Received: 16 February 2020 | Revised: 21 April 2020 | Accepted: 2 May 2020

DOI: 10.1002/cpe.5850

WILEY

SPECIAL ISSUE PAPER

Cost-aware cloud bursting in a fog-cloud environment with real-time workflow applications

Georgios L. Stavrinides | Helen D. Karatza

Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece

Correspondence
Georgios L. Stavrinides, Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki 54124, Greece.
Email: gstavrin@csd.auth.gr

Summary

Cloud bursting is a concept originating from the hybrid cloud computing paradigm. During workload spikes, the local resources of the private cloud are supplemented by resources in the public cloud. This technique could also be applied in a fog computing environment, in order to handle workload fluctuations, by offloading applications to the cloud. Toward this direction, in this article, we propose a strategy for the utilization of supplementary cloud resources, in order to assist in the processing of Internet of Things workflow jobs that arrive dynamically in a fog environment. As the cloud involves higher data transfer latency and monetary cost, our approach takes into account these two factors, in addition to the real-time constraints of the workload. The proposed scheduling heuristic is based on the tradeoff between performance and monetary cost. During resource selection, different contribution factors of these two parameters are assessed. Furthermore, the proposed scheduling method is compared against a baseline policy that utilizes only the fog resources. The simulation experiments were carried out under different sizes of workflow input data and for workloads with soft and hard deadlines.

KEYWORDS

cloud bursting, fog computing, performance, real-time workflows, scheduling, workload offloading

1 | INTRODUCTION

Along with the explosion of the *Internet of Things (IoT)*, *fog computing* has emerged as a new paradigm, supplementing cloud computing. The fog extends the cloud to the network edge, close to where the IoT data are generated, in an attempt to tackle the problem of data transmission latency.[1] As in the case of cloud computing, fog resources can be virtualized. Consequently, a fog node can be a virtual machine (VM).[2,3] However, in contrast to the cloud computing paradigm, the computational capacity of the fog resources is typically limited.[4]

The IoT data often require processing within a deadline, in a *real-time* manner.[5] Examples include connected smart vehicles, fitness tracking, dynamic targeted advertising, automated production lines, traffic control, telemedicine, and healthcare monitoring.[6] IoT data are usually processed by real-time workflow jobs, consisting of tasks with precedence constraints among them. The output data of a workflow task are used as input by other tasks of the workflow. A component task without any parent tasks is called an *entry task*, whereas a task without any child tasks is called an *exit task*.[7]

The deadline of such jobs may be *soft* or *hard*, depending on the criticality of the application. In case soft time constraints are imposed on the workload and a job misses its deadline, its results are still acceptable, but their usefulness decreases over time. For instance, a delayed response to a user's query may be tolerated. However, it degrades the system performance and thus the user experience. On the other hand, in case hard time constraints are imposed on the workload and a job misses its deadline, it is discarded from the system as its results would

be useless. For example, a delayed processing of an autonomous car's sensor data may lead to a serious accident, or even worse, it may cost human lives.[8]

1.1 | Motivation

According to the fog computing paradigm, the main processing of IoT data is typically performed in the fog layer, close to where the data are generated.[9] However, the computational capacity of the fog resources is usually limited. On the other hand, the computational demands and real-time requirements of IoT applications continue to grow at a staggering rate. Cloud bursting is a concept originating from the hybrid cloud computing paradigm. During workload spikes, the local resources of the private cloud are supplemented by resources in the public cloud.[10,11] This technique could also be applied in a fog computing environment, in order to handle workload fluctuations, by offloading applications to the cloud. Consequently, it is imperative to explore alternative strategies that involve the collaboration between the fog and cloud resources. Novel scheduling heuristics should be employed, in order to effectively utilize the supplementary cloud resources, taking into account their higher communication latency, as well as monetary cost.

1.2 | Contribution

Toward this direction, in this article, we propose a strategy for the utilization of supplementary cloud resources, in order to assist in the processing of IoT workflow jobs that arrive dynamically in a fog environment. As the cloud involves higher data transfer latency and monetary cost, our approach takes into account these two factors, in addition to the real-time constraints of the workload. The proposed scheduling heuristic is based on the tradeoff between performance and monetary cost. During resource selection, different contribution factors of these two parameters are assessed. Moreover, the proposed scheduling method is compared against a baseline policy that utilizes only the fog resources. The simulation experiments were carried out under different sizes of entry task input data. Furthermore, this article extends our work in Reference 12, by considering real-time workflows with soft and hard deadlines, instead of only hard deadlines.

The remainder of the article is organized as follows: Section 2 provides an overview of related literature. Section 3 presents the system and workload models, as well as the adopted pricing scheme of the supplementary cloud resources. Section 4 describes the proposed scheduling heuristic. Section 5 presents the employed performance metrics, the experimental setup and the simulation results. Finally, Section 6 concludes the article, providing directions for future research.

2 | RELATED WORK

Supplementing fog resources with cloud resources has been the focus of several recent research efforts.[13-15] Enguehard et al[16] used queuing theory in order to build an analytical framework for evaluating the performance of a cloud offloading strategy in a fog environment. In order to achieve load-balancing, they proposed an approach based on request popularity. Their experiments showed that the proposed strategy performed measurably better than an optimized blind load-balancer. However, even though their approach considered latency constraints, it was only applicable to a single-task application.

Deng et al[17] studied the collaboration between fog and cloud platforms. They investigated the tradeoff between power consumption and transmission latency utilizing an approximate approach. The proposed method decomposed the primal problem into three subproblems of corresponding subsystems, which could be respectively solved. Simulations and numerical results revealed that by sacrificing modest computational resources in order to save communication bandwidth and reduce transmission delay, fog computing can significantly improve the performance of cloud computing. In order to deal with the uncertainty characterizing the run-time of real-time workload in a fog-cloud environment, Li et al[18] proposed a methodology that combined three aspects: task buffering, resource allocation, and task offloading to cloud resources. The simulation results revealed that the particular technique avoided task starvation while meeting the task deadlines. Both of the aforementioned works considered workload consisting of single tasks, without any precedence constraints among them.

Pham et al[19] proposed a workflow scheduling approach based on the collaboration between fog and cloud computing. The major objective of the proposed strategy was to achieve a tradeoff between the application time and the monetary cost of the cloud resources, under user-defined time constraints. The proposed technique was both fog and cloud-aware and suitable for real-time workflows, utilizing idle time slots during the scheduling process. However, it had the following limitations: (i) it was static and thus not practically suitable for the dynamic nature of IoT applications, (ii) only a single workflow application was considered for scheduling, (iii) it did not take into account the communication cost incurred by the transfer of data from the IoT layer to the fog and cloud resources, and (iv) both the performance and monetary cost factors had equal weight during the decision making process.

In Reference 20, on the other hand, we proposed a fog and cloud-aware heuristic, suitable for the dynamic scheduling of multiple real-time workflows, utilizing possible schedule gaps. The proposed approach took into account the communication cost incurred by the transfer of data from the sensors and devices in the IoT layer to the VMs in the fog and cloud layers. However, the proposed approach did not take into account the monetary cost incurred by the utilization of the cloud resources. On the contrary, in Reference 12, we proposed a scheduling strategy that took into account both the communication latency and monetary cost involved with the cloud resources, in addition to the real-time constraints of the workload. The proposed scheduling approach was based on the tradeoff between performance and monetary cost, utilizing different contribution factors of these two parameters during resource selection, as opposed to Reference 19. However, we considered only the case where hard deadlines were imposed on the workload. As in a fog environment workloads may also feature soft deadlines, this article extends our work in Reference 12, by evaluating the performance of the proposed scheduling heuristic for workflows with soft and hard deadlines. In the case of soft time constraints, an important aspect of the system performance is the tardiness of the workload. Consequently, it is one of the adopted performance metrics in this work.

3 | PROBLEM DEFINITION

3.1 | System model

The environment under study is based on a 3-tier architecture. The first tier encompasses the IoT sensors and devices, which transmit data to the fog layer for processing. The second tier comprises the fog layer. It consists of a set $V_{\text{fog}} = \{vm_1^{\text{fog}}, \dots, vm_{v_{\text{fog}}}^{\text{fog}}\}$ of v_{fog} VMs. The third tier represents the cloud layer. There is a set $V_{\text{cloud}} = \{vm_1^{\text{cloud}}, \dots, vm_{v_{\text{cloud}}}^{\text{cloud}}\}$ of v_{cloud} VMs. Each VM in the fog tier has a virtual CPU (vCPU) with operating frequency μ_{fog}, whereas each cloud VM has a vCPU with frequency μ_{cloud}. Each vCPU has its own queue of assigned tasks that need to be processed.

It is assumed that the VMs in both the fog and cloud tiers support the same instruction set and thus require the same number of clock cycles per instruction. This is a reasonable assumption, as the VMs offered by major cloud vendors, such as Amazon Web Services, Microsoft Azure, and Google Cloud Platform, typically use VMs that support the x86-64 instruction set.

The transfer rate r_{IoT} of data between the IoT and fog tiers is uniformly distributed in the range:

$$r_{\text{IoT}} \sim [\overline{r_{\text{IoT}}} \times (1 - H_{\text{IoT}}/2), \overline{r_{\text{IoT}}} \times (1 + H_{\text{IoT}}/2)], \tag{1}$$

where H_{IoT} is the heterogeneity degree of the network that connects the IoT and fog tiers, whereas $\overline{r_{\text{IoT}}}$ is the mean data transfer rate between the two layers.

The data transfer rate between two fog VMs vm_i^{fog} and vm_j^{fog} is denoted by r_{ij}^{fog}. It is uniformly distributed in the range:

$$r_{ij}^{\text{fog}} \sim [\overline{r_{\text{fog}}} \times (1 - H_{\text{fog}}/2), \overline{r_{\text{fog}}} \times (1 + H_{\text{fog}}/2)], \tag{2}$$

where H_{fog} and $\overline{r_{\text{fog}}}$ are, respectively, the heterogeneity degree and the mean data transfer rate of the virtual network in the fog tier.

Similarly, the transfer rate of data between two cloud VMs vm_i^{cloud} and vm_j^{cloud} is denoted by r_{ij}^{cloud}. It is uniformly distributed in the range:

$$r_{ij}^{\text{cloud}} \sim [\overline{r_{\text{cloud}}} \times (1 - H_{\text{cloud}}/2), \overline{r_{\text{cloud}}} \times (1 + H_{\text{cloud}}/2)], \tag{3}$$

where H_{cloud} is the heterogeneity degree and $\overline{r_{\text{cloud}}}$ is the mean data transfer rate of the virtual network in the cloud tier.

The VMs in the fog and cloud tiers are fully connected by a virtual network that connects the two layers over the Internet (e.g., through a site-to-site VPN). The data transfer rate between a fog VM vm_i^{fog} and a cloud VM vm_j^{cloud} is denoted by r_{ij}^{inter} and is uniformly distributed in the range:

$$r_{ij}^{\text{inter}} \sim [\overline{r_{\text{inter}}} \times (1 - H_{\text{inter}}/2), \overline{r_{\text{inter}}} \times (1 + H_{\text{inter}}/2)], \tag{4}$$

where H_{inter} is the heterogeneity degree of the virtual network that connects the fog and cloud tiers, whereas $\overline{r_{\text{inter}}}$ is the mean data transfer rate between the two layers.

There is a central scheduler running on a dedicated node in the fog tier that manages the available resources in both the fog and cloud layers. Furthermore, it is responsible for scheduling the tasks to all of the available VMs, that is, the VMs in the fog layer and the supplementary VMs in the cloud layer. The 3-tier architecture under study is shown in Figure 1.

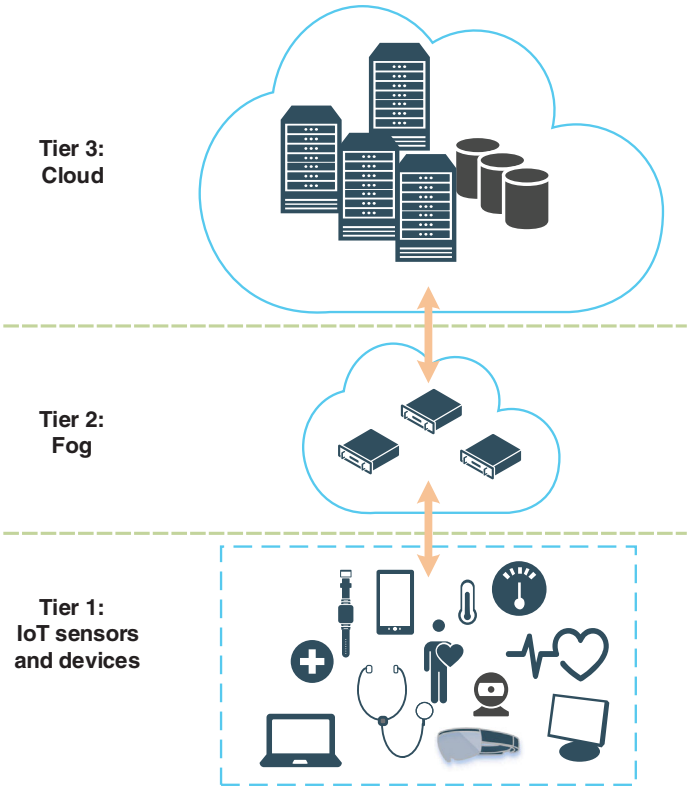

FIGURE 1 The 3-tier architecture under study

3.2 | Workload Model

The IoT sensors and devices generate data that are transmitted to the fog layer, where they are processed by real-time workflow jobs. Consequently, multiple real-time workflow jobs arrive dynamically at the central scheduler, in a Poisson stream with rate λ.

Each workflow job is represented by a *directed acyclic graph* $G=(N,E)$. N is the set of the nodes of the graph and E is the set of the directed edges between the nodes. Each node represents a component task n_i of the workflow, whereas a directed edge e_{ij} between two tasks n_i and n_j represents the data that must be transferred from task n_i to task n_j. Hereafter, the terms workflow and job are used interchangeably. It is noted that the component tasks of a workflow are not preemptible, as preemption in a real-time context may lead to performance degradation.

Each task n_i has a weight w_i that denotes its *computational volume*, which is expressed as the number of clock cycles required to execute the instructions of the particular task. The computational volume of each task is exponentially distributed with mean $\overline{w}$. The *computational cost* of the task n_i on a VM vm_k is given by:

$$Comp(n_i, vm_k) = w_i/\mu_k, \tag{5}$$

where μ_k is vm_k's operating frequency. Specifically, it is equal to μ_{fog} in case vm_k is a fog VM and μ_{cloud} in case vm_k is a cloud VM.

Each edge e_{ij} between two tasks n_i and n_j has a weight z_{ij} that represents its *communication volume*, which is expressed as the number of GB of data required to be transferred between the two tasks. The communication volume of each edge is exponentially distributed with mean $\bar{z}$. The *communication cost* of the edge e_{ij} is incurred when data are transferred from task n_i, scheduled on VM vm_m, to task n_j, scheduled on VM vm_n, and is defined as:

$$Comm((n_i, vm_m), (n_j, vm_n)) = z_{ij}/r_{mn}, \tag{6}$$

where r_{mn} is the data transfer rate between vm_m and vm_n. Specifically, it is equal to r_{mn}^{fog} when both VMs are in the fog tier, whereas it is equal to r_{mn}^{cloud} when both VMs are in the cloud tier. On the other hand, it is equal to r_{mn}^{inter} when the two VMs belong to different tiers. It is noted that in case both tasks n_i and n_j are scheduled on the same VM or on VMs that run on the same physical host, the communication cost of the edge e_{ij} is considered negligible.

Since each workflow operates on data transferred from the IoT layer, its entry tasks require input data that may vary in size. The *input data size* d_i of an entry task n_i is exponentially distributed with mean $\overline{d}$. The communication cost incurred by the transfer of input data from the IoT tier to an entry task n_i scheduled on a fog VM vm_m, is given by:

$$Comm(n_i, vm_m) = d_i/r_{\text{IoT}}, \tag{7}$$

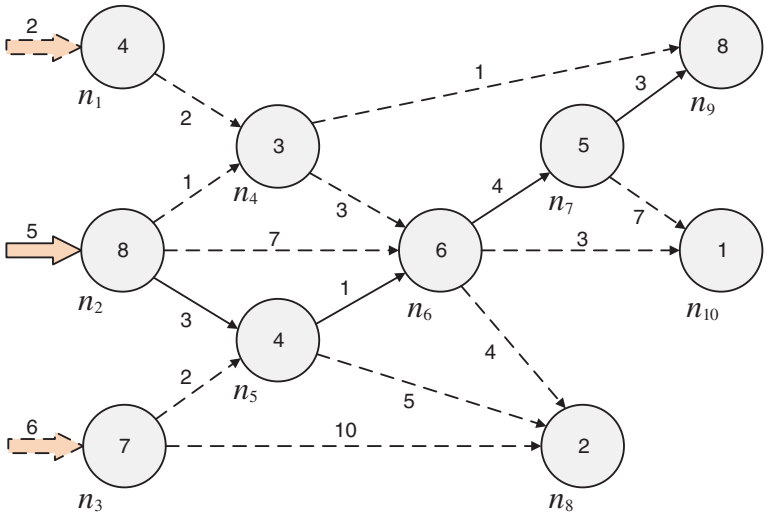

FIGURE 2 A workflow job represented by a directed acyclic graph with three entry tasks and three exit tasks. The number in each node denotes the average computational cost of the represented task. The number on each edge denotes the average communication cost between the two tasks that it connects. The arrows pointing to the entry tasks of the graph denote the initial input data of the job transferred from the IoT layer. The critical path of the graph is marked with solid arrows

where r_{IoT} is the data transfer rate between the IoT and fog tiers. In case the input data must be transferred from the IoT tier to a cloud VM vm_n, they are uploaded to the cloud through the fog layer. The communication cost in this case is:

$$Comm(n_i, vm_n) = d_i/r_{\text{IoT}} + d_i/r_{\text{inter}}, \tag{8}$$

where r_{inter} is the data transfer rate between the fog and cloud tiers.

The length of a path in the graph is the sum of the computational and communication costs of all of the tasks and edges, respectively, on the path, including the input data communication cost of the respective entry task on the particular path. The *critical path length CPL* is the length of the longest path in the graph. Each real-time workflow has an *end-to-end deadline D* within which all of its component tasks should finish execution. It is defined as:

$$D = AT + RD, \tag{9}$$

where *AT* is the *arrival time* of the workflow. *RD* is its *relative deadline* and is uniformly distributed in the range:

$$RD \sim [CPL, 2CPL]. \tag{10}$$

The deadline of a workflow may be soft or hard. In case it is soft, the *tardiness T* of a workflow is the amount of time by which its completion time exceeds its deadline. It is defined as:

$$T = \begin{cases} FT - D, & \text{if } FT > D \\ 0, & \text{otherwise} \end{cases}, \tag{11}$$

where *FT* is the *finish time* of the workflow. In case the deadline of a workflow is hard and is not met, the job is discarded from the system and is considered lost. A workflow is illustrated in Figure 2.

3.3 | Pricing of cloud resources

In contrast to fog resources, the utilization of supplementary cloud resources involves monetary cost. Specifically, the execution of a task on a cloud VM incurs monetary cost at a rate c_{proc} per second. Furthermore, the transfer of data from a fog VM to a cloud VM and vice versa, as well as the transfer of data from the IoT layer to a cloud VM, incurs monetary cost at a rate c_{data} per GB.

4 | SCHEDULING TECHNIQUE

As the real-time workflow jobs arrive dynamically at the central scheduler in the fog layer, their component tasks first enter a global waiting queue, until they become ready to be scheduled. A task becomes ready to be scheduled when all of its predecessor tasks have finished execution.

The ready tasks are scheduled on the available resources of the fog tier, utilizing supplementary resources in the cloud layer. In addition to the real-time constraints of the workload, our approach also takes into account the higher data transfer latency and monetary cost of the cloud resources. The proposed cloud bursting heuristic assesses the tradeoff between performance and monetary cost during the resource selection for each ready task. It consists of two stages: (i) a *task selection stage* and (ii) a *VM selection stage*.

4.1 | Task selection stage

The ready tasks in the global waiting queue are prioritized according to the deadline of their respective job. The task with the earliest respective deadline has the highest priority. Hence, tasks are prioritized according to the *Earliest Deadline First (EDF)* policy. In case two or more tasks have the same priority, the task with the highest average computational cost is selected first. This tie-breaking technique is employed, because when combined with EDF, it gives better results than other methods, such as random selection.[21]

4.2 | VM selection stage

Once a task n_j is selected by the scheduler, it is allocated to the VM vm_k (in the fog or cloud tier) that provides the minimum value for the following *score function*:

$$Score(n_j, vm_k) = \alpha \times EFT'(n_j, vm_k) + \beta \times EMC'(n_j, vm_k), \tag{12}$$

where $EFT'(n_j, vm_k)$ is the *normalized estimated finish time* of task n_j on vm_k, whereas $EMC'(n_j, vm_k)$ is the *normalized estimated monetary cost* of transferring the required input data and executing task n_j on vm_k. Both parameters are normalized in the range [0,1]. The definition of the *EFT* and *EMC* parameters is given below.

The *estimated finish time* $EFT(n_j,vm_k)$ of task n_j on vm_k is an indicator of the potential performance that might be achieved when the particular VM is selected for the execution of the particular task. It is defined as:

$$EFT(n_j, vm_k) = \max\{t_{data}(n_j, vm_k), t_{idle}(n_j, vm_k)\} + Comp(n_j, vm_k), \tag{13}$$

where $t_{data}(n_j,vm_k)$ is the time at which all input data of task n_j will be available on vm_k, whereas $t_{idle}(n_j,vm_k)$ is the time at which vm_k will be able to execute task n_j.

In order to calculate the term $t_{idle}(n_j,vm_k)$, we determine the position in vm_k's queue in which task n_j would be placed in case the particular VM was selected for the execution of the particular task. This position is determined by taking into account n_j's priority and by utilizing possible gaps in vm_k's schedule, as follows:

1. We first find the position at which task n_j would be placed in vm_k's queue, according to its priority.
2. In case all of the required input data of task n_j are available on vm_k, we check whether a schedule gap exists. A schedule gap is formed when the VM is idle and the task n_q placed at the head of the queue is still in the process of receiving its required input data from other hosts. The *capacity* g of the schedule gap is given by:

$$g = t_{data}(n_q, vm_k) - t_{current}, \tag{14}$$

   where $t_{data}(n_q,vm_k)$ is the time at which all of the required input data of task n_q will be received, whereas $t_{current}$ is the current time.
3. If a schedule gap exists, we try to insert task n_j into the gap. In order to do so, the following condition must hold:

$$g \geq w_j/\mu_k, \tag{15}$$

   where w_j is the computational volume of task n_j and μ_k is the operating frequency of vm_k's vCPU. In case we cannot place task n_j into the schedule gap (or a gap does not exist in vm_k's schedule), the position of task n_j in vm_k's queue is determined only by its priority.

The *estimated monetary cost* $EMC(n_j,vm_k)$ of task n_j on vm_k is an indicator of the potential monetary cost that might be incurred when the particular VM is selected for the execution of the particular task. It is defined as:

$$EMC(n_j, vm_k) = Cost_{comp}(n_j, vm_k) + Cost_{comm}(n_j, vm_k), \tag{16}$$

where $Cost_{comp}(n_j,vm_k)$ is the monetary cost required for processing n_j on vm_k, whereas $Cost_{comm}(n_j,vm_k)$ is the monetary cost required for transferring to the particular VM the required input data of task n_j.

As mentioned in Section 3.3, in case vm_k is in the cloud layer, monetary cost will be required for the execution of task n_j. That is:

$$Cost_{comp}(n_j, vm_k) = \begin{cases} c_{proc} \times Comp(n_j, vm_k), & \text{if } vm_k \in V_{cloud} \\ 0, & \text{otherwise} \end{cases}, \tag{17}$$

where c_{proc} is the rate per second at which the processing of a task on a cloud VM is charged, whereas $Comp(n_j,vm_k)$ is the computational cost of task n_j.

Furthermore, in case:

1. vm_k is in the cloud layer and some (or all) of task n_j's parent tasks were executed on fog VMs,
2. vm_k is in the fog layer and some (or all) of task n_j's parent tasks were executed on cloud VMs or
3. vm_k is in the cloud layer and n_j is an entry task of its respective workflow job and thus requires input data from the IoT layer,

the transfer of the required input data entails monetary cost, which is given by:

$$Cost_{comm}(n_j, vm_k) = \begin{cases} c_{data} \times \sum\limits_{n_i \in N_{fog}(n_j)} z_{ij}, & \text{if } vm_k \in V_{cloud} \\ c_{data} \times \sum\limits_{n_i \in N_{cloud}(n_j)} z_{ij}, & \text{if } vm_k \in V_{fog} \\ c_{data} \times d_j, & \text{if } n_j \in N_{entry} \text{ and } vm_k \in V_{cloud} \\ 0, & \text{otherwise} \end{cases}, \tag{18}$$

where c_{data} is the rate per GB at which data transfers in and out of the cloud are charged, z_{ij} is the communication volume of the edge e_{ij} between a parent task n_i and task n_j, $N_{fog}(n_j)$ is the set of n_j's parent tasks that were executed on fog VMs, $N_{cloud}(n_j)$ is the set of n_j's parent tasks that were executed on cloud VMs, and d_j is the input data size of n_j, in case n_j is an entry task.

The normalized values of the *EFT* and *EMC* parameters in Equation (12) are obtained as follows:

$$EFT'(n_j, vm_k) = \frac{EFT(n_j, vm_k) - \min\limits_{vm_m \in V}\{EFT(n_j, vm_m)\}}{\max\limits_{vm_m \in V}\{EFT(n_j, vm_m)\} - \min\limits_{vm_m \in V}\{EFT(n_j, vm_m)\}}, \tag{19}$$

and

$$EMC'(n_j, vm_k) = \frac{EMC(n_j, vm_k) - \min\limits_{vm_m \in V}\{EMC(n_j, vm_m)\}}{\max\limits_{vm_m \in V}\{EMC(n_j, vm_m)\} - \min\limits_{vm_m \in V}\{EMC(n_j, vm_m)\}}, \tag{20}$$

where $V = V_{fog} \cup V_{cloud}$. That is, V is the set of all of the available VMs in the fog and cloud tiers.

The coefficients α and β in Equation (12) are the *EFT* and *EMC contribution factors*, respectively. They indicate the weight of each parameter (*EFT* and *EMC*, respectively) in the final score of the examined VM. Consequently, they indicate the tradeoff between the performance that might be achieved when the particular VM is selected for the execution of the particular task, and the corresponding monetary cost that might be incurred. It is noted that the sum of the contribution factors α and β is equal to 1. The proposed scheduling heuristic is denoted by *MinScore*(α,β), where (α,β) is the pair of values used for the *EFT* and *EMC* contribution factors.

5 | PERFORMANCE EVALUATION

We investigated the impact of the *EFT* and *EMC* contribution factors of our cloud bursting algorithm, MinScore(α,β), on the system performance. The tradeoff between performance and monetary cost was assessed using the following pairs of values for the *EFT* and *EMC* contribution factors: (α,β)=(0.8,0.2), (0.85,0.15), (0.9,0.1), (0.95,0.05), and (1,0). The last pair of values was selected as a reference case, where the only criterion for resource selection was the performance that might be achieved, ignoring completely the monetary cost that might be incurred. Furthermore, the proposed approach was compared against a baseline policy, denoted by *MinEFT-Fog*, that utilized only the resources in the fog layer. The experiments were conducted using simulation, for mean entry task input data size $\bar{d}$ equal to 0.1 and 1 GB and for workflows with soft and hard deadlines.

5.1 | Performance metrics

The following performance parameters were employed:

- *Job Guarantee Ratio*, which is the ratio of the number of workflow jobs that finished execution within their deadline, over the number of all of the jobs that arrived at the central scheduler, during the observed time period.
- *Average Tardiness*, which is the average tardiness of the workflow jobs, during the observed time period.
- *Percentage of Tasks Executed on Cloud*, which is the percentage of the component tasks of all of the workflow jobs that arrived at the central scheduler that were executed on cloud VMs, during the observed time period.
- *Total Cost of Cloud Resources*, which is the total monetary cost for the utilization of the supplementary cloud resources, during the observed time period.

5.2 | Experimental setup

Due to the complexity of the investigated environment and in order to have full control on all of the system and workload parameters, we implemented our own discrete-event simulation program in C++, instead of using one of the simulation packages available, such as iFogSim.[22] The independent replications method was utilized for our simulation experiments. Even though relevant workload traces are generally available, we chose to use synthetic workload in our simulation experiments, in order to obtain unbiased results, not restricted to a particular type of workload.

As the resources in the fog tier are typically limited with low to moderate computational capacity, we considered that there were four physical hosts in the fog layer. Each host had a processor based on the Intel Xeon D-2142 IT processor, with eight cores. Each fog VM was assigned a vCPU mapped to a physical core. On the other hand, as the cloud typically encompasses a greater number of resources, with higher computational capacity, we considered that there were 16 physical hosts in the cloud layer. Each host had a processor based on the Intel Xeon Gold 6144 processor, with eight cores. As in the case of the fog VMs, each cloud VM was assigned a vCPU mapped to a physical core.

The mean computational volume of the tasks was selected to be equal to $\overline{w} = 3.8 \times 10^{11}$ clock cycles, so that on average, a task would take 2 minutes to execute on a VM. The mean edge communication volume $\bar{z}$ was selected to be equal to 1 GB. The arrival rate of the workflows was chosen to be λ=0.0015, in order for the environment under study to be stable. The network heterogeneity degree in all of the three tiers (IoT, fog and cloud) was chosen to be equal to H=0.5, since typically most networks feature moderate heterogeneity. The input parameters used in our simulation model are included in Table 1.

For each set of input parameters, we conducted 30 replications of the simulation program, with different seeds of random numbers in each run. Each replication was terminated when 10^4 workflow jobs had been completed. We found by experimentation that this simulation run length was long enough to minimize the effects of warm-up time. For every mean value, a 95% confidence interval was calculated. The half-widths of all of the confidence intervals were less than 5% of their respective mean values. In order to examine whether the differences between the obtained mean values were statistically significant, a 95% confidence interval was calculated for the difference between each pair of mean values. As the calculated confidence intervals did not include 0, the difference between each pair of mean values was statistically significant.

5.3 | Simulation results

The simulation results with regard to the job guarantee ratio are illustrated in Figure 3. It can be observed that the job guarantee ratio was about the same for workflows with soft and hard deadlines, for each of the mean entry task input data sizes, 0.1 and 1 GB. When the input data size increased from 0.1 to 1 GB, the job guarantee ratio decreased for both types of workflows and for all scheduling heuristics, except for the baseline policy, MinEFT-Fog. In that case, more jobs managed to finish within their deadline, compared to the case where the input data size was 0.1 GB.

This can be explained by the fact that in the MinEFT-Fog case only the computational resources in the fog layer were considered. Consequently, the workflows were considered more computationally intensive, compared to the case of the proposed heuristic, MinScore, where the target environment encompassed more resources, utilizing both the fog and cloud tiers. Therefore, the increase in the input data size did not burden the system performance as much as in the MinScore case, where the workload was considered more communication intensive. On the contrary, it led to larger slack times, allowing more jobs to meet their deadline. The same explanation holds for the average tardiness of the jobs, in case the deadlines imposed on the workload were soft, as shown in Figure 4.

However, in all of the examined scenarios, it can be observed that the proposed cloud bursting heuristic, MinScore, outperformed the baseline policy, MinEFT-Fog, for *EFT* contribution factor equal to or greater than α=0.9. The performance improvement was due to the fact that the proposed scheduling heuristic attempted to assign computationally demanding tasks with low communication requirements to VMs in the cloud, where there

TABLE 1 Input parameters of the simulation model

Parameter	Value
System Model Parameters	
Tier 1: IoT sensors and devices	
IoT-fog mean data transfer rate	$\overline{r_{\text{IoT}}} = 50$ Mbps
IoT-fog network heterogeneity degree	H_{IoT}=0.5
Tier 2: Fog	
Number of fog VMs	v_{fog}=32
Fog VM vCPU frequency	μ_{fog}=1.9 GHz
Fog mean data transfer rate	$\overline{r_{\text{fog}}} = 1$ Gbps
Fog network heterogeneity degree	H_{fog}=0.5
Fog-cloud mean data transfer rate	$\overline{r_{\text{inter}}} = 100$ Mbps
Fog-cloud network heterogeneity degree	H_{inter}=0.5
Tier 3: Cloud	
Number of cloud VMs	v_{cloud}=128
Cloud VM vCPU frequency	μ_{cloud}=3.5 GHz
Cloud mean data transfer rate	$\overline{r_{\text{cloud}}} = 1$ Gbps
Cloud network heterogeneity degree	H_{cloud}=0.5
Cloud VM processing rate	c_{proc}=$0.1 per hour
In/out cloud data transfer rate	c_{data}=$1 per TB
Workload Model Parameters	
Number of completed workflows	10^4
Workflow arrival rate	λ=0.0015
Number of tasks per workflow	$n \sim U[1,128]$
Mean entry task input data size	$\overline{d} = \{0.1, 1\}$ GB
Mean task computational volume	$\overline{w} = 3.8 \times 10^{11}$ clock cycles
Mean edge communication volume	$\overline{z} = 1$ GB
Scheduling Heuristics	
MinScore(α,β), where (α,β)={(0.8,0.2),(0.85,0.15),(0.9,0.1),(0.95,0.05),(1,0)}	
MinEFT-Fog	

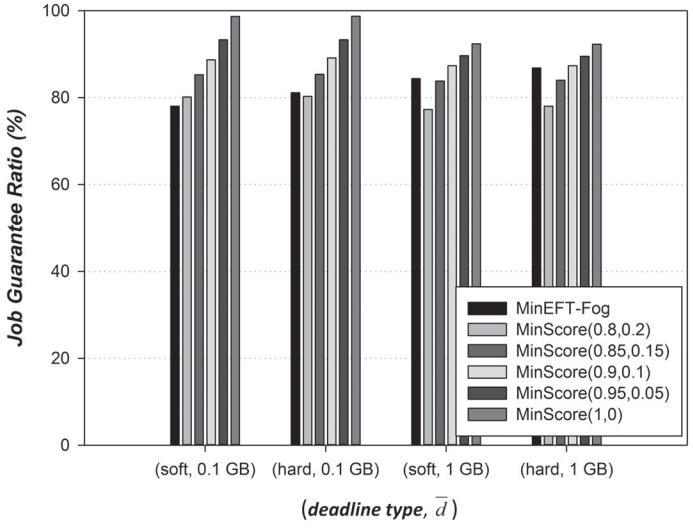

FIGURE 3 Job guarantee ratio (%) for mean entry task input data size $\overline{d} = \{0.1, 1\}$ GB and for workflows with soft and hard deadlines

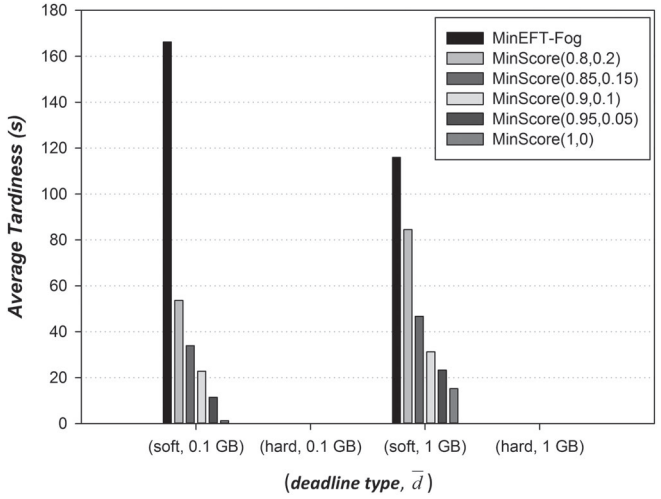

FIGURE 4 Average tardiness (s) for mean entry task input data size $\bar{d} = \{0.1, 1\}$ GB and for workflows with soft and hard deadlines

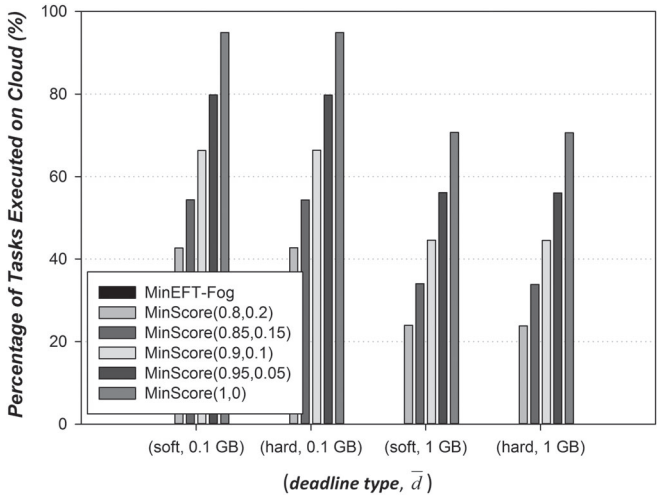

FIGURE 5 Percentage of tasks executed on cloud (%) for mean entry task input data size $\bar{d} = \{0.1, 1\}$ GB and for workflows with soft and hard deadlines

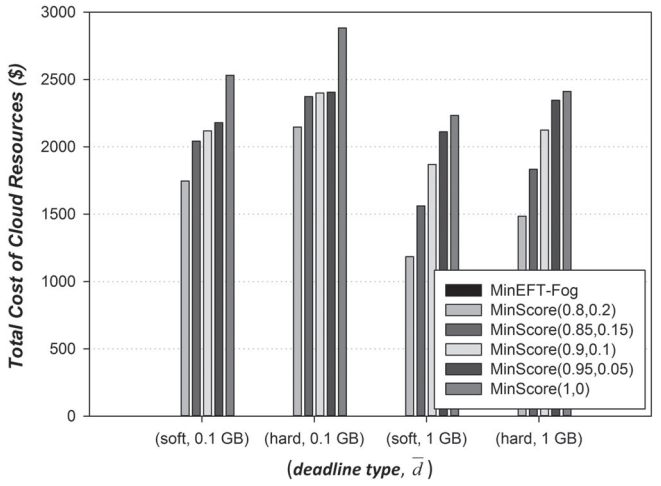

FIGURE 6 Total cost of cloud resources ($) for mean entry task input data size $\bar{d} = \{0.1, 1\}$ GB and for workflows with soft and hard deadlines

was a larger number of VMs with greater computational capacity than in the fog layer. On the other hand, it tried to schedule communication intensive tasks with low computational demands to fog VMs, which were closer to the IoT sources of the generated data, in an attempt to minimize the incurred communication cost. On the contrary, the baseline policy did not utilize any cloud resources.

When the proposed heuristic was employed, the percentage of tasks offloaded to cloud resources increased with the decrease of the *EMC* contribution factor, as shown in Figure 5. However, as the input data increased, fewer tasks were scheduled on the supplementary cloud resources. This was due to the fact that in the 1 GB case, the transfer of the required entry task input data to cloud resources would incur higher monetary cost, as opposed to the 0.1 GB case. Therefore fewer tasks were scheduled on the cloud tier. As Figure 6 shows, the incurred monetary cost for the utilization of the supplementary cloud resources followed a similar pattern with the percentage of tasks executed on cloud.

Comparing MinScore(0.95,0.05), where there was a tradeoff between performance and monetary cost, against MinScore(1,0), where there was no tradeoff, it can be observed that for a small average decrease of 4.23% in job guarantee ratio—but still a significant 10.98% average increase compared to the baseline policy, MinEFT-Fog—an average cost saving of 9.63% could be achieved for the utilization of the cloud resources.

6 | CONCLUSIONS AND FUTURE DIRECTIONS

In this article, we proposed a cloud bursting strategy, MinScore(α,β), for the utilization of supplementary cloud resources, in order to assist in the processing of real-time IoT workflow jobs that arrived dynamically in a fog environment. The proposed scheduling heuristic was based on the trade-off between performance and monetary cost. During resource selection, different contribution factors of these two parameters were assessed. Furthermore, the proposed scheduling algorithm was compared against a baseline policy, MinEFT-Fog, that utilized only the fog resources, under different sizes of entry task input data and for workflows with soft and hard deadlines. Our future work plans include the utilization of dynamic scaling techniques, in order to use more effectively the supplementary cloud resources.

ORCID

Georgios L. Stavrinides https://orcid.org/0000-0001-7289-9682

REFERENCES

1. Cisco. *Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are*. Technical Report C11-734435-00. San Jose, CA: Cisco Systems, Inc; 2015. https://www.cisco.com/c/dam/en_us/solutions/trends/iot/docs/computing-overview.pdf.
2. Mhedheb Y, Jrad F, Tao J, Zhao J, Kolodziej J, Streit A. Load and thermal-aware VM scheduling on the cloud. In: Kolodziej J, Di Martino B, Talia D, Xiong K, eds. *Proceedings of the 13th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'13)*. Lecture Notes in Computer Science 8285. Cham, Switzerland: Springer; 2013:101-114. https://doi.org/10.1007/978-3-319-03859-9_8.
3. OpenFog *OpenFog Architecture Overview*. Technical Report OPFWP001.0216. Needham, MA: OpenFog Consortium Architecture Working Group; 2016.
4. Bittencourt L. F., Diaz-Montes J, Buyya R, Rana O. F., Parashar M. Mobility-aware application scheduling in fog computing. *IEEE Cloud Comput*. 2017;4(2):26-35. https://doi.org/10.1109/MCC.2017.27.
5. Chen Y. *Service-Oriented Computing and System Integration: Software, IoT, Big Data, and AI as Services*. 6th ed. Dubuque, IA: Kendall Hunt Publishing; 2018.
6. Zhu J, Chen Y, Zhang M, et al. An edge computing platform of guide-dog robot for visually impaired. In: Puik E, Chen Y, Lu X, Moergestel L, eds. *Proceedings of the IEEE 14th International Symposium on Autonomous Decentralized Systems (ISADS'19)*. New York, NY: IEEE; ; 2019:23-30.
7. Stavrinides G. L., Karatza H. D. An energy-efficient, QoS-aware and cost-effective scheduling approach for real-time workflow applications in cloud computing systems utilizing DVFS and approximate computations. *Futur Gener Comput Syst*. 2019;96:216-226. https://doi.org/10.1016/j.future.2019.02.019.
8. Stavrinides G. L., Karatza H. D. Performance evaluation of a SaaS cloud under different levels of workload computational demand variability and tardiness bounds. *Simul Model Pract Theory*. 2019;91:1-12. https://doi.org/10.1016/j.simpat.2018.11.006.
9. Zhu J, Hu J, Zhang M, Chen Y, Bi S. A fog computing model for implementing motion guide to visually impaired. *Simul Model Pract Theory*. 2020;101:102015. https://doi.org/10.1016/j.simpat.2019.102015.
10. Guo T, Sharma U, Shenoy P, Wood T, Sahu S. Cost-aware cloud bursting for enterprise applications. *ACM Trans Internet Technol*. 2014;13(3):10:1-10:24. https://doi.org/10.1145/2602571.
11. Lee Y. C., Lian B. Cloud bursting scheduler for cost efficiency. In: Fox G. C., ed. *Proceedings of the IEEE 10th International Conference on Cloud Computing (CLOUD'17)*, 774–777. New York, NY: IEEE; 2017. https://doi.org/10.1109/CLOUD.2017.112.
12. Stavrinides G. L., Karatza H. D. Cost-effective utilization of complementary cloud resources for the scheduling of real-time workflow applications in a fog environment. In: Younas M., Awan I., Hara T., eds. *Proceedings of the 7th International Conference on Future Internet of Things and Cloud (FiCloud'19)*. New York, NY: IEEE; 2019;1–8. https://doi.org/10.1109/FiCloud.2019.00009.
13. Jararweh Y, Doulat A, AlQudah O, Ahmed E, Al-Ayyoub M, Benkhelifa E. The future of mobile cloud computing: integrating cloudlets and mobile edge computing. In: Chatzimisios P, Iossifides A, Mao S, Zheng K, Friderikos V, eds. *Proceedings of the 23rd International Conference on Telecommunications (ICT'16)*. New York, NY: IEEE; 2016:1-5. https://doi.org/10.1109/ICT.2016.7500486.
14. Pham X. Q., Huh E. N. Towards task scheduling in a cloud-fog computing system. In: Miyoshi T., Seok S. J., Yi C. W., eds. *Proceedings of the 18th Asia-Pacific Network Operations and Management Symposium (APNOMS'16)*. New York, NY: IEEE; 2016;1–4. https://doi.org/10.1109/APNOMS.2016.7737240.
15. Shah-Mansouri H, Wong V. W. S. Hierarchical fog-cloud computing for IoT systems: a computation offloading game. *IEEE Internet of Things*. 2018;5(4):3246-3257. https://doi.org/10.1109/JIOT.2018.2838022.
16. Enguehard M, Carofiglio G, Rossi D. A popularity-based approach for effective cloud offload in fog deployments. In: Altman E, Bianchi G, Zinner T, eds. *Proceedings of the 30th International Teletraffic Congress (ITC'18)*. New York, NY: IEEE; 2018:55-63. https://doi.org/10.1109/ITC30.2018.00016.
17. 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 Things J*. 2016;3(6):1171-1181. https://doi.org/10.1109/JIOT.2016.2565516.
18. Li L, Guan Q, Jin L, Guo M. Resource allocation and task offloading for heterogeneous real-time tasks with uncertain duration time in a fog queueing system. *IEEE Access*. 2019;7:9912-9925. https://doi.org/10.1109/ACCESS.2019.2891130.
19. Pham X. Q., Man N. D., Tri N. D. T., Thai N. Q., Huh E. N. A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. *Int J Distrib Sens Netw*. 2017;13(11):1-16. https://doi.org/10.1177/1550147717742073.
20. Stavrinides G. L., Karatza H. D.. A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments. *Multimed Tools Appl*. 2019;78(17):24639-24655. https://doi.org/10.1007/s11042-018-7051-9.

21. Park M, Han S, Kim H, Cho S, Cho Y. Comparison of tie-breaking policies for real-time scheduling on multiprocessor. In: Yang L. T., Guo M, Gao G. R., Jha N. K., eds. *Proceedings of the 2004 International Conference on Embedded and Ubiquitous Computing (EUC'04)*. Lecture Notes in Computer Science. 3207. Berlin, Germany: Springer; 2004:174-182. https://doi.org/10.1007/978-3-540-30121-9_17.
22. 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. *Softw Pract Exp*. 2017;47(9):1275-1296. https://doi.org/10.1002/spe.2509.

How to cite this article: Stavrinides GL, Karatza HD. Cost-aware cloud bursting in a fog-cloud environment with real-time workflow applications. *Concurrency Computat Pract Exper*. 2020;e5850. https://doi.org/10.1002/cpe.5850