Evaluating performance of the Non-linear data structure for job queuing in the Cloud Environment

Sampa Sahoo, Sambit Kumar Mishra, Devang Swami, Md Akram Khan, and Bibhudatta Sahoo
Department of Computer Science & Engineering
National Institute of Technology, Rourkela
Rourkela, Odisha
Email: (sampaa2004, skmishra.nitrkl, swamx.mi, akram.sgsits, bibhudatta.sahoo)@gmail.com

***Abstract*—Cloud Computing era comes with the advancement of technologies in the fields of processing, storage, bandwidth network access, security of the internet, etc. Several advantages of Cloud Computing include scalability, high computing power, on-demand resource access, high availability, etc. One of the biggest challenges faced by Cloud provider is to schedule incoming jobs to virtual machines(VMs) such that certain constraints satisfied. The development of automatic applications, smart devices, and applications, sensor-based applications need large data storage and computing resources and need output within a particular time limit. Many works have been proposed and commented on various data structures and allocation policies for a real-time job on the cloud. Most of these technologies use a queue-based mapping of tasks to VMs. This work presents a novel, min-heap based VM allocation (MHVA) designed for real-time jobs. The proposed MHVA is compared with a queue based random allocation taking performance metrics makespan and energy consumption. Simulations are performed for different scenarios varying the number of tasks and VMs. The simulation results show that MHVA is significantly better than the random algorithm.**

***Keywords*—** Cloud Computing, Energy Consumption, Makespan, Min-heap, Virtualization, VM.

I. Introduction

Cloud computing is a model that must enchanter following three needs: Dynamism, Abstraction and Resource Sharing. For the same, it provides both hardware and software computing resources (e.g. Storage, networks, and processing power) as a service in an on-demand pay-per-use basis over the internet. Usually, a cloud is a three layered stack, bottom most being infrastructure layer, followed by platform layer and finally application layer. The bottom most layer consists of physical resources like routers, switches, power, physical servers, cooling systems, and likes. The platform layer consists of VM containers, and each of them may have same or different operating systems and/or frameworks. Application layer hosts the service that a user may demand such as email, HTTP, SSH, database services and others. A software system as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) make up the service model of cloud [1, 2]. Here, we need to map application tasks or jobs to appropriate VM so that the overall cost of jobs reduces along with processing time. Hence, a major challenge in cloud computing is energy management while guaranteeing user demanded QoS [3]. Increasing energy demands due to an exponential growth of cloud users require to be addressed as energy today means increased carbon footprints [4]. Scheduling (Resource Allocation) is a process in the cloud meant for even distribution of tasks to resources to guarantee user demanded QoS and using the least amount of resources. Now, it is a well-established fact that energy consumption depends on system load. A proper data structure can benefit scheduling to reduce computation time, do a uniform load distribution among VMs and ultimately reducing energy consumption.

In cloud computing, virtual machines (VMs) are made available to the user on lease basis over the Internet. A fixed amount is charged setting aside the VM utilization during the rental period. As the cloud consists of a finite pool of virtualized resources, it requires a scheduling policy that can guarantee maximum use of the resources [5]. The scheduling mechanism may consider various Quality of Services (QoS) metrics like the deadline, low energy consumption, minimum response time, etc. The organization of tasks has a significant impact on scheduling outcome, so an efficient data structure for task organization can lead to a useful scheduling algorithm. Mostly queue data structure is used for job organization whose most of the operations performed in O(n) time (n is the number of elements). So, a data structure whose running time is less than O(n) is preferable. Cloud computing is gaining lots of interest in several domains such as disaster management, healthcare, military [6, 7, 8]. Still, there is a scope to contribute security solutions to protect data in cloud and intermediate edge data center [6]. In the current era, data turns to a new term as big data, which need a unique solution to support 4V's properties to process big data in cloud [7, 8].

The main contributions of the paper are as follows:

1) Propose a min-heap based scheduling algorithm to execute deadline constraint task in the cloud.
2) Demonstrate the applicability of the algorithm with the help of an example and simulation result. The performance metrics used for the evaluation of the algorithm are energy consumption and makespan.

The given work is organized as follows: Section 2 discusses literature that focused on this problem and problem definition. The next section describes scheduling model. Section 4 consists of our proposed algorithm followed by an example. Section 5 shows the simulation results, and finally, the conclusion

is presented in Section 6.

II. Related Work

Scheduling is a colossal issue in the development of cloud system whose primary job is mapping jobs to VMs so that they have a shorter response time, a boost in the utilization of the system following some constraints such as cost, deadline, energy consumption, etc. In [9] researchers proposed a scheduling algorithm based on min-heap i.e. (MHSA) for both homogeneous and heterogeneous distributed environments. The primary goal of the algorithm is to provide efficient scheduling of tasks in both static (i.e. Known task demand) and dynamic (i.e. Varying task demand) environment using migration and rescheduling wherever applicable. Resource allocation problem in an IaaS cloud for the real-time task is presented as a constrained optimization problem [10]. The proposed polynomial time scheduling algorithm with cost performance metric compared with optimal and EDF solution [11]. A comparative study of numerous scheduling algorithms like Round-Robin, Random Resource Selection, Opportunistic Load Balancing, and Minimum Completion Time presented for time bound task in a cloud environment. The performance of the system is measured by throughput, makespan, and total execution cost metrics [12]. The proposed auto-scaling approach dynamically allocate/deallocate tasks to the most cost-efficient VM such that all tasks meet their deadlines. Researchers used workflow DAG as scheduling input, exerted genetic algorithm and backtracking to schedule deadline constrained tasks [13]. Cloud computing state-of-the-art techniques and different energy efficient processes discussed for IaaS cloud [14]. Slack and migration function based scheduling algorithm suggested for aperiodic and periodic task to meet their deadline constraint in a cloud environment. Separate queues used for both the tasks and lower bound on the number of resources defined based on application type, computation, and communication cost [15].

An analysis of resource scheduling in cloud computing discussed that includes taxonomy, challenges, and future research directions [4]. Greedy task scheduling based on most-efficient-server-first compared with a random scheme taking energy consumption as a performance metric. Here the scheduling problem is formulated as an integer programming optimization problem [16].

An optimal level of utilization of VM based scheduling algorithm proposed to reduce energy consumption. Here VM migration is eliminated by taking VM utilization level from unequal level to optimal level to reduce energy consumption in the cloud [3]. This paper represented resource allocation as an optimization problem with minimum CO_2 emission and maximum profit constraint and offered a dynamic voltage scaling based solution [17]. The authors have comprehensively and comparatively studied software and hardware based energy efficient schemes in cloud computing [18]. The survey study highlights the key concepts, architectural principles, state-of-the-art implementation, and research directions of cloud computing [19].

A. Problem Definition

Consider a list of jobs $J_i \in J$ with arrival time (T_i), a time limit (D_i) in seconds and load (L_i) in millions of instructions. Suppose there are m VMs in the system represented as V. Each VM $V_j \in V$ can provide instruction processing at an average rate of S_i (in MIPS). A VM may be in an active state where it consumes $\alpha_j = 10^{(-8)} \times (S_i)^2$ J/MI of energy while it reduced by 40%, i.e. $\beta_j = 0.6 \times \alpha_j$ J/MI when VM is idle. The task with the earliest deadline is always allocated to VM with higher processing speed. For the laid assumption we need to assign a set of jobs J to set of virtual machines V with the objective of minimizing overall energy consumption, maximize makespan and still meeting the target time. We assume that the incoming jobs are stored in min-heap.

III. Scheduling Model

A cloud data center consists of a vast number of VMs created by virtualization technique. A task (real-time/non-real-time) request coming to the cloud data center is assigned to an appropriate VM. The real-time task is characterized by arrival time(T_i), a time limit(D_i) and load(L_i). Its instruction processing rate (S_i) classifies each VM. The primary requirement of real-time task processing in the cloud is to choose a VM that will finish the task before its time limit. An effective data structure for task organization will be useful to speed up the scheduling.

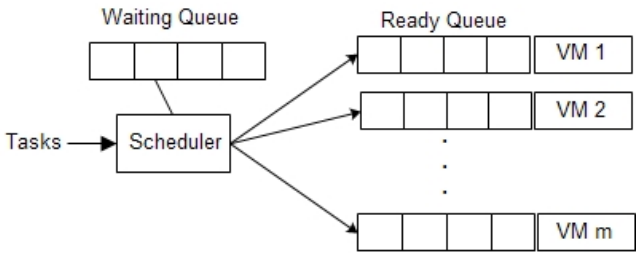

Fig. 1. scheduling model

Fig. 1 demonstrates the scheduling model used in this work. The scheduler takes incoming job requests stored in a waiting queue. The scheduler takes a set of tasks that are less than or equal to some threshold, i.e. window size and schedule them. VM assignment begins with the creation of a min-heap based on the deadline of the number of tasks equal to the window size. Initially, the fastest VM becomes the root node of the min-heap. The allocation is done based on the following principle: *assign fastest VM to the earliest deadline*. Once a task is assigned to appropriate VM, the VM min-heap is heapified after updating selected VM completion time, and min-heap for jobs is heapified again after removing the allocated task. This course continues until there are no more jobs in the waiting queue.

IV. Scheduling Algorithm

The proposed scheduling algorithm use the min-heap data structure for job assignment of incoming user requests to

TABLE I
VIRTUAL MACHINE DETAILS LIST

VM (V_i)	Processing Speed (S_i)
V_1	1500
V_2	1200
V_3	1000

TABLE II
JOB DETAILS LIST

Jobs	T_i	D_i	L_i
J_1	0	30	27000
J_2	0	53	45000
J_3	0	23	18000
J_4	0	27	22500
J_5	0	60	36000
J_6	0	38	18000

suitable VM with an objective of effective job scheduling and decreasing the amount of energy consumed. The reason behind the use of min-heap is that it performs most operations in amortized time complexity in O(log n) whereas most linear data structure does operations in O(1) or O(n) where n is the number of nodes. Algorithm 1 describes the process used to calculate expected time to compute in a VM.

Algorithm 1 Generate ETC Matrix

Input: $J \leftarrow J_1, J_2, ..., J_n$:List of jobs with their deadlines
$L \leftarrow L_1, L_2, ..., L_n$:List of job loads
$V \leftarrow V_1, V_2, ..., V_m$:List of VM's
$S \leftarrow S_1, S_2, ..., S_m$:Processing speed of VM in MIPS
$CT(j,k) \leftarrow 0, 0, 0, ..., 0$:Expected computation time matrix
Output: ETC matrix CT
1: **for** $j = 1$ **to** n **do**
2: **for** $k = 1$ **to** m **do**
3: $CT(j,k) = \frac{L_j}{S_k}$
4: $k = k + 1$
5: **end for**
6: $j = j + 1$
7: **end for**

Algorithm 2 depicts the process to schedule jobs to appropriate VM. Lets we have 3 VMs having different computation speed as shown in Table I. Say the system is in the idle state before the arrival of jobs/tasks as shown in Table II. For simplicity of explanation, we have assumed that all requests arrive at the same time with different time limits and load.

We have applied Algorithm 1 for data of Table I and Table II, thus obtained ETC matrix as shown in Table III.

Let us solve the following allocation problem using linear queue: Here we assume that jobs are linearly queued as per increasing order of their deadlines. So for given dataset, a linear queue would look as follows: $J_3, J_4, J_1, J_6, J_2, J_5$ where $W = 6$. We are using a circular queue to store the list of VM based on decreasing order of their processing speed initially and then by computation time. The VM queue would look as V_1, V_2, V_3. By following the linear queue policy jobs

Algorithm 2 Scheduling Algorithm

Input: $J \leftarrow J_1, J_2, ..., J_n$: List of jobs with their deadlines
$L \leftarrow L_1, L_2, ..., L_n$: List of job loads
$V \leftarrow V_1, V_2, ..., V_m$: List of VM's
$S \leftarrow S_1, S_2, ..., S_m$: Processing speed of VM in MIPS
$CT(j,k)$: Expected computation time matrix
$E(m) \leftarrow 0, 0, 0, ..., 0$: Energy usage matrix
$C(m) \leftarrow 0, 0, 0, ..., 0$: Computation time of VM
$H1 \leftarrow Heap(C(m))$: Min Heap of VM using computation time
$H2 \leftarrow Heap(J(n))$: Min heap of Job using deadline
$w \leftarrow WINDOWSIZE$: Number of tasks taken at time
$LV \leftarrow LV_1, LV_2, LV_3$: Total load in a VM
Output: makespan (ms), energy consumption (E)
Initialize:$E(m) = 0$, $C(m) = 0$, $LV(m) = 0$
1: **while** $i \leq w$ **do**
2: **if** $C(k) \leq D_i$ **then**
3: Assign Root(H1) to Root(H2)
4: Heapify(H1)
5: Heapify(H2)
6: **end if**
7: **while** $k \leq m$ **do**
8: **if** $Root(H2) = V(k)$ **then**
9: $C(k) = C(k) + CT(i,k)$
10: $LV(k) = LV(k) + L(k)$
11: **end if**
12: $k = k + 1$
13: **end while**
14: $i = i + 1$
15: **end while**
16: $ms = Max(C(k))$
17: **for** $p = 1$ **to** m **do**
18: $E(p) = 1.6 \times 10^{(-8)} \times S(p)^2 \times LV(p)$
19: **end for**

TABLE III
ETC MATRIX

Jobs/VM	V_1	V_2	V_3
J_1	18	22.5	27
J_2	30	37.5	45
J_3	12	15	18
J_4	15	18.75	22.5
J_5	24	30	36
J_6	12	15	18

J_3 and J_6 will be allocated to VM V_1, Jobs J_4 and J_2 will be assigned to VM V_2 and Jobs J_1 and J_5 allotted to VM V_3. The jobs J_2 and J_5 placed on VMs V_2 and V_3 respectively cannot be finished by the deadline as shown below:
Processing time for J_4 on V_2 is 18.75 and J_1 on V_3 is 27s,
Processing time for J_2 on V_2 is 37.5 and J_5 on V_3 is 36s,
The total time to complete both the jobs, J_1 and J_5 on V_3 is 63s, but the deadline for J_5 = 60s. Similarly, total completion time of J_4 and J_2 on V_2 is 56.25s but the deadline for J_2 is 53s. If we use linear queue jobs, J_2 and J_5 will miss their deadline. Hence queuing methodology is not able to provide

an optimal solution to the job-placement problem.

Now lets use min-heap to tackle the challenge for the same dataset: construct a min-heap tree of VMs using computation time as key. To break the tie use processing time of VMs. A min-heap of the jobs by using the principle: Job I is child of Job J if and only if deadline of Job I is greater than that of Job J as shown in Fig. 2. Here we assign root node of min-heap

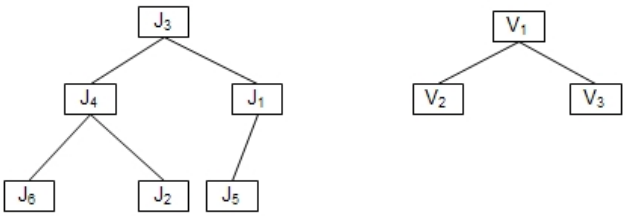

Fig. 2. Initial Min-heap tree for Jobs and VM's

of jobs to the root node of min-heap of virtual machines and heapify both the trees again.

Iteration 1: Job J_3 is dispensed to VM V_1 and expected computing time of V_1 i.e. C(1) becomes 12s. Consequently, on heapifying both the trees, V_2 and job J_4 becomes new root of the corresponding min-heaps as demonstrated in Fig. 3.

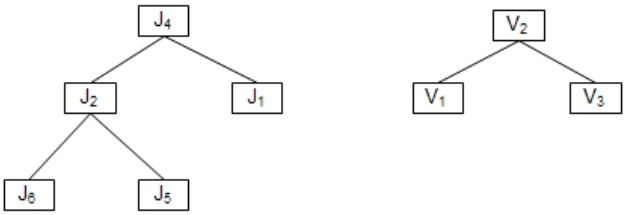

Fig. 3. Min-heap tree after Iteration 1

Iteration 2: Similarly, the system would allocate J_4 to V_2 and update C(2) = 18.75s.

Again heapify both the heaps, next J_1 and V_3 are the root of corresponding trees as presented in Fig. 4.

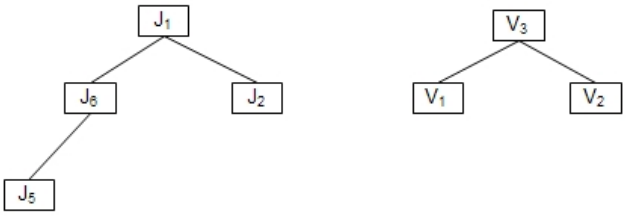

Fig. 4. Min-heap tree after Iteration 2

Repeating the process, we come up with following final assignment: J_3, J_6, and J_5 on V_1, J_4 and J_2 on V_2 and J_1 on V_3. Final total processing time for each VM is C(1) = 48s, C(2) = 56.25s and C(3) = 27s. In this case only J_2 misses its deadline and overall computation time of all the jobs also reduced.

The significant difference between both the data structures is that linear queue does not evenly distribute jobs, it just sequentially allocates them while min-heap can select the best candidate in every iteration because the root of the min-heap tree for VM will always have minimum computation time compared to other members. Table IV indicates how min-heap outperforms linear queue by reducing makespan by 10.7%. Makespan is the maximum processing time among all VM

TABLE IV
PROCESING TIME RESULT MATRIX

DATA STRUCTURE / VM	V_1	V_2	V_3	MAKESPAN
LINEAR QUEUE	24	56.25	63	63
MIN-HEAP	48	56.25	27	56.25

on the system. Hence, it can be mathematically expressed as Max (C(i)) where $i = 1, 2, 3, ..., m$. Considering energy consumption perspective, each VMs consumes E(i) amount of energy in active state and $0.6 \times E(i)$ amount of energy in idle state [20]. It can be rewritten as

$$E(i) = 1.6 \times 10^{-8} \times S_i^2 \tag{1}$$

Unit of E(i) is J/MI (Joules per million instructions), to obtain energy used by one VM we need to multiply no of instructions to E(i). Thus,

$$E'(i) = E(i) \times 1.6 \tag{2}$$

Total energy consumption of whole system is

$$TE = \sum_{i=1}^{m} E(i) \tag{3}$$

Energy consumption by individual virtual machines is revealed in Table V where it can be noted that min-heap uses 15.72% more energy than linear queue to process all requests by deadline.

TABLE V
ENERGY CONSUMPTION MATRIX

DATA STRUCTURE/VM	V_1	V_2	V_3	TOTAL
LINEAR QUEUE	1296	1555.2	1008	3859.2
MHVA	2592	1555.2	432	4579.2

V. Simulation Results

We have performed the simulation is in the cloud computing environment using the CloudSim tool [21]. We have compared our algorithm with the random algorithm in the cloud environment. To simulate our algorithm, we have some assumptions as follows: We have performed the simulation in the cloud computing environment using the CloudSim [21]. We have compared our algorithm with the random algorithm in the cloud environment. Following simulations are made for the simulation:

1) Physical Machines (PMs) are heterogeneous, and each PM has m number of heterogeneous VMs.
2) Tasks are deadline-constrained and independent.
3) Resource requirement of each task is independent of each other.

Various scenarios considered for the evaluation of the algorithms are as follows: Scenario-1: We are fixing the number of tasks to 1000 and varying the number of VMs from 50 up to 500 in the interval of 50. Fig. 5 and Fig. 6 shows the bar-chart for makespan and energy consumption of the system for the MHVA and random algorithms.

Scenario-2: We are fixing the number of VMs to 200 and varying the number of tasks from 200 up to 2000 in the interval of 200. Fig. 7 and Fig. 8 shows the bar-chart for makespan and energy consumption of the system for the MHVA and random algorithms.

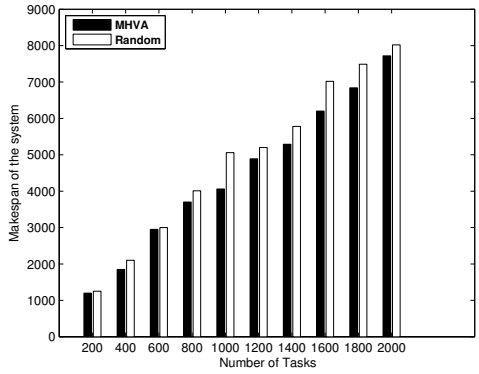

Fig. 5. A bar chart to compare makespan value for Scenario-1

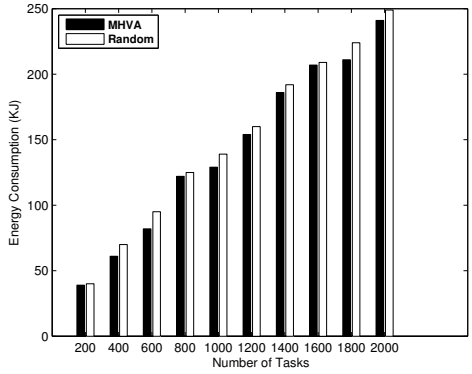

Fig. 6. A bar chart to compare energy consumption for Scenario-1

VI. Conclusion & Future Works

In the following work, the benefits and drawbacks of queue based random algorithm and min-heap based job allocation in the cloud. Both example and simulation results show that MHVA performs better as compared to the random algorithm. From the graph shown in simulation section, it is evident that

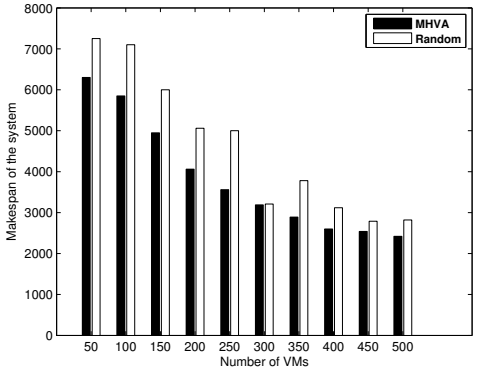

Fig. 7. A bar chart to compare makespan value for Scenario-2

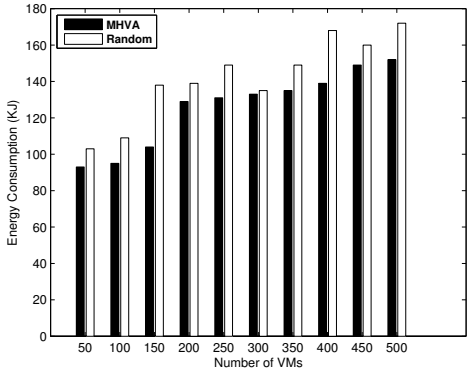

Fig. 8. A bar chart to compare energy consumption for Scenario-2

MHVA performs better as compared to the random algorithm. Devising a data structure that balances the trade-off between dynamic job assignment and reducing energy consumption is an open problem. In future, we are planning to compare scheduling algorithm based on other data structure and design a new job organization so that maximum cloud resources can be utilized.

References

[1] Sampa Sahoo, Syed Nawaz, Sambit Kumar Mishra, and Bibhudatta Sahoo. Execution of real time task on cloud environment. In *2015 Annual IEEE India Conference (INDICON)*, pages 1–5. IEEE, 2015.

[2] Sambit Kumar Mishra, Reenu Deswal, Sampa Sahoo, and Bibhudatta Sahoo. Improving energy consumption in cloud. In *12th IEEE India International Conference 2015 (INDICON 2015), 17-20th December, 2015, At Jamia Millia Islamia, New Delhi, India*. IEEE, 2015.

[3] Seyedmehdi Hosseinimotlagh, Farshad Khunjush, and Rashidaldin Samadzadeh. Seats: smart energy-aware task scheduling in real-time cloud computing. *The Journal of Supercomputing*, 71(1):45–66, 2015.

[4] Zhi-Hui Zhan, Xiao-Fang Liu, Yue-Jiao Gong, Jun Zhang, Henry Shu-Hung Chung, and Yun Li. Cloud

computing resource scheduling and a survey of its evolutionary approaches. *ACM Computing Surveys (CSUR)*, 47(4):63, 2015.

[5] Deepak Puthal, BPS Sahoo, Sambit Mishra, and Satyabrata Swain. Cloud computing features, issues, and challenges: a big picture. In *Computational Intelligence and Networks (CINE), 2015 International Conference on*, pages 116–123. IEEE, 2015.

[6] Deepak Puthal, Surya Nepal, Rajiv Ranjan, and Jinjun Chen. Threats to networking cloud and edge datacenters in the internet of things. *IEEE Cloud Computing*, 3(3): 64–71, 2016.

[7] Deepak Puthal, Surya Nepal, and Jinjun Chen. A secure big data stream analytics framework for disaster management on the cloud. *In 18th IEEE International Conferences on High Performance Computing and Communications*, pages 1218–1225, 2016.

[8] Deepak Puthal, Surya Nepal, Rajiv Ranjan, and Jinjun Chen. Dlsef: A dynamic key-length-based efficient real-time security verification model for big data stream. *ACM Transactions on Embedded Computing Systems (TECS)*, 16(2):51, 2016.

[9] Paulo HR Gabriel, Marcelo K Albertini, Antonio Castelo, and Rodrigo F de Mello. Min-heap-based scheduling algorithm: an approximation algorithm for homogeneous and heterogeneous distributed systems. *International Journal of Parallel, Emergent and Distributed Systems*, 31(1):64–84, 2016.

[10] Sambit Kumar Mishra, Priti Paramita Parida, Sampa Sahoo, Bibhudatta Sahoo, and Sanjay Kumar Jena. Improving energy usage in cloud computing using dvfs. 2016.

[11] Karthik Kumar, Jing Feng, Yamini Nimmagadda, and Yung-Hsiang Lu. Resource allocation for real-time tasks using cloud computing. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–7. IEEE, 2011.

[12] Isam Azawi Mohialdeen. Comparative study of scheduling al-gorithms in cloud computing environment. *Journal of Computer Science*, 9(2):252, 2013.

[13] Ming Mao and Marty Humphrey. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, page 49. ACM, 2011.

[14] Si-Yuan Jing, Shahzad Ali, Kun She, and Yi Zhong. State-of-the-art research study for green cloud computing. *The Journal of Supercomputing*, 65(1):445–468, 2013.

[15] Florin Pop, Ciprian Dobre, Valentin Cristea, Nik Bessis, Fatos Xhafa, and Leonard Barolli. Deadline scheduling for aperiodic tasks in inter-cloud environments: a new approach to resource management. *The Journal of Supercomputing*, 71(5):1754–1765, 2015.

[16] Ziqian Dong, Ning Liu, and Roberto Rojas-Cessa. Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers. *Journal of Cloud Computing*, 4(1):1, 2015.

[17] Saurabh Kumar Garg, Chee Shin Yeo, Arun Anandasivam, and Rajkumar Buyya. Energy-efficient scheduling of hpc applications in cloud computing environments. *arXiv preprint arXiv:0909.1146*, 2009.

[18] Tarandeep Kaur and Inderveer Chana. Energy efficiency techniques in cloud computing: A survey and taxonomy. *ACM Computing Surveys (CSUR)*, 48(2):22, 2015.

[19] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1(1):7–18, 2010.

[20] Ting Shi, Mei Yang, Xiang Li, Qing Lei, and Yingtao Jiang. An energy-efficient scheduling scheme for time-constrained tasks in local mobile clouds. *Pervasive and Mobile Computing*, 27:90–105, 2016.

[21] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. Modeling and simulation of scalable cloud computing environments and the cloudsim toolkit: Challenges and opportunities. In *High Performance Computing & Simulation, 2009. HPCS'09. International Conference on*, pages 1–11. IEEE, 2009.