Section A-Research paper

Adaptive Workload Scheduling Design with Task Level SLA in Parallel Computational Framework

[1]Jagadevi Bakka,[2]Sanjeev c Lingareddy,

Abstract

In the recent years, many researches have showed a great deal of interest to improve the scheduling of workload in the cloud platforms. On the other hand, to carry out the execution of the scientific workloads in the cloud environment, it consumes much time and is expensive; hence, it is neither time efficient nor cost-efficient. Due to this reason, many research studies have been carried, by which the researchers tend to reduce the processing time and make a cost-efficient method as the users are charged based on the usage. Very few studies have been done to optimize the cost with processing time and energy parameters together in order to meet the Service Level Agreement (SLA) and Quality of Service (QoS) of the workload task. Hence, in this paper, we present an Adaptive Workload Scheduling (AWS) model that ensures the Task Level SLA (TLS) prerequisites in a heterogeneous distributed-computing environment. This AWS-TLS model approach reduces the amount of energy and time needed to execute a given workloads. Cybershake and Inspiral scientific workload has been utilized for the studying proposed AWS-TLS model. When model was compared with the standard workload scheduling approach, our model reduced the consumption of cost, energy and time.

Keywords: Cloud computing, MapReduce, Quality of services, Resource utilization, Service level agreement, Task scheduling, Workloads.

[1]Associate Professor, Dept. of CSE**,** East Point College of Engineering and Technology, Bangalore, India
[2]Prof. & HOD, Dept of CSE, Sri Venketeshwara College Of Engineering, Bangalore, India

1 INTRODUCTION

There are many scientific complex workloads, which are widely used for the scientific and business investigation [1]. Some of these examples include astronomy workloads known as LIGO and Montage, earthquake detection workload known as CyberShake and genome sequencing workloads known as SIPHT and Epigenomics [2]. There are other scientific workloads, which are widely used for various purposes. These complicated scientific workloads are run on a variety of platforms, including Amazon EC2, Pegasus, and MapReduce [3], [4]. For the execution of the sophisticated scientific workloads, cloud platforms offer high quality computing and storage resources, including networks, services, and applications [5]. In the recent time, different scientific fields like the physics, bio-informatics and astronomy are currently using the cloud resources for modelling the different scientific complex workloads to provide better solutions for the realtime problems [6].

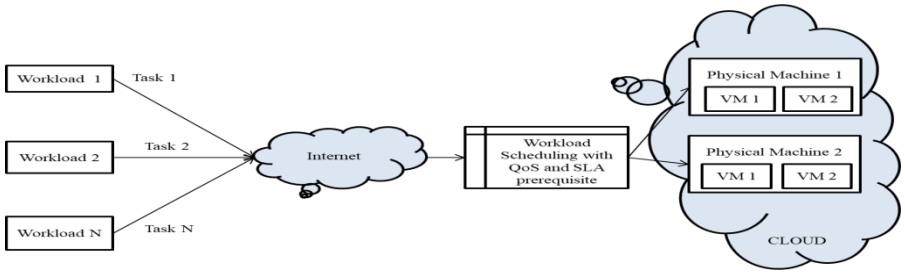

Fig. 1. Basic architecture of workload scheduling using cloud.

The complex workloads can be described using a Directed Acyclic Graph (DAG) in which the vertices of the DAG describe the dependencies and the edges of the DAG describe the various tasks [7-8].In the recent years, there has been an increasing adoption of cloud services for different applications uses. Due to

this, most of the researchers are using the cloud services to schedule the workload [9]. A straightforward architecture for scheduling workloads in a cloud environment is shown in Figure 1. However, creating an effective workload-scheduling model by examining the existing models poses a number of difficulties, such as performing increasingly complicated scientific workloads, which takes more time to complete and is more expensive to execute. When things must be completed by a certain date, it becomes more difficult. Various studies have been done by the various researchers where they have developed various heuristic algorithms to provide an optimal solution for such problems. Further, the heuristics model are not time efficient. Hence, most of the researchers have failed to get an optimal solution, which affects the SLA violation and QoS. Additionally, it is said that workload scheduling is an NP-hard problem [10]. In reality, workload scheduling makes it exceedingly difficult to optimize both time and cost [11]. For example, if a scheduling model aims to cut costs, it will take longer to accomplish a particular task. This is due to the relationship between time and cost. The cost and makespan optimization challenge still exists since many existing models do not take virtual machine selection policy into account while constructing schedules [12]. [13]. This research provides an adaptive workload scheduling method that ensures SLA at the task level to address the aforementioned issue. The AWS model provides energy minimization with performance assurance at the task level.

The significance of AWS-TLS is given below:

1. The AWS-TLS provide effective workload scheduling that minimize energy dissipation and meet task-dead line prerequisite.
2. The AWS-TLS reduces energy dissipation reduce makespan, and cost for executing data-intensive workload in comparison with standard model.
3. The AWS-TLS is very efficient is execution of scientific workflow that is CPU, I/O, and memory intensive in nature.
4. Works well to provide task level SLA for both smaller and larger task.

2 LITTERATURE SURVEY

The state-of-art scheduling of complex-workload is studied in this study [12], [13] to determine its advantages and drawbacks. For heterogeneous computation platforms, [14] focuses on optimizing cost and energy and jointly to create scheduling of workload. Here a min function is used to save energy consumption and satisfy work deadlines, considering that job information is scattered geographically. Considered diverse deadlines and classified by deadline small to large here. Finally, a method of adaptive search is proposed for the selection of an efficient timetable for the execution of the process. According to [15], increased computing costs for service supply are directly related to an increase in energy consumption. The most important metrics in service supply are reliability and timeliness. In order to reduce the amount of energy required to run workloads, they came up with a scheduling architecture namely energy-min scheduling (EMS) that met both the reliability and the timeliness requirements.

When it comes to dealing with cloud computing's uncertain resource availability, a tradeoff was modelled in [16]. Multi-objective optimization models of cost and make-span are combined here. Various levels of interruption are explored, and the results reveal that existing models perform better [17]. An evolutionary computing model, NPSO (Nested-Particle-Swarm-Optimization), as well as a faster version, FNPSO (FastNested-Particle Swarm Optimization), were designed in [18] with the goal of improving the execution of complex workloads. In comparison to the NPSO model, the FNPSO is greatly reduced. Heterogeneous earliest finish time (HEFT) and Q-Learning (QL) were coupled to create an efficient scheduling system called QL-HEFT in [19]. In order to speed up computations, the QL-HEFT has been designed. QL's reward function is modified based on HEFT's rising ranks. This helps the Q-Learning system learn more efficiently. To begin, the QL determines the most efficient task sequence and then selects the best machine for the job based on the earliest completion time. Contention awareness was taken into consideration in [20] when designing a scheduling system for workload execution.

In [21], they have proposed a workload scheduling approach based on an evolutionary computing model for meeting deadlines while minimizing costs, namely DCOH. It was also upgraded with multi-objective parameter optimization under hybrid cloud platform to improve DCOH. Scheduling workload applications to fulfil application deadlines and costs is now possible in [22]. During resource allocation, budget and cost ratio are utilized to correlate budget and timeline constraints. Here, they improved priority selection design for job ordering. Certain decisions are discarded in order to increase success rate (i.e., reliability). According to [23], a cloud-based

scheduling strategy must meet user deadline requirements and service level agreements (SLAs). They used a multi-cloud platform to suit the performance and cost requirements of the stream workload application. Using a multi-cloud platform and a fault-tolerant scheduling design, create [24-25]. In addition, the model ensures that the reliability criterion is met and that the cost is decreased. An analysis of failure and reliability rates was performed using a continuous probability distribution. Once the cost of executing on the multi-cloud platform has been calculated, the next step is to define a fault-tolerant workload scheduling design that ensures reliability, reduces cost, reduces execution time and is cost effective. However, the application's cost constraints could not be met due to a lack of load balancing. The following part proposes a cost and performance-aware scheduling strategy for heterogeneous cloud environments to address the aforementioned challenges.

3 ADAPTIVE WORKLOAD SCHEDULING DESIGN WITH TASK LEVEL SLA IN PARALLEL COMPUTATIONAL FRAMEWORK

In this section, we present our model for scheduling the workload, which provides a better performance by keeping the SLA violations constant at the task-level in a parallel-computational architecture. For monitoring the process, we design and implement an adaptive workload scheduling, also known as AWS-TLS, which takes into account several restrictions including task dependency, priority, parallel computing as well as various parameters like energy consumption and make-span minimization. A scientific complex workload has a various kind of tasks and huge amount of data. Hence, to execute these large data, a significant number of clusters are required for the execution of each task. Further, there are many tasks which are running parallel in the clusters, which needs a well-organized scheduling and efficient use of resources. There are mainly two key elements to solve the problem of workload scheduling, which can be broken down into two separate issues: workload scheduling and optimal-task scheduling. Moreover, the workload scheduling has to be done in a way where different constraints can be fulfilled like task priority, energy constraint, task monitoring, makespan constraint and parallel computation. Hence, by using a multi-level scheduling workload model, these issues can be addressed by executing and scheduling each tasks at each level, as well as monitoring them with time and energy constraints. For the initial stage, let's consider a large number of tasks o running parallelly in each cluster, each having a certain size $\mathbb{P}_1, \mathbb{P}_2, \ldots, \mathbb{P}_o$ and set of constraints with the necessity for executing a task $_1, s_2, \ldots, s_o$. Moreover, to optimize the energy consumption F with respect to the task scheduling on the given processor in a data center is to compute the various power supplies $q_1, q_2, \ldots, q_o$ on a processor n is such a way that the length of scheduling is reduced and the energy consumed should not exceed the energy consumption F. Furthermore, it has been determined that most of the recent works have only focused on the single constraints, which is impractical for real-time models and does not provide a feasible optimal solution. Hence, multiobjective constraint at different levels is considered for the constructing adaptive workload scheduling with task-level SLAs in section B.

3.1 Resource Usage Estimation Metric:

In this given section, we present a system model, which provides an task and power model. In this section, the main concern is to reduce the power consumption for the given computational framework. Thus, we first describe the power consumption using the given Equation (1).

$$\mathrm{q} = \mathrm{g}\mathbb{b}\mathrm{DW2} \quad (1)$$

In Equation (1), q is considered the power consumption and is calculated to an approximate value. g is used describe the clock frequency, describes the load capacity that it can handle and W is used to describe the voltage. Further, $W \propto g^{\delta}$ is used to describe the relationship between the voltage and the frequency of the clock in which δ is considered to be a constant. Moreover, the speed consumed for the execution is represented using which is proportional to the frequency of clock g. Assume two scenarios in which $W = cg^{\phi}$ and $T = dg$. In these scenarios let c and d be a constant value. Using this scenario, the power consumption can be evaluated using the given below equations.

$$\mathrm{q} = \mu\mathrm{Tb} \quad (2)$$

$$\mathrm{b} = 2\alpha + 1 \quad (3)$$

$$\mu = \mathrm{bc2CD}\,(\mathrm{d}2\alpha + 1) - 1 \quad (4)$$

Equation (2) has been generated by using the equation (1). The evaluation of b and μ is calculated by utilizing the Equation (3) and Equation (4) respectively.

In a Directed Acyclic Graph, each tasks are first designed and any of the parallel task denoted by o can be represented using $H = (W, F)$. In $H = (W, F)$,

W is used to indicate the tasks and W = {1,2,3, … . , o} and F represents the priority of the task. Further, if any two tasks, consider task 1 as j and task 2 as k, then the relationship between the two tasks j and k can be represented using the $arc(j, k)$ which explains the task k cannot be executed until and unless the task j has been completed and finished. Furthermore, $\mathbb{P}_j$ represents the resources required by the task j for the execution of the task, and is calculated using the given equation X_j= $\mathbb{P}_j s_j$. For each task processor is considered. For example, if we want to execute a task which has been given an n processor from the data center, then it executes the task and also provides parallel computation. Further, to evaluate the parallel computation, consider an application v which is running parallel and has some tasks H_1,H_2,….., H_v for the given processor n. In this application, the multiple tasks are also designed under the single task using the multi-level method. In addition, the power supplied to the task j of the application can be represented using q_j and can be evaluated using the equation T_j= $q_j^{1/b}$ which is also used to determine the execution speed. Further, q_j= T_j^b and time required for the execution is evaluated using the Equation (5). During the execution of the tasks the speed of the processor remains the same. The energy required for the execution of the task is given using the Equation (6).

$$\text{uj} = \left({}^{\text{sj}}/_{\text{Qi}}\right)\text{b} - 1 \tag{5}$$

$$\text{fj} = \text{TJB} - 1\text{Xj} \tag{6}$$

In Equation (6), X_j represents the amount of work that has to be executed by the task j. Furthermore, it should be taken into consideration that in a real processor, the clock frequency and the execution speed can only take finite values, as the X_j is the executed work, which has been executed earlier.

3.2 SLA constraint modelling:

We determine the lower bound in this section. Consider the parameter X, which may be represented by the equation, as the quantity of work completed for a particular parallel task o. (7)

$$\begin{aligned} \text{X} = \text{x1} + \text{x2} + \cdots + \text{xo} \\ = \mathbb{P}1\,\text{s1} + \mathbb{P}1\,\text{s2} \\ + \cdots + \mathbb{P}\text{oso} \end{aligned} \tag{7}$$

F' and U' are used to represent the lowest energy and ideal length needed for an optimal schedule, respectively. In a multi-level workload-scheduling paradigm, the lower bound that helps to shorten the makespan can be evaluated by taking into account all of these characteristics.

$$\begin{aligned} nX\ b1/(b-1) \\ U' \geq (()) \\ \tilde{F} \quad n \end{aligned} \tag{8}$$

According to equation (8), the lower bound shortens the makespan and the supplied Equation (9) below shows the lower bound to cut back on energy consumption.

$$\begin{aligned} X\ b \\ F' \geq n\,()\,(\bar{U}b-1)-1 \\ n \end{aligned} \tag{9}$$

Both the Equation (8) and Equation (9) can be used for any dependent, independent and parallel task.

3.3 Adaptive Workload Scheduling Method:

An adaptive workload scheduling approach has been shown in this section. The AWS method starts by taking into considers $_c$ as the amount of time needed to complete job O_{kd}. The tasks that are not scheduled in this part are examined and monitored to determine whether they are available for execution, that is, to determine whether the resources needed to complete the work are readily available. As a result, this includes more tasks than the earlier approaches [15], [24]. The task's scheduling also aids in increasing the processor's efficiency. The equation, which states that when the job O=1, then, can be used to represent the task scheduling

$$\begin{aligned} N \\ Q*(O,N) = \sum nsn \\ n = 1 \end{aligned} \tag{10}$$

Suppose the task O is more than 1 then it is represented using the Equation (11)

$$\begin{aligned} Q*(O,N) = \sum Nn \\ = 1\,sn(n + Q*(O \\ - 1,N - n)) \\ + (\sum n \\ > N\,sn)Q*(O \\ - 1,N) \end{aligned} \tag{11}$$

If the size of the first task is large, then the resources, which are available, are used and all the other tasks are terminated. After this all the other remaining task $o-1$ are scheduled. Further average scheduling after the above situations is $Q^*(O-1, N)$

3.4 Adaptive Workload Scheduling with Task Level SLA Method:

In this given section, we develop and design a scheduling mechanism for the task level SLA in distributed computing platform, which monitors and reduces the consumption of energy and makespan

during parallel computation. Assume the parallel task as which have priority constraint. Further, consider two tasks $j$and k, also consider a cardinality order $\mathcal{C}$which is denoted by using the DAG for the parallel task o. Then, we can say that there is a relationship between the two tasks j and k, which can be represented as $j\mathcal{C}k$ such that the task k can only start the execution only when the task j has been completed. A task-level scheduling model contains different stages with w as the number of stages; moreover, the task that have no priority constraint like the first task takes the stage 1. In addition, a particular task j is considered at Stage m. Suppose, if the number of nodes start from the staring task j, then it is represented as $1 \leq m \leq w$. This method schedules the various task using various stages, i.e., Stage 1, Stage 2, Stage 3, ….,. Stage w. Moreover, the Stage m + 1 never allows the execution of another task until the task m at the Stage m has been completed. For all the stages, same procedure is used. Also, the residual energy $\tilde{F}$ is then allocated to the Stage v, in which the Stage 1 consumes F_m and$F_1 + F_2 + \cdots + F_w = \tilde{F}$. Further, the processor$n_1$ is considered for the first task in the Stage 1 and the execution of the task is represented using the $s_{m,1}$,$s_{m,2}$, …….,$s_{m,n}$. Using this the total amount of work can be evaluated by the given Equation (12)

$$X1 = \pi m,1sm,1 + \pi m,2sm,2 + \cdots + \pi m,omss,om \quad (12)$$

When scheduling workloads, it's important to keep in mind a variety of factors, such as the order in which tasks are completed and the amount of energy and time each task consumes.

3.5 F.Adaptive Workload Scheduling with Task Level SLA Assurance Algorithm:

This section presents the algorithm of adaptive workload scheduling assuring SLA at task level. The AWS-TLS algorithm is described in Algorithm 1.

Algorithm 1. The algorithm of adaptive workload scheduling assuring SLA at task level.
Step 1. Design a task and power model
Step 2. Compute the given lower performance constraint limits
Step 3. Mechanism for adaptive task scheduling in which tasks that have not been scheduled are tested to see if the necessary resources are available for their execution.
Step 4. Design the tasks in Directed Acyclic Graph. The number of levels in the DAG is w, where w is the number of levels with varying energy and makespan SLA constraint in the DAG.
Step 5. Non-urgent tasks fall under the category of level 1.
Step 6. AWS-TLS organizes tasks into levels 1, 2, 3, 4, etc., and keeps track of all of them.
Step 7. Furthermore, unless the task in level m is finished, m + 1 level cannot be executed. The monitoring process is the same for each level m.
Step 8. As task are independent are different level as a result, they are scheduled using adaptive task scheduling technique.
The AWS-TLS model is evaluated by using the data-intensive scientific workloads.

4 RESULT AND DISCUSSION

This section examines the effectiveness of the proposed AWS-TLS over the current EMS-RTPW [15] approach in terms of makespan, energy efficiency, and cost effectiveness. Java programming is used to implement AWS-TLS and EMS utilising cloudsim [2], [3]. The CPU and I/O consuming character CPU, memory, and cybershake intensive nature Workload-scheduling model validation is done using Inspiral. Metrics like makespan, energy utilisation, and cost effectiveness are utilised to verify workload-scheduling models.

4.1 Makespan performance:

In this section the makespan for completing execution of small to extra-large workload of both Inspiral and Cybershake is studied. In Fig. 2, the makespan attained for executing Inspiral using AWS-TLS and EMS-RTPW considering varied workload is graphically shown. The Inspiral workload task size of small, medium, large, and extra-large is equal to 30, 50, 100, and 1000, respectively. An average makespan efficiency improvement of 83.07% is achieved using AWS-TLS over EMS-RTPW. In Fig. 3, the makespan attained for executing Cybershake using AWS-TLS and EMS-RTPW considering varied workload is graphically shown. The Cybershake workload task size of small, medium, large, and extra-large is equal to 30, 50, 100, and 1000, respectively. An average makespan efficiency improvement of 78.99% is achieved using AWS-TLS over EMS-RTPW.

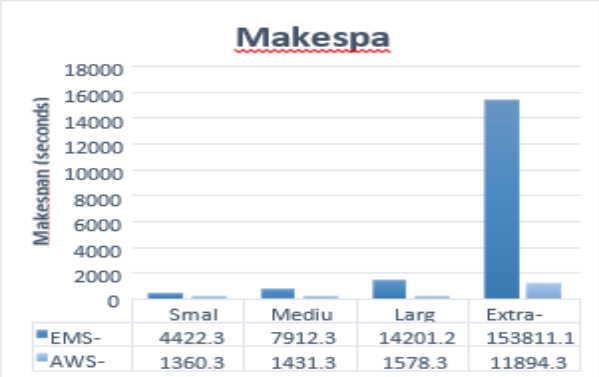

Fig. 2. Makespan efficiency with different Inspiral workload size.

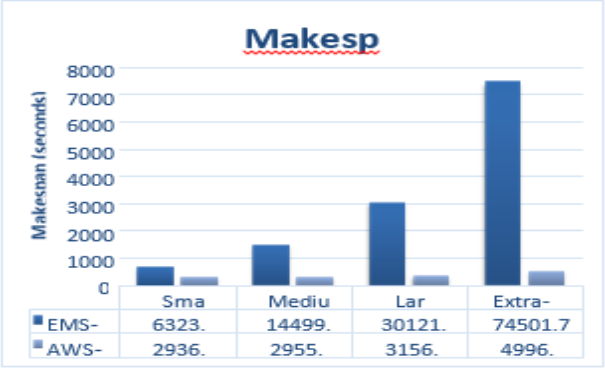

Fig. 3. Makespan efficiency with different Cybershake workload size.

4.2 Energy efficiency:

In this section, the energy efficiency for completing execution of small to extra-large workload of both Inspiral and Cybershake is studied. In Fig. 4, the energy consumed for executing Inspiral using AWS-TLS and EMS-RTPW considering varied workload is graphically shown. The Inspiral workload task size of small, medium, large, and extra-large is equal to 30, 50, 100, and 1000, respectively. An average energy consumption reduction of 44.85% is achieved using AWS-TLS over EMS-RTPW. In Fig. 5, the energy consumed for executing Cybershake using AWS-TLS and EMS-RTPW considering varied workload is graphically shown. The Cybershake workload task size of small, medium, large, and extra-large is equal to 30, 50, 100, and 1000, respectively. An average energy consumption reduction of 24.35% is achieved using AWS-TLS over EMS-RTPW.

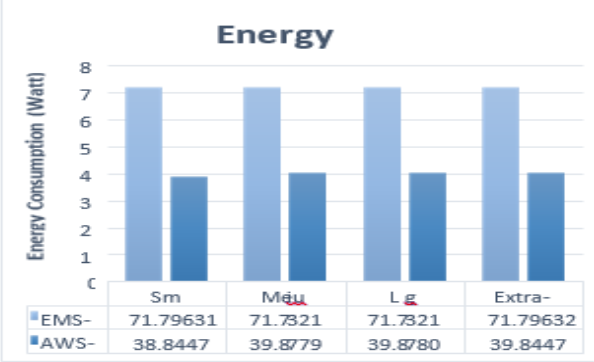

Fig. 4. Energy efficiency with different Inspiral workload size.

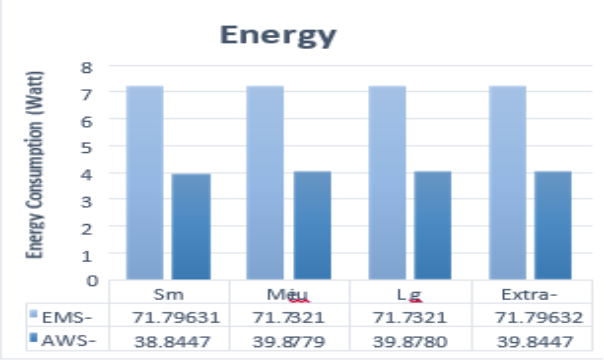

Fig. 5. Energy efficiency with different Cybershake workload size.

4.3 Cost efficiency:

In this section the cost efficiency for completing execution of small to extra-large workload of both Inspiral and Cybershake is studied. In Fig. 6, the cost incurred for executing Inspiral using AWS-TLS and EMS-RTPW considering varied workload is graphically shown. The Inspiral workload task size of small, medium, large, and extra-large is equal to 30, 50, 100, and 1000, respectively. An average cost reduction of 82.88% is achieved using AWS-TLS over EMS-RTPW. In Fig. 7, the cost incurred for executing Cybershake using AWS-TLS and EMS-RTPW considering varied workload is graphically shown. The Cybershake workload task size of small, medium, large, and extra-large is equal to 30, 50, 100, and 1000, respectively. An average cost reduction of 78.68% is achieved using AWS-TLS over EMS-RTPW.

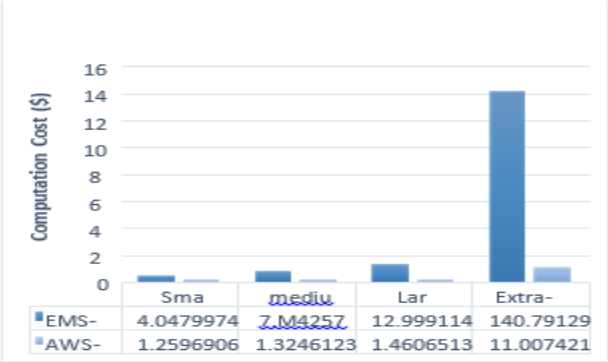

Fig. 6. Cost efficiency with different Inspiral workload size.

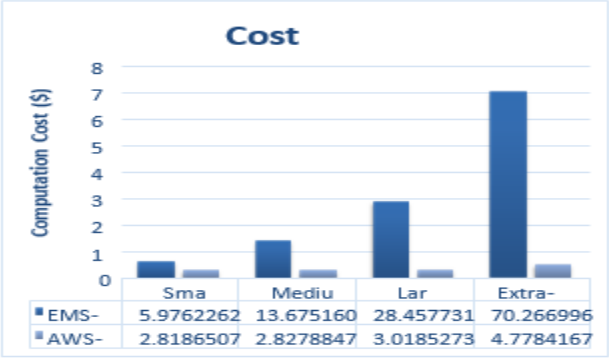

Fig. 7. Cost efficiency with different Cybershake workload size.

5 CONCLUSION

In this research an effective workload scheduling is presented that assures the task level Service Level Agreement (SLA). None of the existing approaches had considered workload scheduling at task level SLA till yet. In this work, an adaptive scheduling technique is established that can decrease the energy consumption as well as maintain the performance on high level, which reduces the execution cost significantly. The AWS- TLS model is very much effective in provisioning CPU, memory as well as I/O intensive execution that leverages for distributed platform for computing such as cloud environment. Further, experiments shows the efficient outcomes of AWS-TLS model with respect to energy efficiency with an improvisation of 44.85% and 25.35% is experienced by AWS-TLS over EMS-RTPW for Inspiral and Cybershake workload, respectively. The makespan for execution of workload is reduced by 83.07% and 78.99% by AWS-TLS over EMS-RTPW for Inspiral and Cybershake workload, respectively. Similarly, cost for execution of workload is reduced by 82.88% and 78.68% by AWS-TLS over EMS-RTPW for Inspiral and Cybershake workload, respectively. In future the proposed scheduling model will be tested with more diverse workload dataset. Alongside would consider leveraging multi-cloud and edge-cloud platform to further reduce cost and delay of execution, respectively.

REFERENCES

[1] Lei Wu, Ran Ding, Zhaohong Jia, Xuejun Li, "Cost-Effective Resource Provisioning for Real-Time Workload in Cloud", Complexity, vol. 2020, Article ID 1467274, 15 pages, 2020. https://doi.org/10.1155/2020/1467274.

[2] L. Chen, X. Li, Y. Guo and R. Ruiz, "Hybrid Resource Provisioning for Cloud Workloads with Malleable and Rigid Tasks," in IEEE Transactions on Cloud Computing, vol. 9, no. 3, pp. 10891102, 1 July-Sept. 2021, doi: 10.1109/TCC.2019.2894836.

[3] J. Wang, X. Li, R. Ruiz, J. Yang and D. Chu, "Energy Utilization Task Scheduling for MapReduce in Heterogeneous Clusters," in IEEE Transactions on Services Computing, vol. 15, no. 2, pp. 931944, 1 March-April 2022, doi: 10.1109/TSC.2020.2966697.

[4] Z. Wen, R. Qasha, Z. Li, R. Ranjan, P. Watson and A. Romanovsky, "Dynamically Partitioning Workload over Federated Clouds for Optimising the Monetary Cost and Handling Run-Time Failures," in IEEE Transactions on Cloud Computing, vol. 8, no. 4, pp. 1093-1107, 1 Oct.-Dec. 2020, doi: 10.1109/TCC.2016.2603477.

[5] Nasr AA, El-Bahnasawy NA, Attiya G, El-Sayed A (2019) Cost-effective algorithm for workload scheduling in cloud computing under deadline constraint. Arab J Sci Eng 44(4):3765–3780.

[6] Zhou X, Zhang G, Sun J, Zhou J, Wei T, Hu S (2019) Minimizing cost and makespan for workload scheduling in cloud using fuzzy dominance sort based heft. Futur Gener Comput Syst 93:278–289.

[7] Rodriguez MA, Buyya R (2017) Budget-driven scheduling of scientific workloads in iaas clouds with fine-grained billing periods. ACM Trans Auton Adapt Syst (TAAS) 12(2):1–22.

[8] Anwar N, Deng H (2018) Elastic scheduling of scientific workloads under deadline constraints in cloud computing environments. Futur Int 10(1):5.

[9] Ismayilov G, Topcuoglu HR (2020) Neural network based multi-objective evolutionary algorithm for dynamic workload scheduling in cloud computing. Futur Gener Comput Syst 102:307–322

[10] Manasrah AM, Ba Ali H (2018) Workload scheduling using hybrid ga-pso algorithm in cloud computing. Wirel Commun Mob Comput 2018.

[11] Yassir S, Mostapha Z, Claude T (2017) Workload scheduling issues and techniques in cloud computing: A systematic literature review. In: International Conference of Cloud Computing Technologies and Applications. Springer. pp 241–263.

[12] Qin Y, Wang H, Yi S, Li X, Zhai L (2020) An energy-aware scheduling algorithm for budgetconstrained scientific workloads based on multi-objective reinforcement learning. J Supercomput 76(1):455–480

[13] Konjaang JK, Xu L (2020) Cost optimised heuristic algorithm (coha) for scientific workload scheduling in iaas cloud environment. In: 2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS). IEEE Computer Society. pp 162–168.

[14] X. Li, W. Yu, R. Ruiz and J. Zhu, "Energy-aware cloud workload applications scheduling with geodistributed data," in IEEE Transactions on Services Computing, doi: 10.1109/TSC.2020.2965106.

[15] B. Hu, Z. Cao and M. Zhou, "Energy-Minimized Scheduling of Real-Time Parallel Workloads on Heterogeneous Distributed Computing Systems," in IEEE Transactions on Services Computing, doi: 10.1109/TSC.2021.3054754.

[16] T. Pham and T. Fahringer, "Evolutionary Multi-objective Workload Scheduling for Volatile Resources in theCloud," in IEEE Transactions on Cloud Computing, doi: 10.1109/TCC.2020.2993250.

[17] H. Li, B. Wang, Y. Yuan, M. Zhou, Y. Fan and Y. Xia, "Scoring and Dynamic Hierarchy-Based NSGA-II for Multiobjective Workload Scheduling in the Cloud," in IEEE Transactions on Automation Science and Engineering, doi: 10.1109/TASE.2021.3054501.

[18] A. Song, W. Chen, X. Luo, Z. Zhan and J. Zhang, "Scheduling Workloads with Composite Tasks: A Nested Particle Swarm Optimization Approach," in IEEE Transactions on Services Computing, doi: 10.1109/TSC.2020.2975774.

[19] Tong, Zhao & Deng, Xiaomei & Chen, Hongjian & Mei, Jing & Liu, Hong. (2020). QL-HEFT: a novel machine learning scheduling scheme base on cloud computing environment. Neural Computing and Applications. 32. 1-18. 10.1007/s00521-019-04118-8.

[20] Q. Wu, M. Zhou and J. Wen, "Endpoint Communication Contention-Aware Cloud Workload Scheduling," in IEEE Transactions on Automation Science and Engineering, doi: 10.1109/TASE.2020.3046673.

[21] Zhou, Junlong & Wang, Tian & Cong, Peijin & Lu, Pingping & Wei, Tongquan & Chen, Mingsong. (2019). Cost and Makespan-Aware Workload Scheduling in Hybrid Clouds. Journal of Systems Architecture. 100. 10.1016/j.sysarc.2019.08.004.

[22] G. Wang, Y. Wang, M. S. Obaidat, C. Lin and H. Guo, "Dynamic Multiworkload Deadline and Budget Constrained Scheduling in Heterogeneous Distributed Systems," in IEEE Systems Journal, doi: 10.1109/JSYST.2021.3087527.

[23] M. Barika, S. Garg, A. Chan and R. Calheiros, "Scheduling Algorithms for Efficient Execution of Stream Workload Applications in Multicloud Environments," in IEEE Transactions on Services Computing, doi: 10.1109/TSC.2019.2963382.

[24] X. Tang, "Reliability-Aware Cost-Efficient Scientific Workloads Scheduling Strategy on MultiCloud Systems," in IEEE Transactions on Cloud Computing, doi: 10.1109/TCC.2021.3057422.

[25] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, ”Characterizing and profiling scientific workloads,” Future Gen. Comput. Syst., vol. 29, no. 3, pp. 682-692, Mar. 2013. https://confluence.pegasus.isi.edu/display/pegasus/Workload Generator. Accessed 18 July 2020.