TCA: A multi constraint real-time task scheduling algorithm for heterogeneous cloud environment

Sampa Sahoo[1], Sahil Kumar Sahu[2], Tanmay Kumar Rath[2] Bibhudatta Sahoo[1], Ashok Kumar Turuk[1]
[1]Department of Computer Science and Engineering, National Institute of Technology, Rourkela, India
[2]Department of Computer Science and Engineering, College Of Engineering and Technology, Bhubaneswar, India
Email:(sampaa2004, sahilsahu19970, tanmayrath98, bibhudatta.sahoo, akturuk)@gmail.com

***Abstract*—Cloud computing is an emerging computing paradigm, where cloud resources are available according to the pay-per-use pricing model and can be scaled dynamically depending on the application needs. Noticeably, many real-time applications that demand both time and functional correctness are moving to cloud. So, it requires efficient use of cloud resources to support applications need. In this context, task scheduling is a well-known technique to achieve the performance improvement of applications running in clouds. Again, execution time and execution cost play a vital role in deciding an appropriate VM for execution of a real-time task. In this paper, we formulate the real-time task scheduling problem as a multi-constraint optimization problem with time and cost constraint. Further, proposed a solution through time and cost-efficient best fit ($TCA : BF$) and first fit ($TCA : FF$) scheduling algorithms. Extensive simulation is performed to validate the superiority of the proposed approach compared to some existing ones.**

***Index Terms*—Cloud computing, Deadline, Execution time, Execution cost, Real-time task**

I. Introduction

Nowadays, many applications like healthcare system, video streaming, transaction system demand output within a specified timing constraint or deadline. The infrastructure used to develop such systems must provide high computing capability, ample storage volume, and reliable communication so that the timing constraint of the task can be met. For example, a health monitoring, and analysis framework enables a user (healthcare professionals and patients) to collect and disseminate health data anytime and from anywhere. The pervasive healthcare system is helpful for persons who need continuous monitoring but leave far away from their service provider, individuals facing difficulty in attending frequent therapy sessions, etc. Despite several advantages, the framework designed for healthcare system faces several challenges concerning scalability and economy [1]. Cloud computing plays a vital role in overcoming these issues. The evolution of cloud computing and virtualization technologies facilitates services with varying needs to operate in a virtualized environment.

Cloud computing is a widely used computing paradigm, combined with the benefits of reducing cost by sharing computing and storage resources, reliability, security, and scalability, etc. Cloud computing is a variation of utility and distributed computing which follows a pay-as-you-go pricing model to deliver services to the user. [3]. A cloud service model can be Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). IaaS deals with cloud infrastructure and associated middleware, PaaS provides an abstract platform to develop applications, and SaaS provides support for remote software Services [4].

Task scheduling is an efficient technique to achieve a performance improvement of the cloud system. The task scheduling decision must guarantee efficient use of cloud infrastructure with minimum task execution cost possible while assuring the application's quality of service (QoS) constraints. But, such decisions are generally hard to take as it depends on many factors like cost, energy consumption, pricing model, etc. Although there are an infinite amount of cloud resources, the economic cost of leasing cloud resources is a major concern for cloud users and cloud service providers [10], [11]. Usually, the cost incurred for usage of resources is a function of the time unit [11]. Cloud users are charged by an hourly-based, monthly or yearly subscription-based pricing model. Various tasks coming to the cloud demand a particular type of Virtual Machine (VM). For instance, Amazon EC2 offers many VM instances specific for computing-intensive, memory-intensive, and storage-intensive task [12]. The cloud scheduler must match the heterogeneous tasks to the best-fit VM instance for execution. The best-fit VM instance is chosen based on the objective of task scheduling like energy minimization, cost minimization, time minimization and so on.

Task priority or urgency strategy based on the weight of task attributes has a notable capability for efficient task scheduling and improved throughput of the system. There are several methods to assign a priority value to a task. In [13] the priority of a request is defined concerning task load and its execution time. Each task is assigned a priority level such as low, medium and high through a polynomial weighting scheme in [14]. Tasks are prioritized according to QoS deficit, and the process is better known as Largest Deficit First (LDF) [15]. This paper proposes a time and cost efficient (TCA) scheduling algorithm to meet the challenges as mentioned earlier for the execution of heterogeneous real-time tasks in the cloud environment. Our approach took into account the heterogeneity of computing resources (VM) as well as task and modeled task scheduling problem as a multi-constraint optimization problem. Specifically, we used the weighted sum method for the selection of best fit VM to execute a task. Our main contribution includes

- We proposed a scheduling algorithm named TCA to minimize the execution cost and execution time of a task

while meeting the deadline constraint. It considers both task and VM heterogeneity for mapping a task to an appropriate VM.
- We analyze the proposed algorithm through extensive simulations and experiments. We consider guarantee ratio, average execution cost and time under various scenarios to show the effectiveness of TCA over some existing schemes.

The paper is structured as follows: Section II reviews the work done by various researchers. Cloud system model is discussed in Section III. Section IV explains the TCA scheduling algorithm. Numerical results and conclusions are presented in Section V and Section VI respectively.

II. Related work

A variety of task scheduling methods have been proposed by researchers, in this paper, we mainly focus on task with deadline constraint. Authors in [1] designed a middleware for ECG data analysis to maximize cloud resource utilization. Sahni *et al.* [3] presented a cost-effective scheduling algorithm for scientific workflow in the public cloud. They considered the performance variation of VM and instance acquisition delay to schedule workflow with the aim of minimizing the overall execution cost while guaranteeing user-defined deadline. Sahoo *et al.* [4] presented best fit Earliest Deadline First (best fit) and first fit Earliest Deadline First (first fit EDF) algorithms for real-time tasks in the cloud, aiming maximized guarantee ratio, utilization of VM and throughput. Stavrinides *et al.* [5] proposed a heuristic to schedule real-time workflow in the cloud. They employed bin packing techniques in their scheduling approach to meet the objectives (i) minimized execution time and cost, (ii) applications deadline satisfaction. Calheiros *et al.* [6] designed an architecture for dynamic cloud resource provisioning and cost-efficient scheduling of deadline-based applications. Further, authors have proposed a novel billing strategy based on user's share on the utilization of cloud resources. Su *et al.* [7] proposed a scheduling algorithm that uses Pareto-dominance concept to select the most cost-efficient VM. Van den Bossche *et al.* [8] presented a cost-efficient scheduling mechanism for deadline constrained batch type applications. Hu *et al.* [9] presented Flutter a scheduling mechanism for big data jobs in order to reduce completion time and network cost.

Li *et al.* [13] introduced a cost-efficient scheduling technique for hybrid cloud workload to meet the response time and deadline constraints. Besides queuing analysis is done to allocate resources based on predictions of resource request of interactive services. In [14] evaluation and comparison of several dynamic mapping methods for priority and deadline based tasks in a heterogeneous working environment. Du *et al.* [15] studied different scheduling policies to support soft real-time application in the cloud. They developed outer bound on feasible QoS region and inner bound on policies based on dynamically prioritizing application's task. In [16] authors have discussed MapReduce framework and scheduling algorithms for the real-time task. Security is also one of the most significant challenges in the cloud environment. In this context, Li *et al.* [10] proposed SCAS, security and cost aware scheduling algorithm based on Particle Swarm Optimization (PSO) for scientific workflow in clouds. Researchers in [2] presented an entropy-based technique to deploy workflow reliably.

TABLE I
Task Scheduling Objectives

Research Works	**Objectives**	**Constraints**
Sahni *et al.* [3]	• Minimize execution cost	Deadline
Stavrinides *et al.* [5]	• Minimize execution time and cost	Deadline
Calheiros *et al.* [6]	• Minimize cost	Deadline
Van den Bossche *et al.* [8]	• Minimize cost	Deadline
Hu *et al.* [9]	• Minimize network cost and completion time	NA
Li *et al.* [10]	• Minimize execution cost	Deadline, Risk rate
Li *et al.* [13]	• Minimize cost	Deadline, Response time

Table I shows various scheduling objectives and constraints considered by researchers. From the literature survey, we can infer that time and cost are important parameters, specifically for deadline based tasks. Most of the paper give solution for single objective or constraint real-time task scheduling problem, and very few articles focus on the multi-constraint task scheduling problem. In this regard, we designed an algorithm for the multi-constraint real-time task scheduling problem.

III. System Model

Here, we assume a Heterogeneous Cloud System (HCS) consists of n number of VMs and m number of real-time tasks in a particular time instant.

In this study, we consider a set $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ of heterogeneous VMs. The VM heterogeneity is explained through different speed and cost. Each VM v_j is characterized by its speed $sp(v_j)$, which is measured in terms of Million Instructions Per Second (MIPS) and execution cost ζ_j per time unit.

Let $\mathcal{T} = \{t_1, t_2, ..., t_m\}$ is the task set. Each task t_i is characterized by arrival time $a(t_i)$, length or size $l(t_i)$ and deadline $dl(t_i)$. The length of task is measured in terms of Million Instructions (MI). The task heterogeneity is a function of its size, arrival time and deadline etc. Let $st(i,j)$ is the start time of task t_i on VM v_j. It is computed as

$$st(i,j) = max\{a(t_i), ft(p,j)\} \quad (1)$$

where $ft(p, j)$ is the finish time of task t_p on v_j. We assume that t_p is immediate predecessor of task t_i on v_j. The execution time $et(i, j)$ of a task t_i on VM v_j is calculated as

$$et(i, j) = \frac{l(t_i)}{sp(v_j)} \tag{2}$$

The finish time $ft(i, j)$ of t_i on v_j can be computed as

$$ft(i, j) = st(i, j) + et(i, j) \tag{3}$$

The $ft(i, j)$ of a task t_i determine whether the task t_i meet it's timing constraint or not. If $ft(i, j) \leq dl(t_i)$, then task's deadline can be met otherwise not.

The scheduler shown in Fig. 1 comprises of a task priority calculator, a real-time controller, and a resource allocator. The primary job of the scheduler is to allocate VMs to incoming tasks from the users. The resource allocator uses this information to assign VMs for executing the task.

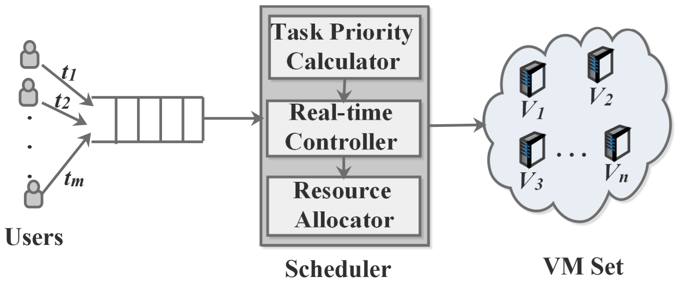

Fig. 1. Scheduling Model

When a new task arrives, the scheduler performs the following steps:

- First, the real-time priority calculator assigns different priority values to the incoming task according to the deadline and worst-case execution time ($WCET$).
- The tasks sorted by the priority value facilitate the scheduling operation.
- The real-time controller decides whether the task can meet its deadline or not. If there is no VM that can finish the task within its timing constraint, then the real-time controller informs the resource allocator to add new VMs.
- If task's timing requirement can be met then the resource allocator assign task to an appropriate VM.

Both real-time controller and resource allocator work together, first to meet task's timing requirement and then reduce the execution cost by assigning it to the best fit VM.

IV. Time and Cost Efficient Algorithm

The scheduling algorithm presented in Algorithm 1 works in two phases. The first phase (*Phase 1:*) consists of selecting a task based on the priority value assigned by the task priority calculator module. The second phase (*Phase 2:*) is about VM selection based on the utility function. Details of both the phases are discussed below:

A. Phase 1:

In this phase scheduler first assigns priority to the incoming tasks, where priority $pr(t_i)$ of the task t_i is calculated as

$$pr(t_i) = \frac{1}{sl(t_i)} \tag{4}$$

where slack time $sl(t_i)$ is computed as

$$sl(t_i) = |dl(t_i) - WCET| \tag{5}$$

$WCET$ is defined as the maximum execution time of t_i on a machine, i.e., $WCET = max(et(i, j))$. A lower value of denominator in Equation 4 indicates a higher priority for task t_i.

Algorithm 1 : Pseudo code of $TCA : BF$

```
Input: Arriving task set $\mathcal{T}$, Speed and execution cost of VM set $\mathcal{V}$;
Output: $SCH$; /* Schedule plan for arriving task */
1: $f_{TAG} \longleftarrow FALSE$;
2: Phase 1:
3: for each task $t_i$ do
4:     Compute priority value $pr(t_i)$ using Equation 4;
5: end for
6: Sort the tasks by their priority value in descending order.
7: Phase 2:
8: for each task $t_i$ in the sorted task list do
9:     for each VM $v_j$ in the system do
10:        Compute start time $st(i, j)$ by Equation 1, execution time $et(i, j)$ by Equation 2 and finish time $ft(i, j)$ by Equation 3;
11:    end for
12:    for each VM $v_j$ in the system do
13:        if $ft(i, j) \leq dl(t_i)$ then
14:            $f_{TAG} \longleftarrow TRUE$;
15:            Compute utility function $\mathcal{U}(i, j)$ using Equation 8;
16:        end if
17:    end for
18:    if $f_{TAG} == FALSE$ then
19:        Add new VMs.
20:    end if
21:    if $f_{TAG} == TRUE$ then
22:        Based on utility function select best fit $v_j$ for task $t_i$;
23:        $SCH \longleftarrow \langle t_i, v_j \rangle$;
24:    end if
25: end for
```

B. Phase 2:

In this phase, an appropriate VM is chosen for executing a task from a set of VMs by solving our multi-constraint optimization problem. To solve the problem, we define a utility function based on both execution cost and execution time. But, both the time and cost constraints considered in our problem has a different measurement unit. Both metrics are normalized to get a unit less value to eliminate computational problems caused by different measurement units. The normalized values are computed as

$$n\Gamma(i,j) = \frac{\Gamma_{mx}}{et(i,j)} \quad and \quad n\zeta(i,j) = \frac{\zeta_{mx}}{\zeta(i,j)} \tag{6}$$

where Γ_{mx} represents the maximum execution time of a task t_i on VM v_j, whereas ζ_{mn} indicates maximum execution cost of a task t_i on VM v_j. Let ζ_j is the processing cost of v_j per time unit. Then the cost incurred for executing t_i on v_j is calculated as

$$\zeta(i,j) = \zeta_j \times et(i,j) \tag{7}$$

The utility function for VM selection for task t_i is formulated as

$$\mathcal{U}(i,j) = \alpha \times n\Gamma(i,j) + (1-\alpha) \times n\zeta(i,j) \tag{8}$$

A high value of α indicates execution time aware scheduling whereas a low value indicates cost-aware scheduling. Here, we give equal priority to both time and cost, i.e., $\alpha = 0.5$.

The time and cost-efficient scheduling algorithm $TCA : BF$ (Algorithm 1) used a heuristic approach to assign each task to a VM in a way to aggressively guarantee task's deadlines while improving time and cost efficiency. In the first phase, it computes the priority value for each task t_i and sorts them based on this value (see lines 3-6). In the second phase, it computes the start time, execution time and finishes time of task t_i on each VM v_j. If deadline constraint of task t_i can be satisfied by a VM, then it calculates the utility function for executing task t_i on VM v_j (see lines 9-17). If t_i cannot be assigned to available VMs, then add new VMs (see lines 18-20). If t_i can be allocated, then it selects best fit VM to execute t_i for $TCA : BF$. Similarly, for $TCA : FF$ it selects first fit VM to execute t_i. The final assignment is stored in SCH (see lines 21-24).

Theorem 1. The time complexity of $TCA : BF$ algorithm is $O(mlog(m) + mn)$, where m is the number of tasks, n is the number of VMs.

Proof. The time complexity of calculating a task's priority value is $O(m)$ (lines 3-5). It takes $O(mlog(m))$ to sort tasks in descending order (line 6). The time complexity of obtaining task t_i's start time, execution time and finish time on all the VMs is $O(n)$ (lines 9-17). The time complexity of finding the best fit (minimum utility value) VM is $O(n)$. Rest lines are computed in $O(1)$ computation time. So, the time complexity of $TCA : BF$ algorithm is $O(m) + O(mlog(m) + O(m)(O(n) + O(n)) = O(mlog(m) + mn)$. □

V. Numerical Results

To illustrate the performance improvement gained by TCA we quantitatively compare it with $HEFT$ and CAA algorithms. For both the algorithms tasks are assigned priority based on Equation 4. The algorithms are briefly described as follows:

Heterogeneous Earliest Finish Time ($HEFT$). selects a VM to execute a task, which has the earliest finish time.

Cost Aware Algorithm (CAA). Differing from $TCA : BF$, it selects a VM that has the least execution cost for a task.

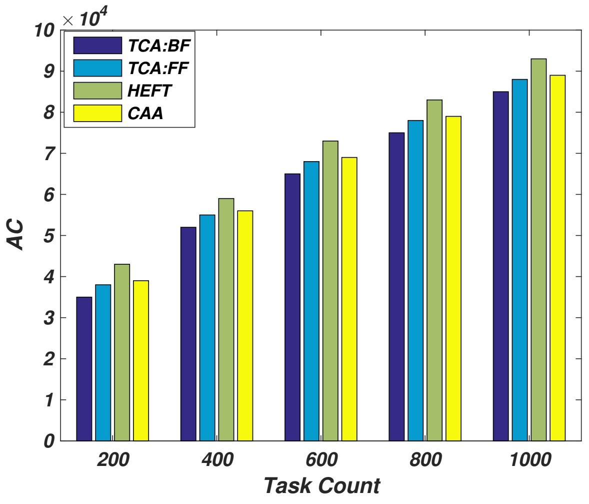

Fig. 2. AC Vs Task Count

We use guarantee ratio (GR), average cost (AC) and average execution time (AET) as the performance metrics to evaluate the system performance. The performance metrics are defined as follows:

GR. It is defined as the ratio between total task count satisfying deadline constraint and total task count.

AC. It is defined as the average cost required to execute a set of task satisfying the specified constraints.

AET. It is the average execution time of all the tasks in the system.

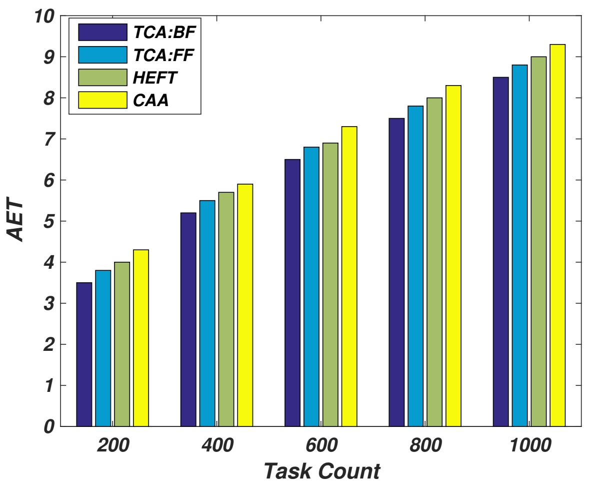

Fig. 3. AET Vs Task Count

We performed a group of experiments to show the performance comparisons of the algorithms in terms of performance metrics. The detailed simulation settings are given as follows. We assume that the task arrival rate follows a Poisson distribution. The deadline of the task is calculated as $dl(t_i) = a(t_i) + baseD$ where, $baseD$ is in uniform distribution $U(5, 10)$. Size of the task is in uniform distribution $U(3000 - 8000)$ MI. Each VM speed is set in the range $[2000 - 4000]$ MIPS. The cost of a VM is set between $[0.40 - 5.50]$\$ [2]. We assume that the cost of the VM is a function of its speed that means higher speed VM is costlier as compared to lower speed VM.

We can infer from Fig. 2 that as the task count increases the average cost value also increases. The increase in task count raises the busy time of a VM which in turns increases the computation cost. Besides, it can be found that $TCA : BF$ and $TCA : FF$ performs better than the other algorithms. This can be attributed to the use of utility function. Utility function helps to select a VM that is best suited concerning both cost and time as opposed to time only and cost only algorithms.

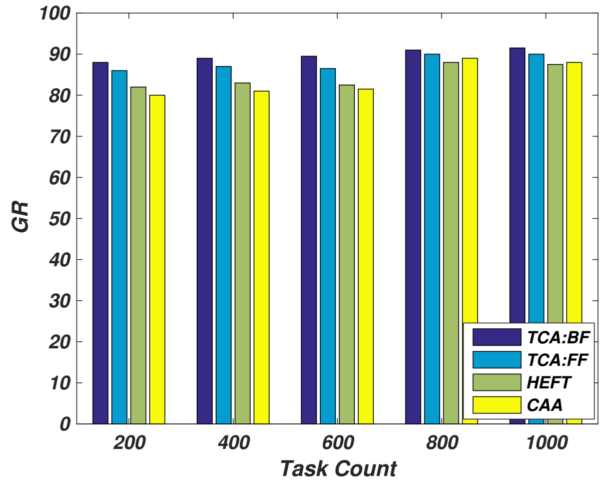

Fig. 4. GR Vs Task Count

From Fig. 3, it can be observed that the average execution time for $TCA : BF$ and $TCA : FF$ is less compared to other algorithms. As the task count grows, the average execution time also rises. But $TCA : BF$ and $TCA : FF$ have better performance, as VM selection depends on both time and cost. Since $HEFT$ selects VM based on earliest execution time, it has reduced AET value compared to CAA.

Fig. 4 demonstrates that all the algorithms have very little variation in guarantee ratio regardless of the task count. This is because infinite cloud resources enable the scheduler to start up new VMs as the task count increases. However, infinite cloud resources don't guarantee the successful completion of all the tasks. Besides, it can be observed that $TCA : BF$ and $TCA : FF$ have better guarantee ratio than other algorithms.

VI. Conclusion

Driven by the popularity of moving to the cloud, an increasing number of real-time (deadline-based) applications are hosted in the cloud. However, the heterogeneity in security, reliability, and cost of a VM in the cloud system, make the scheduler struggle in choosing an optimal VM for executing a user task. In this paper, we addressed the muti-constraint problem concerned with response time and execution cost for the successful execution of a set of real-time tasks. To achieve the objectives, we introduced time and cost-efficient scheduling algorithm named $TCA : BF$ and $TCA : FF$, where VM selection is made based on the solution of the multi-constraint problem. In our future work, we will focus on the Pareto-optimal concept to solve a multi-constraint problem in the HCS.

References

[1] S. Pandey, W. Voorsluys, S. Niu, A. Khandoker, and R. Buyya, "An autonomic cloud environment for hosting ECG data analysis services," in Future Generation Computer Systems, vol. 28, no. 1, pp. 147-154, 2012.
[2] Z. Wen, J. Caa, P. Watson and A. Romanovsky, "Cost Effective, Reliable and Secure Workflow Deployment over Federated Clouds," in IEEE Transactions on Services Computing, vol. 10, no. 6, pp. 929-941, 2017.
[3] J. Sahni and D. P. Vidyarthi, "A Cost-Effective Deadline-Constrained Dynamic Scheduling Algorithm for Scientific Workflows in a Cloud Environment," in IEEE Transactions on Cloud Computing, vol. 6, no. 1, pp. 2-18, 2018.
[4] S. Sahoo, S. Nawaz, S. K. Mishra and B. Sahoo, "Execution of real time task on cloud environment," in Annual IEEE India Conference (INDICON), pp. 1-5, 2015.
[5] G. L. Stavrinides, and H. D. Karatza, "A cost-effective and qos-aware approach to scheduling real-time workflow applications in paas and saas clouds," in 3rd International Conference on Future Internet of Things and Cloud (FiCloud), pp. 231-239, 2015.
[6] R. N. Calheiros, and R. Buyya, "Cost-effective provisioning and scheduling of deadline-constrained applications in hybrid clouds," in International Conference on Web Information Systems Engineering, pp. 171-184, 2012.
[7] S. Su, J. Li, Q. Huang, X. Huang, K.Shuang, and J. Wang, "Cost-efficient task scheduling for executing large programs in the cloud," in Parallel Computing, vol.39, no. (4-5), pp. 177-188, 2013.
[8] R. Van den Bossche, K. Vanmechelen and J. Broeckhove, "Cost-Efficient Scheduling Heuristics for Deadline Constrained Workloads on Hybrid Clouds," in IEEE Third International Conference on Cloud Computing Technology and Science, pp. 320-327, 2011.
[9] Z. Hu, B. Li and J. Luo, "Time- and Cost- Efficient Task Scheduling across Geo-Distributed Data Centers," in IEEE Transactions on Parallel and Distributed Systems, vol. 29, no. 3, pp. 705-718, 1 March 2018.
[10] Z. Li, J. Ge, H. Yang, L. Huang, H. Hu, H. Hu, and B. Luo, "A security and cost aware scheduling algorithm for heterogeneous tasks of scientific workflow in clouds," in Future Generation Computer Systems, vol. 65, pp. 140-152, 2016.
[11] H. M. Fard, T. Fahringer and R. Prodan, "Budget-Constrained Resource Provisioning for Scientific Applications in Clouds," in IEEE 5th International Conference on Cloud Computing Technology and Science, pp. 315-322, 2013.
[12] Amazon EC2, 2015. http://aws.amazon.com/ec2/.
[13] J. Li, S. Su, X. Cheng, M. Song, L. Ma, and J. Wang, "Cost-efficient coordinated scheduling for leasing cloud resources on hybrid workloads," in Parallel Computing, vol. 44, pp. 1-17, 2015.
[14] J. K. Kim, S. Shivle, H. J. Siegel, A. A. Maciejewski, T. D. Braun, M. Schneider, ... and A. Kaul, "Dynamically mapping tasks with priorities and multiple deadlines in a heterogeneous environment," in Journal of parallel and distributed computing, vol. 67, no. 2, pp. 154-169, 2007.
[15] Y. Du, and G. D. Veciana, "Scheduling for cloud-based computing systems to support soft real-time applications," in ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS), vol. 2, no. 3, p. 13, 2017.
[16] S. Sahoo, B. Sahoo, A. K. Turuk, and S. K. Mishra, "Real time task execution in cloud using mapreduce framework," in Resource Management and Efficiency in Cloud Computing Environments, IGI Global, pp. 190-209, 2017.