2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST), June 10-11, 2020, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran.

Cost-Aware Task Scheduling in Fog-Cloud Environment

Tina Samizadeh Nikoui
Department of Computer Engineering
Science and Research Branch
Islamic Azad University
Tehran, Iran
Email: tina.samizadeh@srbiau.ac.ir

Ali Balador
Malardalen University
RISE SICS Vasteras
Sweden
Email: ali.balador@mdh.se

Amir Masoud Rahmani
Department of Computer Engineering
Science and Research Branch
Islamic Azad University
Tehran, Iran
Email: rahmani@srbiau.ac.ir

Zeinab Bakhshi
Malardalen University
Sweden
Email: zeinab.bakhshi@mdh.se

***Abstract*—Cloud computing provides computing and storage resources over the Internet to provide services for different industries. However, delay-sensitive applications like smart health and city applications now require computation over large amounts of data transferred to centralized cloud data centers which leads to drop in performance of such systems. The new paradigms of fog and edge computing provide new solutions by bringing resources closer to the user and provide low latency and energy efficiency compared to cloud services. It is important to find optimal placement of services and resources in the three-tier IoT to achieve improved cost and resource efficiency, higher QoS, and higher level of security and privacy. In this paper, we propose a cost-aware genetic-based (CAG) task scheduling algorithm for fog-cloud environments, which improves the cost efficiency in real-time applications with hard deadlines. iFogSim simulator, which is an extended version of CloudSim is used to deploy and test the performance of the proposed method in terms of latency, network congestion, and cost. The performance results show that the proposed algorithm provides better efficiency in terms of the cost and throughput compared to Round-Robin and Minimum Response Time algorithms.**

Keywords—Fog computing, Cloud computing, Task-scheduling, Internet of things.

I. INTRODUCTION (*HEADING 1*)

For delay-sensitive applications, late responses may lead to fault in the system or even system failure. For example, in an smart-health application that controls patient's condition, late responses may endanger human life [1]. This becomes even more critical when we consider a large amount of data generated by IoT devices. For instance, in smart city applications, IoT devices generate about one million records per second [2]. Considering high density of sensors in a number of applications and low accuracy of measurement, the possibility of duplicate and error-prone data in IoT environments is significant [1]. Therefore, this duplicated and error-prone data might increase the size of unnecessary data. Transferring huge amount of data and requests to the cloud may result in low utilization of network resources, high transmission delay, financial costs and processing overhead, and network congestion. According to Bonomi et al. [3], fog computing as a promising solution is a highly virtualized platform that provides processing, storage, and networking capacities between IoT devices and cloud computing resources and generically, but not exclusively, fog is located at the edge of network.

A common architecture of fog computing consists of three main layers: device, fog and the cloud layer. Fog computing provides an intermediate layer between device and cloud layers, which includes large number of fog nodes that have the capability of processing, networking, et. Since the fog nodes are supposed to be located near the device layer, requests with high requirements on delay can be processed at the fog layer and other requests can be sent to the cloud. Moreover, since it provides local processing capabilities, it can reduce the bandwidth usage and financial cost. Fog layer filters, analyzes and preprocesses the received requests and data. If the fog layer has enough available resources and can execute a task, the task is scheduled in the fog layer. Otherwise, if there are not enough resources to execute the task, it will be sent to the cloud layer.

If neither of them (i.e. cloud and fog) can respond at the specified time, the task will be rejected. So, it is important to find optimal placement of services and resources in the three

978-1-7281-7551-5/20/$31.00 ©2020 IEEE

tire IoT architecture to achieve improved cost and resource efficiency, higher QoS, and

higher level of security privacy. Moreover, the fog layer may be organized homogeneously or heterogeneously. In the homogeneous structure, the fog sources have the same processing, storage, and bandwidth capabilities, but in the heterogeneous case, resources have different capabilities. In case of heterogeneous fog devices, task scheduling has more complexity since resources have different capabilities in terms of processing speed, commutations and etc., which leads to various combinations (task and resource) [4]. Apart from response time and deadline for each task, other parameters, such as resource efficiency, energy, and financial cost may also need to be considered by task scheduling algorithms.

Considering several parameters along with the heterogeneity of tasks and resources imposes more complexity of the decision-making process. With increasing number of tasks and resources, possible solutions grow exponentially and there is no deterministic polynomial algorithm to solve this problem, and it is considered as a NP-Hard problem [4]. To solve this type of problems, evolutionary algorithms such as particle swarm optimization (PSO), genetic algorithms (GA) bees life algorithm (BLA) and ant colony optimization (ACO) can be applied. Due to global optimization and search ability of GA, it is widely used to solve task scheduling problems [5], [6], [7]. As [8] shows, GA algorithm performs well in task scheduling problem. The interplay and cooperation between the fog and the cloud has recently received considerable attention.

In this paper, we consider task scheduling in a cloud-fog computing system, where fog nodes will collaborate with the rented cloud nodes for efficiently executing users' large-scale offloading applications. The proposed algorithm decides whether to execute a task in the fog layer or send it to the cloud layer. The main contributions of this work are as follow:

- Reduce the cost of executing tasks in the fog/cloud layer, which includes the cost of bandwidth used (for sending/receiving requests and responses) and the cost of using computational resources.
- Increasing the success rate of responding to tasks within deadlines.

The rest of the paper is organized as follows: related studies are reviewed in Section II. A comprehensive description of proposed cost aware scheduling algorithm is provided in III. Simulation settings are expressed in Section IV. The results of our simulation studies are presented and discussed in Section V. The paper ends with conclusions drawn from the results in Section VI.

II. Background and Related Works

Task scheduling problem in cloud computing has been studied for several years. Recently, there are several studies, where new proposals are provided considering fog computing. In this section, we provide a review of scheduling models, used metrics and related studies.

A. Background ting a Template (Heading 2)

In a common task scheduling problem, a set of tasks should be scheduled on distributed nodes with one or more objectives. In this paper, the term "task" is used to indicate an atomic unit of processing, while the term "job" refers to a set of tasks. In the scheduling problem, each task has main parameters including resource requirements, size of task, ready time and deadline. Deadline is the time the t_i should be completed and ready time is the time that t_i is ready for execution [9]. Considering dependency in a task set, two tasks t_i and t_j are dependent if execution of task t_j should be started after completion of task t_i. In contrast, if tasks have no dependency relation with each other, tasks are considered independent and can be scheduled independently and the ready time for all tasks are defined based on their entrance time. This type of tasks is also called Bag-of-Tasks (BoT) [10].

Task scheduling approaches are motivated by one or more objectives and it is essential to evaluate different approaches to estimate their effectiveness. In this regard, key metrics that can be considered in scheduling approaches in distributed systems such as cloud and fog environment, are as follows [11], [12]:

- Makespan: total time taken to process a set of tasks for its complete execution. However in dynamic task scheduling, makespan is unusable, because the task stream is continually arriving and it is impossible to define the completion time for the set of tasks.
- Efficiency: proportion of execution time to total makespan.
- Throughput: total number of tasks completed successfully in a certain time period.
- Waiting time: time after submission and before the execution, also it includes the time that a task spends to obtain other resources or waits for some events.
- Execution time: time that a task is running and utilizes its resources.
- Response time: it refers to the time interval between submission and completion time for each task. In another word, it can be calculated as the summation of waiting time and execution time.
- Cost: total payment for usage of resources, and additional cost such as energy cost.

B. Related Works

Task scheduling in distributed computing systems, such as cloud and fog computing has been addressed in many different studies. Heuristics and meta-heuristics algorithms GA, PSO, BLA, and ACO can be used for scheduling and resource allocation problems while satisfying the requirements (i.e. lower makespan, higher efficiency in terms of energy, cost, utilization, etc.). In the following, some related studies are reviewed.

In a work by Nikoui et al. [13], the authors used GA algorithm to minimize the energy consumption of scheduling in

cloud computing systems. This paper assumes that central cloud broker uses GA to schedule a number of tasks on a set of virtual machines (VM) with the objective of optimizing energy consumption and the execution time. Results show that the proposed approach decreased the energy consumption in comparison with RR algorithm.

Another GA-based scheduling algorithm was proposed by Binh et al. [14] to decrease makespan and financial cost in fog networks. The authors presented a central task scheduling model, using a GA-based algorithm. In this paper, scheduler node is responsible for assigning arrival tasks to resources (cloud or fog nodes), considering two metrics: makespan and financial cost (cost of processing, memory and bandwidth). Moreover, weighting coefficients are included to provide more flexibility and trade-off between requirements. For evaluation, 10 heterogeneous fog nodes and 5 cloud nodes are simulated in iFogSim [15]. Based on conducted experiments, the proposed approach performs better in terms of makespan and financial cost compared to BLA. However, time-sensitivity of tasks and deadline are not included in the proposed approach.

Apart from GA-based algorithms, Bees life algorithms have been also applied to solve scheduling problems. For instance, in [16], BLA is applied to a central scheduler in heterogeneous fog computing systems, while considering time of execution and allocated memory. Job scheduler breaks down the received jobs to a number of tasks and they are assigned to worker fog nodes. Completed tasks and outcomes are combined together in the central scheduler, and final results are sent to end-devices. Experiments showed better results in terms of memory and time of execution compared to PSO and GA. The authors assumed fog nodes are heterogeneous, however, the cloud nodes are not included in the problem definition.

Some studies have selected hybrid approaches. For example, in [17], a combination of PSO and ACO algorithms has been used to schedule tasks in smart production line. Using this approach helps to compensate low accuracy characteristic of PSO, and slow search speed of ACO. Energy consumption and time (execution and transmission time) are included to increase efficiency of smart production line. Evaluations was conducted thorough simulation including 300 tasks and 10 fog nodes. Results show that the proposed algorithm performs better in terms of energy, time, and success rate. Similar to [16], the cloud nodes are not included in the problem definition. In addition, waiting time is not also included in response time calculation.

A multi-level architecture that consists of fog and cloud nodes is considered in [18]. This paper presents an ACO-based approach, called DATS-ACO, for heterogeneous fog computing systems to increase the profit of service providers. The authors considered task scheduling as a multi-dimensional 0-1 knapsack problem, where incoming tasks are assigned to a task queue to be scheduled; and the scheduler makes decision to execute tasks on suitable IoT devices or locally, or even sends them to cloud servers. Results show DATS-ACO has better results in terms of number of executed tasks and profit compared to First-Come-First-Served (FCFS) and Min-Min algorithms. However, since simulation setup includes only one datacenter and one fog node, collaboration between multiple fog nodes is not addressed.

Another cost-aware scheduling algorithm based on a multi-layer architecture is presented in [19] for time-constrained workflows. The proposed algorithm consists of two main phases. In the first phase, PSO generates solutions based on problem objectives (financial cost, computation and communication time). The second phase resolves conflicting tasks (conflicting tasks refers to set of tasks that are assigned to a single resource and have overlap based on their start to finish times), the authors applied Min-Min algorithm to calculate earliest start and finish times of conflicting tasks. For comparison, the authors compared the proposed approach with two strategies, the first one only uses the fog nodes and the second one only uses cloud layer resources. Experimental results showed that presented approach performs better in terms of execution time and cost. However, this approach does not consider waiting time, which has a critical impact on response time.

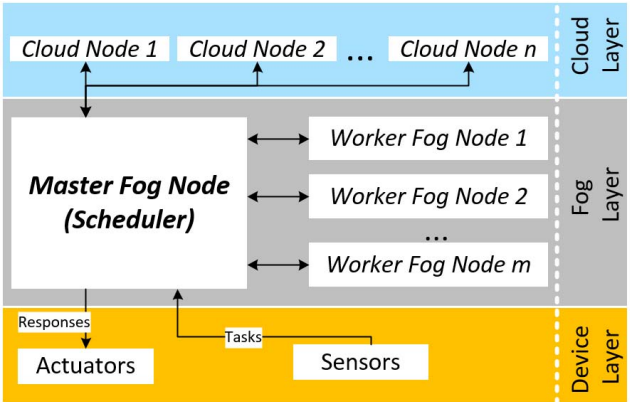

Fig. 1. Fog-cloud based task scheduler model.

As it was mentioned above, cloud computing scheduling and resource allocation approaches present good ideas, but they do not consider the load balancing between fog and cloud computing, so they are not applicable for a scenario where both cloud and fog architectures collaborate. Other approaches that are presented for fog computing do not present a cost-aware task scheduling approach for time-constrained tasks in a fog-cloud computing ecosystem that includes heterogeneous and collaborative resources. Also, a comprehensive and realistic performance evaluation, including different scenarios and configurations, was missing in the literature.

III. COST-AWARE TASK SCHEDULING

This section provides a description of the proposed approach that focuses on task scheduling in fog-cloud computing systems to increase the efficiency, in terms of response time and financial cost of processing.

Task scheduling is a decision-making process that is used on a regular basis in many manufacturing and services industries [20] and deals with the allocation of resources to tasks over a defined period of time in order to optimize one or more objectives. In distributed systems, such as cloud and fog

computing, there are different types of resources, and assigning resources to each task should be considered.

In this paper, to provide a trade-off between cost of processing and meeting the deadlines of time-constrained tasks, we present a genetic-based algorithm that is run in a central scheduling node. The system model of scheduling is presented in Figure 2. At the first step, task request is forwarded to MasterFogNode, which is in charge of running scheduling algorithm. The scheduler estimates the completion time of tasks with respect to networking and processing capability of available resources and requirements of tasks. According to the output of the estimation and scheduling algorithm, if a/the task cannot be completed on time, it will be rejected, otherwise it will be sent to corresponding resources (i.e. worker fog nodes or cloud nodes). Each node should process receiving tasks and send results back to MasterFogNode. The corresponding responses will be sent to the requester.

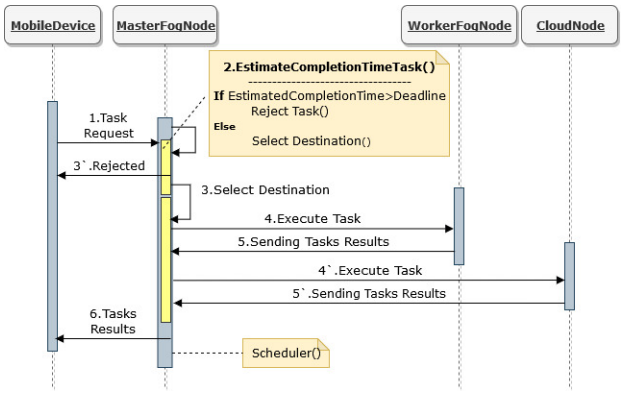

Fig. 2. System model of scheduling.

To provide an optimal solution, The Master Fog node uses Cost-Aware Genetic algorithm (CAG) to assign task set $T=\{t_1,t_2,\ldots,t_n\}$ to fog and cloud nodes $D=\{d_1,d_2,d_3,..,d_m\}$. Pseudo code of CAG is presented in Figure 3. A List of incoming tasks and available processing nodes are used as the input for this algorithm. Each task has main characteristics such as entrance time, number of instructions, upload and result data size, and deadline. Also, tasks are assumed as BoT. Since BoT applications consist of independent tasks with no communication [10] and there is no dependency between different tasks, the ready time for all tasks are defined based on their entrance time and they can be scheduled independently.

Similar to a task, each processing node has characteristics, such as processing rate, network and storage capabilities, and cost of processing. According to the CAG Algorithm, the first step is to define the representation method (chromosome encoding). Each possible solution in GA is represented by a chromosome (individual). To represent chromosomes, we use integer coding that includes the resource identifications. A set of chromosome in a generation is called a population [21].

In this work, the first generation of chromosome (initial population) is generated randomly. Another important step is definition of fitness function, which is used to measure the fitness of solutions by taking a candidate chromosome as input and producing how suitable the solution is with respect to the objectives of problem. Therefore, fitness function defines which chromosomes will survive or reproduced to next generations; and search process based on fitness function tries to provide the most suitable individuals. With the objectives of decreasing cost and increasing the success rate, we define the fitness function in (1), where the Cost shows total financial cost that is spent for execution of tasks, and *SuccessRate* shows the fraction of successfully completed tasks.

$$f(x) = \frac{Cost}{SuccessRate}. \qquad (1)$$

Similar to a task, each processing node has characteristics, such as processing rate, network and storage capabilities, and cost of processing.

According to CAG Algorithm which is shown in Figure 3, the first step is to define the representation method (chromosome encoding). Each possible solution in GA is represented by a chromosome (individual). To represent chromosomes, we use integer coding that includes the resource identifications. A set of chromosome in a generation is called a population [21].

Data: $T = \{t_1, t_2, .., t_n\}$ and $D = \{d_1, d_2, .., d_m\}$ such that t_i is a task, and d_j is an available node.
Result: Mapping List between tasks and resources
Function *Scheduler (list of incoming tasks and available resources):***is**
  Chromosome encoding;
  Set Fitness Function;
  Set Population Size;
  Generate initial population;
  repeat
    Apply Genetic Operators;
    Evaluate fitness Function $f(x)$;
  until *Termination Criteria*;
  Get Best Chromosome;
  Mapping List=Best Chromosome;
end

Fig. 3. Pseudo Code of CAG Algorithm

In this work, the first generation of chromosome (initial population) is generated randomly. Another important step is definition of fitness function, which is used to measure the fitness of solutions by taking a candidate chromosome as input and producing how suitable the solution is with respect to the objectives of problem. Therefore, fitness function defines which chromosomes will survive or reproduced to next generations; and search process based on fitness function tries to provide the most suitable individuals.

With the objectives of decreasing cost and increasing the success rate, we define the fitness function in (1), where the Cost shows total financial cost that is spent for execution of tasks, and *SuccessRate* shows the fraction of successfully completed tasks. Where T_L is the latency and T_p is the processing time and is calculated as shown in (3). In this equation, $I(t_i)$ indicates the size of task (number of instructions)

and d*i* shows the index of computing node. The waiting time of each task to get the processor (T_W) can be calculated using (4). In this equation, t_i and t_j are executed on the same nodes and t_j is executed before t_i. For simplicity, we assume that each of the nodes has a single processor.

$$T_P(t_i) = \frac{I(t_i)}{ProcessingRate(d_i)} + T_W(t_i) \quad (3)$$

$$T_W(t_i) = \sum_{j=1}^{k} T_P(t_j) \quad (4)$$

To calculate the latency (T_L) for task t_i, propagation time *PropagationTime* and transmission time $T_N(t_i)$ should be included (5). According to [22], propagation time that is required for a bit to travel from the source to the destination, can be calculated by dividing the distance between two points to the speed of propagation. Also, data transmission time $T_N(ti)$ is calculated by (6). Where $S_i(t_i)$ and $S_o(t_i)$ show upload data size and output data size of the task t_i and B_d shows the bandwidth between the two points (i.e. master node and destination worker node). Therefore, data transmission time is dependent to data size and bandwidth.

$$T_L(t_i) = T_N(t_i) + 2 \cdot PropagationTime(t_i) \quad (5)$$

$$T_N(t_i) = \frac{S_i(t_i)}{B_d} + \frac{S_o(t_i)}{B_d} \quad (6)$$

The cost of processing is calculated in (7), in the C_p and C_B show the cost of processing per unit of time in destination node and cost of network bandwidth .

$$Cost = \sum_{i=1}^{n} T_P(t_i) \cdot C_p + T_L(t_i) \cdot C_B. \quad (7)$$

IV. SIMULATION SETTINGS

The iFogSim [15] simulator, which is an extended version of Cloudsim [23] was found to be suitable, as it is specially designed for resource management in fog environment, and parameters, such as latency, network congestion, energy consumption and cost have been considered. We extended iFogSim by adding some classes for the genetic algorithm and new scheduling approaches.

The interval between incoming tasks are generated using exponential distribution with rate of 0.02. Specifications of resources and cost are configured based on [14]. Also, considering QoS of different IoT industry application [24], deadline of tasks are defined in range of 50 to 250 milliseconds. Size of tasks are also defined based on [14] and generated randomly. The Network latency values are configured based on [25]. The characteristics of workload, resources and parameters that are used in the genetic algorithm are mentioned in Table I.

A. Simulation Scenario

To evaluate the performance of the proposed algorithm in case of increasing the number of fog nodes in simulation experiments, the number of tasks and cloud nodes are kept constant (1000 tasks, 5 cloud nodes), while number of heterogeneous worker fog nodes varies between 2 and 20. The characteristics of the workload and nodes are kept constant in all simulations. Also, to achieve a normal value, each of the experiments ran 40 times and the average values are provided. Round-Robin (RR) is a simple but well-known algorithm that is one of the most common algorithms for resource allocation [26]. RR has been used by other papers in the literature for comparison [27], [28], [29], [30]. In addition to RR, Minimum Response Time (Minimum Completion Time) [31], which assigns each tasks to the resource with minimum completion time is used for comparison. In addition, to ensure system stability over time, simulations are repeated with different simulation duration.

TABLE I. CONFIGURATION PARAMETERS

Parameter	Value
Size of tasks (Million Instructions)	[1-100]
I/O data of a tasks (KB)	[1-10]
Deadlines of tasks (ms)	[50-250]
Processing rate fog nodes (MIPS)	[500-2000]
Processing rate cloud nodes (MIPS)	[3000-10000]
Bandwidth in fog environment (Mbps)	20
Bandwidth in cloud environment (Mbps)	100
Network latency from fog to cloud layers (ms)	100
Network latency from device to fog layers (ms)	10
Network latency in fog layer (ms)	[5-10]
Processing cost at cloud per time unit	[1- 2]
Processing cost at fog per time unit	[0.2-0.5]
Communication cost at fog per data unit	[0.01-0.02]
Communication cost at cloud per data unit	[0.05-0.01]
Crossover probability	0.09
Mutation probability	0.03
Population size	50

B. Evaluation metrics

To evaluate the proposed approach, we consider the following metrics:

- Success rate: the fraction of tasks that are completed within deadline. This metric is calculated by dividing the number of successfully completed tasks to total number of tasks.
- Cost: the financial cost that is spent for execution of tasks by (7).
- Cost per Instruction: the financial cost that is spent for a single instruction. This metric is calculated by dividing the total financial cost to total number of instructions for successful tasks.

V. RESULTS AND DISCUSSIONS

In this section, we investigate the performance of CAG algorithm in terms of *Cost* and *SuccessRate* and *CostPerInstruction*.

As shown in Figure 4, the experimental evaluation shows that the CAG algorithm has higher success rate (0.591-0.808) than the RR (0.415-0.701) and Minimum Response Time algorithm (0.479-0.709). Experiments show that when there are 2 fog nodes, about 591, 479 and 415 tasks are completed successfully in CAG, RR and Minimum Response Time algorithms, respectively.

As it was expected, the success rate improves with the increasing number of computing resources. Since, increasing number of computational resources at the fog layer (20 fog nodes), 808, 709 and 688 tasks are completed in CAG, RR and Minimum Response Time algorithm respectively.

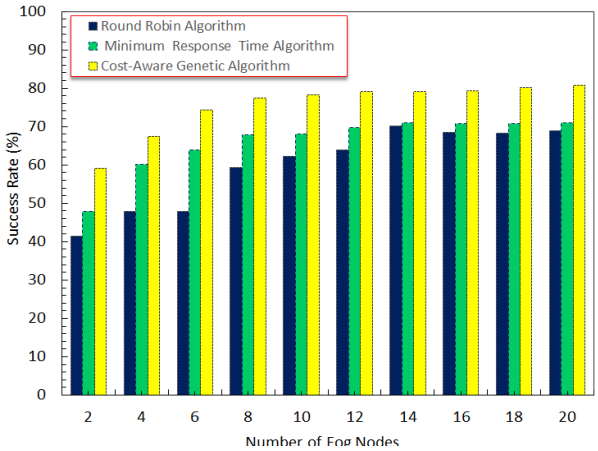

Fig. 4. Success Rate.

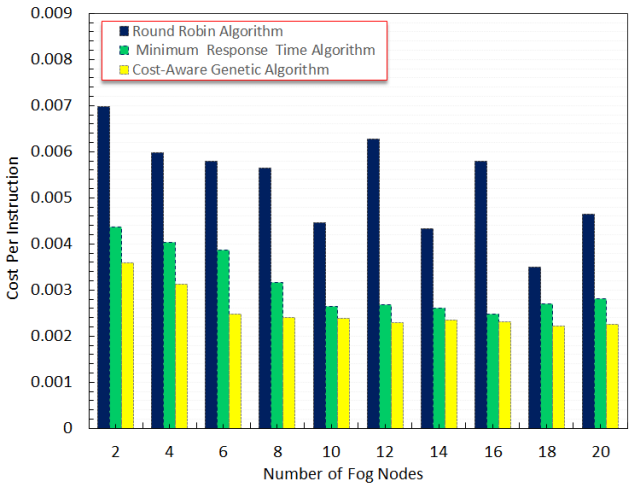

Fig. 5. Total Cost of Processing.

The underlying reasons are (*i*) increasing the number of computational resources reduces the waiting time for tasks and response time will be decreased. So, there is an increment in number of tasks that can be completed within deadline; (*ii*) another reason is related to lower latency of added resources (fog nodes) because they are located in the fog layer and response time is reduced due to shorter latency. However, RR and Minimum Response Time algorithms does not consider a global perspective of problem and assign tasks to resources one by one. Therefore, the success rate in RR and Minimum Response Time algorithms are lower than CAG algorithm.

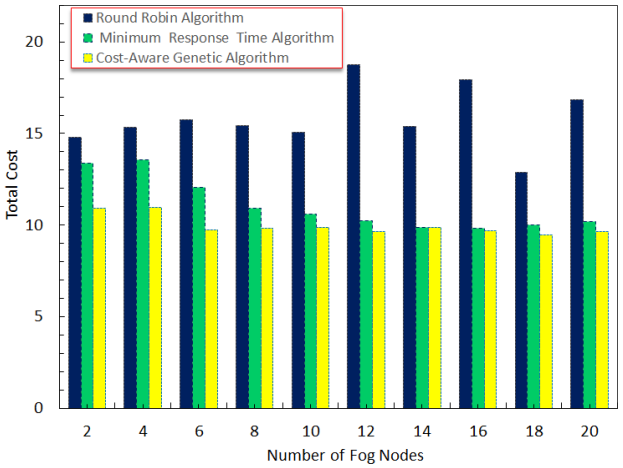

Fig. 6. Cost per Instruction.

Another metric that is included in evaluations is the Cost, which is depicted in Figure 5. Evaluation shows that RR has higher cost than the others. Also, Minimum Response Time compared to CAG has higher cost in most cases. For instance, in case of having 14 fog nodes, Minimum Response Time algorithm has the lower cost (9.8712) compared to CAG (9.8713). One reason is related to number of rejected tasks (i,e. if task can not be completed in deadline, it will be rejected by the scheduler). In addition, instructions that are belonged to missed tasks has a negative impact on cost. Considering these issues, *CostperInstruction* is included in our analysis. The measured values in experiments is presented in Figure 6.

This value is in range of ($3{:}4 \times 10^{-3}$ - $6{:}9 \times 10^{-3}$) for RR, (2.2×10^{-3} -$3{:}5 \times 10^{-3}$) for CAG algorithm and ($2{:}4 \times 10^{-3}$ - $4{:}3 \times 10^{-3}$) for Minimum Response Time algorithm. It can be concluded that a tradeoff between cost and success rate can be achieved using CAG algorithm.

VI. CONCLUSIONS

Fog computing paradigm made a significant impact on business models and the interplay and cooperation between the fog and the cloud has recently received considerable attention. Efficient task scheduling is a critical challenge for a cloud-fog ecosystem. In this paper, we consider task scheduling in a cloud-fog computing system, where fog nodes will collaborate with the rented cloud nodes for efficiently executing users' large-scale offloading applications. This work proposes a genetic-based algorithm that is called CAG, considering the trade-off between throughput and cost for hard real-time applications. To evaluate the performance of the proposed approach, we made some modifications in iFogSim simulator and evaluated CAG in different cases to investigate the impact of changes on the performance. Results show that CAG performs better than RR and Minimum Response Time algorithms in terms of financial cost and success rate. For the future works, we intend to evaluate the performance of

proposed approach under realistic workloads and applications. Another potential topic for future study is workflow scheduling and considering dependent tasks in such an environment.

ACKNOWLEDGMENT

This work was partially supported by the CelticNext projects RELIANCE (C2017/3-8) and Health5G (C2017/3-6), and Knowledge Foundation (KKS) via the ELECTRA project. Zeinab Bakhshi is also funded by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement n0 764785 through the FORA project.

REFERENCES

[1] T. Samizadeh Nikoui, A. Rahmani, and H. Tabarsaied, Data Management in Fog Computing: Principles and Paradigms, pp. 171–190. 01 2019.

[2] Y. Qin, Q. Z. Sheng, N. J. Falkner, S. Dustdar, H. Wang, and A. V. Vasilakos, "When things matter: A survey on data-centric internet of things," Journal of Network and Computer Applications, vol. 64, pp. 137 – 153, 2016.

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pp. 13–16, ACM, 2012.

[4] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, and R. Buyya, "An online algorithm for task offloading in heterogeneous mobile clouds," ACM Trans. Internet Technol., vol. 18, pp. 23:1–23:25, Jan. 2018.

[5] A. Bose, T. Biswas, and P. Kuila, "A novel genetic algorithm based scheduling for multi-core systems," in Smart Innovations in Communication and Computational Sciences, pp. 45–54, Springer, 2019.

[6] P. Rekha and M. Dakshayini, "Efficient task allocation approach using genetic algorithm for cloud environment," Cluster Computing, pp. 1–11, 2019.

[7] A. Jooyayeshendi and A. Akkasi, "Genetic algorithm for task scheduling in heterogeneous distributed computing system," International Journal of Scientific & Engineering Research, vol. 6, no. 7, pp. 1338–1345, 2015.

[8] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bol ¨ oni, M. Maheswaran, ¨ A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," Journal of Parallel and Distributed computing, vol. 61, no. 6, pp. 810–837, 2001.

[9] W. Shih, J. S. W. Liu, J. Chung, and D. W. Gillies, "Scheduling tasks with ready times and deadlines to minimize average error," SIGOPS Oper. Syst. Rev., vol. 23, pp. 14–28, July 1989.

[10] W. Cirne, F. Brasileiro, L. Costa, D. Paranhos, E. Santos-Neto, N. Andrade, C. De Rose, T. Ferreto, M. Mowbray, R. Scheer, et al., "Scheduling in bag-of-task grids: The PAUA case," in ´ 16th Symposium on Computer Architecture and High Performance Computing, pp. 124–131, IEEE, 2004.

[11] A. Burkimsher, I. Bate, and L. S. Indrusiak, "A survey of scheduling metrics and an improved ordering policy for list schedulers operating on workloads with dependencies and a wide variation in execution times," Future Generation Computer Systems, vol. 29, no. 8, pp. 2009–2025, 2013.

[12] S. H. H. Madni, M. S. A. Latiff, M. Abdullahi, M. J. Usman, et al., "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment," PloS one, vol. 12, no. 5, 2017.

[13] T. S. Nikoui, S. Jabbehdari, and A. Bagheri, "Providing a cloud broker-based approach to improve the energy consumption and achieve a green cloud computing," International Journal of Computer Applications, vol. 138, no. 1, 2016.

[14] H. T. T. Binh, D. B. Son, P. A. Duc, B. M. Nguyen, et al., "An evolutionary algorithm for solving task scheduling problem in cloud-fog computing environment," in Proceedings of the Ninth International Symposium on Information and Communication Technology, pp. 397–404, ACM, 2018.

[15] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "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, vol. 47, pp. 1275–1296, jun 2017.

[16] S. Bitam, S. Zeadally, and A. Mellouk, "Fog computing job scheduling optimization based on bees swarm," Enterprise Information Systems, vol. 12, no. 4, pp. 373–397, 2018.

[17] J. Wang and D. Li, "Task scheduling based on a hybrid heuristic algorithm for smart production line with fog computing," Sensors, vol. 19, no. 5, p. 1023, 2019.

[18] J. Fan, X. Wei, T. Wang, T. Lan, and S. Subramaniam, "Deadlineaware task scheduling in a tiered iot infrastructure," in GLOBECOM 2017 - 2017 IEEE Global Communications Conference, pp. 1–7, Dec 2017.

[19] R. Ding, X. Li, X. Liu, and J. Xu, "A cost-effective time-constrained multi-workflow scheduling strategy in fog computing," in International Conference on Service-Oriented Computing, pp. 194–207, Springer, 2018.

[20] M. L. Pinedo, Parallel Machine Models (Deterministic), pp. 113–150. Cham: Springer International Publishing, 2016.

[21] K.-L. Du and M. Swamy, "Search and optimization by metaheuristics," Techniques and Algorithms Inspired by Nature; Birkhauser: Basel, Switzerland, 2016.

[22] A. B. Forouzan, Data communications & networking (sie). Tata McGraw-Hill Education, 2007.

[23] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities," in 2009 International Conference on High Performance Computing Simulation, pp. 1–11, June 2009.

[24] N. Kherraf, H. A. Alameddine, S. Sharafeddine, C. M. Assi, and A. Ghrayeb, "Optimized provisioning of edge computing resources with heterogeneous workload in iot networks," IEEE Transactions on Network and Service Management, vol. 16, no. 2, pp. 459–474, 2019.

[25] R. Mahmud, F. L. Koch, and R. Buyya, "Cloud-fog interoperability in iot-enabled healthcare solutions," in Proceedings of the 19th international conference on distributed computing and networking, pp. 1–10, 2018.

[26] V. Meena, S. Niveditha, S. Arthika, N. Ilakkiyaa, V. Kalpana, and J. S. Kumar, "Optimal scheduler algorithm with least makespan and communication time for offloaded tasks in mobile cloud computing," in 2018 2nd International Conference on Inventive Systems and Control (ICISC), pp. 1306–1310, IEEE, 2018.

[27] S. Rasheed, N. Javaid, S. Rehman, K. Hassan, F. Zafar, and M. Naeem, "A cloud-fog based smart grid model using maxmin scheduling algorithm for efficient resource allocation," in International Conference on Network-Based Information Systems, pp. 273–285, Springer, 2018.

[28] S. Nazir, S. Shafiq, Z. Iqbal, M. Zeeshan, S. Tariq, and N. Javaid, "Cuckoo optimization algorithm based job scheduling using cloud and fog computing in smart grid," in International Conference on Intelligent Networking and Collaborative Systems, pp. 34–46, Springer, 2018.

[29] S. Rehman, N. Javaid, S. Rasheed, K. Hassan, F. Zafar, and M. Naeem, "Min-min scheduling algorithm for efficient resource distribution using cloud and fog in smart buildings," in International Conference on Broadband and Wireless Computing, Communication and Applications, pp. 15–27, Springer, 2018.

[30] Y. Nan, W. Li, W. Bao, F. C. Delicato, P. F. Pires, Y. Dou, and A. Y. Zomaya, "Adaptive energy-aware computation offloading for cloud of things systems," IEEE Access, vol. 5, pp. 23947–23957, 2017.

[31] T. D. Braunt, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswarans, A. I. Reuthert, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgeno, et al., "A comparison study of eleven static heuristics for mapping a class of independent tasks onto ileterogeneous distributed computing systems," ECE Technical Reports, p. 19, 2000.