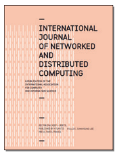
International Journal of Networked and Distributed Computing
Vol. **8**(**3**); June (2020), *pp.* 131–138
DOI: https://doi.org/10.2991/ijndc.k.200515.003; ISSN 2211-7938; eISSN 2211-7946
https://www.atlantis-press.com/journals/ijndc



Research Article

Towards a Task and Resource Aware Task Scheduling in Cloud Computing: An Experimental Comparative Evaluation

Muhammad Ibrahim[1,*], Said Nabi[1], Abdullah Baz[2], Nasir Naveed[1], Hosam Alhakami[3]

[1]*Department of Computer Science, Virtual University of Pakistan, Rawalpindi, Pakistan, Rawalpindi, 44000, Pakistan*
[2]*Department of Computer Engineering, College of Computer and Information Systems, Umm Al-Qura University, Makkah, Saudi Arabia*
[3]*Department of Computer Science, College of Computer and Information Systems, Umm Al-Qura University, Makkah, Saudi Arabia*

ARTICLE INFO

Article History

Received 03 January 2020
Accepted 26 February 2020

Keywords

Cloud computing
resource allocation
task scheduling
scheduling algorithms
load balancing
performance evaluation
load imbalance

ABSTRACT

Cloud computing has been considered as one of the large-scale platforms that support various type of services including compute, storage, compute, and analytic to the users and organizations with high agility, scalability, and resiliency intact. The users of the Cloud are increasing at an enormous rate which also resulted in issues related to handling and scheduling the users' requested workload effectively and efficiently on the available Cloud resources. The aim of the Cloud service providers is to maximize resource utilization and in turn increased revenue generation. In the last few years, Cloud Task scheduling has been considered as an important area of research for the researchers. As different scheduling heuristics are associated with different underlying assumptions; thus, performing a precise comparison cannot be guaranteed. This work empirically compares and provides an insight into the performance of some renown state-of-the-art task scheduling heuristics concerning the Makespan, average resource utilization ratio, Throughput. Those approaches include task-aware, resource-aware, and some hybrid approaches. The experiments were then extended by evaluating the performance using average response time for all the compared approaches. The simulation experiments are conducted by utilizing Heterogeneous Computing Scheduling Problems (HCSP) and Google Cloud Jobs (GOCJ) benchmark datasets using CloudSim a renowned simulation tool for Cloud. Based on the findings of the comparative analysis and results discussion, we have highlighted some important aspects of the underlying approaches and for future work we will propose a task-cum-resource aware task scheduling approach.

1. INTRODUCTION

Cloud computing platform has attracted users to provide them high processing and huge memory-intensive services in a pay as you go manner. In the case of Cloud computing, the computing and storage resources are provisioned to the Cloud users from a pool of configured resources according to the user requirements [1–3].

In most of the cases, the efficient mapping of tasks [4] to the Virtual Machines (VMs) lead to the user satisfaction level, minimizing the task execution time, and improvement in terms of energy consumption. The task Scheduling approaches are classified into either static or dynamic type. In the case of static task scheduling, the tasks to VM mapping are planned and the scheduling is performed according to the planned mapping. However, for the dynamic scheduling approaches, the scheduling primarily relies on the run-time dynamics of the scheduled tasks according to the user requirements thus may require VM migration [5–7]. In the case of the static scheduling, VM migration is avoided and ultimately results in reducing the overhead and task execution time. Moreover, the static task scheduling approaches produce improved turnaround time and optimum Quality of Service (QoS) due to the availability of the computing resources for task execution [1]. With all its benefits, the static scheduling can suffer from inefficient and under-utilization of the Cloud resource utilization due to run-time workload and the computing environment [8]. In the context of Cloud computing, the scheduling is employed at two levels that is: scheduling of VMs on the Physical hosts and the scheduling of tasks on the available VM resources.

To evaluate the performance of the task scheduling heuristics various metrics are considered. These parameters include Makespan, Throughput, Average Response Time (ART), Average Resource Utilization Ratio (ARUR).

Many researchers have proposed numerous task scheduling heuristics [9–17] keeping in view different objectives. However, the question about selecting an appropriate scheduling approach for a given application (i.e., task computation requirements and the available resources) to accomplish the required scheduling objectives like reduced response time and energy consumption, higher ARUR, etc.) is still complex [4,18,19]. As different scheduling heuristics are associated with different underlying assumptions; thus, performing a precise comparison cannot be guaranteed. In this research work, we have considered some renown state-of-the-art static task scheduling approaches [i.e., Proactive Simulation-based Scheduling and Load Balancing (PSSLB) [20], Minimum Completion Time (MCT) [13], Max-Avg [12], MinMin [11], Load Balance Improved MinMin (LBIMM) [10], Task-Aware Scheduling Algorithm (TASA) [17], Resource-Aware Scheduling Algorithm (RASA) [15], and Sufferage [16]].

[*]*Corresponding author. Email: ibrahimmayar@vu.edu.pk*

To evaluate the performance of each of the state-of-the-art task scheduling heuristic approaches, several metrics are considered as provided in Table 1. Table 1 delineates two types of information (i.e., Task scheduling approach and parameters used for the performance evaluation). To quantify the performance of the proposed approaches, different researchers (i.e., [21,22]) considered the parameters reported in Table 1. From the existing comparative analysis work, it can be seen that most of the researchers opted Makespan and ARUR metrics to measure the execution time and resource utilization of the compared contemporary scheduling heuristics.

One similar study performed a comparative analysis considering Minimum Execution Time (MET), first come first serve, MinMin, MCT, Sufferage, and Max–Min scheduling approaches [8]. After an in-depth comparative analysis, the authors concluded that hybrid scheduling approaches may produce improved results and thus lead optimization of workload scheduling in the context of Cloud computing [8]. Keeping in view these points, this research also consider hybrid task scheduling heuristics like TASA [17], RASA [15], LBIMM [10], and PSSLB [20] for comparative evaluation in the context of Cloud computing. For instance, the RASA approach is based on combination of MinMin and Max–Min, TASA utilizes the concept of MinMin and Sufferage, whereas the LBIMM is an improved version of the MinMin task scheduling approach. The PSSLB task scheduling approach is the modified and improved version of the Max-Min task scheduling algorithm.

The contributions of this work are presented as follows: An in-depth comparative analysis of task aware, resource-aware, and hybrid task scheduling approaches by utilizing two prominent benchmarks Dataset (i.e., GOCJ [24] and HCSP [25] instances) implemented using CloudSim [25]. The experiments are performed using four evaluation metrics that are: ARUR, Makespan, Throughput, and ART. Based on the result discussion and comparative analysis, we have provided several recommendations for the Cloud users and service provides in terms of optimum response time and better resource utilization respectively.

The rest of the work is outlined as follows: Section 2 discuss the task scheduling heuristics working and corresponding evaluation metrics utilized for comparative analysis. The details regarding experimental configuration and Dataset selection is presented in Section 3. The experimental evaluation, Result discussion and recommendations are delinted in Section 4 and finally, the conclusions and future work are explained in Section 5.

Table 1 | Task scheduling approaches parameters comparison

Task scheduling approach	Evaluation metrics
MCT [13]	Makespan, Throughput
MinMin [11]	ARUR, Makespan, Throughput
Max–Min [14]	ARUR, Makespan, Throughput
Max-Avg [12]	Makespan, ARUR
LBIMM [10]	ARUR, Makespan, Throughput, Average VIP task completion time
PSSLB [20]	ARUR, Makespan, Throughput
PSSELB [20]	ARUR, Makespan, Throughput
RASA [15]	ARUR, Makespan, Throughput, load imbalance
SLA-MinMin [9]	ARUR, Makespan, Gain, Penalty
Sufferage [16]	Makespan, Throughput
TASA [17]	ARUR, Makespan, Throughput
SLA-MCT [9]	Makespan, Throughput, Gain, Penalty
Heuristic evaluation [23]	Cost, Makespan, Throughput
Analysis of 10 scheduling approaches	ARUR, Makespan, Throughput

2. COMPARISON FRAMEWORK

This Section contemplates the working of state-of-the-art static Cloud task scheduling heuristics (considered for comparative analysis in this study) descriptively.

Minimum completion time [13] is designed for reducing the execution and response time of tasks. For this purpose, the MCT-based approach tries to assign tasks to the faster VMs. This approach selects a single task and compute expected completion time on each VM and assign the task to the VM that gives minimum expected completion time. Therefore, most of the time faster VMs are overloaded; however, the slower VMs remain idle.

MinMin [11] algorithm uses MCT as a base algorithm. This algorithm is best suited for the situation where there are fewer small size tasks and higher number of larger size tasks. The MinMin heuristic favors smaller tasks and penalizes large size tasks. The working procedure of the MinMin algorithm consists of two steps. In the first step, the smallest task is selected among the set of tasks and computes expected completion time on every VM. In the second step, a VM that gives minimum expected completion time is selected for scheduling the tasks.

To improve Makespan and resource utilization of Cloud, the authors have proposed a Max-Avg task scheduling heuristic in Maipan-uku et al. [12]. The proposed approach uses Max–Min as a base algorithm that completes their scheduling decision in two phases. In the first phase, the proposed algorithm selects the largest task, compute their completion time on every VM, and identify VM with minimum completion time. In the second phase, average completion time is calculated. In case, the average completion time of tasks is less than completion time of the largest task, then the smallest task is identified and assigned to VM that executes that task in minimum time. If average completion time is greater than the completion time of the selected task, then the largest task is assigned to VM that completes them in the smallest possible time.

A LBIMM (Cloud) tasks scheduling heuristic has been proposed in Chen et al. [10]. The proposed algorithm is modified form of MinMin scheduling algorithm that maps users tasks to VM based on user priority. LBIMM provides user choice to select type of services like time and cost etc. they needed and schedule tasks according to the user choice.

Proactive simulation-based scheduling and load balancing algorithm has been proposed in Alaei and Safi-Esfahani [20]. This approach divides a batch of input tasks into two halves. The first half of the tasks is directly mapped using pooling method i.e., in reactive manner. However, the second half of tasks are mapped proactively using simulation feedback. The reason for mapping first half of tasks is to reduce dependency on historical scheduling information and start scheduling based on minimum available information.

Resource aware scheduling algorithm is a hybrid algorithm proposed in Parsa and Entezari-Maleki [15]. RASA combines best practices of Max–Min and MinMin tasks scheduling heuristics. The proposed approach uses Max–Min and MinMin scheduling approaches alternatively based on the count of tasks. If the number of tasks is even then Max–Min scheduling heuristic is used; however, MinMin scheduling algorithm is considered for VM tasks mapping if the number of tasks is odd.

To reduce waiting time of the tasks and improve completion time of Cloud datacenter, a TASA has been proposed in Taherian Dehkordi and Khatibi Bardsiri [17]. TASA algorithm combines best features of Sufferage and MinMin algorithms. MinMin scheduling algorithm helps in reducing tasks allocation time for smaller tasks while Sufferage scheduling heuristic helps in reduction of waiting time of the larger tasks. Therefore, this algorithm creates a balance in executing both smaller and larger size tasks.

Sufferage algorithm [16] computes completion time of all tasks on every VM and store them in the form of matrix. The proposed algorithm identify task with minimum completion time and task with second minimum completion time. After that, it computes Sufferage value for these two tasks and select task with high Sufferage value and assign to VM that finishes their execution in minimum time. Sufferage algorithm gives high priority to the task with higher waiting time.

3. EXPERIMENTAL SETUP

To investigate the performance of any scheduling approach for Cloud computing, experimental, analytical, or simulation approach can be used. One problem with the experimental techniques is the high cost and complexity in configuration which in turn may require an expert to setup the test bed. Additionally, executing simulation experiments on the real Clouds may require subscription fee, thus performing the experiments several times may incur substantial monitory cost. On the other hand, the analytical approaches may not be able to investigate the proposed scheduling heuristics in all aspects. Thus, the simulation techniques are best choice for experimental evaluation of the underlying scheduling heuristics using wide variety of configurations. The Cloudsim [25] simulation platform was utilized for the experiments in this research work.

To investigate and compare the performance of different scheduling algorithms, three important factors need to be considered; firstly, the choice of the simulation tool utilized for experimentation; secondly, the dataset selected for the experimental analysis; and thirdly, the performance metrics considered for the evaluation. In this particular research, we have considered two renowned benchmark datasets known as HCSP proposed by Braunt et al. [26] and GOCJ proposed in Hussain and Aleem [24]. The HCSP model is based on the expected time to compute matrix having m number of tasks and n number of VMs. The HCSP dataset used in this research is comprised of four instances (i.e., c-hilo, c-lohi, i-hilo, i-lohi) having a different level of task and resource heterogeneity is given below:

hilo: High heterogeneity of Cloudlet with low heterogeneity of VMs.

lohi: Low heterogeneity of Cloudlets and high heterogeneity of VMs.

4. EXPERIMENTAL EVALUATION

The choice of the right scheduling heuristic depends on several factors including resource utilization, Makespan, Throughput, response time, energy consumption, and SLA, etc. From the end-user perspective, the main important considerations are related to the response time and Makespan keeping intact the quality of service (SLA); however, the Cloud Service Providers (CSPs) are more concerned about the maximum useful utilization of the available resources and the revenue generation. Keeping in view, these objectives, the requirement is to develop and implement scheduling approaches that are capable of handling a large number of scheduling requests and efficient resource utilization of the available pool of resources. In this work, these areas are highlighted about the trade-off among the available scheduling approaches to help the CSPs in the right decision making. The discussion concerning the obtained results is delineated below.

4.1. Results and Discussion

The comparative investigation is performed in terms of four parameters (Makespan, ARUR, Throughput, and ART. Following are the formulas pertaining to each of metrics used:

1. **Makespan**: The Makespan is defined as the maximum time required to complete the execution of the task on each of the VM as given in Equation (1)

$$\text{Makespan} = \text{Max}\{\text{CT}_j\} = \text{Max}\{\text{CT}_1, \text{CT}_2, \ldots, \text{CT}_m\} \tag{1}$$

   where CT_j shows completion time of VM_j and m represent number of VMs.

2. **ARUR**: The ARUR corresponds to average resource utilization by each of the heuristic during the complete execution of all the tasks on the available VMs as given in Equation (2).

$$\text{ARUR} = \frac{\text{avgMakespan}}{\text{Makespan}} \tag{2}$$

   where avgMakespan is computed in Equation (3)

$$\text{avgMakespan} = \frac{\sum_{j=1}^{m} \text{Makespan}_j}{m} \tag{3}$$

3. **Throughput**: The throughput is defined as the number of tasks executed per unit time. The throughput can be measured as:

$$\text{Throughput} = \frac{\text{Number of tasks}}{\text{Makespan}} \tag{4}$$

4. **ART**: The response time is the time between the request from end-users for the task execution to the result response received at the received side in the form of task results. The ART is calculated as follows:

$$\text{ART} = (\text{Finish}_{\text{Time}} - \text{Arrival}_{\text{Time}}) + \text{Waiting}_{\text{Time}} \tag{5}$$

Figure 1 shows comparison results of ARUR using four instances i.e., i-hilo, i-hilo, i-lohi, and i-lohi of HSCP dataset for state-of-the-art static task scheduling heuristics. These algorithms include LBIMM, Max-Avg, MCT, MinMin, PSSLB, RASA, Sufferage, and TASA. Experimental results plotted in Figure 1 shows that TASA task scheduling algorithm outperformed LBIMM, Max-Avg, MCT, MinMin, RASA, and Sufferage in terms of ARUR for all the four instances of HCSP datasets. Its because, TASA is task-aware and adopt scheduling strategy based on tasks nature. The ARUR value of PSSLB is comparable with TASA for i-hilo and i-hilo instances of HCSP dataset. Similarly, the ARUR value of PSSLB is lower than TASA algorithm for c-lohi and i-lohi; however, higher than LBIMM, Max-average, MCT, MinMin, PSSLB, RASA, and Sufferage. The obtained results reveals that the Sufferage and MCT algorithms have lower ARUR values as compared with TASA, LBIMM, Max-Avg, MCT, MinMin, and RASA. The reason is that MCT overloads faster VMs while slower machines remain idle. On the other hand, Sufferage is also MCT-based and involve comparatively more complex computations. Experimental results asserts that TASA and PSSLB are more scalable as compared with other state-of-the-art task scheduling approaches.

Makespan is the overall execution time and one of the most important parameters especially for Cloud service users in Cloud computing environment. Lower value of Makespan show better performance and vice versa. The results presented in Figure 2 reveals that TASA task scheduling algorithm outperformed Sufferage, PSSLB, MCT, Max-Avg, LBIMM, MinMin, and RASA. The reason behind is that TASA is a task-aware scheduling algorithm that combines best features of MinMin and Sufferage. MinMin algorithms favors smaller tasks and Sufferage enables to execute highly waited tasks in priority base. PSSLB has comparable Makespan to that of TASA and results in improved Makespan than the other state-of-the-art tasks scheduling algorithms. Results delineate that MCT has better performance in term of Makespan than LBIMM, RASA, Max-average, Sufferage, and MinMin for i-lohi and i-lohi instances of HCSP dataset, however, poor performance for i-hilo and i-hilo. This shows that MCT has better performance for datasets with small size tasks and poor performance for datasets with large size tasks. Due to the high scheduling overhead, Sufferage algorithm has poor performance in term of Makespan as compared with other tasks scheduling algorithms. The overall results show that PSSLB and TASA has better performance on all datasets and can be easily generalized and scaled-up.

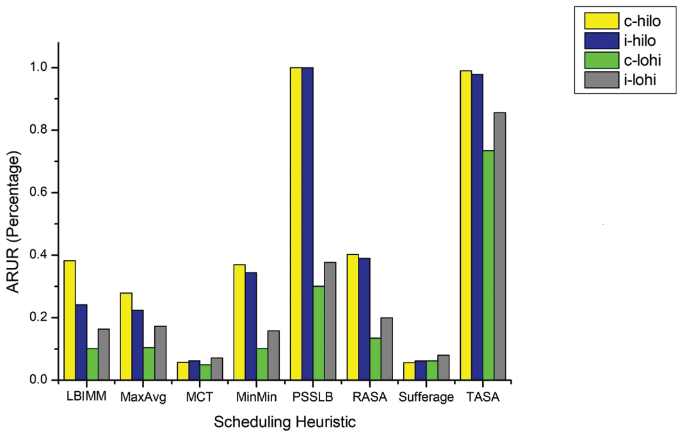


Figure 1 | ARUR comparison on HCSP dataset.

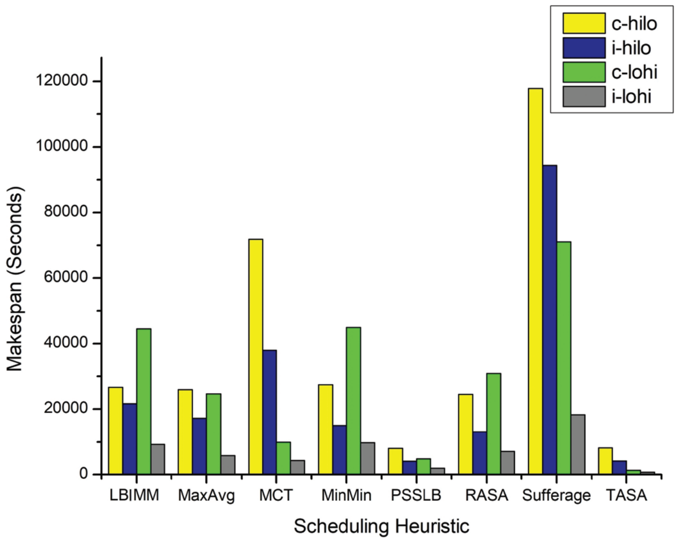


Figure 2 | Makespan comparison on HCSP dataset.

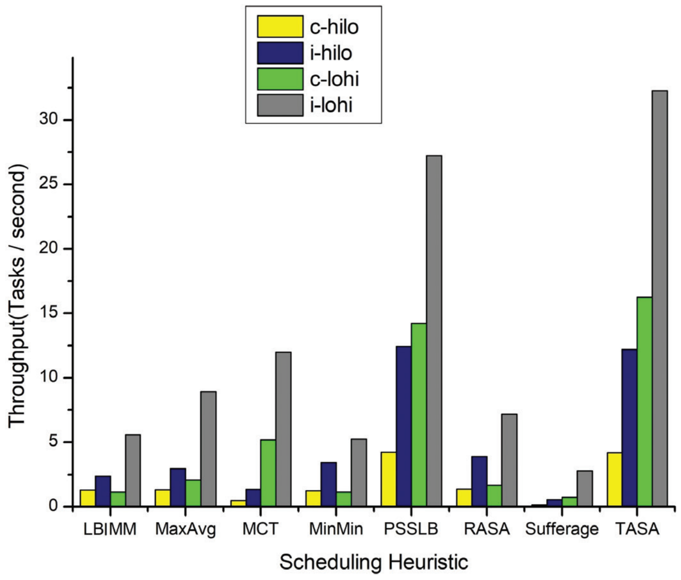


Figure 3 | Throughput comparison on HCSP dataset.

Figure 3 contemplates the results concerning the throughput for all the compared scheduling heuristics (selected in this study). The throughput is the reciprocal of the Makespan, the lower the value of Makespan, the higher the value of throughput will be and vice versa. Similar to the Makespan results, higher throughput is observed for the (i.e., on c-lohi and i-lohi) for the TASA and PSSLB as compared with the rest of the task scheduling algorithms. For Sufferage and MCT, lower throughput is seen for the hilo instances (i.e., c-hilo and i-hilo). However, for lohi dataset instances, the MCT and Sufferage have attained improved throughput. As the lohi dataset has mostly smaller size tasks, thus, the MCT assigns the tasks to the powerful VMs and completing the execution of a higher number of jobs within shorter Makespan. This means for the dataset with most tasks having a smaller size, the MCT will assure minimum SLA violation. Similarly, the PSSLB is best for the datasets having larger size tasks.

The effectiveness and performance of any scheduling algorithm depends on the Makespan and resource utilization. The resource utilization is directly affected by how the available tasks are mapped to the available resources. The more balanced the share of load on each of the VM the more likely is to maximize the resource utilization which ultimately lead to reduced Makespan with better response time.

The result of the ART are plotted in Figures 4 and 5. In Figure 4, the results of hilo instances dataset are explained. The MCT attains the best ART results as compared with the rest of the scheduling approaches. This is because MCT harnesses the power of the fastest machines for each of the scheduling requests. Thus, MCT comes up

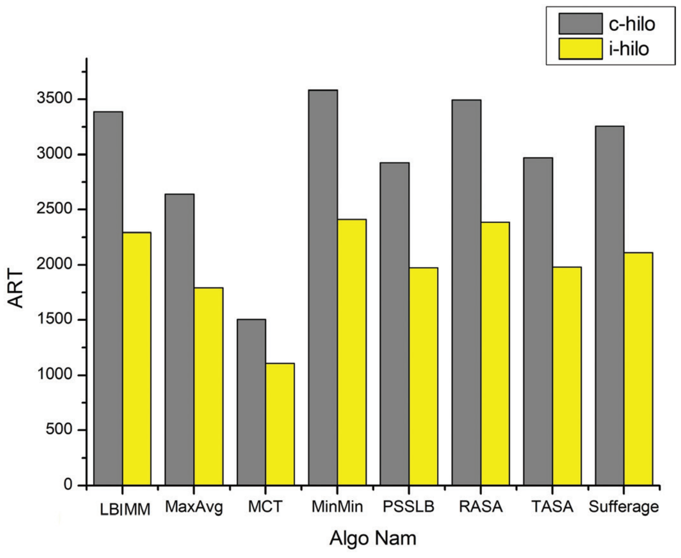


Figure 4 | Average response time comparison on HCSP hilo instances.

with the best response time. However, leads to inefficient resource utilization as most of the slower machines remain idle during the task scheduling by the scheduler for MCT. The TASA, PSSLB, and Max-Avg showed the second leading performance results in terms of ART for both c-hilo and i-hilo dataset instances. The Sufferage scheduling approach has shown improved ART for the i-hilo instance. Figure 5 delineates the ART results for all the compared contemporary approaches on the lohi datasets. Once again, the MCT outperformed the rest of the compared scheduling heuristics in terms of ART. The MCT has been able to reduce the ART by 60% as compared with the TASA and PSSLB and about 400–600% as compared with the rest of the approaches. The TASA and PSSLB again revealed very good ART results as compared with the rest of the compared scheduling approaches.

To see the behavior of each of the scheduling approach in more depth, another set of experiments was conducted using GOCJ dataset instance. Again, the same set of performance metrics (i.e., ARUR, Makespan, Throughput, and ART) were considered for performance investigation. Figure 6 plots the results regarding average resource utilization (ARUR) for all the compared contemporary approaches. The worst performance is observed for the MCT and Sufferage scheduling heuristics. The MCT and Sufferage use a greedy approach for each of the incoming tasks to complete the execution of the task by selecting the most powerful VMs from the available resources. In this way, the completion of the tasks is ensured up to some extent; however, the resources of the low powers VMs are mostly idle during the task scheduling and thus resulted in poor ARUR. Likewise the results of the HCSP dataset, the TASA scheduling approach outperformed all the compared approaches by achieving 98% resource utilization on the GOCJ dataset. The PSSLB that is a modified form of Max–Min has been able to achieve 80% of resource utilization and thus been considered as the second highest achiever.

The Makespan is an another very important performance metrics used by almost every proposed task scheduling approach for performance evaluation. Thus, the second set of experiments was conducted for all of the compared contemporary approaches and

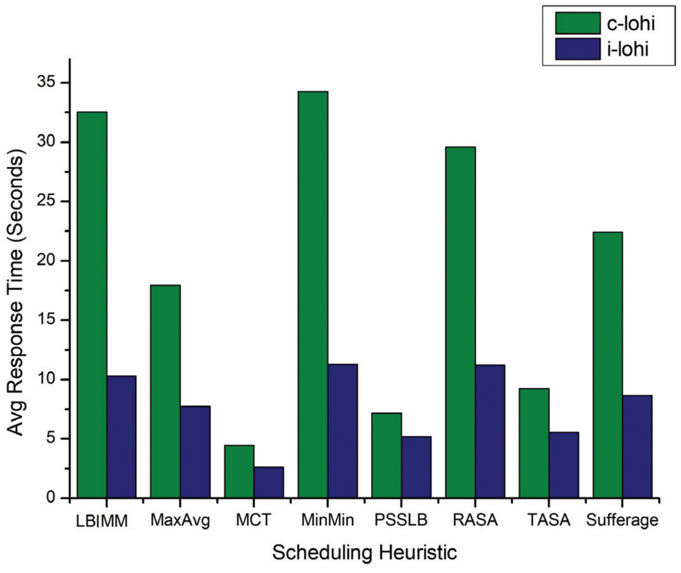


Figure 5 | Average response time comparison on HCSP lohi instances.

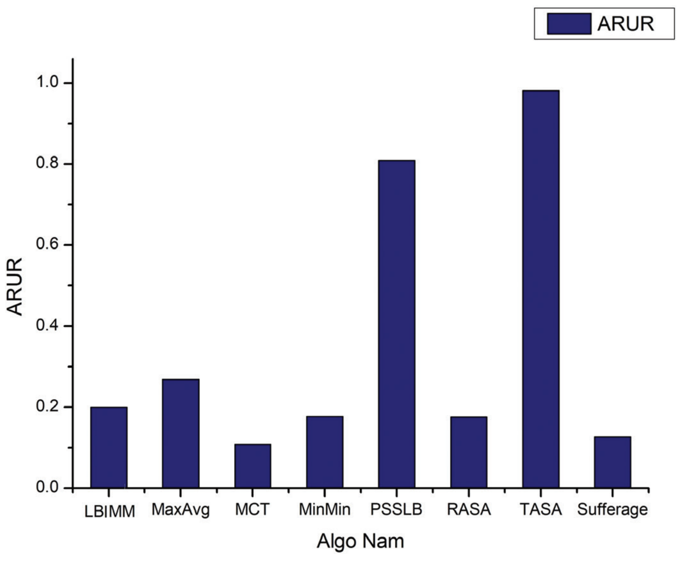


Figure 6 | ARUR on GOCJ dataset.

results were obtained for Makespan. These results were obtained and plotted in Figure 7. The results are the average of multiple runs. The PSSLB and TASA approaches have been able to reduce the Makespan by four times as compared with the Max-Avg scheduling approach and eight to 10 times as compared with the rest of heuristics. The MCT has been able to achieve the second-best performance as compared with the rest of the scheduling approaches except TASA and PSSLB. This is the reason why most of the researchers consider the MCT approach while proposing their own task scheduling approaches. From this discussion, it can be inferred that the TASA approach is considered an optimal choice for the Cloud service providers as well as for the end users.

To dive into more depth of the performance evaluation, the results pertaining to the throughput metric are also gathered and plotted in Figure 8. The throughput shows the execution of Cloudlets per second. Likewise, Makespan results, the TASA has been able to attain the highest throughput as compared with the rest of the approaches followed by the PSSLB which is the second-best optimal scheduling approach. Similarly, the MCT due to its greedy nature toward harnessing the fastest machines has shown moderate performance as compared with the rest of contemporary approaches other than PSSLB and TASA.

From the end-user point of view, the Makespan and response time metrics are very important. The response time is the time between the request from end-users for the task execution to the result response received at the receiver side in the form of task results. The fourth performance metric considered for the performance evaluation of the compared available approaches is the ART.

Figure 9 contemplates the ART results concerning all the contemporary task scheduling approaches. The RASA, MinMin, and LBIMM resulted in poor ART as compared with the rest of the approaches. The MCT approach has been able to dramatically reduce the response time and this behavior is due to the reason for harnessing the powerful VMs for each of the incoming task requests. However, this has resulted in two major issues: First, most of the low power VMs remain idle during the task scheduling execution. Second, the overall average resource utilization is hurt and seems to be several times lower as compared with the other approaches. The TASA and PSSLB are the most consistent approaches showing a moderate performance for the ART and thus are considered to be the most optimal choice for diverse datasets.

From the above-given results and discussion pertaining to the performance investigation of the contemporary scheduling approaches, we have highlighted and extracted several important points.

The PSSLB and TASA showed scalable performance concerning the Makespan, ARUR, ART, and throughput for both of the benchmark datasets.

The CSPs are more interested in higher resource utilization. In case, the resources are not efficiently utilized may also contribute to higher Makespan and more energy consumption. Therefore, more care is required while addressing the resource utilization problem. To achieve higher resource utilization, the tasks need to assigned to the VMs keeping in view the workload properties, the existing tasks to VMs mapping, etc. The resource and task-aware mapping

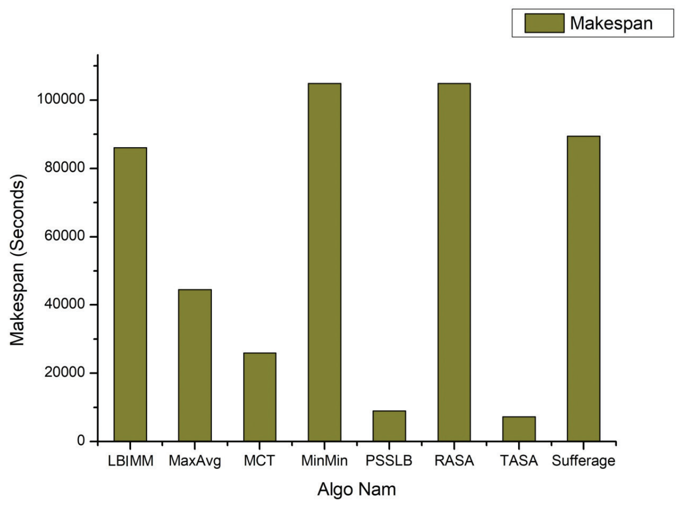


Figure 7 | Makespan on GOCJ dataset.

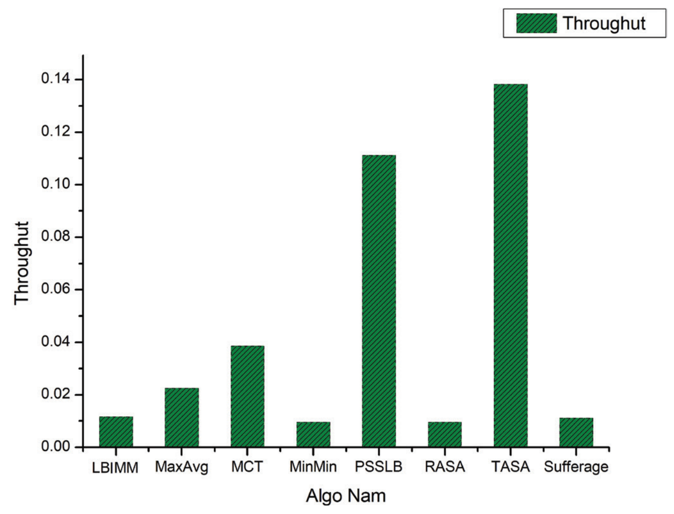


Figure 8 | Throughput on GOCJ dataset.

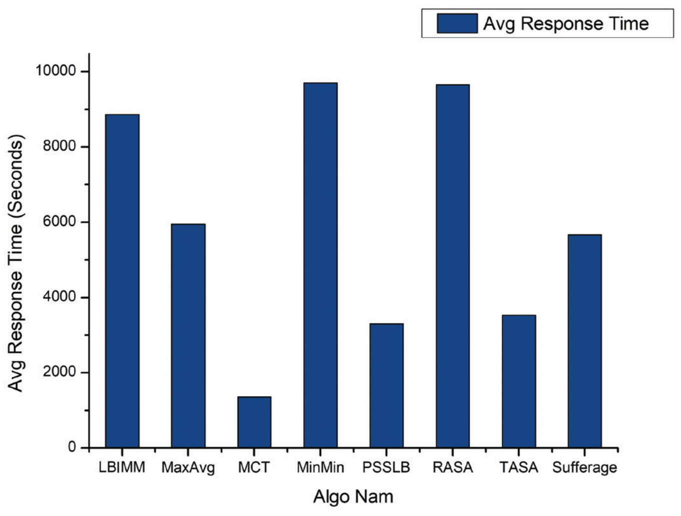


Figure 9 | ART on GOCJ dataset.

of Cloudlets on the available VMs in a load-balanced manner will lead to eliminate the problem of under-utilized/overloaded VMs ultimately resulting in reducing high energy costs.

Moreover, the CSPs can increase their revenue with reduced energy consumption in the CDCs by careful scheduling, reduced Maksespan and resource utilization efficiency.

The reduction in execution time will also result is Cloud end-user satisfaction.

5. CONCLUSION AND FUTURE WORK

In this work, the performance of several renowned state-of-the-art static task scheduling approaches is evaluated empirically. Two widely used datasets were utilized for the experimental analysis (i.e., GOCJ, and HCSP). The Cloudsim simulation tool was utilized for the comparative investigation. The workload used for the simulations was configured with VMs and tasks having different heterogeneity levels. The results have shown the usefulness of TASA and PSSLB and remained the top performer concerning resource utilization and Makespan. The results have shown that the ARUR of PSSLB is reduced to 31% and 39%. Moreover, the TASA scheduling approach outperformed and showed scaleable performance against all of the compared scheduling heuristics for both HCSP and GOCJ datasets. The MCT and Sufferage have shown improved ART results. The LBIMM, MinMin, Max-Avg, and RASA resulted in a moderate performance with respect to the performance metrics. The results advocates the use of TASA and PSSLB approach to be used for a variety of tasks. The MCT and Sufferage approaches are useful when main goal is to reduce the average response time.

In our future work, we plan to compare the available contemporary scheduling approaches with respect to energy consumption and QoS. The future work will also propose and implement a hybrid task scheduling approach and compare against the state-of-the-art approaches concerning ARUR, Makespan, Throughput, ART, and energy consumption.

CONFLICTS OF INTEREST

The authors declare they have no conflicts of interest.

REFERENCES

[1] M. Ibrahim, M.A. Iqbal, M. Aleem, M.A. Islam, Sim-cumulus: an academic cloud for the provisioning of network-simulation-as-a-service (NSaaS), IEEE Access. 6 (2018), 27313–27323.

[2] M.A. Iqbal, M. Aleem, M.A. Islam, M. Ibrahim, S. Anwar, Amazon cloud computing platform EC2 and VANET simulations, Int. J. Ad Hoc Ubiquit. Comput. 30 (2019), 127–136.

[3] F. Durao, J.F.S. Carvalho, A. Fonseka, V.C. Garcia, A systematic review on Cloud computing, J. Supercomput. 68 (2014), 1321–1346.

[4] S. Groot, Research on efficient resource utilization in data intensive distributed systems, Ph.D. dissertation, University of Tokyo, Japan, 2013.

[5] A. Beloglazov, R. Buyya, Managing overloaded hosts for dynamic consolidation of virtual machines in Cloud data centers under quality of service constraints, IEEE Trans. Parallel Distrib. Syst. 24 (2012), 1366–1379.

[6] H. Jin, W. Gao, S. Wu, X. Shi, X. Wu, F. Zhou, Optimizing the live migration of virtual machine by CPU scheduling, J. Netw. Comput. Appl. 34 (2011), 1088–1096.

[7] V. Sivagami, K. Easwarakumar, An improved dynamic fault tolerant management algorithm during vm migration in Cloud data center, Future Gen. Comput. Syst. 98 (2019), 35–43.

[8] B. Jennings, R. Stadler, Resource management in clouds: survey and research challenges, J. Netw. Syst. Manage. 23 (2015), 567–619.

[9] S.K. Panda, P.K. Jana, SLA-based task scheduling algorithms for heterogeneous multi-Cloud environment, J. Supercomput. 73 (2017), 2730–2762.

[10] H. Chen, F. Wang, N. Helian, G. Akanmu, User-priority guided MinMin scheduling algorithm for load balancing in cloud computing, 2013 National Conference on Parallel computing technologies (PARCOMPTECH), IEEE, Bangalore, India, 2013, pp. 1–8.

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

[12] J. Maipan-uku, A. Muhammed, A. Abdullah, M. Hussin, Max-average: an extended max-min scheduling algorithm for grid computing environtment, J. Telecommun. Electron. Comput. Eng. 8 (2016), 43–47.

[13] N.A. Mehdi, A. Mamat, Z.T. Abdul-Mehdi, Minimum completion time for power-aware scheduling in cloud computing, 2011 Developments in E-systems Engineering, IEEE, Dubai, UAE, 2011.

[14] Y. Mao, X. Chen, X. Li, Max–min task scheduling algorithm for load balance in cloud computing, Proceedings of International Conference on Computer Science and Information Technology, Springer, New Delhi, 2014, pp. 457–465.

[15] S. Parsa, R. Entezari-Maleki, RASA: a new grid task scheduling algorithm, Int. J. Digit. Content Technol. Appl. 3 (2009), 152–160.

[16] E.K. Tabak, B.B. Cambazoglu, C. Aykanat, Improving the performance of independent task assignment heuristics MinMin, MaxMin and Sufferage, IEEE Trans. Parallel Distrib. Syst. 25 (2013), 1244–1256.

[17] S. Taherian Dehkordi, V. Khatibi Bardsiri, TASA: a new task scheduling algorithm in cloud computing, J. Adv. Comput. Eng. Technol. 1 (2015), 25–32.

[18] A.R. Arunarani, D. Manjula, V. Sugumaran, Task scheduling techniques in Cloud computing: a literature survey, Future Gen. Comput. Syst. 91 (2019), 407–415.

[19] A. Hussain, M. Aleem, M.A. Iqbal, M.A. Islam, A rigorous evaluation of state-of-the-art scheduling algorithms for Cloud computing, IEEE Access 6 (2018), 75033–75047.

[20] N. Alaei, F. Safi-Esfahani, RePro-active: a reactive–proactive scheduling method based on simulation in cloud computing, J. Supercomput. 74 (2018), 801–829.

[21] Y. Wang, J-T. Zhou, Y. Jiao, X. Song, Comparative analysis of evolutionary algorithms based on swarm intelligence for qos

optimization of cloud services, 2019 IEEE 23rd International Conference on Computer Supported Cooperative Work in Design (CSCWD), IEEE, Porto, Portugal, 2019, pp. 434–439.

[22] B. Li, Y. Pei, H. Wu, B. Shen, Heuristics to allocate high-performance Cloudlets for computation offloading in mobile ad hoc clouds, J. Supercomput. 71 (2015), 3009–3036.

[23] S.H.H. Madni, M.S.A. Latiff, M. Abdullahi, M.J. Usman, Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment, PLoS ONE 12 (2017), e0176321.

[24] A. Hussain, M. Aleem, GoCJ: Google cloud jobs dataset for distributed and cloud computing infrastructures, Data 3 (2018), 38.

[25] A. Hussain, M. Aleem, M.A. Iqbal, M.A. Islam, Investigation of cloud scheduling algorithms for resource utilization using CloudSim, Comput. Inform. 38 (2019), 525–554.

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