the number of modules per device for the simultaneous method is greater than the other two. The number of modules in the cloud for the first-in-first-out method was higher than the other two.

2.4 Heuristic Methods

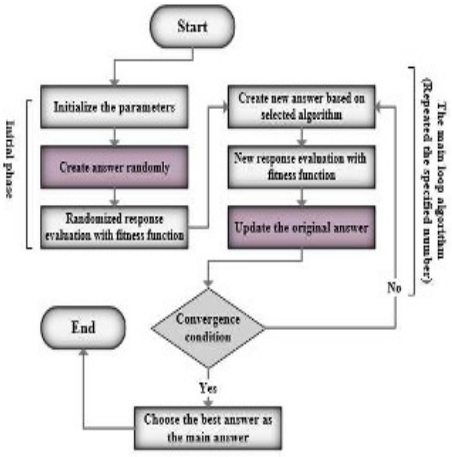
In heuristic algorithms, the answers are obtained by a number of rules. In the classical type of these algorithms, there are methods such as first, best and worst fit. Major resource allocation [28] issues in Fog can be solved by such methods. In [38], the input tasks are programmed by a heuristic algorithm, which aims to reduce the cost of executing the tasks. Performance and cost improved, according to the results. In [39], parallel tasks were implemented by instantaneous execution on a network of different sources by a heuristic. Researchers selected the frequency of sources using nonlinear programming (NLP).

In [40], a knapsack-based scheduler for parallel transmission of video content is presented. The researchers implemented the max-min method on high-powered computers for mapping tasks in a number of sectors. The analysis of the results proves that their proposed method has improved at runtime and number of segments. Researchers in [41] solved the task scheduling problem in fog-based IoT applications by knapsack. They optimized knapsack by symbiotic organism search. The results revealed superiority than other methods. In another study [42], a backpack algorithm with dynamic programming was used to solve the resource allocation problem with the aim of reducing runtime and cost. Their major achievements have been the use of low-capacity resources and effective quantities for time-lapse parameters, network congestion, and precise job size.

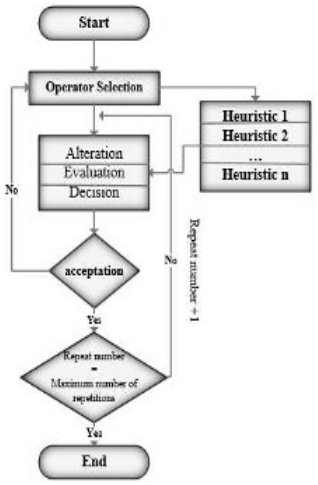
The general process of heuristic algorithms is shown in Figure3a. The flowchart of population-based algorithms is similar to Figure 3b, with the difference that instead of a solution, is created a set of solution. Hyper-heuristic algorithms (Figure 3c) are the exploratory search method in which automation, often combined with the techniques of machine learning, is the process of selecting, combining, producing or modifying several simple heuristics for solving computational problems.



(a) Heuristic Algorithm -



(b) Meta-Heuristic Algorithm



(c) Hyper-Heuristic Algorithm

Figure 3. Flowchart of approximate algorithms

2.5 Meta-heuristic Methods

One of the extra-heuristic algorithms to solve the resource allocation problem is the PSO optimization method presented in [11]. One of the features of this method is the provision of elastic and different sources with infinite resources as well as changing VM operation. One of the problems with this approach is the computational overhead for resource providers, which increased by the number of VMs and tasks [11]. A hybrid algorithm involving PSO and cats' optimization for resource allocation and VM management in the cloud has reduced the average response time, also it has increased resource utilization by up to 12% in compared with other benchmark algorithms [43]. ACO-based scheduling ACO has been used as another evolutionary algorithm for scheduling and resource allocation in the cloud, which has been more effective in the loading of resources and reducing of request failures. It used the fitness function based on trusted values and deadlines. As results, ACO minimized the throughput and the number of request failure and maximized the computation power [10]. The different scheduling problem solved in [44] by knapsack and optimized by ant colony optimization (ACO). In [45], resource allocation in the cloud is solved by the KnapGA genetic backpack algorithm. Its graceful function includes CPU utilization, network power, disk input, and output times. The researchers were able to reduce the energy consumption and migration of VMs.

Another meta-heuristic approach presented in [46] is the Bee-Based Algorithm for allocating resources to tasks in the fog network. Their algorithm as BLA is based on the optimized distribution of tasks in the fog nodes. The researchers using BLA find an optimal tradeoff between runtime and memory allocation for mobile users. The results show that runtime and memory allocation values by BLA are lower than GA and PSO algorithms.

The authors in [47] studied the resource allocation based on the meta-heuristic methods in the clouds. Each algorithm has some advantages and disadvantages. Scheduling solutions have issues like resource scaling, failure handling, security and storage-aware, dependent tasks, data transfer cost, dynamic resource provisioning for the IoT. In [48], resources are allocated to tasks in FC by NSGA-II method. This work simulated in MATLAB. They only compare their method with random allocation method. Their scheduling method reduced the latency and improve the stability of the task execution.

Studies show that much has been done in the field of cloud computing in the Cloudim simulator [49]. Of course, a number of FC-related work has been done on Cloudim or different programming frameworks. iFogsim, as successful development of Cloudsim, is very applicable to FC scheduling and resource management algorithms. The analysis of the heuristic algorithms proves that these methods have a long runtime and are not suitable for delay-sensitive scheduling problems.

We compare the mentioned scheduling methods by the problem, algorithm, objectives, framework, and environment in Table 1. The problems are categorized by task/job scheduling and resource provisioning.

Table 1. Summary of scheduling algorithms

Algorithm	Problem	Objectives	Framework	Environment
Knapsack [40]	Task scheduling	Complete time	Cloud	Matlab
Knapsack [42]	Task scheduling	Deadline and cost	Cloud	Cloudsim
KnapGA [45]	Task scheduling	I/O rate, migration count, and host occupation	Cloud	Simulation
ACO [10]	Task scheduling	Trust value and deadline	Cloud	Simulation
PSO [11]	Resource provisioning	Deadline and cost	Cloud	Simulation
DP [33]	Task scheduling	Deadline and cost	Cloud	Fabric compiler
MOO [34]	Task scheduling	Earliest finish time and completion time	Cloud	Real
Bidding strategy [35]	Task scheduling	Deadline, cost, and reliability	Cloud	Cloudsim
Backward shifting [36]	Resource provisioning	Host failure and task starting time	Cloud	Cloudsim
PSO [43]	Task Scheduling	Utilization of VMs and Response time	Cloud	Python
ACO [44]	Task scheduling	Start time	Smart Grid	Matlab
Heuristic [38]	Task scheduling	Makespan and execution cost	Cloud-Fog	Cloudsim
NSGA-II [48]	Resource scheduling	latency and stability	Fog	Matlab

DOI: https://doi.org/10.30564/jeisr.v1i1.1135

Mobility aware [37]	Task scheduling	Costs and QOS	Fog	iFogsim
NLP [39]	Task scheduling	Energy of thread execution	Fog	Simulation
BLA [46]	Job scheduling	Run time and memory allocation	Fog	BLA with C++
HHS [26]	Task scheduling	Security, CPU, and bandwidth	Fog	iFogsim
GKS [27]	Task scheduling	Cost and Energy	Fog	iFogsim
KnapSOS [41]	Task scheduling	CPU and bandwidth	Fog	iFogsim

3. Mobile Computing

The executing computation tasks of mobile applications on MDs requires a lot of resources, so they can offload to the Cloud. But Cloud is far from MDs and has challenges as high delay and power consumption. In general, mobile computing falls into the following three categories.

(1) MCC: Some processes on mobile devices require robust resources so they must be sent to the cloud data center. If a large number of users are logged in with delay-sensitive applications and want to cloud the data, then there is the problem of bandwidth [50, 2].

(2) MEC: In this type, tasks are performed near the mobile device [51]. Mobile edge computing reduces network congestion and performs tasks more efficiently [2].

(3) MFC: Since the cloud is far away from mobile devices [52], it is possible to send delay-sensitive tasks to the fog layer. As a result, it will save time and energy [53].

MFC can speed up the transfer of tasks to data centers [54]. This inexpensive and low-latency network architecture can be used as an infrastructure for the IoT.

4. Offloading

The issue of offloading in mobile computing has become an attractive topic for study and research in recent years. This issue has optimization goals that are outlined below. The objectives of offloading and scheduling in the reviewed articles are as shown in Figure 4.

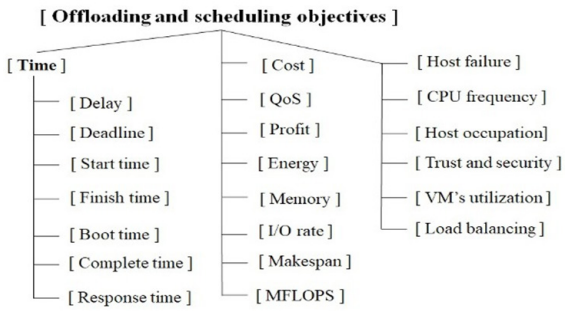


Figure 4. The offloading and scheduling objectives

4.1 MCC-based Offloading

Researchers in [55] addressed the problem of offloading and allocation of resources at MCC with the aim of reducing energy consumption. This has limitations such as response time, execution time and cost. The problem-solving method has been a greedy algorithm that has been able to optimize the target criteria. In [56], the authors developed an optimal pricing model (OPS) by examining the behavior of mobile users. This has led to a compromise between energy consumption and time delays.

The authors in [57] calculated the waiting time of cloud data centers and presented an offloading algorithm as HCOA based on the PSO. In [58], the ant colony-based offloading method as CMSACO is presented. The objectives of this method are profit, deadline, task dependency, resource differences, and load balancing. The analysis of results proves that total profits, time spent completing tasks, and network resource consumption have improved.

4.2 MEC-based Offloading

Various algorithms for resource allocation and offloading have been proposed in the MEC [59,60]. The authors addressed this in [60] and were able to reduce the cost of mobile devices by proving the Nash equilibrium.

In [61], the authors propose a functional architecture for different methods of offloading on mobile and IoT devices. In [62], the problem of offloading and allocating resources solved by the maximum greedy algorithm called by DGMS. They first introduced the policy of collecting energy for wireless devices from the environment. Then Lyaponuv optimization method for loading is presented. The results show the superiority of the proposed method over-centralized and random planning methods.

4.3 MFC-based Offloading (Module Placement)

In [63], the problem of offloading mobile device codes in MFC is solved. The researchers compared their approach to VM-based methods and container models. The proposed model performs better than others in the criteria of time delay, memory consumption, and image size and energy consumption. Another method of solving the offloading problem in FC is queue theory [53]. The authors proposed a multi-objective optimization method (MOIPM). They were able to improve energy consumption, delay in execution and cost.

In [64], the load of informed social calculation is provided for MFC. The proposed method is based on different queue models and energy collection models. Their algorithm has been able to reduce the running cost by solving the Nash generalized equilibrium problem. In [65],

 DOI: https://doi.org/10.30564/jeisr.v1i1.1135

researchers provided a task offloading method in MFC by classification and regression tree. They could optimize their method by Markov chain process.

The offloading algorithm presented in [66] is based on machine learning. The proposed method reduces the space of the answers by applying the Markov process and deep reinforcement learning. This method, called DQLCM, has been able to minimize latency.

4.4 MCC/MEC/MFC-based Offloading

Many studies have been presented in integrated architectures [67, 68]. The researchers [68] presented an integrated architecture of the cloud, edge, and IoT that reduced energy consumption. In [69], the offloading and scheduling problem is solved using the complex nonlinear programming method (integer). The researchers were able to reduce the most weight-related to cost, energy consumption and time delay criteria. In [70], the problem of offloading application code in hybrid cloud, edge and fog architecture is provided. The proposed method, SIMD, improves energy consumption, time, and the number of executable instructions as well as the migration overhead.

The mentioned offloading algorithms are summarized in Table 2. The problems are categorized by task and code offloading.

Table 2. Summary of offloading algorithms

Algorithm	Problem	Objectives	Framework	Environment
GABTS [55]	Task offloading and scheduling	Energy, response time, deadline, and cost	MCC	C++
OPS [56]	Task offloading and scheduling	Energy and delay	MCC	ThinkAir
HCOA [57]	Task offloading and scheduling	Energy	MCC	Simulation
CMSACO [58]	Multi-Task offloading	Profit and completion time	MCC	Simulation
W5 [61]	Task offloading	g CPU, memory, and network usage	MEC/MCC	Real
JCORAO [60]	Task offloading and scheduling	Deadline and cost	MEC	Hetnet simulation
DGMS [62]	Multi-Task offloading	Energy, battery, and CPU frequency	MEC	Simulation
Unikernel [63]	Code Offloading	Boot time, memory, and energy	MFC	Android-x86
MOIPM [53]	Task offloading	Energy, delay, and cost	MFC	Simulation
MPMCP [65]	Task offloading	Power, QoS, and security	MFC	Cloudsim
GNEP [64]	Task offloading	Cost	MFC	Simulation
DQLCM [66]	Task offloading	Delay and Energy	MFC	Real
MINP [69]	Task offloading and scheduling	Delay, Energy, and Cost	MFC/MCC	Matlab
SIMD [70]	Code Offloading	Energy, execution time, and MFLOPS	FC/MEC/MCC	Real

5. Conclusion and Suggestion

In this paper, we survey recent researches of computation offloading and scheduling in Cloud, edge, and FC. Challenges include time delay, energy, cost, trust, QoS, stability, memory, and security. Our categorizations are based on the issue, proposed strategy, objectives, framework, and test environment of various works. Moreover, based on the analysis, we propose machine learning algorithms to make smart distributed computing environments. We propose the machine learning methods to intelligent task scheduling and offloading in distributed computing. To simulate Cloud and FC applications, the Cloudsim and iFogsim libraries are closer to the real environment.

Studies show that fog computing has been more effective than cloud computing in implementing IoT scheduling and resource management algorithms. Things like power consumption, time delays, and optimizing server shutdown times are much easier in fog computing than cloud computing. Heavy processing is done instead of moving to cloud data centers in fog devices near the end users' location. Based on this study, the following can be suggested for future research:

(1) Applying Machine Learning Techniques: Extensive methods of artificial intelligence and machine learning can be very useful in various applications of IoT. Artificial intelligence methods have shown to be highly effective in a variety of issues. In the scheduling problem, applying these algorithms will help the system learn more and provide better solutions. By focusing on new classifications and adjusting the parameters of these algorithms according to the workflows and resources, it is possible to strive for better performance of the scheduling algorithms.

(2) Applications: One of the case studies could be the definition of new application models in IoT with different computations in fog computing. These include medical care, smart home, smart city, transportation system, car park, instant video analysis, traffic lights management, computer games, big data analytics, energy industry, water, and soil management.

(3) Objective Criteria: Many parameters can be investigated, some of which are: time (start, end, completion,

DOI: https://doi.org/10.30564/jeisr.v1i1.1135

wait, delay, and deadline), energy consumption, renewable energy, bandwidth, network resource consumption, efficiency, entry rate tasks, accuracy, cost, quality of service, location awareness, mobility, and inter-network connectivity.

(4) Security services: Considering more services in the security overhead model in addition to the three authentication, confidentiality and integration services as well as analyzing these issues in managing and allocating resources to input tasks can be introduced as another area of research.

(5) Scheduling Challenges in the IoT: Research on topics such as load balancing, load prediction, reliability and fault tolerance in offloading, resource provisioning, software-defined networking, network function virtualization and scheduling with fog architecture are considered to be very suitable developments that are at the beginning of the road today and very They have research. The following is a brief explanation of each:

A. Load balancing in fog computing is related to the placement of modules in the appropriate fog device, which requires the development of new methods and models.

B. Forecasting on the IoT can be a very useful topic. Since in dynamic systems, the rate of entry of tasks into the system varies, so it is very difficult to predict. On the other hand, its prediction can be considered as a pre-processing operation in scheduling and resource allocation operations and can improve it. Artificial intelligence methods work very well in this regard. For example, neural network-based approaches such as deep learning and reinforcement can be effective.

C. Trust in fog computing or IoT can be followed by block chain technology because there is no centralized server in distributed systems and so it is very close to the architecture of the hub. On trust, the goal is that new modules, before moving on to the new haze tool, ask their neighbors to trust the haze tool, which can be the subject of much research with mathematical modeling and intelligent algorithms.

D. Fault tolerance is also one of the most commonly used fields in fog computing. There are various methods in this area that examine the error and the solution to reduce or reduce it before and after it occurs.

E. Resource provisioning is one of the special challenges in fog computing that differs with resource allocation. Here are some ways to create or restore resources in the system.

F. Software-defined networks can be explored in fog computing. In this regard, the control section of the network is separated from its forwarding section, and the expressions in the network (such as router and switch) become ineffective decision-making tools and only perform tasks based on the flow tables that the controller communicates to them. Therefore, the programming algorithms of this section are very useful.

G. Network function virtualization is able to implement network elements as software components. Each of these components was traditionally implemented as a separate hardware device. For example, in a network, firewall, router, and load balancing tools have been the norm. Obviously, having separate devices for each application is very costly and has many management complexities. In fog architecture, setting up such sections in virtual terms is very useful and new.

(6) New architectures: Investigating new architectures such as cloud computing increases productivity and reduces time delays in IoT applications and can therefore extend the boundaries of knowledge. These include the dew architecture or network slicing. Given the growing number of network tools, this can be of interest to network architecture professionals.

(7) Development of new software frameworks: Due to the high cost of network tools and the large geographical location needed to test the methods, researchers can develop new simulation environments in fog architecture. Special frameworks can be designed for specific applications that are close to the IoT.

References

[1] Gupta, H. Vahid Dastjerdi, A. Ghosh, S. K. Buyya, R. 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, 2017, 47(9): 1275-1296.

[2] Gusev, M. Dustdar, S. Going back to the roots-the evolution of edge computing, an iot perspective, IEEE Internet Computing, 2018, 22(2): 5-15.

[3] Dizdarević, J., Carpio, F., Jukan, A., Masip-Bruin, X. A survey of communication protocols for internet of things and related challenges of fog and cloud computing integration. ACM Computing Surveys (CSUR), 2019, 51(6): 116.

[4] Gill, S. S., Chana, I., Singh, M., Buyya, R. RADAR: Self-configuring and self-healing in resource management for enhancing quality of cloud services. Concur rency and Computation: Practice and Experience, 2019, 31(1): e4834.

[5] Aazam, M. St-Hilaire, M. Lung, C. H. Lambadaris, I. Pre-fog: Iot trace based probabilistic resource estimation at fog, in: Consumer Communications and Networking Conference (CCNC), 2016 13th IEEE

 DOI: https://doi.org/10.30564/jeisr.v1i1.1135

Annual, Las Vegas, NV, USA, 9-12, IEEE, 2016: 12-17.

[6] Mahmud, R. Kotagiri, R. Buyya, R. Fog computing: A taxonomy, survey and future directions, in: Internet of Everything, Springer, 2018: 103-130.

[7] Rahmani, A. M. Gia, T. N. Negash, B. Anzanpour, A. Azimi, I. Jiang, M. Liljeberg, P. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: a fog computing approach, Future Generation Computer Systems, 2017: 641-658.

[8] Yousefpour, A., Fung, C., Nguyen, T., Kadiyala, K., Jalali, F., Niakanlahiji, A., Jue, J. P. All one needs to know about fog computing and related edge computing paradigms: A complete survey. Journal of Systems Architecture, 2019.

[9] Satyanarayanan, M. Bah, P. Caceres, R. Davies, N. The case for vm-based cloudlets in mobile computing, IEEE pervasive Computing, 2009, 8(4): 14-23.

[10] Gupta, P. Ghrera, S. P. Trust and deadline aware scheduling algorithm for cloud infrastructure using ant colony optimization, in: Innovation and Challenges in Cyber Security (ICICCS-INBUSH), International Conference on, Noida, India, 3-5. IEEE, 2016: 187-191.

[11] Rodriguez, M. A. Buyya, R. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds, IEEE Transactions on Cloud Computing, 2014, 2(2): 222-235.

[12] Yakubu, J., Christopher, H. A., Chiroma, H., Abdullahi, M. Security challenges in fog-computing environment: a systematic appraisal of current developments.Journal of Reliable Intelligent Environments, 2019: 1-25.

[13] Wang, T., Liang, Y., Jia, W., Arif, M., Liu, A., Xie, M. Coupling resource management based on fog computing in smart city systems. Journal of Network and Computer Applications, 2019, 135: 11-19.

[14] Chen, N. Chen, Y. Smart city surveillance at the network edge in the era of iot: Opportunities and challenges, in Smart Cities, , 2018: 153-176.

[15] Hosseinian-Far, A. Ramachandran, M. Slack, C. L. Emerging trends in cloud computing, big data, fog computing, iot and smart living, in Technology for Smart Futures, Springer, 2018: 29-40.

[16] Umang, N., Bierlaire, M., & Erera, A. L. Real-time management of berth allocation with stochastic arrival and handling times. Journal of Scheduling, 2017, 20(1): 67-83.

[17] Zhen, L., Liang, Z., Zhuge, D., Lee, L. H., & Chew, E. P. Daily berth planning in a tidal port with channel flow control. Transportation Research Part B: Methodological, 2017, 106: 193-217.

[18] Dulebenets, M. A. A comprehensive multi-objective optimization model for the vessel scheduling problem in liner shipping. International Journal of Production Economics, 2018, 196: 293-318. .

[19] Xiang, X., Liu, C., & Miao, L. Reactive strategy for discrete berth allocation and quay crane assignment problems under uncertainty. Computers & Industrial Engineering, 2018, 126: 196-216.

[20] Dulebenets, M.A. A Delayed Start Parallel Evolutionary Algorithm for Just-in-Time Truck Scheduling at a Cross-Docking Facility. International Journal of Production Economics. 2019, 212: 236-258.

[21] Dulebenets, M.A. A Comprehensive Evaluation of Weak and Strong Mutation Mechanisms in Evolutionary Algorithms for Truck Scheduling at Cross-Docking Terminals. IEEE Access. 2018, 6: 65635-65650.

[22] Serrano, C.; Delorme, X.; Dolgui, A. Scheduling of truck arrivals, truck departures and shop-floor operation in a cross-dock platform, based on trucks loading plans. International Journal of Production Economics. 2017, 194: 102–112.

[23] Khalili-Damghani, K.; Tavana, M.; Santos-Arteaga, F.J.; Ghanbarzad-Dashti, M. A. A customized genetic algorithm for solving multi-period cross-dock truck scheduling problems. Measurement. 2017, 108: 101–118.

[24] Ertem, M., Ozcelik, F., & Saraç, T. Single machine scheduling problem with stochastic sequence-dependent setup times. International Journal of Production Research, 2019, 1-17.

[25] Dong, C.; Li, Q.; Shen, B.; Tong, X. Sustainability in Supply Chains with Behavioral Concerns. Sustainability, 2019, 11: 4071.

[26] Rahbari, D., Kabirzadeh, S., and Nickray, M.. A security aware scheduling in fog computing by hyper heuristic algorithm. In Intelligent Systems and Signal Processing (ICSPIS), 2017 3rd Iranian Conference on. IEEE, 87-92.

[27] Rahbari, D., Nickray, M. Low-latency and energy-efficient scheduling in fog-based IoT applications. Turkish Journal of Electrical Engineering & Computer Sciences, 2019, 27(2): 1406-1427.

[28] Rodriguez, M. A. Buyya, R. A taxonomy and survey on scheduling algorithms for scientific workflows in iaas cloud computing environments, Concurrency and Computation: Practice and Experience, 2017, 29 (8): 1-23.

[29] Raidl, G. R., Puchinger, J., Blum, C. Metaheuristic Hybrids. In Handbook of Metaheuristics. Springer, Cham, 2019: 385-417.

[30] Frincu, M. E. Genaud, S. Gossa, J. Comparing pro-

DOI: https://doi.org/10.30564/jeisr.v1i1.1135

visioning and scheduling strategies for workflows on clouds, in: Parallel and Distributed Processing Symposium Workshops and PhD Forum (IPDPSW), 2013 IEEE 27th International, Cambridge, MA, USA, 20-24 May, IEEE, 2013, pp. 2101-2110.

[31] Singh, S., & Chana, I. A survey on resource scheduling in cloud computing: Issues and challenges. Journal of grid computing, 2016, 14(2): 217-264.

[32] Singh, P., Dutta, M., & Aggarwal, N. A review of task scheduling based on meta-heuristics approach in cloud computing. Knowledge and Information Systems, 2017, 52(1): 1-51.

[33] Malawski, M. Figiela, K. Bubak, M. Deelman, E. Nabrzyski, J. Scheduling multi-level deadline-constrained scientific workflows on clouds based on cost optimization, Scientific Programming 2015, 5-5.

[34] Durillo, J. J., Prodan, R. Multi-objective workflow scheduling in amazon ec2, Cluster computing, 2014, 17 (2): 169-189.

[35] Poola, D., Ramamohanarao, K., Buyya, R. Fault-tolerant workflow scheduling using spot instances on clouds, Procedia Computer Science, 2014, 29: 523-533.

[36] Zhu, X., Wang, J., Guo, H., Zhu, D., Yang, L. T., Liu, L. Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds, IEEE Transactions on Parallel and Distributed Systems, 2016, 27 (12): 3501-3517.

[37] Bittencourt, L. F., Diaz-Montes, J., Buyya, R., Rana, O. F., Parashar, M. Mobility-aware application scheduling in fog computing, IEEE Cloud Computing, 2017, 4(2): 26-35.

[38] Pham, X. Q., Huh, E. N. Towards task scheduling in a cloud-fog computing system, in: Network Operations and Management Symposium (APNOMS), 2016 18th AsiaPacific, Kanazawa, Japan, 5-7, IEEE, 2016: 1-4.

[39] Zahaf, H. E., Benyamina, A. E. H., Olejnik, R., Lipari, G. Energy-efficient scheduling for moldable real-time tasks on heterogeneous computing platforms, Journal of Systems Architecture, 2017, 74: 46-60.

[40] Lao, F. Zhang, X. Guo, Z. Parallelizing video transcoding using map-reduce-based cloud computing, in: Circuits and Systems (ISCAS), 2012 IEEE International Symposium on, Seoul, South Korea, 20-23 May, IEEE, 2012: 2905-2908.

[41] Rahbari, D., Nickray, M. Scheduling of fog networks with optimized knapsack by symbiotic organisms search. In 2017 21st Conference of Open Innovations Association (FRUCT). IEEE, 2017: 278-283.

[42] Rodriguez, M. A. Buyya, R. A responsive knapsack-based algorithm for resource provisioning and scheduling of scientific workflows in clouds, in: Parallel Processing (ICPP), 2015 44th International Conference on, Beijing, China, 1-4, IEEE, 2015: 839-848.

[43] Guddeti, R.M., Buyya, R., et al. A hybrid bio-inspired algorithm for scheduling and resource management in cloud environment', IEEE Transactions on Services Computing, 2017.

[44] Rahim, S., Khan, S. A., Javaid, N., Shaheen, N., Iqbal, Z., Rehman, G. Towards multiple knapsack problem approach for home energy management in smart grid, in: Network-Based Information Systems (NBiS), 2015 18th International Conference on, Taipei, Taiwan, 2-4, IEEE, 2015: 48-52.

[45] Chen, S. Wu, J. Lu, Z. A cloud computing resource scheduling policy based on genetic algorithm with multiple fitness, in: Computer and Information Technology (CIT), 2012 IEEE 12th International Conference on, Chengdu, China, 27-29, IEEE, 2012: 177-184.

[46] Bitam, S. Zeadally, S. Mellouk, A. Fog computing job scheduling optimization based on bee's swarm, Enterprise Information Systems 0, 2017: 1-25.

[47] Sheff, I., Magrino, T., Liu, J., Myers, A. C., van Renesse, R. Safe serializable secure scheduling: Transactions and the trade-off between security and consistency, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24-28, ACM, 2016: 229-241.

[48] Sun, Y. Lin, F. Xu, H. Multi-objective optimization of resource scheduling in fog computing using an improved nsga-ii, Wireless Personal Communications, 2018: 1-17.

[49] Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., Buyya, R. Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evalu-ation of resource provisioning algorithms, Software: Practice and experience, 2011, 41 (1): 23-50.

[50] Fernando, N. Loke, S. W. Rahayu, W. Mobile cloud computing: A survey, Future generation computer systems, , 2013, 29(1): 84-106.

[51] Mach, P. Becvar, Z. Mobile edge computing: A survey on architecture and computation offloading, IEEE Communications Surveys & Tutorials, 2017, 19(3): 1628-1656.

[52] Li, C., Xue, Y., Wang, J., Zhang, W., Li, T. Edge-oriented computing paradigms: A survey on architecture design and system management, ACM Computing Surveys (CSUR), 2018, 51(2): 39.

[53] Liu, L., Chang, Z., Guo, X., Mao, S., Ristaniemi, T.

 DOI: https://doi.org/10.30564/jeisr.v1i1.1135

Multiobjective optimization for computation offloading in fog computing, IEEE Internet of Things Journal, 2018, 5(1): 283-294.

[54] Roman, R., Lopez, J., Mambo, M. Mobile edge computing, fog et al.: A survey and analysis of threats and challenges, Future Generation Computer Systems, 2018, 78: 680-698.

[55] Tang, C., Wei, X., Xiao, S., Chen, W., Fang, W., Zhang, W., Hao, M. A mobile cloud-based scheduling strategy for industrial internet of things, IEEE Access, 2018, 6: 7262-7275.

[56] Shah-Mansouri, H., Wong, V. W., Schober, R. Joint optimal pricing and task scheduling in mobile cloud computing systems, IEEE Transactions on Wireless Communications, 2017, 16(8): 5218-5232.

[57] Zhang, J., Zhou, Z., Li, S., Gan, L., Zhang, X., Qi, L., Xu, X. Dou, W. Hybrid computation offloading for smart home automation in mobile cloud computing, Personal and Ubiquitous Computing, 2018, 22(1): 121-134.

[58] Wang, T., Wei, X., Tang, C., Fan, J. Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints', Peer-to-Peer Networking and Applications, 2017, 11(4): 793-807.

[59] Wang, Z., Zhao, Z., Min, G., Huang, X., Ni, Q., Wang, R. User mobility aware task assignment for mobile edge computing, Future Generation Computer Systems, 2018, 85: 1-8.

[60] Zhang, J., Xia, W., Yan, F., Shen, L. Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing, IEEE Access, 2018, 6: 19324-19337.

[61] Elazhary, H. H., Sabbeh, S. F. The w5 framework for computation offloading in the internet of things, IEEE Access, 2018.

[62] Chen, W., Wang, D. Li, K. Multi-user multi-task computation offloading in green mobile edge cloud computing, IEEE Transactions on Services Computing, 2018.

[63] Wu, S., Mei, C., Jin, H. Wang, D. Android unikernel: Gearing mobile code offloading towards edge computing, Future Generation Computer Systems, 2018.

[64] Liu, L. Chang, Z. Guo, X. Socially-aware dynamic computation offloading scheme for fog computing system with energy harvesting devices, IEEE Internet of Things Journal, 2018.

[65] Rahbari, D., Nickray, M. Task offloading in mobile fog computing by classification and regression tree. Peer-to-Peer Networking and Applications, 2019, 1-19.

[66] Tang, Z., Zhou, X., Zhang, F., Jia, W., Zhao, W. Migration modeling and learning algorithms for containers in fog computing, IEEE Transactions on Services Computing, 2018.

[67] Mohan, N. Kangasharju, J. Placing it right! optimizing energy, processing, and transport in edge-fog clouds, Annals of Telecommunications, 2018: 1-12.

[68] Lyu, X., Tian, H., Jiang, L., Vinel, A., Maharjan, S., Gjessing, S., Zhang, Y.Selective offloading in mobile edge computing for the green internet of things, IEEE Network, 2018, 32(1): 54-60.

[69] Du, J., Zhao, L., Feng, J., Chu, X. Computation offloading and resource allocation in mixed fog/cloud computing systems with min-max fairness guarantee, IEEE Transactions on Communications, 2017.

[70] Shuja, J., Gani, A., Ko, K., So, K., Mustafa, S., Madani, S.A. Khan, M.K. Sim-dom: A framework for simd instruction translation and offloading in heterogeneous mobile architectures, Transactions on Emerging Telecommunications Technologies, 2018, 29(4): e3174.

DOI: https://doi.org/10.30564/jeisr.v1i1.1135