AN INTEGRATED FRAMEWORK FOR SLA-AWARE MULTI-OBJECTIVE TASK SCHEDULING IN CLOUD COMPUTING

Yadaiah Balagoni[1]
Department of CSE, Mahatma Gandhi Institute of Technology, Hyderabad, Telangana, India.
byadaiah_cse@mgit.ac.in.
Dr.R.Rajeswara Rao[2]
Department of CSE, JNTUK University College of engineering, Vizianagaram, Andhra Pradesh, India.
raob4u@jntukucev.ac.in

Abstract---**Cloud services are offered to consumers based on Service Level Agreements (SLAs) signed between Cloud Service Provider (CSP) and consumer. Due to on-demand provisioning of resources there is exponential growth of cloud consumers. Job scheduling is one of the areas that has attracted researchers to improve performance of cloud management system. Along with the on premise infrastructure, Small and Medium Enterprises (SMEs) also depend on public cloud infrastructure (leading to hybrid cloud) for seamless continuity of their businesses. In this context, ensuring SLAs and effective management of hybrid cloud resources are major challenging issues to be considered. Hence, there is a need for an effective scheduling algorithm which considers multiple objective functions like SLA (deadline), cost and energy while making scheduling decisions. Most of the state of the art schedulers in hybrid cloud environment considered single objective function. However, in real world, it is inadequate for scheduling effectiveness. To overcome this problem, we proposed an integrated framework which ensures SLAs (deadline), cost effectiveness and energy efficiency with an underlying scheduling algorithm known as SCE-TS. This algorithm is evaluated with different workloads and SLAs using a cloud platform. The empirical study revealed that the proposed framework improves scheduling efficiency in terms of meeting SLAs, cost and energy efficiency. It is evaluated and compared with the state of the art and found to be effective in making scheduling decisions in cloud environment.**

Keywords–Cloud computing, Service Level Agreements (SLA), task scheduling, cost-effectiveness, energy efficiency.

I. INTRODUCTION

Scheduling tasks in cloud computing has its impact on QoS of cloud services. It also leads to level of satisfaction in consumers besides making the cloud platform reliable. Given a set of jobs with different SLAs like deadline, it is essential to see that the jobs are finished in stipulated deadline. Efficient scheduling has its impact on the resource utilization as well. In cloud environment virtual machine (VM) instances are the primary resource units for rendering cloud services. Since cloud computing is on top of virtualization technology, for effective utilization of cloud resources it is very much essential to develop effective algorithms for VM scheduling. Many researchers contributed towards defining algorithms for scheduling jobs on VMs. Some of them like [2] used heuristic combination of multiple objective functions for scheduling decisions, before scheduling of any task on a VM. Some researchers considered cost parameter to improve scheduling efficiency as explored in [1]. Other researchers focused on energy efficiency as in [3].Heuristic functions are employed in [6] for workflow scheduling while Differential Evolution

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

Algorithm (DEA) is used for optimizations in scheduling. Another important optimization is Cuckoo PSO with multiple objective functions as discussed in [8]. Yet another optimization algorithm known as Ant Colony Optimization (ACO) is explored in [10] with specific enhancements for scheduling jobs.

A common thread found in many research articles is that, they focused on SLAs, with single objective functions like cost and energy efficiency either in public or private cloud environment. However, there was a little research found on a comprehensive scheduling framework that considers a hybrid cloud with multiple objectives like SLAs, cost and energy efficiency. Therefore, in this paper, we proposed a comprehensive framework that considered both private and public resources (hybrid cloud) while scheduling tasks. Accordingly, an integrated algorithm is proposed based on the prior methods defined by the authors of this paper.

The individual methods are found in [1], [2], [3] and [4] are integrated and provided a framework by considering SLA(deadline), cost and energy efficiency in hybrid cloud environment. Our contributions in this paper are summarized as follows.

- An integrated framework is proposed to exploit local and public cloud resources in a hybrid cloud environment. It considers task scheduling with an integrated approach which includes SLA (deadline), cost and energy efficiency.
- An algorithm named SLA-aware, Cost-effective and Energy-efficient Task Scheduling (SCE-TS) to enhance efficiency in scheduling decisions.
- A prototype application is built with Cloud-Sim, a cloud computing simulation framework, with Hadoop MapReduce programming paradigm. The empirical results revealed the usefulness of the proposed framework.

The remainder of the paper is structured as follows. Section II reviewed literature on the scheduling algorithms in cloud computing. Section III presents the proposed framework and underlying algorithms. Section IV provides experimental results. Section V concludes the paper and provides directions for future scope of the research.

II. RELATED WORK

Many researchers contributed to the scheduling enhancements in cloud computing. Yadaiah and Rajeswra Rao [1] proposed scheduling algorithm which is cost effecting and SLA-aware. They employed private and public schedulers for ensuring that tasks are scheduled in either in private or public cloud based on resource availability to reduce cost and meet SLAs. Yadaiah and Rao [2] on the other hand studied locality, load and prediction to have multiple heuristic functions to meet scheduling objectives. Yadaiah and Rao [3] also proposed a scheduling algorithm that employs optimal VM migration to achieve SLA-aware scheduling. Yadaiah and Rao [4] proposed a green computing concept by considering slack time to manipulate CPU frequency to reduce energy consumption besides meeting SLAs. Gabi *et al*. [5] proposed an algorithm based on orthogonal taguchi-based cat that makes use of Swarm Optimization (SO). They achieved better system utilization. However, they desired to continue to have multi-objective optimization in future.

Abazari *et al*. [6] focused on workflow scheduling in cloud by defining a heuristic algorithm. They implemented an algorithm using WorkflowSim with security as well. Han *et al*. [7] proposed a Differential Evolution Algorithm (DEA) along with many optimization policies for task scheduling. They called it bio-objective task scheduling, which is a meta-heuristic based solution. They intended to improve it by considering dependent tasks. Jacob and Pradeep [8] proposed a Cuckoo PSO for multi-objective task scheduling with optimizations. Madni *et al*. [13] also investigated Cuckoo search optimization for resource allocation and effective scheduling with multi-objective functions. They

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

could minimize makespan and deadline violation rate besides making it cost effective. Sobhanayak *et al*. [9] proposed a task scheduling algorithm based on bacteria foraging algorithm that considers multiple objective functions and meet SLAs. They intended to improve it with SOA based approach. Reddy *et al*. [10] used modified ACO for task scheduling implementation. It was a multi-objective task scheduling. While scheduling they intended to improve the characterization of behaviours of VMs with resource allocation.

Zuo L, Shu L, Dong S, Zhu C, Hara T [11] proposed multi-objective scheduling on monolithic cloud environment by considering resource cost the algorithm. They used improved ant colony optimization technique intended to improve performance of the system. Midya *et al*. [12] focused on multi-objective optimization of resource allocation and task scheduling in cloud architecture associated with vehicular network. Naik *et al*. [14] on the other hand focused on selection of VMs for task scheduling with multiple objectives. They used different kinds of optimizations to achieve improvements. Huang *et al*. [15] proposed a PSO based optimization algorithm for task scheduling. By dynamically changing mutation rate they could optimize task allocation. Liu *et al*. [16] proposed a scheduling algorithm for Mobile Cloud Computing (MCC) environments. They used an adaptive weight adjustment approach to meet deadlines. Abdullahi *et al*. [17] employed chaotic optimization strategy for efficient symbiotic organisms search algorithm to overcome scheduling problems in virtual environments. They could improve QoS with optimal scheduling.

Sreenu and Malempati [18] investigated on optimizations in task scheduling. They employed an enhanced fractional grey wolf optimizer for task scheduling with multiple objectives. They could improve efficiency in reaching deadlines and reducing cost and energy consumption. Manikandan and Pravin [19] proposed a hybrid task scheduling with many objective functions. They analysed profits for assigning specific VM as part of scheduling. Xiao and Li [20] proposed a chemical reaction based optimization algorithm for task scheduling by using Directed Acyclic Graph (DAG). From the literature, it is understood that there are many approaches that focused on multi-objective scheduling. However, it is learnt that there is need for an integrated approach that considers hybrid cloud in order to see optimal scheduling of tasks keeping SLA (deadline), cost and energy conservation in mind. Such integrated algorithm is proposed in this paper.

III. PROPOSED INTEGRATED FRAMEWORK FOR TASK SCHEDULING

In cloud environment Virtual Machine (VM) resources, running on geographically distributed data centres, are the primary resources which caters the service requests made by cloud users. So in order to improve the cloud management system performance need of an effective task scheduling algorithms are very much essential. This section provides an integrated approach in task scheduling, which considers SLA (deadline), cost effectiveness and energy efficiency. The framework considers scheduling of both local (private) and public cloud infrastructure based on the availability of resources. It also considers usage of spot instances along with regular instances while scheduling of public cloud resources to make the algorithm a cost effective one. An algorithm named SLA-aware, Cost-effective and Energy-efficient Task Scheduling (SCE-TS) is defined to realize the integrated scheduling algorithm. The below figure shows architecture of the cloud environment on which the proposed algorithm runs.

A. The Framework

The proposed framework is outlined in Figure 1. The framework outlines the overall modus operandi involved in integrated scheduling. More details of the framework are provided in Section 3.2 and 3.3.The given user jobs with specified deadline are scheduled efficiently considering both private and public cloud resources.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

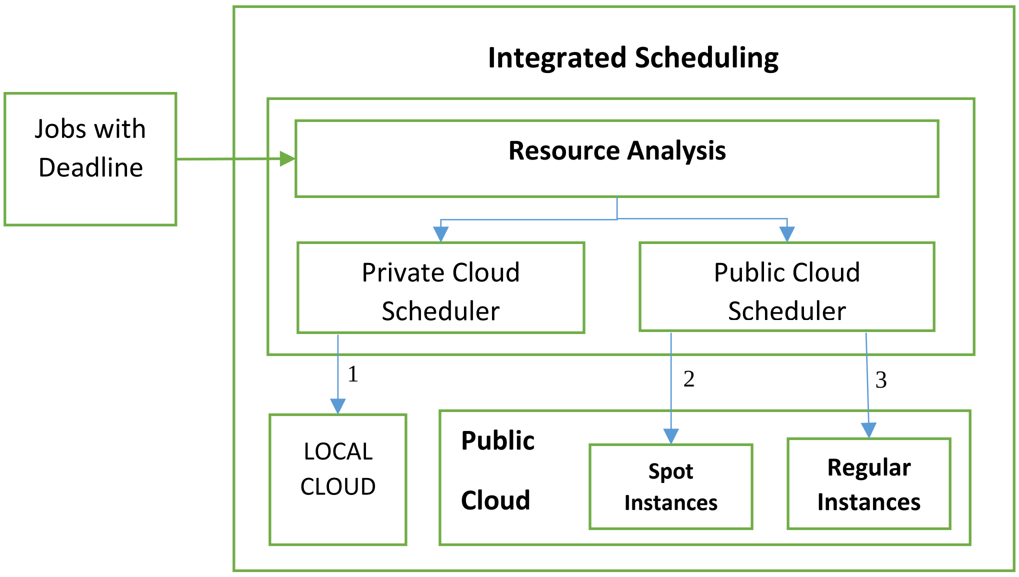


Fig 1: Flow of integrated approach for scheduling

As per the framework, jobs from multiple users along with their deadlines are submitted for scheduling. Upon receiving the jobs, the algorithm finds the availability of local (private) resources that can meet deadline. If localresources are available, algorithm invokes Private Scheduler for scheduling. When there exist multiple resources which can satisfy SLA, scheduler analyses further to find the best resource among them for scheduling. Towards this end, it considers all resources and starts an iterative process for every resource. Table 1 show the notations used in the proposed algorithm.

Table 1: Notations used in the algorithms

Notation	Description
J	Set of jobs $J_1, J_2, \dots, J_n$ arrived at any time interval
n(J_i)	Number of tasks waiting in a given resource queue
$TWT(J_i)$	task waiting timeof J_i
$TET(J_i)$	task execution time of J_i
$TCT(J_i)$	task completion time of J_i

for a resource queue R, the scheduler computes the average waiting time of a task TWT (J) using the equation (1).

$$TWT(J_i) = \frac{\sum_{i=1}^{n} StartTime(J_i) - SubmitTime(J_i)}{n(J_i)} \qquad (1)$$

Where n(J_i) represents the total number of tasks waiting in the resource queue of R_i. The actual execution time of the task is computed by using equation (2).

$$TET(J_i) = CompletionTime(J_i) - StartTime(J_i) \qquad (2)$$

Finally the Completion time of the task is computed by using equation (3).

$$TCT(J_i) = TWT(J_i) + TET(J_i) \qquad (3)$$

The expected completion time of a jog/application is defined as sum of the task completion time and provisioning time of the VMs which is shown equation (4).Here $PROV\_TIME(VM)$ represents the provisioning time of the virtual machine.

$$ECT_{TASK} = TCT(J) + PROV\_TIME(VM) \qquad (4)$$

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

If the $ECT_{Task} \leq D_{task}$ then the task is scheduled in the private cloud. Most of the state of the art scheduling algorithms rejected the tasks which cannot meet their deadline. Whereas the proposed algorithm invokes public cloud scheduler for scheduling of such tasks. It uses the locality, load and prediction functional values for identification of feasible resources in public cloud environment. The locality function is defined in equation (5).

$$LO(R_i) = Min_{R_i}\left(\frac{Task\_Read\_From\_User + Task\_Written\_To\_VM}{Transfer\_Duration(J_i)}\right) \quad (5)$$

Here, $Task\_Read\_From\_User$ represents the time consumed to read the task J_i from user, $Task\_Written\_To\_VM$ represents the time taken to write the task J_i to the VM, and $Transfer\_Duration(J_i)$ gives the task transfer time from user environment to virtual machine environment. The normalized value of the locality values is denoted as $0 \leq LO(R_i) \leq 1$. Similarly, the load value of a resource is computed by estimating the expected average task completion time for all waiting tasks in the resource queue is denoted in Equation (6). The normalized load value is in the form of shown as $0 \leq L(R_i) \leq 1$

$$L(R_i) = \frac{\sum_{i=1}^{n} TCT(J_i)}{n} \quad 6)$$

The prediction function can predict the data-center resource requirements in future as a probability distribution. The memory usage of the resource Ri is predicted over a time series $T = \{t_1, t_2, \ldots, t_k\}$ is denoted in Equation (7).

$$P(R_i) = \{P_{R_i}(t_1), P_{R_i}(t_2), \ldots, P_{R_i}(t_k)\} \quad (7)$$

The average value of predicted memory consumption in time domain is specified as regression model as in Equation (8).

$$P_{R_i}(t_i + 1) = \Phi_1 \times P_{R_i}(t_i) + \Phi_2 \times P_{R_i}(t_i) + \delta_{t_i+1} \quad (8)$$

Here, Φ_1 and Φ_2 denotes the error residuals. Normalization of $P(R_i)$ is denoted as $0 \leq P(R_i) \leq 1$. Equation (9) computes the task transfer time of any task J_i .

$$TTT(J_i) = \sum_{l=1}^{p}\left(latency_l + \frac{J_i^{Size}}{bandwidth_l}\right) \quad (9)$$

Finally the utility values of resources are estimated using the equation (10).

$$\hat{U}(R_i) = LO(R_i) + L(R_i) + P(R_i) \quad (10).$$

The VM which is having the least utility value is selected for scheduling in the public cloud environment.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

B. Flow Chart of Scheduling Algorithm in Hybrid Cloud

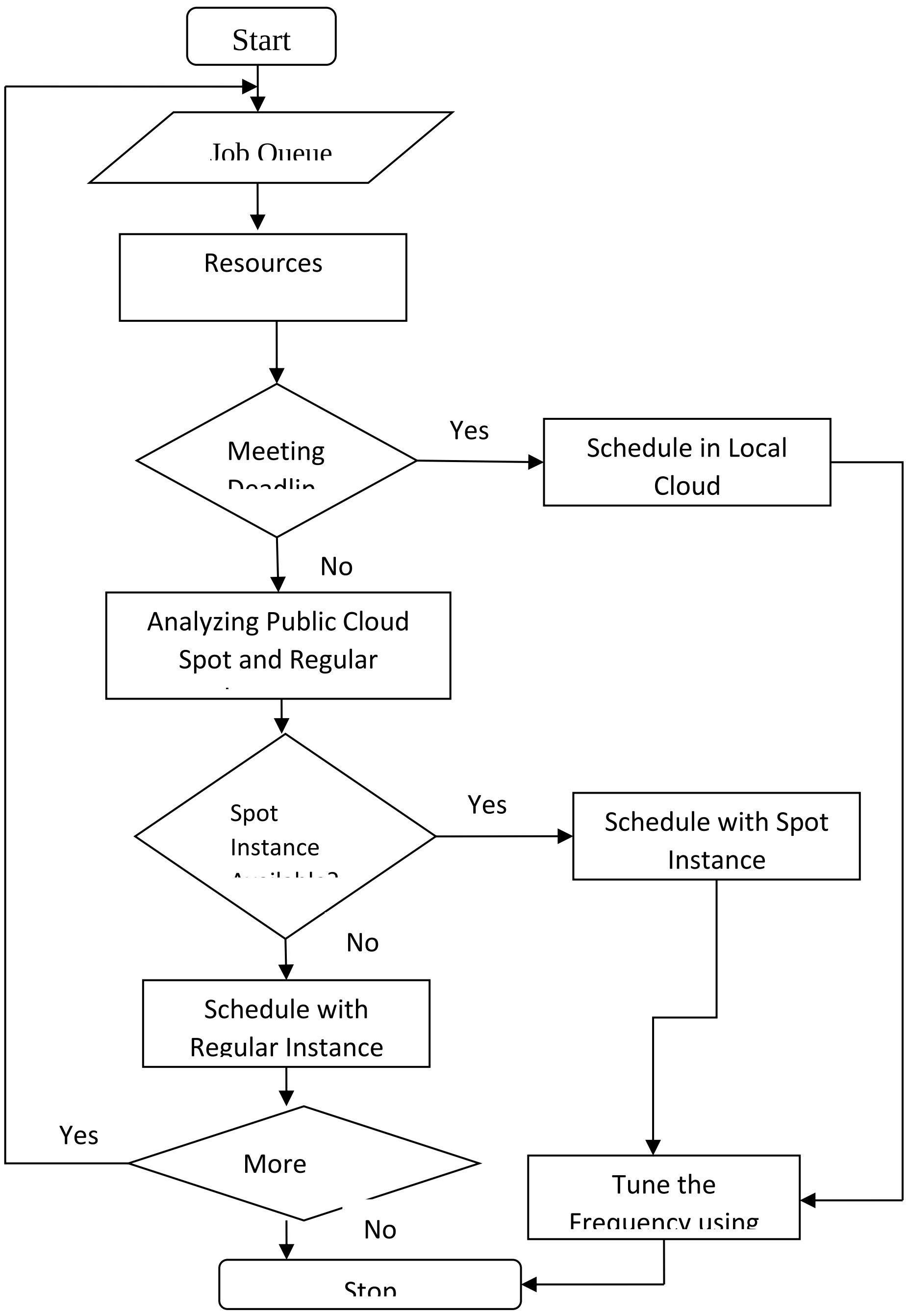


Fig2: Shows the flow of activities as part of scheduling

The proposed methodology for an integrated scheduling algorithm is graphically shown in Figure 2.The flowchart provides outline of the functioning of the proposed scheduling algorithm. The jobs provided by consumers are kept in a job queue. Then an iterative process begins to ensure job scheduling with multiple objective functions. First, it analyses local resources. If the local resources are able to meet deadline of given tasks, they are scheduled in the on premise infrastructure. If there is no possibility to schedule in local infrastructure, jobs that cannot be scheduled here, are transferred

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

to public cloud. Prior to scheduling in public cloud, various factors like cost and energy along with SLA are considered. Spot instances bidding by scheduler has resulted in minimization of cost. Tasks are scheduled to either spot instances or regular instances based on the cost function and availability of spot instances. While scheduling energy efficiency of underlying DVFS enabled machines with appropriate frequency tuning is considered. This process is continued until all jobs are scheduled.

C. An Integrated Task Scheduling Algorithm

An integrated scheduling algorithm is considered that considers different objective functions in hybrid cloud environment to ensure that the task scheduling is efficient and improves the state of the art. The individual approaches followed in [1], [2] and [3] have certain limitations as they are not holistic or integrated. The proposed algorithm SLA-aware, Cost-effective and Energy-efficient Task Scheduling (SCE-TS) overcomes the problem and provides task scheduling with efficiency. It simultaneously considers usage of hybrid cloud resources in cost effective manner with energy efficiency besides meeting SLAs.

```
Algorithm:SLA-aware, Cost-effective and Energy-efficient Task Scheduling (SCE-TS)
Inputs:
  Set of  all tasks sorted on Earliest Dead-line First(EDF) T={$t_1$, $t_2$, …, $t_k$}
  Set of all local resources $VM_{pri}$={$vm_1$, $vm_2$,  …, $vm_m$}
  Set of all public resources $VM_{pub}$={$vm_1$, $vm_2$, …, $vm_n$}

Output: SLA (deadline)-Aware Scheduling, with optimal costand energy.

    1.  Start
    2.  Initialize vector $VM_{available}$ for private resources
    3.  Initialize rejected tasks vector
    Get available resources from private cloud which may satisfy SLA (deadline)
    4.  For each task t in T
    5.      For each resource  r in $VM_{pri}$
    6.   runtime = ExecuteRuntimeEstimator(t,r)
    7.   IF runtime<=deadline of t Then
    8.             Add r,  to $VM_{available}$
    9.  End If
    10.   End For
    11.   IF $VM_{available}$ is not empty Then
    12.  ScheduleWithPrivateCloudScheduler(t, deadline(t), $VM_{available}$)
    13.   Else
    14. ScheduleWithPublicCloudScheduler(t, deadline(t))
    15. End For
    16. End
```

Algorithm 1: SLA-aware, Cost-effective and Energy-efficient Task Scheduling.

As presented in Algorithm 1, the proposed integrated algorithm followed a comprehensive approach in task scheduling. It takes set of tasks, set of local VM resources and set of public VM resources as inputs. After scheduling activity, it results in efficient scheduling that meets SLA (deadline) besides reducing cost by exploring spot instances and private resources and improves

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

energy efficiency by considering the time gap between task actual completion time and estimated completion time. It is achieved by influencing CPU frequency.

When CPU frequency is reduced, energy efficiency is increased and at the same time, it considers the time gap aforementioned to see that the slow processing will not cause violation of deadline. Steps 4 to10 follow an iterative approach to know runtime estimation of given task for given resource. If there are available resources in the private cloud, scheduling is made in the local infrastructure. If not scheduling is made with public cloud. In public cloud spot instances are exploited for cost reduction and the time gap between task estimated completion time and actual completion time to ensure energy efficient task scheduling. The algorithm 1 invokes the following two algorithms appropriately based on the availability of local resources to meet deadline.

```
Algorithm: PrivateCloudScheduler(task t, deadline, $VM_{available}$)
1.Start
2. Initialize resources vector V
3.For each resource r in $VM_{available}$
4. Get the resource queue of r
5.Estimate TWT of t using Eq. (1)
6.Estimate the TET of t using Eq. (2)
7.Compute TCT of t using Eq. (3)
8.Find expected completion time(ECT) using Eq.(4)
9. Add r and its ECT to V
10.End For
11. Assign t to r with minimum ECT
12.Tune the frequency of r for energy efficiency using ConservativeGoverner (as per procedure in Section 4)
13.End.
```

Algorithm 2: Private cloud scheduler

This scheduler as shown in Algorithm 2 estimates task waiting time, task execution time and tack completion time and considers energy as well prior to scheduling tasks in thhe available local resources.

```
Algorithm: PublicCloudScheduler(task t, deadline)
1.Start
2.Initialize resource vector $VM_{availlable}$
3.   For each resource r in $VM_{pub}$
4.        runtime = ExecuteRuntimeEstimator(t,r)
5.          IF runtime<=deadline of t Then
6.          Add  r to $VM_{available}$
7.         End If
8.    End For
9.For each resource r in $VM_{available}$
10. Estimate Locality function for r using eq(5)
```

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

```
11.Estimate Load  function for r usngine.q(6)
12. Estimate the prediction function for r using e.q(8)
13.Compute the Utility value of the resource r using Eq. (10)
14. End For
15.Assign t to r with minimum Utility value of the resource.
16.Tune the frequency of r for energy efficiency using
 ConservativeGoverner (as per procedure in Section 4)
17. End
```

Algorithm 3: Public cloud scheduler

As presented in Algorithm 3, the public cloud scheduler considers public cloud resources that cloud meet deadline. Besides it considers spot instances to reduce cost and the time gap between task completion time and estimated completion time to influence CPU frequency for energy efficiency.

IV. FREQUENCY TUNING FOR ENERGY AWARE SCHEDULING WITH DVFS

There are many approaches for energy aware scheduling in cloud. They include switching off underutilized hosts, migration mechanisms and DVFS. In this paper DVFS is employed to achieve energy efficient scheduling. This approach can dynamically adjust voltage and frequency of CPU in accordance with load and the SLA (deadline). DVFS can work with five modes of operations. They are known as Performance, PowerSave, UserSpace, Conservative and OnDemand. There is fixed frequency associated with three of these modes. Highest frequency is used by Performance mode. Lowest frequency used by PowerSave mode and UserSpace lets user to determine one of the available frequencies for CPU. The last two modes namely Conservative and OnDemand are dynamic in nature. They help in varying CPU frequency based on load. Each mode has its governor. The Conservative governor has one or two thresholds to be used along with the modes. They are known as down_threshold and up_threshold. On the other hand, the OnDemand mode makes use of only one threshold that helps in performance by adjusting frequencies. With conservative mode, CPU frequency is decreased when CPU load is less than the threshold.

A lower frequency of CPU has benefits related to power saving. Low frequency implies weaker voltage and influences reduction of CPU power consumption. However, it also slows down computation capability of CPU. This feature is utilized in this paper for improving energy efficiency with both deadline-aware and energy-aware approach. When deadline of given task is less than the expected task completion time and actual completion time, the Conservative mode is used appropriately to tune CPU frequency in such a way that the task is competed in stipulated time besides reducing power consumption of underlying CPU. This is achieved with the DIFS module introduced in Cloud-Sim.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

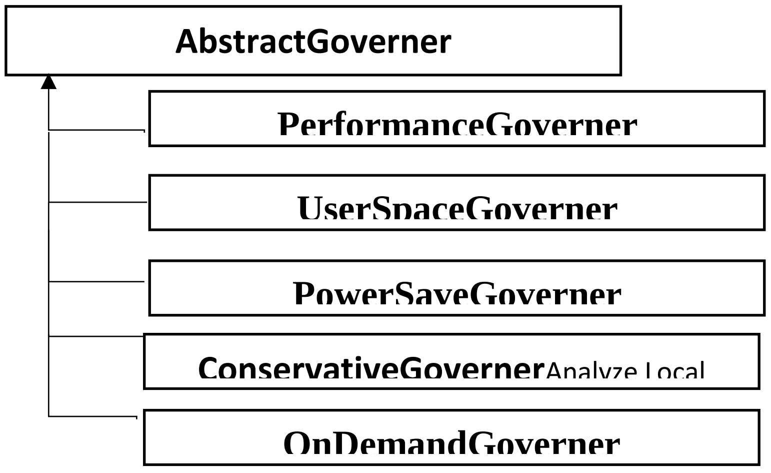


Fig3: Shows DVFS API in CloudSim [1].

V. EPERIMENTAL RESULTS

Experiments are made with Cloud-Sim framework. The results are observed in terms of number of jobs arrived, number of IO related jobs, number of CPU related jobs, how many jobs are able to meet deadline and % of jobs meeting deadline. These results are observed with different runtime estimators that provide estimation on job completion time. Different runtime estimators used are known as Markov, User and Average. In addition to this cost is analysed when spot instances are used along with regular instances and when only regular instances are used. The deadline, cost and energy performance of the integrated algorithm are compared with the state of the art.

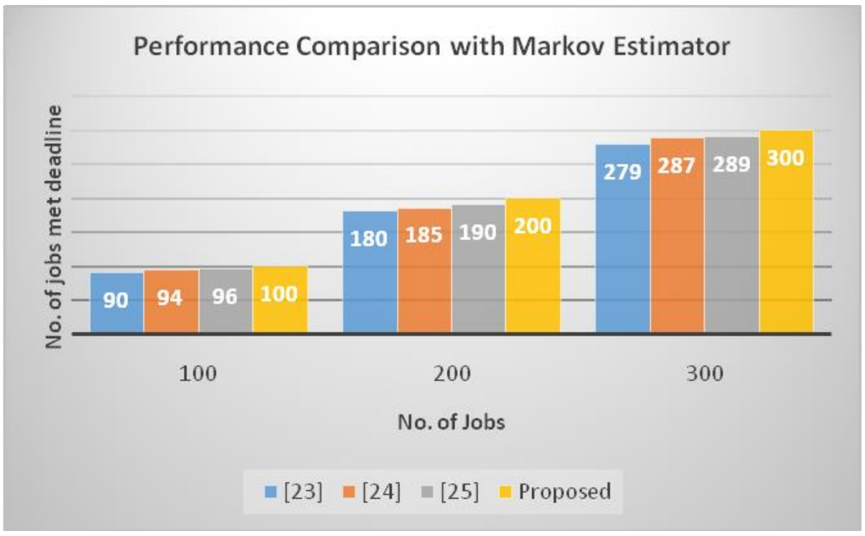


Fig 4: Results of integrated scheduler with Markov estimator

As presented in Figure 4, the performance of integrated scheduling algorithm (proposed) is compared with many state of the art scheduling algorithms. Algorithms found in the literature [23], [24] and [25] along with the proposed integrated algorithm are used in the graph. The number of jobs used for experiments is shown in horizontal axis while the number of jobs met deadline is shown in vertical axis. Three experiments are made with 100 jobs, 200 jobs and 300 jobs respectively. When 100

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

jobs are given as input, the scheduling algorithm in [23] exhibited 90 jobs meeting deadline, [24] 94 jobs, [25] 96 jobs and the proposed exhibited all 100 jobs meeting deadline. When 200 jobs are given as input, the scheduling algorithm in [23] exhibited 180 jobs meeting deadline, [24] 185 jobs, [25] 190 jobs and the proposed exhibited all 200 jobs meeting deadline. When 300 jobs are given as input, the scheduling algorithm in [23] exhibited 279 jobs meeting deadline, [24] 287 jobs, [25] 289 jobs and the proposed exhibited all 300 jobs meeting deadline. Thus the results revealed that the proposed method outperformed the existing schemes. As presented in Figure 5, the performance of integrated scheduling algorithm (proposed) is compared with many state of the art scheduling algorithms. Algorithms found in the literature [23], [24] and [25] along with the proposed integrated algorithm are used in the graph.

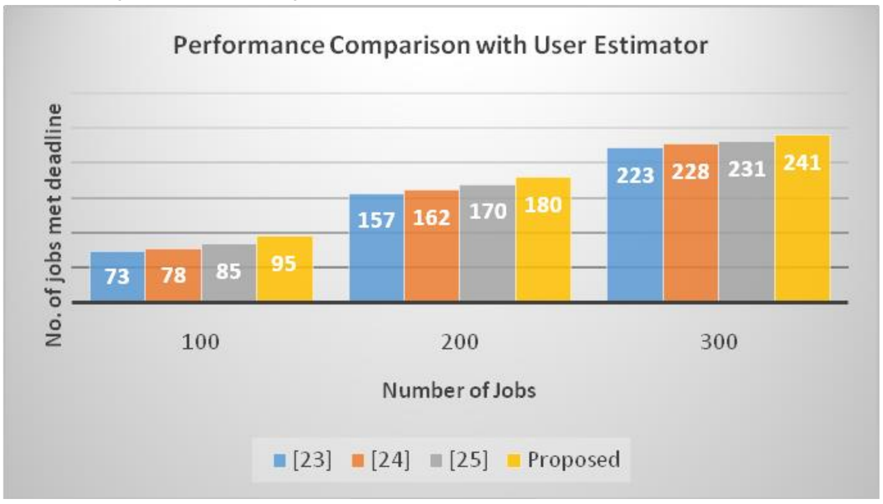


Fig 5: Results of integrated scheduler with User estimator

The number of jobs used for experiments is shown in horizontal axis while the number of jobs met deadline is shown in vertical axis. Three experiments are made with 100 jobs, 200 jobs and 300 jobs respectively. When 100 jobs are given as input, the scheduling algorithm in [23] exhibited 73 jobs meeting deadline, [24] 78 jobs, [25] 75 jobs and the proposed exhibited 95 jobs meeting deadline. When 200 jobs are given as input, the scheduling algorithm in [23] exhibited 157 jobs meeting deadline, [24] 162 jobs, [25] 170 jobs and the proposed exhibited all 180 jobs meeting deadline. When 300 jobs are given as input, the scheduling algorithm in [23] exhibited 223 jobs meeting deadline, [24] 228 jobs, [25] 231 jobs and the proposed exhibited all 241 jobs meeting deadline. Thus the results revealed that the proposed method outperformed the existing schemes.

As presented in Figure 6, the performance of integrated scheduling algorithm (proposed) is compared with many state of the art scheduling algorithms. Algorithms found in the literature [23], [24] and [25] along with the proposed integrated algorithm are used in the graph. The number of jobs used for experiments is shown in horizontal axis while the number of jobs met deadline is shown in vertical axis. Three experiments are made with 100 jobs, 200 jobs and 300 jobs respectively. When 100 jobs are given as input, the scheduling algorithm in [23] exhibited 85 jobs meeting deadline, [24] 89 jobs, [25] 91 jobs and the proposed exhibited 93 jobs meeting deadline.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

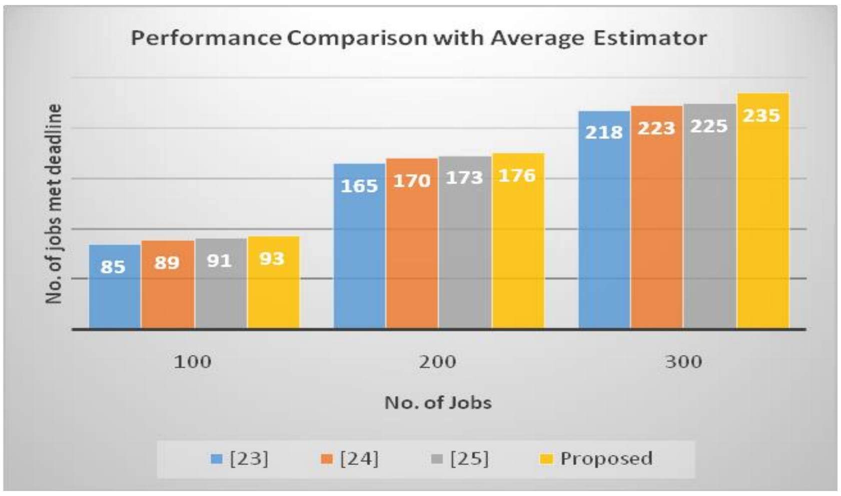


Fig 6: Results of integrated scheduler with Average estimator.

When 200 jobs are given as input, the scheduling algorithm in [23] exhibited 165 jobs meeting deadline, [24] 170 jobs, [25] 173 jobs and the proposed exhibited all 176 jobs meeting deadline. When 300 jobs are given as input, the scheduling algorithm in [23] exhibited 218 jobs meeting deadline, [24] 223 jobs, [25] 225 jobs and the proposed exhibited all 235 jobs meeting deadline. Thus the results revealed that the proposed method outperformed the existing schemes.

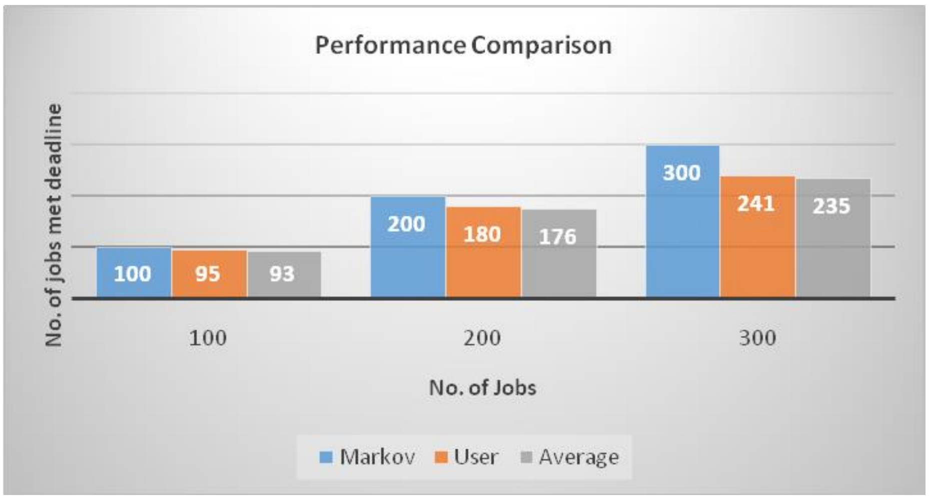


Fig 7: Performance comparison with different estimators

As presented in Figure 7, the summary of results of integrated scheduling algorithm when Markov, User and Average estimators are used are provided. All jobs could be scheduled with Markov estimator as their estimated completion time is less than deadline. There are some jobs rejected in case of User and Average estimator. The results thus reveal the performance improvement of integrated scheduler with Markov estimator.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

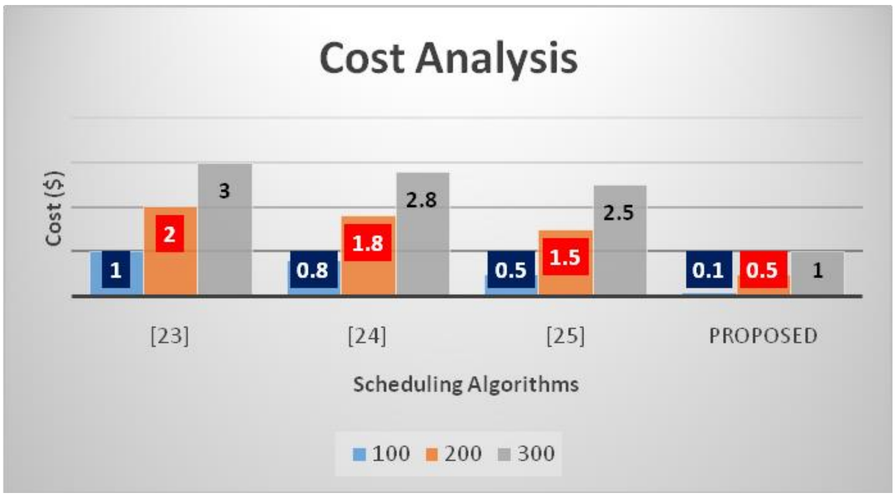
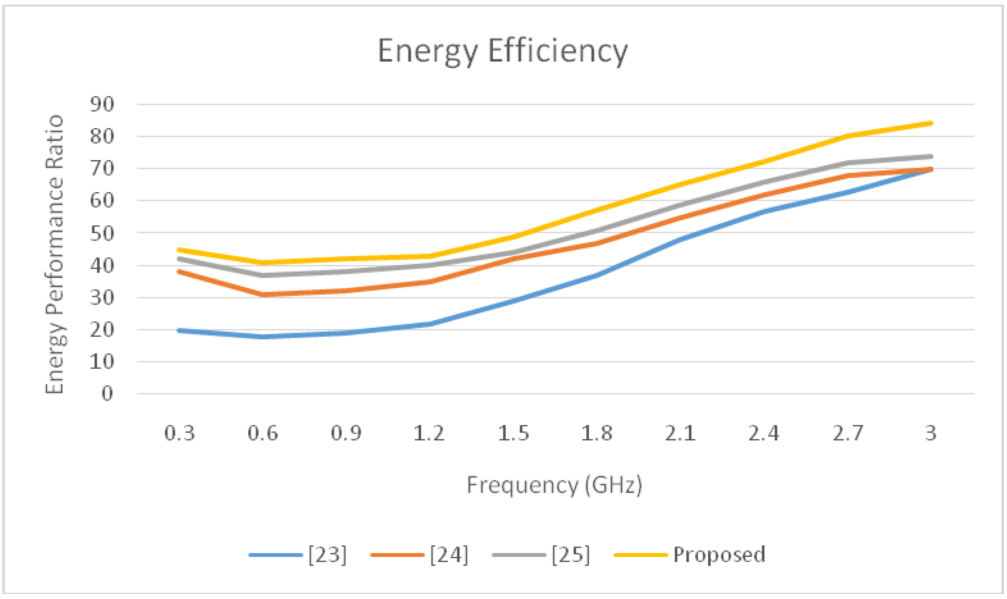


Fig 8: Performance comparison in terms of cost.

As presented in Figure 8, the performance of integrated scheduling algorithm is compared with many state of the art scheduling algorithms in terms of cost. Algorithms found in the literature [23], [24] and [25] along with the proposed integrated algorithm are shown in horizontal axis. The cost of jobs execution is shown in vertical axis.

Three experiments are made with 100 jobs, 200 jobs and 300 jobs respectively. When 100 jobs are given as input, the scheduling algorithm in [23] exhibited cost as 1$, [24] 0.8$, [25] 0.5$ and the proposed algorithm showed cost as 0.1$. When 200 jobs are given as input, the scheduling algorithm in [23] exhibited cost as 2$, [24] 1.8$, [25] 1.5$ and the proposed algorithm showed cost as 0.5$. When 300 jobs are given as input, the scheduling algorithm in [23] exhibited cost as 3$, [24] 2.8$, [25] 2.5$ and the proposed algorithm showed cost as 1$. Thus the results revealed that the proposed method outperformed the existing schemes.

As presented in Figure 9, the performance of integrated scheduling algorithm is compared with many state of the art scheduling algorithms in terms of energy efficiency. Algorithms found in the literature [23], [24] and [25] along with the proposed integrated algorithm are evaluated here. Frequency of VM resource is shown in horizontal axis. The energy performance is shown in vertical axis. The frequency is measured in MHz. Different frequencies from 0.3 to 3 are provided in X axis with 0.3 interval.



ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

Fig 9: Performance comparison in terms of energy efficiency

There is relationship between frequency tuning and the power consumption. Higher the frequency, higher is power consumption. Thus by reducing frequency, power consumption is optimized besides meeting deadline (as frequency tuning is made keeping deadline in mind).

VI. CONCLUSION AND FUTURE WORK

In this paper, we proposed an integrated framework for SLA (deadline) aware multi-objective (cost and energy) task scheduling in cloud computing in hybrid cloud environment. Unlike most of the state of the art scheduling algorithms that used single objective for job scheduling, the proposed framework considers a heuristic combination of multiple objective functions for effective scheduling decisions. The framework is aimed at satisfying SLAs besides ensuring cost effectiveness and energy efficiency of hybrid cloud infrastructure. It is needed in the contemporary era where SMEs look at hybrid cloud as an ideal cloud deployment model. An algorithm known as Multi-Objective Task Scheduling (MOTS) is proposed. It usesdifferent runtime estimators for an effective estimation oftaskcompletiontime that includes provisioning delays. Towards cost effectiveness, the algorithm uses spot instances along with regular instances. For energy efficiency, it considers Dynamic Voltage Frequency Scaling (DVFS) enabled machines with appropriate frequency tuning. Thus the algorithm makes optimal scheduling decisions that meet the aforementioned objectives leading to an equilibrium state which satisfies consumers and also CSP. Experiments are made with a cloud test bed. The empirical results revealed that the proposed integrated framework outperforms the state of the art. In our futurework, the proposed framework will be evaluated further with real cloud environments. Another direction for future work is to define scheduling algorithm for IoT based applications.

REFERENCES

[1] Yadaiah Balagoni &Dr.R.Rajeswara Rao,(2016). A cost-effective SLA-aware scheduling for hybrid cloud environment. 2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). P1-7.

[2] Yadaiah Balagoni &Dr.R.Rajeswara Rao, (2017). Locality-Load-Prediction Aware Multi-Objective Task Scheduling in the Heterogeneous Cloud Environment. Indian Journal of Science and Technology, 10(9), 1–9.

[3] Yadaiah Balagoni and Dr.R.Rajeswara Rao,(2017). SLA-Aware Scheduling Of VMs For Load Balancing In Cloud Using A Three Phase Optimal VM Migration Technique. International Journal of Latest Trends in Engineering and Technology. 10 (2), p217-227.

[4] Yadaiah Balagoni and RamisettyRajeswara Rao. (2018). SAGS: A SLA-Aware Green Scheduling in Heterogeneous Cloud Using Hadoop YARN. *International Journal of Intelligent Engineering and Systems*. 11 (6), p1-10.

[5] Gabi, D., Ismail, A. S., Zainal, A., Zakaria, Z., & Abraham, A. (2016). *Orthogonal Taguchi-based cat algorithm for solving task scheduling problem in cloud computing. Neural Computing and Applications.* P1-19.

[6] Abazari, F., Analoui, M., Takabi, H., & Fu, S. (2018). *MOWS: Multi-Objective Workflow Scheduling in Cloud Computing based on Heuristic Algorithm. Simulation Modelling Practice and Theory.* P1-14.

[7] Han, P., Du, C., & Chen, J. (2018). *A DEA Based Hybrid Algorithm for Bi-objective Task Scheduling in Cloud Computing. 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS).* P1-5.

[8] Prem Jacob, T., & Pradeep, K. (2019). *A Multi-Objective Optimal Task Scheduling in Cloud Environment Using Cuckoo Particle*

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

Swarm Optimization. Wireless Personal Communications. P1-17.

[9] Srichandan, S., Ashok Kumar, T., &Bibhudatta, S. (2018). *Task scheduling for cloud computing using multi-objective hybrid bacteria foraging algorithm. Future Computing and Informatics Journal.* P1-30.

[10] Gogi Reddy Narendrababu Reddy and SingamsettyPhanikumar. (2018). Multi Objective Task Scheduling Using Modified Ant Colony Optimization in Cloud Computing. *International Journal of Intelligent Engineering and Systems*. 13 (3), p1-9.

[11] Zuo L, Shu L, Dong S, Zhu C, Hara T (2015) A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. IEEE Access 3:2687–2699

[12] Midya, S., Roy, A., Majumder, K., &Phadikar, S. (2018). *Multi-objective optimization technique for resource allocation and task scheduling in vehicular cloud architecture: A hybrid adaptive nature inspired approach. Journal of Network and Computer Applications, 103, 58–84.*

[13] Madni, S. H. H., Latiff, M. S. A., Ali, J., &Abdulhamid, S. M. (2018). *Multi-objective-Oriented Cuckoo Search Optimization-Based Resource Scheduling Algorithm for Clouds. Arabian Journal for Science and Engineering.* P1-18.

[14] Naik, K., Meera Gandhi, G., &Patil, S. H. (2018). *Multiobjective Virtual Machine Selection for Task Scheduling in Cloud Computing. Advances in Intelligent Systems and Computing, 319–331.*

[15] Huang, C.-L., Jiang, Y.-Z., Yin, Y., Yeh, W.-C., Chung, V. Y. Y., & Lai, C.-M. (2018). *Multi Objective Scheduling in Cloud Computing Using MOSSO. 2018 IEEE Congress on Evolutionary Computation (CEC).* P1-8.

[16] Liu, L., Fan, Q., &Buyya, R. (2018). *A deadline-constrained multi-objective task scheduling algorithm in Mobile Cloud environments. IEEE Access, 1–20.*

[17] Abdullahi, M., Ngadi, M. A., Dishing, S. I., Abdulhamid, S. M., & Ahmad, B. I. (2019). *An efficient symbiotic organisms search algorithm with chaotic optimization strategy for multi-objective task scheduling problems in cloud computing environment. Journal of Network and Computer Applications.* P1-55.

[18] Sreenu, K., & Malempati, S. (2018). *MFGMTS: Epsilon Constraint-Based Modified Fractional Grey Wolf Optimizer for Multi-Objective Task Scheduling in Cloud Computing. IETE Journal of Research, 1–15.*

[19] Manikandan, N., & Pravin, A. (2019). *LGSA: Hybrid Task Scheduling in Multi Objective Functionality in Cloud Computing Environment. 3D Research, 10(2).* P1-16.

[20] Xiao, X., & Li, Z. (2019). *Chemical Reaction Multi-objective Optimization for Cloud Task DAG Scheduling. IEEE Access, 1–8.*

[21] Archana N. Jethava and Megha R. Desai. (2019). Optimizing Multi Objective Based Dynamic Workflow Using ACO and Black Hole Algorithm in Cloud Computing. *IEEE*, p1-4.

[22] Ismayilov, G., &Topcuoglu, H. R. (2019). Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing. Future Generation Computer Systems. P1-16.

[23] Jiang, W. Z., & Sheng, Z. Q. (2012). A New Task Scheduling Algorithm in Hybrid Cloud Environment. 2012 International Conference on Cloud and Service Computing. p45-49.

[24] Javanmardi, S., Shojafar, M., Amendola, D., Cordeschi, N., Liu, H., & Abraham, A. (2014). Hybrid Job Scheduling Algorithm for Cloud Computing Environment. Proceedings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA 2014, p43–52.

[25] [11]A. M. Senthil Kumar and M. Venkatesan. (2019). Multi-Objective Task

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731

Scheduling Using Hybrid Genetic-Ant Colony Optimization Algorithm in Cloud Environment. *springer*, p1-14.

.

ISSN (Print): 2204-0595
ISSN (Online): 2203-1731