IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 31, 2020, accepted June 15, 2020, date of publication July 6, 2020, date of current version July 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3007201

An In-Depth Empirical Investigation of State-of-the-Art Scheduling Approaches for Cloud Computing

MUHAMMAD IBRAHIM[1], SAID NABI[1], ABDULLAH BAZ[2], (Senior Member, IEEE), HOSAM ALHAKAMI[3], MUHAMMAD SUMMAIR RAZA[1], ALTAF HUSSAIN[4], KHALED SALAH[5], (Senior Member, IEEE), AND KARIM DJEMAME[6], (Member, IEEE)

[1]Department of Computer Science, Virtual University of Pakistan, Rawalpindi 46300, Pakistan
[2]Department of Computer Engineering, College of Computer and Information Systems, Umm Al-Qura University, Makkah 21955, Saudi Arabia
[3]Department of Computer Science, College of Computer and Information Systems, Umm Al-Qura University, Makkah 21955, Saudi Arabia
[4]Department of Computer Science, KICSIT Campus, IST, Islamabad 47330, Pakistan
[5]SEECS Department, Khalifa University, Abu Dhabi, United Arab Emirates
[6]School of Computing, University of Leeds, Leeds LS2 9JT, U.K.

Corresponding author: Muhammad Ibrahim (ibrahimmayar@vu.edu.pk)

This work was supported by the Deanship of Scientific Research at Umm Al-Qura University under Grant 19-COM-1-01-0015.

ABSTRACT Recently, Cloud computing has emerged as one of the widely used platforms to provide compute, storage and analytics services to end-users and organizations on a pay-as-you-use basis, with high agility, availability, scalability, and resiliency. This enables individuals and organizations to have access to a large pool of high processing resources without the need for establishing a high-performance computing (HPC) platform. From the past few years, task scheduling in Cloud computing is reckoned as eminent recourse for researchers. However, task scheduling is considered an NP-hard problem. In this research work, we investigate and empirically compare some of the most prominent state-of-the-art scheduling heuristics in terms of Makespan, Average resource utilization (ARUR), Throughput, and Energy consumption. The comparison is then extended by evaluating the approaches in terms of individual VM level load imbalance. After extensive simulation, the comparative analysis has revealed that Task Aware Scheduling Algorithm (TASA) and Proactive Simulation-based Scheduling and Load Balancing (PSSLB) outperformed as compared to the rest of the approaches and seems to be optimal choice keeping in view the trade-of between the complexities involved and the performance achieved concerning Makespan, Throughput, resource utilization, and Energy consumption.

INDEX TERMS Cloud computing, resource allocation, task scheduling, scheduling algorithms, load balancing, performance evaluation, load imbalance.

I. INTRODUCTION

For the last few years, task scheduling in Cloud computing got attention from the research community. The Cloud computing platform is deemed to provide high processing and huge memory-intensive services to the users in pay as you use manner. The resources in the Cloud computing are distributed across different locations in the form of large-scale data centers connected in a distributed way. In the Cloud computing environment, the resources are provisioned (to the users in an on-demand fashion) in the form of virtual machines (VMs) from a pool of configured resources (Physical Hosts (PHs)) keeping in view the requirements of the intended users [1], [2]. The Cloud service owners provide guaranteed services called Quality of Service (QoS) to the users to meet the requirements of their customers [3]. These QoS contracts are formally maintained in the form of Service Level Agreements (SLA). To meet the SLA requirements of the users, the resources are provisioned aggressively from the Cloud Data Centers(CDCs) [4], [5]. Allocating resources in this way may lead to in-efficient resource allocation which will have a devastating impact on resource utilization across the Cloud. Moreover, will lead to workload imbalance across the CDCs causing data and computing skewness problem [6]. This will further result in under-utilization of some of the resources and on the other hand the waiting time of some of the jobs will also be increased [7]. To cope with these issues, the VM Migrations approaches are proposed. The VM

The associate editor coordinating the review of this manuscript and approving it for publication was P. K. Gupta.

TABLE 1. Task scheduling heuristics evaluation metrics.

Scheduling Algorithm	Evaluation Parameters	Core Parameter
MCT [15]	Makespan, Throughput	Makespan
MinMin [11]	ARUR, Makespan, Throughput	Makespan
MaxMin [16]	ARUR, Makespan, Throughput	Makespan
MaxAvg [13]	Makespan, ARUR	ARUR
LBIMM [10]	ARUR, Makespan, Throughput	Average VIP task completion time
PSSLB [12]	ARUR, Makespan, Throughput	ARUR
RALBA [17]	ARUR, Makespan, Throughput, load imbalance	ARUR, load imbalance
RASA [18]	ARUR, Makespan, Throughput, load imbalance	ARUR, load imbalance
SLA-MinMin [7]	ARUR, Makespan	Gain, Penalty
Sufferage [23]	Makespan	load imbalance ratio
TASA [24]	ARUR, Makespan, Throughput	ARUR Makespan
SLA-Ralba [26]	ARUR, Makespan, Throughput, SLA	SLA, Makespan
SLA-MCT [7]	Makespan, Throughput,	Gain, Penalty
Heuristic Evaluation [25]	Cost, Makespan, Throughput, Load Imbalance	Load Imbalance

migration approaches are useful in achieving load balancing, fault tolerance, scheduling optimization, and energy-aware resource/task scheduling. VM migration [8], [9] provides the ability to reduce the downtime pertaining to the overloaded VMs. With all its benefits, VM migration suffers from several issues including overhead involved and memory consumed during the migration phase. To mitigate these issues, a number of approaches are being introduced for VM migration. However, the required efficiency will not be achieved until and unless the VM migration is integrated with an efficient approach. One such approach will be to provide a load-balanced task distribution across the Cloud.

In the recent few years, the researchers have focused on introducing Cloud scheduling heuristics focusing on load-balanced provisioning of tasks with the aim of producing efficient resource utilization. The Min-Min [11] scheduling algorithm was introduced to give priority to the tasks with a smaller size. However, the tasks having larger size had to wait for longer times before executing them on the available resources thus leading to inefficient resource utilization. In [10], Chen *et al.* extended the existing Min-Min [11] approach by the proposal of two efficient load balancing heuristics (namely Load-Balanced Improved Min-Min (LBIMM) and User Priority-aware Load-Balanced Improved Min-Min (PA-LBIMM)). The PA-LBIMM considers the users' priority while keeping the load balanced scheduling thus provide improved results in terms of achieving the required level of SLA and resource utilization. This approach also assures a gain in the required level of throughput, better resource utilization, and better response time for all the required tasks. The authors in [12] introduced and implemented a RePro-active scheduling framework that utilizes the existing knowledge related to the tasks with the aim of load-balanced scheduling. The Max-Average scheduling heuristics was introduced that extended and improved the Max-Min algorithm with an aim of providing load balanced task scheduling across the CDCs [13]. Chauhan and Joshi proposed two different QoS aware heuristics (i.e., QWMTS (QoS Weighted Mean Time Min-Min Max-Min Selective) and QWMTM (QoS Guided Weighted Mean Time-Min)) for load-balanced task scheduling across the CDCs. The authors in [14] designed and implemented a load-balanced algorithm by extending the Max-Min strategy that keeps the status of each of the tasks. This information is used as an input for predicting the real-time load on each of the VM across the Cloud. The SLA-Min-Min and SLA-MCT approaches are presented in [7] that consider the cost and task execution time as the QoS parameters for guaranteed scheduling.

The task scheduling in the Cloud computing is NP-hard in nature. This means a single metric does not guarantee the efficiency and effectiveness of any task scheduling approach. Several metrics are utilized to measure the efficiency and optimality of the task scheduling approaches. Each evaluation metric measures the performance of the task scheduling approaches from a different perspective. Thus, it is not trivial to decide which task scheduling approach is the optimal choice. Keeping in view these points, in this research, we have selected few renowned state-of-the-art static task scheduling heuristics (i.e., task aware, resource-aware, and hybrid approaches) for an in-depth empirical investigation concerning five performance metrics (i.e., ARUR, Makespan, Throughput, Energy consumption, and Individual level VM load imbalance). In order to quantify the performance of each of the state-of-the-art approaches (compared in this work), several parameters are utilized as reported in Table 1. The Table provides three types of information that are: heuristic name, parameters used for evaluating the performance, and one main parameter that meets the objectives of the proposed scheduling approaches. In order to see the in-depth analysis of the proposed approaches, several approaches (i.e., [19], [21]) have provided a comparative analysis of the available contemporary approaches by employing the parameters given in Table 1. Moreover, it is observed that most of the approaches considered makespan and ARUR as the performance metrics to measure the load balancing and resource utilization attained by the scheduling approaches which are measured as:

$$Makespan = Max\{CT_j\} = Max\{CT_1, CT_2, \ldots, CT_m\} \quad (1)$$

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

$$ARUR = \frac{avgMakespan}{Makespan} \quad (2)$$

where avgMakespan is computed in Equation 3

$$avgMakespan = \frac{\sum_{j=1}^{m} Makespan_j}{m} \tag{3}$$

The throughput can be measured as:

$$Throughput = \frac{numberoftasks}{Makespan} \tag{4}$$

From the comparative analysis of the state-of-the-art scheduling approaches (provided in various studies), it is observed that a higher value of ARUR is achieved, however, there is a need to investigate and address the machine level load-imbalance for improving the profitability of the cloud resources. Moreover, the energy consumption also need to be investigated for all the compared approaches. The contributions of this work are presented as follows: Empirical evaluation and comparative analysis of state-of-the-art static task scheduling approach using one of the most prominent Cloud benchmarks Dataset (i.e., HCSP instances) implemented using CloudSim [20]. In the last portion of this work, we outlined few recommendations for the cloud service provides as well as for the cloud users based on the comparative analysis results in terms of machine-level load-balancing, energy consumption, and corresponding resource utilization.

The rest of this paper is organized as follows: Section II delineates the aspirations of this study to highlight the motivations of this work. In Section III, we discuss the task scheduling algorithms working and corresponding parameters utilized for performance investigation. Section IV provides the details regarding experimental configuration setup, Dataset selection, and obtained results. The Result discussion and recommendations are delineated in Section V and finally, Section VI reports the conclusions and future work of this study.

II. OBJECTIVES

Task scheduling and load balancing have become one of the most important and attractive areas of research in the Cloud computing domain. A number of Cloud task scheduling heuristics have been proposed by different researchers. These scheduling heuristics have different scheduling objectives which include minimization of makespan, improving utilization of Cloud resources, SLA, and enhancement of throughput in the Cloud. Some of these approaches focus on reducing makespan while other approaches focus on improving resource utilization and/or throughput. However, these tasks scheduling algorithms have various levels of resource utilization, throughput, and makespan [22]. Moreover, various researchers have used different datasets, different number and combination of VMs with respect to their computation capabilities. This arise the need to comprehensively investigate and empirically examine state-of-the-art tasks scheduling heuristics using available datasets in terms of tasks heterogeneity and computation requirements, and VMs computation capability. The aim of this research is to see the effectiveness of state-of-the-art Cloud tasks scheduling heuristics in terms of makespan, ARUR, Throughput, and energy consumption. Some of the researchers have used homogeneous workload and Cloud environment and others have utilized heterogeneous Cloud resources and workload. However, in a real Cloud environment, both workload and computation resources are heterogeneous in nature. Therefore, we have used four sets of HCSP datasets each of which has workload with a different number of tasks and level of heterogeneity. Similarly, Virtual Machines (VMs) with different computation capabilities have been used. The objective of using different datasets and computing resources is to see how each of the algorithms behave with different datasets. This research will also highlight the load imbalance for each individual VM on various datasets. Moreover, we will identify the maximum, minimum, average load-imbalance behavior of each of the tasks scheduling heuristics.

For this, the computation share of each VM needs to find out which is calculated by using VMs computation power in Million Instructions per Second (MIPS) and total computation requirements of workload using equation 5.

$$Comp - Share_j = | \sum_{i=1}^{n} Clt_i * \frac{VM_j}{\sum_{k=1}^{m} VM_k} \tag{5}$$

where Clt_i shows computation requirements of $Cloudlet_i$ in terms of Million Instructions (MIs), VM_k and VM_j represent computation power of VM_j and VM_k. Comp-$Share_j$ corresponds to the computation share of VM_j. The difference between the computation share of VM and the load assigned by different algorithms is identified and categorized as load-imbalance.

III. TASK SCHEDULING APPROACHES

This section of the study discusses and critically investigate the working of some of the most prominent state-of-the-art static task scheduling algorithms as following.

Minimum Completion Time (MCT) [15] based scheduling heuristics select tasks in the given order and assign them to the VMs that execute the task in minimum time. For this, the MCT based approach scans all VMs for each task and calculate their completion time. On each scheduling decision, VM load/ready time is updated. MCT can reduce makespan but overload faster VMs and slower VMs remain idle. This approach results in poor resource utilization.

Min-Min [11] tasks scheduling heuristics uses basic principles of MCT for tasks to VM mapping. Min-Min scheduling heuristics calculate Expected Execution Time (EET) for all tasks on each of the VM. The Min-Min algorithm selects the task with minimum EET and assigns the task to the resource which executes that task in minimum time. After mapping a task on VM, the load of that VM is updated and the mapped task is removed from the task list. Min-Min based scheduling heuristic favors smaller tasks and penalize tasks with the larger size. This may result in reduced resource utilization and throughput.

Max-Min [16] scheduling heuristic is an MCT-based approach. This approach first calculates the execution time of all tasks on each VM and then selects the row with the highest execution time. In the already selected row, the Max-Min approach selects VM with minimum execution time and map tasks on that VM. On each scheduling decision, the VM load is updated. Tasks execution is started after mapping all the tasks on the respective VMs. Max-Min based scheduling heuristics favor larger tasks and penalizing smaller tasks.

Max-Average (MaxAvg) is Max-Min based improved task scheduling heuristic proposed in [13] and complete their scheduling in two phases. In the first phase, the proposed approach uses the Max-Min approach to calculate expected completion time for a set of tasks on each VM. This approach selects the largest task and identifies the VM(s) that execute the selected task in Minimum Completion Time (MCT). In case, there is more than one candidate VMs that can execute the task with MCT, the proposed approach will assign the task to the resource with the least usage. To identify the resource with the least usage, it calculates the average completion time of all the resources. However, if average completion time is less than the MCT of the smallest task, then it selects the largest task and assigns the task to VM that executes them in minimum time. Otherwise, select the smallest tasks and assign the task to VM with MCT. This scheduling heuristic performs better for smaller datasets while high makespan and reduces throughput for larger tasks. Min-Min [11] scheduling heuristic reduces the makespan as compared to other tasks scheduling heuristics. However, the main issue with the Min-Min algorithm is poor resource utilization which is one of the key requirements of Cloud Service Provider (CSP). Authors in [10] have proposed an improved load balancing scheduling approach namely Load Balanced Improved Min-Min (LBIMM). This algorithm is based on Min-Min task scheduling heuristics. Moreover, this approach also can schedule users' jobs according to their priority like job execution time and cost. The proposed approach provides a user with the option to select the type of services they need. To evaluate the proposed technique parameters like makespan, resource utilization and average completion time of high priority jobs are considered.

PSSLB is a proactive simulation-based scheduling and load balancing algorithm proposed in [12]. This algorithm sorts a batch of incoming tasks in descending order of their size and calculates completion time of every task on all the available resources (i.e., VMs). PSSLB scheduling heuristic store ETC (expected time completion) in the form of a matrix, where each row represents the completion time of a task on every VM. The heuristic then selects the tasks with minimum completion time.

The authors in [18] proposed Resource Aware Scheduling Algorithm (RASA). RASA exploits the strengths of Min-Min and Max-Min tasks scheduling heuristics. A set of tasks are provided as an input to RASA that calculates the execution time of every task on each VM and stores it in the form of a matrix. This approach uses Min-Min tasks scheduling heuristic if the number of tasks is odd. For an even number of tasks, this approach applies the Max-Min based approach to all remaining tasks. RASA provides a fair task allocation criteria for small as well as large size tasks. However, RASA leads to load imbalance if there are a higher number of large size tasks.

Sufferage [23] task scheduling heuristic uses MCT as a base approach and calculate execution time for each task on the available resources (VMs). sufferage scheduling classifies MCT and second MCT (second least execution time) for each task and then determine sufferage value. Sufferage value is identified by finding the difference between two tasks i.e task with MCT (minimum completion time) and task with second MCT. Sufferage produces high makespan and lower throughput for larger datasets having a substantial number of larger tasks.

Task Aware Scheduling Heuristic (TASA) [24] works in two phases that are; In the first phase, TASA uses Min-Min scheduling heuristic which ultimately favors smaller jobs. In the second phase, sufferage task scheduling heuristic is used to select a suitable VM for the selected task.

IV. EXPERIMENTAL SETUP

To evaluate the performance of any scheduling heuristics for Cloud computing, three different approaches (i.e., experimental, analytical, and simulation) can be used. The experimental techniques are expensive and are difficult to set up and may require an expert to design the testbeds. Moreover, utilizing the real Cloud platform works on a pay as you use model, thus executing simulation multiple times may result in high monitory cost. Whereas, the analytical techniques are often limited in evaluating the proposed scheduling heuristics. For the last few decades, the simulation approaches are extensively used to evaluate the performance of the underlying scheduling approaches using a wide variety of configurations. To investigate the performance of the available contemporary approaches (chosen in this work), a renowned simulation platform Cloudsim [20]is utilized. The computing powers of VMs are assigned in ascending order of their VM-ID. The simulation experiments were executed on a workstation equipped with Intel Core i5-8500 Quad-core processor (3.0 GHz clock speed) and 8 GBs of main memory.

A. DATASET SELECTION

To analyze the performance of state-of-the-art heuristic algorithms, we chose the heterogeneous Computing Scheduling Platform (HCSP) dataset proposed by Braunt *et al.* [27]. This model is based on Expected Time to Compute (ETC) matrix with m number of tasks and n number of VMs. The instance with size 1024 x 32 is considered for the evaluation of the compared heuristic approaches [28]. Each dataset instance denotes 1024 tasks also called Cloudlets (tasks) and 32 virtual machines (VMs). Four different instances (i.e., c-hilo, i-hilo, c-lohi, and i-lohi) are utilized. These instances are based on task heterogeneity and resource heterogeneity as described below;

TABLE 2. HCSP dataset specification.

Dataset	c-hilo	i-hilo	c-lohi	i-lohi
Number of VMs	32	32	32	32
Number of Tasks	1024	1024	1024	1024
VM power in MIPS	900-4500	900-4500	100-7000	100-7000
Task size in MIPS (Range)	32859-17487787559	200-2704624	100-58220	20-20619

TABLE 3. VM CPU specification.

VM-IDs	VM CPU MIPS			
	c-hilo	c-lohi	i-hilo	i-lohi
0	900	100	900	100
1	920	120	900	150
2	970	140	1000	200
3	1040	170	1020	450
4	1095	320	1030	500
5	1130	350	1400	550
6	1190	490	1420	600
7	1250	500	1550	850
8	1320	510	1580	900
9	1380	800	1590	920
10	1430	1150	1750	950
11	1510	1690	1770	970
12	1560	2050	1800	1000
13	1590	2090	1950	1020
14	1640	2300	1970	1050
15	1700	2550	2150	1300
16	1770	2570	2190	1330
17	1850	2800	2210	1450
18	1950	3000	2250	1500
19	2070	3050	2350	1550
20	2250	3100	2660	1620
21	2300	3350	2720	1700
22	2370	3760	2770	1950
23	2420	3820	2810	2200
24	2540	3950	3250	2450
25	2850	4000	3270	2700
26	3350	4150	3500	3450
27	3780	4250	3800	3600
28	3850	4280	3850	3950
29	4070	5100	3970	4450
30	4220	5950	4020	5700
31	4500	7000	4500	7000

hilo: Heavy set of tasks with the light capacity of resources
lohi: Light set of tasks and high capacity of resources

The details regarding the utilized dataset is also available at [29]. In the case of hilo datasets, most of the Cloudlets are heavier and the available VM resources are lighter in terms of the available CPU power. Whereas, the lohi datasets contain Cloudlets having smaller sizes and the resources (VMs) are powerful in terms of the CPU MIPS [29]. Only a single data center is considered in the simulation that is comprised of 32 servers. Each of the server is equipped with a RAM of 2GB and CPU of power 10,000 MIPS. The VMs are provisioned resources from this available pool of the server resources. In Table 2, the details about the VMs CPU specification in MIPS (million instructions per second) and task size (required MIPS) for all of the four datasets are provided. Also, the details regarding each VMs' CPU power (in MIPS) are reported in Table 3.

B. EXPERIMENTAL EVALUATION

The effectiveness of the scheduling algorithms depends on a number of factors that are: Execution time (Makespan), the average resource utilization (ARUR), throughput, the level of SLA violation, etc. From the users' point of view, the execution time with minimum SLA violation is of primary concern. On the other hand, the service providers' focus is to obtain maximum profitability from their services. Thus to achieve this, there is a need to develop and deploy schedulers that can handle a large number of requests and efficiently utilize the resources. This work specifically targets these areas that can provide performance trade-off of different scheduling heuristics to enable the service providers for better decision making. The obtained results and their discussion is presented below.

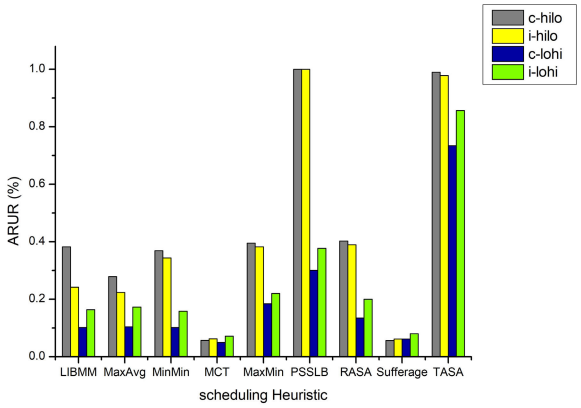


FIGURE 1. ARUR comparison.

The results in Figure 1 demonstrate the average resource utilization on all the contemporary scheduling approaches (i.e., LIBMM, MaxAvg, MaxMin, etc.) compared in this research. The obtained results reveal several important peculiarities. The PSSLB and TASA approach lead to higher ARUR (0.97 and 0.99) as compared to the other compared approaches. The TASA and PSSLB approaches have achieved 155 %, 223 %, 162 %, 10 times, 148 %, 155 %, 11 times and 157 %, 227 %, 167 %, 10 times, 150 %, 157 %, 11 times as compared to LIBMM, MaxAvg, MinMin, MCT, MaxMin, RASA, and Sufferage respectively for the c-hilo dataset. This improvement is due to the balanced task mapping over the available resources. Sufferage and MCT resulted in poor resource utilization by acheiving only 10 or less than 10 % average resource utilization. This behavior is due to the fact that MCT burdens the resources (VMs) that provide minimum completion time for the Cloudlets while slower VMs remain idle. Likewise MCT, the sufferage under-utilizes the slower VMs resulting in poor resource utilization. The attained ARUR also depends on the available tasks in the datasets. For c-hilo and i-hilo dataset, the PSSLB and TASA attained higher ARUR. However, the ARUR value

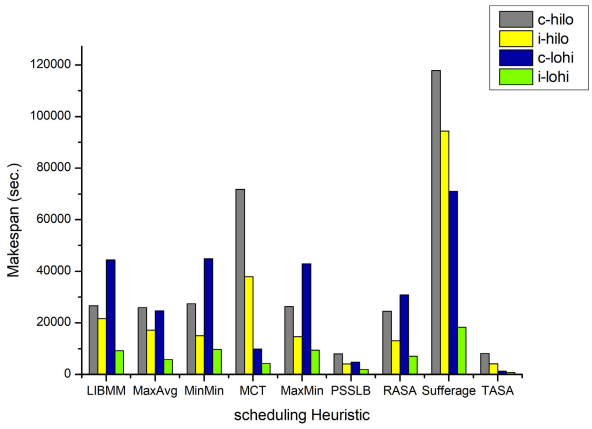


FIGURE 2. Makespan comparison.

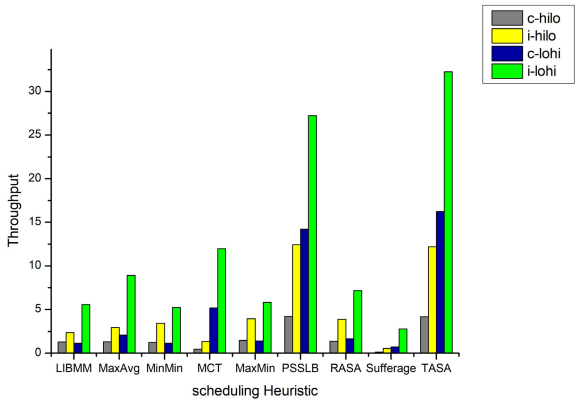


FIGURE 3. Throughput comparison.

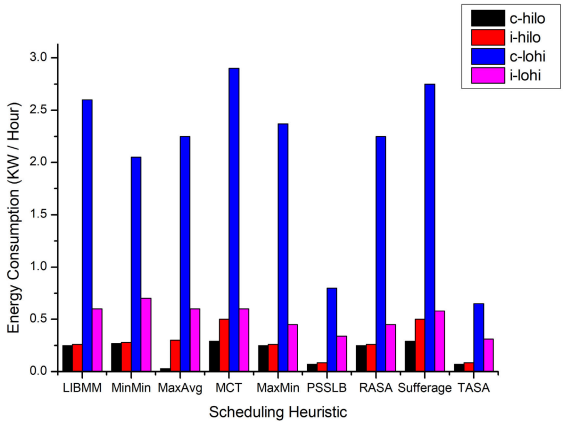


FIGURE 4. Energy consumption comparison.

is dropped (from 0.98 and 0.97 to 0.33 and 0.39 respectively) for the PSSLB on the c-lohi and i-lohi datasets. On the other hand, TASA exhibited consistent behavior for all the datasets utilized. This means that task nature also impacts the behavior of the scheduling approach used. From the given discussion, it is concluded that TASA provides more stable and scale-able performance as compared to PSSLB and other compared approaches.

The mapping of tasks on the available VMs in a proper way has a clear impact on the overall execution time for task scheduling. The more balanced the mapping of tasks is the less the overall makespan for the available Cloudlets (i.e., tasks). To obtain an insight into the performance of the available scheduling heuristics (i.e., LIBMM, MaxAvg, MaxMin etc) with respect to makespan the simulation is executed multiple times, by employing *HSPC* [28] datasets (i.e., c-hilo, c-lohi, i-hilo and i-lohi). The results regarding the attained makespan on each of the scheduling approach (utilized in this research) are reported in Figure 2. The obtained results show that all the algorithms lead to reduced makespan for i-lohi dataset. For all the reported makespan results, the worst value of makespan is observed for *sufferage* on all the available datasets. The *TASA* and *PSSLB* outperformed as compared to other approaches (utilized for comparison in this research) in terms of makespan. For the hilo dataset instances (i.e., c-hilo and i-hilo), the TASA and PSSLB shown similar performance and shown an improvement of 170 %, 160 %, 157 %, 6 times, 170 %, 140 %, 10times as compared to LIBMM, MaxAvg, MinMin, MCT, MaxMin, RASA, and Sufferage respectively.

The in-depth investigation shows that *TASA* been able to produce the best makespan (10,000 and 5000 seconds) for all the available datasets. This is due to the fact that *TASA* maps the tasks in a way that keeps all the resources busy evenly during the execution of tasks on the available resources. The obtained results assert that *TASA* can be considered as the fittest candidate for the Cloud service providers. This concludes that understanding the characteristics of tasks is important for the design and development of any scheduling heuristic.

The best case and worst case throughput results for all the compared scheduling heuristics on the available HSPC datasets are plotted in Figure 3. Higher throughput value shows better resource utilization across the available VMs. The obtained results show a substantial increase (minimum of 115 % and 160 % improvement against MCT and 9 times, 11 times improvement against sufferage) in throughput on h-lohi and i-lohi for the PSSLB and TASA of the available the contemporary scheduling approaches used in this study. The worst performance is observed for MCT and sufferage on the hilo datasets (i.e., c-hilo and i-hilo). However, higher throughput is observed for MCT on the lohi datasets as compared to all the approaches except PSSLB and TASA. In case of lohi dataset, most of the jobs are of small size. Therefore, the MCT map tasks to the faster machines thus leading to completing a higher number of tasks within shorter makespan. This means that for jobs with small size, the MCT will be able to satisfy the SLA in an ideal way. On the other hand, the PSSLB is ideal for datasets with larger size jobs. The obtained results reveal that TASA shows consistent behavior for all the datasets, achieving higher throughput for both hilo and lohi datasets.

For the last few years, the energy consumption has been considered a vital metric for evaluating the performance of any Cloud scheduling approach. The energy consumption

TABLE 4. Percentage load imbalance (VM-level) traces for c-hilo 1024 × 32.

VM ID	LIBMM	Max Avg	MaxMin	MCT	MinMin	PSSLB	Sufferage	RASA	TASA
0	1.07	0.61	1.15	1.35	1.15	0.00	1.15	1.40	0.01
1	1.33	0.14	1.43	1.38	1.43	0.00	1.38	2.16	0.00
2	3.35	1.99	3.48	1.45	3.48	0.00	3.48	1.48	0.00
3	0.79	0.64	0.92	1.56	0.92	0.00	1.56	0.23	0.01
4	1.85	1.32	1.92	1.64	1.92	0.00	1.92	3.35	0.00
5	2.62	0.92	2.73	1.69	2.73	0.02	1.69	0.56	0.01
6	1.58	0.29	1.70	1.78	1.71	0.00	1.71	2.07	0.02
7	**4.22**	0.37	1.19	1.87	1.19	0.00	**47.66**	2.12	0.04
8	0.04	1.05	0.08	1.98	0.08	0.00	0.08	0.09	0.01
9	0.67	0.25	0.77	2.07	0.77	0.00	2.07	1.64	0.03
10	1.71	1.87	1.84	2.14	1.84	0.00	1.84	0.33	0.01
11	1.25	0.43	1.30	2.26	1.30	0.00	1.30	0.59	0.01
12	1.72	0.83	1.82	2.34	1.82	0.00	1.82	0.48	0.03
13	3.21	0.39	3.30	2.38	3.30	0.00	2.38	1.78	0.00
14	1.59	1.94	1.43	2.46	1.43	0.00	2.46	0.47	0.01
15	0.25	0.75	0.08	2.55	0.08	0.00	2.55	2.39	0.01
16	0.77	2.19	0.62	2.65	0.62	0.00	0.62	0.24	0.03
17	0.01	1.32	0.09	2.77	0.09	0.00	2.77	0.81	0.00
18	0.23	0.79	0.17	2.92	0.17	0.00	0.17	1.31	0.01
19	0.68	0.21	0.56	3.10	0.56	**0.06**	0.56	0.76	0.00
20	1.74	2.26	1.59	3.16	1.59	0.00	1.59	1.30	0.02
21	0.06	1.88	0.01	3.21	0.01	0.00	0.01	0.07	0.02
22	1.53	2.16	1.45	2.97	1.45	0.00	3.55	0.14	0.02
23	1.69	1.99	1.59	2.81	1.59	0.00	1.59	1.71	0.02
24	0.87	1.01	0.97	3.37	0.97	0.00	0.97	0.16	0.02
25	0.70	0.61	0.60	3.45	0.60	0.00	4.26	2.64	0.02
26	3.09	2.02	3.02	4.11	3.02	0.00	5.02	2.17	0.01
27	3.94	1.75	**3.80**	1.88	3.80	1.20	0.00	1.60	0.00
28	3.17	6.84	3.12	1.43	3.12	0.00	5.77	3.68	0.00
29	1.86	2.69	1.81	0.60	1.81	0.03	6.10	2.94	0.03
30	3.70	**13.96**	3.63	**49.89**	3.64	0.00	6.32	**4.00**	**0.06**
31	1.21	5.67	1.22	16.58	1.22	0.00	6.74	0.77	0.05
-		-	-	-	-	-	-	-	-
Avg. load Imbalance	1.64	1.91	1.54	3.59	1.54	0.004	3.26	1.41	0.02

has been completely ignored by most of the authors while investigating and comparing their proposed task scheduling approaches performance against other state-of-the-art approaches. Thus, in this study we also attempted to analyze and compare the performance of the contemporary task scheduling approaches considered for comparison in this study. The energy model used in [30] is considered for empirical investigation. The results regarding the energy consumption for all the task scheduling approaches are reported in Figure 4. The best and worst results for all the dataset instances are shown obtained and plotted in Figure 4. Again the TASA and PSSLB approaches outperformed against the rest of the compared approaches concerning the energy consumption and lead to an average improvement of 130 %, 110 %, 125 %, 190 %, 140 %, 110 %, 197 % as compared to LIBMM, MaxAvg, MinMin, MCT, MaxMin, RASA, and Sufferage respectively. For the hilo dataset instances, the minimum energy consumption observed is 0.052 and 0.054 for the TASA and PSSLB respectively while higher energy values are reported for the MCT and sufferage approaches. The MinMin, MaxAvg, and RASA have shown moderate performance concerning the energy consumption for both the hilo and lohi datasets. For MCT and sufferage, higher energy consumption is observed for both the datasets and this is due to the reason that these approaches overload some of the faster VMs while the slower VMs remain idle. In most of the studies, it is reported that an idle machine consumes 70% of its resources even with no process running on it. Thus, for MCT and sufferage, the energy cost of these idle VMs leads to an increase in the overall energy consumption.

The effectiveness and performance of any scheduling algorithm depend 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 balance the share of the load on each of the resource the more likely is to maximize the resource utilization which ultimately leads to reduced makespan with the better response time. To obtain an insight into the performance of each scheduling algorithm, a number of experiments are performed using four sets of HCSP datasets (i.e., h-hilo, h-lohi, i-hilo, i-lohi). The obtained results pertaining to the percentage individual machine level and average load imbalance for all the compared scheduling heuristics (used in this study) are reported in Table 4. The MCT and sufferage lead to a higher level of

TABLE 5. Percentage load imbalance (VM-level) traces for c-lohi 1024 × 32.

VM ID	LIBMM	Max Avg	MaxMin	MCT	MinMin	PSSLB	RASA	Sufferage	TASA
0	4.51	1.26	4.56	0.13	4.56	**0.55**	4.21	4.56	0.02
1	7.35	4.01	7.41	0.15	7.41	0.54	3.09	0.15	0.02
2	1.32	0.54	1.40	0.18	1.40	0.55	3.21	1.40	0.03
3	2.02	1.16	2.06	0.21	2.06	0.57	2.98	0.21	0.02
4	1.66	1.36	1.73	0.40	1.73	0.39	3.43	1.73	0.03
5	3.17	1.88	3.19	0.44	3.19	0.37	0.66	0.44	0.04
6	3.89	2.77	3.91	0.62	3.91	0.20	2.36	3.91	0.03
7	2.54	0.95	2.59	0.63	2.59	0.20	6.51	**48.84**	0.03
8	2.51	1.88	2.58	0.64	2.58	0.19	2.26	2.58	0.01
9	2.32	1.45	2.39	1.01	2.39	0.04	3.75	1.01	0.02
10	0.18	1.13	0.14	1.45	0.14	0.05	2.29	0.14	0.04
11	6.53	0.98	6.56	2.13	6.56	0.08	0.28	6.56	0.03
12	0.51	1.70	0.51	2.58	0.51	0.09	1.76	0.51	0.01
13	2.35	2.21	2.40	2.63	2.40	0.10	1.13	2.63	0.01
14	0.07	1.00	0.06	2.89	0.06	0.11	0.95	2.89	0.01
15	0.57	1.37	0.59	3.21	0.59	0.12	0.17	3.21	0.01
16	0.52	2.21	0.53	3.23	0.53	0.12	1.01	0.53	0.02
17	0.16	2.33	0.12	3.52	0.12	0.13	1.01	3.52	0.01
18	0.32	1.37	0.37	3.78	0.37	0.14	1.52	0.37	0.02
19	0.89	2.97	0.91	3.84	0.91	0.14	1.69	0.91	0.03
20	1.97	2.23	1.97	3.90	1.97	0.14	1.49	1.97	0.00
21	2.45	2.97	2.43	4.05	2.43	0.16	1.24	2.43	0.00
22	2.85	4.30	2.84	4.50	2.84	0.17	1.32	4.73	0.01
23	3.22	3.96	3.24	4.53	3.24	0.18	3.17	3.24	0.03
24	1.34	2.25	1.40	4.76	1.40	0.18	3.05	1.40	0.04
25	3.65	4.63	3.70	4.63	3.70	0.19	2.94	5.03	0.01
26	0.86	2.65	0.89	4.73	0.89	0.19	2.21	5.22	0.03
27	0.88	0.95	0.92	2.40	0.92	0.20	3.51	0.92	0.01
28	3.00	0.77	3.05	1.47	3.05	0.20	2.73	5.39	0.00
29	4.15	1.45	4.64	3.87	4.24	0.24	5.20	6.42	0.05
30	6.08	**28.53**	5.74	**71.81**	6.15	0.28	**4.84**	7.49	0.03
31	**6.97**	7.23	**7.10**	0.69	**7.10**	0.32	2.27	8.81	**0.06**
-		-	-	-	-	-	-	-	-
Avg. load Imbalance	*2.52*	*3.01*	*2.56*	*4.53*	*2.56*	*0.22*	*2.45*	*4.35*	*0.02*

load imbalance (i.e., 49.89% and 47.66%), where the MCT and sufferage under-utilize the slower and faster machines respectively. LIBMM and RASA produced a moderate level of load-balanced scheduling that resulted in better resource utilization. The Min-Min and RASA results in overloading the faster VMS slightly while leading to a load imbalance on the slower VMs (i.e.,). The PSSLB and TASA produce very balanced scheduling on all the VMs with negligible load imbalance on a few VMs (i.e., vm5, vm19, and vm29 for PSSLB and vm0, vm3, vm5, etc. for TASA). This, in turn, results in high resource utilization as shown in Figure 2. From the obtained results it has reveled that PSSLB and TASA outperformed as compared to other compared approaches by producing almost a completely load-balanced schedule on the available VMs.

To examine the behavior of each scheduling algorithm, the experiments were extended and executed for c-lohi dataset. In lohi dataset, most of the tasks are of smaller size and the VMs are high processing. The simulation was then performed several times with different seed values and results were obtained for each of the scheduling approaches. The results regarding individual machine level load imbalance (percentage) and average load imbalance for all the heuristics are presented in Table 5. The best and worst-case performance achieved by each of the scheduling approaches in terms of load imbalance is shown as bold in the table. Again, the PSSLB and TASA shown consistent load-balanced mapping (i.e., on most of the VMs 0% overload/under-load and only few VMs less than 0.06% for TASA and 0.22% average load imbalance for PSSLB) of the tasks on the available resources and therefore, outperformed as compared to all of the contemporary approaches (considered for comparison in this paper). The MaxAvg and LIBMM overloaded 6 to 9% of the faster VMs whereas a slight under-utilization is observed for the rest of the VMs. Similarly, the MCT severely overloaded the faster machines while mapping zero percent of tasks to the slower VMs (i.e., VM0 to VM20). Likewise MCT, the sufferage overloaded 58% of the VMs while the remaining VMs were idle during the course of scheduling. This imbalance leads to the under-utilization of resources as can be seen in Figure 1. These results also support the use of PSSLB and TASA as good candidates for scheduling tasks with most of the tasks having a smaller size.

TABLE 6. Percentage load imbalance (VM-level) traces for i-hilo 1024 × 32.

VM ID	LIBMM	Max Avg	MaxMin	MCT	MinMin	PSSLB	RASA	Sufferage	TASA
0	3.07	0.53	3.21	1.22	3.21	**0.00**	1.51	3.21	0.01
1	1.25	0.87	1.59	1.22	1.59	0.00	2.65	1.22	0.06
2	2.69	1.17	3.00	1.35	3.00	0.05	1.56	3.00	0.02
3	1.61	0.57	1.94	1.38	1.94	0.01	2.78	1.38	0.01
4	1.62	1.29	2.00	1.39	2.00	0.03	1.60	2.00	0.00
5	0.69	0.81	0.99	1.89	0.99	0.06	1.36	1.89	0.03
6	0.51	0.21	0.76	1.92	0.79	0.00	0.68	0.79	0.06
7	8.90	0.87	0.22	2.10	0.22	0.00	1.54	**45.94**	0.02
8	0.79	0.87	1.12	2.14	1.12	0.00	0.01	1.12	0.06
9	0.17	0.80	0.20	2.15	0.20	0.00	0.97	2.15	0.05
10	0.70	1.07	1.02	2.37	1.02	0.00	1.04	1.02	0.01
11	0.76	0.68	1.08	2.40	1.08	0.00	0.66	1.08	0.03
12	0.86	1.88	0.46	2.44	0.46	0.00	0.50	0.46	0.06
13	0.76	1.58	0.32	2.64	0.32	0.00	0.00	2.64	0.02
14	1.21	1.05	1.26	2.67	1.26	0.00	0.49	2.67	0.05
15	0.61	0.89	0.16	2.91	0.16	0.00	0.66	2.91	0.07
16	0.73	1.46	0.42	2.96	0.42	0.00	0.33	0.42	0.04
17	0.62	1.42	0.25	2.85	0.25	0.00	0.24	2.99	0.04
18	0.22	1.41	0.10	2.96	0.10	0.09	0.17	0.10	0.03
19	0.60	1.67	0.31	3.10	0.31	0.02	0.17	0.31	0.05
20	0.68	1.92	0.37	3.44	0.37	0.02	0.14	0.37	0.00
21	0.84	1.80	0.56	3.28	0.56	0.10	1.33	0.56	0.02
22	0.15	1.31	0.28	3.07	0.28	0.00	0.72	3.75	0.02
23	0.61	1.00	0.44	2.60	0.44	0.00	0.87	0.44	0.06
24	2.45	1.84	2.17	3.73	2.17	0.00	0.86	2.17	0.07
25	1.39	1.89	1.09	2.73	1.09	0.00	1.34	4.42	0.07
26	2.59	1.63	2.22	3.61	2.22	0.00	1.28	4.74	0.01
27	0.85	0.04	0.66	0.29	0.66	0.00	2.71	0.66	0.09
28	2.77	5.95	2.56	3.35	2.56	0.00	0.96	5.21	0.02
29	1.77	6.27	1.63	2.53	1.63	0.00	1.54	5.37	0.02
30	2.65	**17.35**	2.45	**44.68**	2.47	0.00	**2.83**	5.44	0.10
31	**2.78**	4.87	**2.69**	15.67	**2.69**	0.00	3.17	6.09	**0.02**
-		-	-	-	-	-	-	-	-
Avg. load Imbalance	*1.50*	*2.09*	*1.17*	*4.16*	*1.17*	*0.01*	*1.15*	*3.64*	*0.04*

To examine the performance of the available contemporary scheduling heuristics, another set of experiments was performed on i-hilo and i-lohi datasets with tasks having inconsistent heterogeneity. Similarly, the VMs utilized for the experiments are also inconsistent in terms of heterogeneity. Table 6 presents the best and worst-case results of individual VM level load imbalance and average load imbalance for all the compared scheduling heuristics. The PSSLB and TASA achieved almost balanced task distribution on all the available VMs. For PSSLB, 75% of the VMs got exactly the same share of workload as provided in Eq 5 that shows the distribution of tasks equally among all the VMs. On the other hand, TASA leads to an impressive average load balance scheduling (96%) of the tasks on the available resources. The Max-Avg slightly overloads some of the slower and faster VMs. However, it leads to an improvement of 57% in terms of average load imbalance. The worst-case load distribution is observed for MCT and sufferage. MCT overloads the faster VMs and sufferage overloads 33% and under-utilizes 24% of the VMs whereas the remaining VMs are idle during the course of workload execution. The LIBMM and RASA have shown a moderate load-balanced task distribution among the available resources (i.e., VMs). The obtained results advocate the use of PSSLB and TASA as top candidates for scheduling jobs across the Cloud data centers.

The final set of simulation experiments is executed using i-lohi dataset to empirically investigate the performance of the compared approaches. The results in terms of ARUR, Makespan, Throughput, and load imbalance are obtained where the results concerning the first three parameters are plotted in Figure 1, Figure 2, and Figure 3 respectively and discussed accordingly. Table 7 reports the results to appertain to the individual machine load imbalance and the average load imbalance. PSSLB and TASA have shown superior performance as compared to other state-of-the-art compared approaches (used in this study) by producing almost a complete load-balanced scheduling of tasks on the available resources. MCT and sufferage are unable to map the tasks on the available VMS in a load-balanced way thus resulted in a higher level of load imbalance. LIBMM, Maxmin, MinMin, and RASA resulted in a moderate level of load distribution.

All these results assert that PSSLB and TASA remarkable performance against all the compared contemporary approaches for all the datasets showing consistent performance

TABLE 7. Percentage load imbalance (VM-level) traces for i-lohi 1024 × 32.

VM ID	LIBMM	Max Avg	MaxMin	MCT	MinMin	PSSLB	RASA	Sufferage	TASA
0	2.89	0.33	3.07	0.17	3.07	**0.45**	1.83	3.07	0.04
1	3.05	2.34	3.26	0.26	3.26	0.38	3.28	0.26	0.04
2	4.01	3.46	4.17	0.34	4.17	0.30	3.10	4.17	0.03
3	2.82	0.09	2.97	0.77	2.97	0.01	2.86	0.77	0.02
4	2.31	2.00	2.40	0.86	2.40	0.01	2.50	2.40	0.04
5	4.01	3.45	4.13	0.95	4.13	0.01	6.81	0.95	0.01
6	0.45	0.17	0.61	1.03	0.61	0.01	2.03	0.61	0.03
7	3.08	0.73	0.09	1.46	0.09	0.02	2.35	**48.47**	0.03
8	1.83	0.74	1.99	1.55	1.99	0.02	2.74	1.99	0.00
9	1.02	0.40	1.13	1.58	1.13	0.02	2.65	1.58	0.01
10	0.92	1.00	1.11	1.63	1.11	0.02	0.70	1.11	0.02
11	0.86	0.19	0.96	1.67	0.96	0.02	0.11	0.96	0.02
12	0.35	0.75	0.49	1.72	0.49	0.02	1.03	0.49	0.03
13	0.93	0.42	1.03	1.76	1.03	0.02	0.82	1.76	0.01
14	4.45	0.67	4.57	1.81	4.57	0.02	0.03	1.81	0.01
15	0.62	0.75	0.49	2.24	0.49	0.03	0.13	2.24	0.02
16	0.18	1.42	0.30	2.29	0.30	0.03	0.86	0.30	0.02
17	0.89	1.57	0.99	2.22	0.99	0.03	2.43	2.50	0.01
18	2.28	1.95	2.35	2.56	2.35	0.03	0.02	2.35	0.03
19	0.81	0.88	0.91	2.65	0.91	0.03	0.72	0.91	0.02
20	0.87	1.60	0.78	2.78	0.78	0.03	1.20	0.78	0.03
21	0.13	1.65	0.07	2.77	0.07	0.03	0.62	0.07	0.01
22	1.05	1.49	1.03	3.05	1.03	0.04	1.89	3.36	0.02
23	1.53	1.59	1.42	3.23	1.42	0.04	1.87	1.42	0.00
24	2.41	2.52	2.27	3.75	2.27	0.05	2.09	2.27	0.01
25	2.35	2.20	2.24	3.58	2.24	0.05	2.83	4.64	0.01
26	3.41	2.27	3.35	4.15	3.35	0.07	2.24	5.94	0.03
27	0.22	1.75	0.19	1.48	0.19	0.07	1.24	0.19	0.01
28	4.17	6.27	4.15	0.50	4.15	0.08	4.29	6.80	0.06
29	2.50	0.55	2.50	2.70	2.50	0.09	5.18	7.66	0.07
30	8.20	**13.52**	8.23	**55.32**	8.24	0.11	**8.55**	9.81	0.07
31	**9.68**	11.12	**9.80**	2.21	**9.80**	0.14	5.18	12.05	**0.08**
-		-	-	-	-	-	-	-	-
Avg. load Imbalance	*1.64*	*1.91*	*1.54*	*3.59*	*1.54*	*0.00*	*3.26*	*1.41*	*0.02*

in terms of ARUR, Makespan, Throughput, and load imbalance. More in-depth analysis explicates that TASA is the leading algorithm and is considered to be the most appropriate choice for a variety of tasks whether small size or large and for datasets either consistent heterogeneous or inconsistent heterogeneous.

V. RESULT DISCUSSION AND RECOMMENDATIONS

Following the experimental performance evaluation, the result discussion is delineated to extract important findings regarding the scheduling approaches and the nature of datasets. The behavior of a number of scheduling approaches (used in this study) was empirically investigated using a diverse set of large size HCSP heterogeneous datasets. Among the compared scheduling heuristics, the PSSLB and TASA have shown substantial performance with respect to ARUR, Makespan, and throughput. Moreover, these scheduling approaches lead to map the resources in a completely balanced way resulting in 97 to 99% resource utilization. Among PSSLB and TASA, the TASA has shown consistent performance for all the datasets; however, the performance in terms of attained ARUR of PSSLB dropped to 30% and 38% for the c-lohi and i-lohi datasets respectively. RASA has been able to perform well in terms of ARUR for the c-hilo dataset as compared to other scheduling heuristics except for PSSLB and TASA. MCT and sufferage lead to poor resource utilization due to the fact that MCT overloads faster VMs while slower VMs remain idle. Similarly, sufferage like MCT under-utilized some of the slower machines resulting in poor resource utilization causing high Makespan. The under-utilization of the resources also results in an increase in the overall energy consumption. LIBMM and MaxAvg overloaded 6 to 9% of the faster VMs whereas a slight under-utilization is observed for the rest of the VMs. The TASA and PSSLB has been able to dramatically reduce Makespan as well as the energy consumption and also resulted in almost completely load-balanced scheduling of the tasks over the available resources. The TASA approach has shown consistent performance for the diverse nature of utilized dataset instance and thus outperformed all the compared task scheduling approaches concerning the utilized performance metrics.

From the Cloud service providers point of view, the resource utilization is considered as a more crucial

parameter for the selection of scheduling approach. Moreover, poor resource utilization may lead to a load imbalance with high energy consumption. The problem concerning poor resource utilization needs to be addressed in a careful manner. For efficient resource utilization, the task should be mapped to the available resources keeping in view the available computing resources, workload properties, and already mapped tasks to the resources. The resource and task aware mapping of cloudlets on the available VMs in a load-balanced manner will lead to eliminating the problem of under-utilized/overloaded VMs ultimately resulting in reducing higher energy costs.

The obtained results reveal several important facts about the available contemporary scheduling approaches. With similar ARUR values, different Scheduling approaches employ different workload distribution on the available resources. This behavior further affects the Makespan and Throughput achieved. LIBMM and RASA have attained 0.39 and 0.41 ARUR (5% improvement) respectively using the c-hilo dataset; however, the average load imbalance faced by LIBMM and RASA is 1.64 and 1.41 respectively (14.5% improvement). This means that achieving the same ARUR, does not mean that load imbalance can be the same. Similarly, the same trend of different load imbalance produced by the scheduling heuristics having almost similar results for ARUR is observed in the simulation experiments done using HCSP datasets. In summary, TASA scheduling heuristic outperformed and shown scale-able performance as compared to the available approaches for all the utilized HCSP datasets. After careful empirical investigations, this research recommends the following important points for the Cloud service providers and researchers working in the task scheduling area.

The execution time (Makespan) and resource utilization are affected by the level of load-balanced workload distribution among the available resources. The load-balanced mapping of tasks is possible only and only if the nature of tasks and resources is to understand properly. TASA, a task aware scheduling algorithm endorsed this fact by attained almost balanced load scheduling for all the datasets.

To distinguish between different heuristic approaches, a number of datasets with diverse nature should be used to evaluate their performance.

Moreover, Cloud service providers can expand their revenue-generating with balanced energy consumption in the Cloud data centers by load-balanced workload scheduling, smaller Maksespan, and efficient resource utilization.

The reduction in execution time can also lead to satisfying Cloud service users.

VI. CONCLUSIONS AND FUTURE WORK

This research work evaluated several renowned state-of-the-art approaches rigorously by employing one of the most prominent benchmark dataset instances(i.e., HCSP) and considered a widely used simulation platform namely Cloudsim for evaluation comparison. The workload used for the experiments has been configured with different heterogeneity of the tasks and VMs. Among the available compared approaches, PSSLB and TASA have been able to attain the highest resource utilization with minimal makespan for HCSP c-hilo and i-hilo datasets. It was shown that the ARUR of PSSLB reduced to 30% and 38% (from 99% for c-hilo and i-hilo datasets) for the c-lohi and i-lohi datasets respectively. On the other hand, it was demonstrated that TASA scheduling heuristic outperformed and scale-able performance as compared to the compared approaches for all the utilized HCSP datasets. Moreover, the TASA and PSSLB also resulted in a very low energy consumption as compared to the rest of the approaches. MCT and sufferage are unable to map the tasks on the available VMS in a load-balanced way thus resulted in a higher level of load imbalance resulting in an excessive energy cost. LIBMM, Maxmin, MinMin, and RASA resulted in a moderate level of load distribution. The TASA and PSSLB have shown substantial performance concerning the energy consumption for all the dataset instances.

As a future, we plan to compare the available contemporary scheduling approaches with respect to SLA violation. The future work also aims to propose a task cum resource-aware scheduling approach that will exploit the nature of the presented workload and efficiently map the tasks on the available Cloud resources.

ACKNOWLEDGMENT

The authors would like to thanks the Deanship of Scientific Research at Umm Al-Qura University for supporting this work by Grant code 19-COM-1-01-0015.

REFERENCES

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

[2] M. A. Iqbal, M. Aleem, M. A. Islam, M. Ibrahim, and S. Anwar, "Amazon cloud computing platform EC2 and VANET simulations," *Int. J. Ad Hoc Ubiquitous Comput.*, vol. 14, no. 3, pp. 127–136, 2018.

[3] A. Beloglazov and 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.*, vol. 24, no. 7, pp. 1366–1379, Jul. 2013.

[4] C. S. Yeo and R. Buyya, "Service level agreement based allocation of cluster resources: Handling penalty to enhance utility," in *Proc. IEEE Int. Conf. Cluster Comput.*, Sep. 2005, pp. 1–10.

[5] F. Durao, J. F. S. Carvalho, A. Fonseka, and V. C. Garcia, "A systematic review on cloud computing," *J. Supercomput.*, vol. 68, no. 3, pp. 1321–1346, Jun. 2014.

[6] S. Groot, "Research on efficient resource utilization in data intensive distributed systems," Ph.D. dissertation, Dept. Comput. Sci., Univ. Tokyo, Tokyo, Japan, 2013.

[7] S. K. Panda and P. K. Jana, "SLA-based task scheduling algorithms for heterogeneous multi-cloud environment," *J. Supercomput.*, vol. 73, no. 6, pp. 2730–2762, Jun. 2017.

[8] H. Jin, W. Gao, S. Wu, X. Shi, X. Wu, and F. Zhou, "Optimizing the live migration of virtual machine by CPU scheduling," *J. Netw. Comput. Appl.*, vol. 34, no. 4, pp. 1088–1096, Jul. 2011.

[9] V. M. Sivagami and K. S. Easwarakumar, "An improved dynamic fault tolerant management algorithm during VM migration in cloud data center," *Future Gener. Comput. Syst.*, vol. 98, pp. 35–43, Sep. 2019.

[10] H. Chen, F. Wang, N. Helian, and G. Akanmu, "User-priority guided min-min scheduling algorithm for load balancing in cloud computing," in *Proc. Nat. Conf. Parallel Comput. Technol. (PARCOMPTECH)*, Feb. 2013, pp. 1–8.

[11] S. Rehman, N. Javaid, S. Rasheed, K. Hassan, F. Zafar, and M. Naeem, "Min-min scheduling algorithm for efficient resource distribution using cloud and fog in smart buildings," in *Proc. Int. Conf. Broadband Wireless Comput., Commun. Appl.* Cham, Switzerland: Springer, 2018, pp. 15–27.

[12] N. Alaei and F. Safi-Esfahani, "RePro-active: A reactive–proactive scheduling method based on simulation in cloud computing," *J. Supercomput.*, vol. 74, no. 2, pp. 801–829, Feb. 2018.

[13] J. Y. Maipan-Uku, A. Muhammed, A. Abdullah, and M. Hussin, "Max-average: An extended max-min scheduling algorithm for grid computing environtment," *J. Telecommun., Electron. Comput. Eng.*, vol. 8, no. 6, pp. 43–47, Sep. 2016.

[14] A. Arunarani, D. Manjula, and V. Sugumaran, "Task scheduling techniques in cloud computing: A literature survey," *Future Gener. Comput. Syst.*, vol. 91, pp. 407–415, Feb. 2019.

[15] N. A. Mehdi, A. Mamat, A. Amer, and Z. T. Abdul-Mehdi, "Minimum completion time for power-aware scheduling in cloud computing," in *Proc. Develop. E-Syst. Eng.*, 2011, pp. 484–489, doi: 10.1109/dese.2011.30.

[16] Y. Mao, X. Chen, and X. Li, "Max–min task scheduling algorithm for load balance in cloud computing," in *Proc. Int. Conf. Comput. Sci. Inf. Technol.* New Delhi, India: Springer, 2014, pp. 457–465.

[17] A. Hussain, M. Aleem, A. Khan, M. A. Iqbal, and M. A. Islam, "RALBA: A computation-aware load balancing scheduler for cloud computing," *Cluster Comput.*, vol. 21, no. 3, pp. 1667–1680, Sep. 2018.

[18] S. Parsa and R. M. Entezari-Maleki, "RASA: A new grid task scheduling algorithm," *Int. J. Digit. Content Technol. Appl.*, vol. 3, no. 4, pp. 152–160, 2009.

[19] Y. Wang, J.-T. Zhou, Y. Jiao, and X. Song, "Comparative analysis of evolutionary algorithms based on swarm intelligence for QoS optimization of cloud services," in *Proc. IEEE 23rd Int. Conf. Comput. Supported Cooperat. Work Design (CSCWD)*, May 2019, pp. 434–439.

[20] A. Hussain, M. Aleem, M. A. Iqbal, and M. A. Islam, "Investigation of cloud scheduling algorithms for resource utilization using CloudSim," *Comput. Informat.*, vol. 38, no. 3, pp. 525–554, 2019.

[21] B. Li, Y. Pei, H. Wu, and B. Shen, "Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds," *J. Supercomput.*, vol. 71, no. 8, pp. 3009–3036, Aug. 2015.

[22] A. Hussain, M. Aleem, M. A. Islam, and M. A. Iqbal, "A rigorous evaluation of state-of-the-art scheduling algorithms for cloud computing," *IEEE Access*, vol. 6, pp. 75033–75047, 2018.

[23] E. K. Tabak, B. B. Cambazoglu, and C. Aykanat, "Improving the performance of IndependentTask assignment heuristics MinMin,MaxMin and sufferage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 5, pp. 1244–1256, May 2014.

[24] D. S. Taherian and V. K. Bardsiri, "TASA: A new task scheduling algorithm in cloud computing," *J. Adv. Comput. Eng. Technol.*, vol. 1, no. 4, pp. 25–32, 2015.

[25] S. H. H. Madni, M. S. A. Latiff, M. Abdullahi, S. M. Abdulhamid, and M. J. Usman, "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment," *PLoS ONE*, vol. 12, no. 5, May 2017, Art. no. e0176321.

[26] A. Hussain, M. Aleem, M. A. Iqbal, and M. A. Islam, "SLA-RALBA: Cost-efficient and resource-aware load balancing algorithm for cloud computing," *J. Supercomput.*, vol. 75, pp. 6777–6803, Jun. 2019.

[27] T. D. Braunt, H. J. Siegel, N. Beck, L. L. Boloni, M. Maheswarans, A. I. Reuthert, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgeno, and R. F. Freund, "A comparison study of eleven static heuristics for mapping a class of independent tasks onto ileterogeneous distributed computing systems," Purdue Univ., West Lafayette, IN, USA, ECE Tech. Rep. 19, 2000.

[28] *HCSP Dataset*. Accessed: Mar. 20, 2020. [Online]. Available: https://github.com/chgogos/hcsp/tree/master/1024x32

[29] *HCSP Dataset VM Specification*. Accessed: Apr. 26, 2020. [Online]. Available: http://dx.doi.org/10.21227/px5b-b729

[30] A. Marahatta, Y. Wang, F. Zhang, A. K. Sangaiah, S. K. S. Tyagi, and Z. Liu, "Energy-aware fault-tolerant dynamic task scheduling scheme for virtualized cloud data centers," *Mobile Netw. Appl.*, vol. 24, no. 3, pp. 1063–1077, 2019.

MUHAMMAD IBRAHIM received the Ph.D. degree in computer science from the Capital University of Science and Technology, Islamabad, in 2019. He is currently a Lecturer with the Department of Computer Science, Virtual University of Pakistan, Rawalpindi Campus. He has published a number of articles in top rank journals, such as *Cluster Computing* and IEEE Access, and several conference papers. His research interests include cloud computing, fog computing, the IoT, and blockchain.

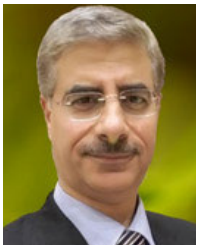
SAID NABI is currently pursuing the Ph.D. degree in computer sciences with the Capital University of Science and Technology (CUST). He has been serving as an Instructor of IT and computer science with the Department of Computer Sciences, Virtual University of Pakistan, Rawalpindi Campus. His research interests include cloud computing (cloud jobs/applications and resource scheduling, cloud load-balancing, optimization, SLA, and Quality of Services (QoS) aware cloud scheduling).

ABDULLAH BAZ (Senior Member, IEEE) received the B.Sc. degree in electrical and computer engineering from UQU, in 2002, the M.Sc. degree in electrical and computer engineering from KAU, in 2007, and the M.Sc. degree in communication and signal processing and the Ph.D. degree in computer system design from Newcastle University, in 2009 and 2014, respectively. He was the Vice-Dean and the Dean of the Deanship of Scientific Research with UQU, from 2014 to 2020. He is currently an Assistant Professor with the Computer Engineering Department, the Vice-Dean of DFMEA, the General Director of the Decision Support Center, and the Consultant of the University Vice Chancellor, UQU. His research interests include data science, ML, AI, VLSI design, EDA/CAD tools, coding and modulation schemes, image and vision computing, and computer system and architecture. Since 2015, he has been serving as a Review Committee Member of the IEEE International Symposium on Circuits and Systems (ISCAS) and a member of the Technical Committee of the IEEE VLSI Systems and Applications. He has served as a Reviewer in a number of journals, including the IEEE Internet of Things, *IET Computer Vision*, *Artificial Intelligence Review*, and *IET Circuits, Devices and Systems*.

HOSAM ALHAKAMI received the B.Sc. degree in computer science from King Abdulaziz University, Saudi Arabia, in 2004, the M.Sc. degree in internet software systems from Birmingham University, Birmingham, U.K., in 2009, and the Ph.D. degree in software engineering from De Montfort University, in 2015. From 2004 to 2007, he has worked with Software Development Industry, where he implemented several systems and solutions for a national academic institution. His research interests include algorithms, semantic web, and optimization techniques. He focuses on enhancing real-world matching systems using machine learning and data analytics in a context of supporting decision-making.

MUHAMMAD SUMMAIR RAZA received the Ph.D. degree from the National University of Science and Technology (NUST), in 2018. He is currently working as an Assistant Professor with the Computer Science Department. He has published various research papers in national/international level journals and conferences.

ALTAF HUSSAIN received the B.S. degree (Hons.) in computer science from NWFP AUP, Pakistan, and the M.S. degree in computer software engineering from the National University Science and Technology, Islamabad, Pakistan, in 2010. He is currently pursuing the Ph.D. degree with the Capital University of Science and Technology, Islamabad. He is working as the Head of Department with the Department of Computer Science, KICSIT Campus, IST, Islamabad, Pakistan. His research interests include software testing, data mining and distributing computing comprises performance analysis, and cloud computing.



KHALED SALAH (Senior Member, IEEE) is currently a Professor with the ECE Department, Khalifa University of Science, Technology, and Research (KUSTAR). He has over 170 publications in areas related to the IoT, blockchain, cybersecurity, and cloud computing. He is on editorial boards of a number of WOS-listed journals, including *IET Networks*, *IET Communications*, Wiley's SCN, Wiley's IJNM, Elsevier's JNCA, J.UCS, and AJSE.

KARIM DJEMAME (Member, IEEE) received the Ph.D. degree from the University of Glasgow, U.K., in 1999. He is currently a Professor with the School of Computing, University of Leeds. He sits on a number of international programme committees for cloud middleware, computer networks, and performance evaluation. He was the investigator of various e-science/cloud projects, including DAME, BROADEN, AssessGrid, ISQoS, STRAPP, OPTIMIS, and ASCETiC. He is currently involved in various research projects, including TANGO. His main research interests include grid/cloud computing, system architectures, resource management, and energy efficiency. He is a member of the BCS.