IJAIA

Volume 1 | Number 1 | June 2025 | Pages XXX-XXX

International Journal of Artificial Intelligence Applications

ISSN (online): 0000-0000
Homepage: ijaia.com

Modelling and Simulating Data Centers in Cloud Computing using GreenCloud Simulator

Eslam Alsubehat[1], Shoroq Al-Sharari[2]

[1]Department of Computer Science, Al Hussein Bin Talal University, Ma'an, Jordan
E-mail: Islam.A.Subaihat@ahu.edu.jo
[2] Department of Software Engineering, Al Hussein Bin Talal University , Ma'an, Jordan
E-mail: shoroq.al-sharari@ahu.edu.jo.

Received: 01, 2025 *Revised: 02, 2025* *Accepted: 03, 2025* *Available online: 06, 2025*

***Abstract*—** Cloud computing is an important source of computing worldwide because it can serve customers as needed and without additional costs. Moreover, it serves the customer at the lowest cost and fastest time by providing computing sources in various forms. Aside from providing millions of users the means to use offered services through their own computers, terminals, and mobile devices, studying cloud computing on real cloud systems is difficult at times. Thus, researchers used a specialized program called cloud simulator to study cloud computing, which in turn studies cloud computing from different perspectives, such as energy consumption, cloud services, and resource management. In this paper, we used Green Cloud simulator to model and simulate cloud data centers (DCs). Through this simulator, we presented an experimental comparative study among common task scheduling algorithms in cloud computing (i.e., green, power-saver, random, and round-robin schedulers). These algorithms are discussed and analyzed briefly. The metrics used to evaluate the task scheduling algorithms include (1) server loads, (2) DC loads, and (3) number of servers used.

***Keywords*—** Cloud computing, Data Center, Simulation, scheduling.

1. INTRODUCTION

Organizations are currently focused on attaining an enduring information and communications technology technique for their business processes. The major motivation for such intent is to reduce their carbon impact and environmental influences, as well as their operational costs. In this context, cloud computing offers a useful means to achieve these goals. Cloud computing is a promising technology that is becoming increasingly prevalent because it facilitates access to computing resources, such as programs, storages, expert services, video games, films, and music, whenever necessary. These resources are provided such that cloud clients do not need to be aware of how or from where they are obtaining these materials. Instead, clients only need to be concerned with acquiring broadband connectivity to the cloud.

Data centers (DCs) possess powerful computing and storage capabilities. Important domains, such as particle physics, scientific computing and simulation, Earth observation, and oil prospecting, are supported by DCs. Numerous densely packed blade servers are utilized by DCs to maximize management efficiency and space utilization. The energy consumed by DCs remarkably increases with the quantity and scale of servers, that is, the amount of such energy is directly related to the number of hosted servers and their respective workloads [1].

* Corresponding author
DOI: https://doi.org/

Considering the increased development of cloud computing and the amount of energy consumed by it, many researchers seek to find solutions that will reduce the amount of energy consumed by cloud computing infrastructure. However, the complexity of a cloud computing environment is an obstacle faced by numerous researchers when conducting studies on a real cloud computing infrastructure. Cloud simulation software is used to overcome this obstacle. This study aims to evaluate and compare common task scheduling algorithms (i.e., green, power-saver, random, and round-robin schedulers) used in cloud system [2].

2. CLOUD COMPUTING

Many experts have defined cloud computing from different aspects. The most common definitions of cloud computing are listed as follows:

- [3] "Clouds are a large pool of easily usable and accessible virtualized resources (such as hardware, development platforms and/or services). These resources can be dynamically re-configured to adjust to a variable load (scale), allowing also for optimum resource utilization. This pool of resources is typically exploited by a pay-per-use mode which guarantees are offered by the Infrastructure Provider by means of customized SLA".
- [4] "Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".
- [5] "Cloud is a parallel and distributed computing system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements (SLA) established through negotiation between the service provider and consumers."

The definitions presented by [3][4][5] are explanations that are most relevant to this research. Hence, cloud computing can possibly be depicted as a set of DCs that connect to the Internet to offer their services. These DCs are based on the virtualization of their infrastructure such that they have virtual machines (VMs) as basic units of computation. In general, they offer hardware (i.e., VM for computations) or software services. These services are provided by mutual agreement through a SLA contract and charged on the basis of a per-use pricing method. Based on the above dentitions, imagining the need for a scheduling algorithm that attempts to find suitable physical machines that can meet the client's requirements in the DC becomes possible. The DC needs to have enough resources, such as bandwidth (BW), which is important in handling user's tasks. Considering that cloud computing is a business model, the SLA agreement should be considered. Now, forming the job's life cycle by using three steps starting from the signing of the SLA contract, then finding the best cloud provider, and ending with managing the resources inside the cloud provider becomes possible. This work has built its framework and drawn its assumptions based on the provided definitions.

Figure 1 depicts the general view of cloud computing. In this figure, several cloud service providers offer various services through the Internet. Clients around the world are connected and send their jobs to the cloud providers via the Internet. However, this variation and multiplicity of services increase the size of search space for clients. Therefore, new methods are needed to orchestrate the services to save clients' time. To understand the architecture of these

services in more detail, the next section shows the components of cloud computing and their relation to this work.

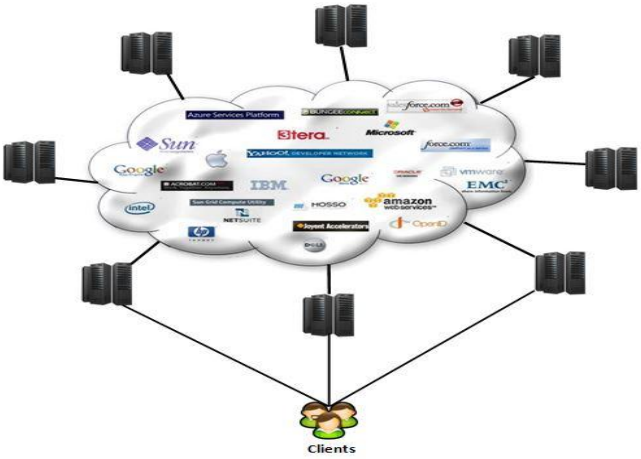

Figure 1 General View of Cloud Computing

2.1. Cloud Computing: Architecture and Provisioning

Cloud computing has emerged as a computing infrastructure that enables the rapid delivery of computing resources as a utility in a dynamically scalable, virtualized manner. The advantages of cloud computing over traditional computing include agility, low entry cost, device independence, and scalability (Tsai et al., 2010).

Cloud models use the DC as a basic unit in their architecture [6][5]. These models can be viewed as a collection of massively distributed DCs [7]. In other words, cloud models are a set of cloud service providers located around the world that offer services via their DCs.

A DC [8], or server farm as it is sometimes called, is a massive, centralized repository for data storage, computation, and management. It is a farm for hosting huge number of servers or processing elements, clusters, and/or huge amounts of storage to serve customer requests.

These DCs are connected to the Internet and merge with other components to form the cloud paradigm. The essential components that make up cloud computing are listed in the following paragraphs [9][10][11](Foster et al., 2008; Riml et al., 2009; Oliveira et al., 2010):

2.1.1. Clients:

A cloud client generally consists of computer software or hardware that relies on the use of cloud computing for the delivery of applications or one that is designed specifically for the delivery of cloud services. Examples include computers, phones and other devices, operating systems, and browsers.

2.1.2. Services:

Services refer to software systems that are designed to support interoperable machine-to-machine interaction over a network that may be directly accessed by other cloud computing components, software (e.g. software plus services), or end users.

2.1.3. Application:

Cloud applications leverage the use of cloud computing in their software architecture, which typically eliminates the underlying need to run the application on the user's own

computer. This service alleviates the burden of software maintenance, ongoing operation, and ongoing support. This kind of service is also called the software as a service.

2.1.4. Platform:

Cloud platforms, or platform as a service, deliver a computing platform that consumes cloud infrastructure while supporting cloud applications. They facilitate the deployment of many applications without the same complexity and cost of purchasing and managing whatever underlying layers of hardware and software that would be required. Platforms in cloud computing fall into a few different categories, namely, services, solution stacks, and structured storage.

2.1.5. Storage:

Cloud storage refers to the delivery of data storage as a service (including database-like services) and is often billed on a utility computing basis (e.g., per gigabyte per month).

2.1.6. Infrastructure:

Cloud computing infrastructure, or infrastructure as a service (IaaS), involves the delivery of a computer infrastructure as a service that is typically a platform virtualization environment.

Figure 2 summarizes the aforementioned cloud comportments in the shape of a stacked layer. The scope of this work falls within the infrastructure layer, such that the cloud providers offer VMs as computation units to clients with urgent tasks that need to be executed. The use of IaaS to execute clients' task is inspired by the cloud characteristics described in the next section.

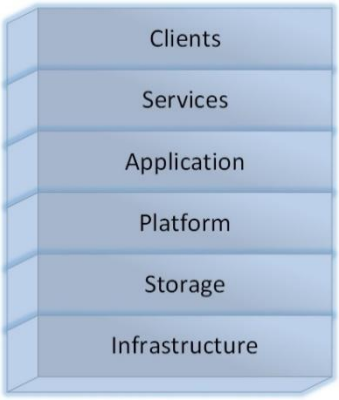

Figure 2 Cloud Computing Stack Layers

2.2. Cloud deployment model

Cloud computing is classified into three types [12] (Peng et. al, 2009):

2.2.1. Private Cloud

This classification offers hosted services within the organization. Firewall and security procedures protect the hosted services from being accessed by unauthorized users.

2.2.2. Public Cloud

This classification offers hosted services that can be accessed by any organization or individual.

2.2.3. Hybrid Cloud

This classification is a combination of the two aforementioned types, where private cloud can be linked to one or more external cloud services.

3. SYSTEM MODEL

The system model is developed based on [13][14][15]. Figure 3 depicts this model, which mainly consists of the cloud users, incoming tasks, VMs and server controllers, scheduling algorithm controllers, and servers.

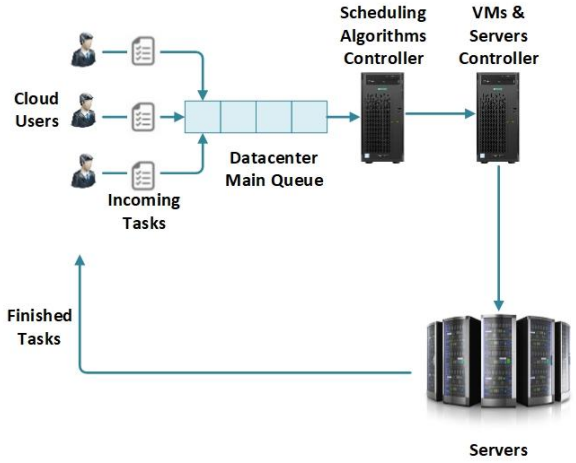

Figure 3 System Model

1. User represents the cloud user who will send tasks to the cloud computing DC.
2. Task refers to the task sent by cloud users to the cloud computing DC. Each task has the following elements: size, maximum completion time, and ID number.
3. DC Main Queue holds the waiting tasks for a period of time < task deadline.
4. Scheduling Algorithm Controller determines the scheduling algorithm that will be used to handle the incoming tasks (e.g., round-robin, green, random, and power-saver schedulers).
5. VM and server controller handle the received tasks after accepting the VM status and decision from the scheduling algorithm controller.
6. Servers create necessary VM and execute the tasks of users.
7. Finished tasks send the completed tasks to users.

4. SIMULATION TOOL

The architecture and main features of the cloud computing simulator used in this study are explained in this section. Social networking, content delivery, web hosting, and real-time instrumented data processing are examples of traditional and emerging cloud-based applications. These types of applications possess different compositions, configurations, and deployment requirements. Quantifying the performance of scheduling and allocation policies in real cloud environments under different conditions and various applications and service models is extremely difficult due to the following reasons. First, users have heterogeneous and conflicting quality of service requirements, and second, clouds have varying demands, supply

patterns, and system sizes. When real infrastructure, such as Amazon EC2, is adopted, experiments are limited to the infrastructure scale. Thus, reproducing the results becomes challenging. This situation arises because the conditions that prevail in an Internet-based environment cannot be controlled by resource allocation developers and application scheduling algorithms [16]. Therefore, we used the GreenCloud simulator, which can be applied to develop novel solutions for the monitoring, resource allocation, workload scheduling, and optimization of communication protocols and network infrastructure. The GreenCloud simulator is an extension of the well-known NS2 network simulator and is released under the General Public License Agreement [17].

4.1. Simulator Architecture

Figure 2 shows the structure of the GreenCloud simulator using a three-tier DC architecture.

The main components of this simulator are listed as follows [17]:

1. Servers that form DC in the cloud are used to run tasks.
2. Switches and links constitute the network topology and the resulting connections by providing different cabling solutions.
3. Workloads are considered objects that model various cloud user services, such as instant messaging and social networks.

4.2. Simulator Implementation

In this experiment, GreenCloud was used to test, evaluate, and compare the adopted and proposed algorithms. Implementation was realized by modifying the original source code of the simulator. The original source code was written in C++ and the Tool Command Language and was based on the NS2 network simulator. Eclipse Standard editor version 4.4 was used for the modification.

Figure 4 provides a general view of the simulation steps. The GreenCloud simulator is set up and installed during the pre-simulation phase, and the simulator configurations are read from the files. In the next step, the DC is created, and the cloud network is developed. This step requires the simulation configuration settings that represent the network and the servers' specifications. Notably, each server may have its own specifications for forming a heterogeneous paradigm. Subsequently, the simulator initiates an event for the arrival of each task to the system. After triggering the events, the simulator begins to execute the scheduling algorithm to map the tasks onto appropriate VMs. Then, the simulator begins monitoring the execution of tasks and recording the ending time of execution and the consumed energy in special tracing files. Simulation stops when all the tasks have passed through the GreenCloud simulator, and then the post-simulation phase begins. This phase involves reading the tracing files and sending the results to an Excel sheet for analysis.

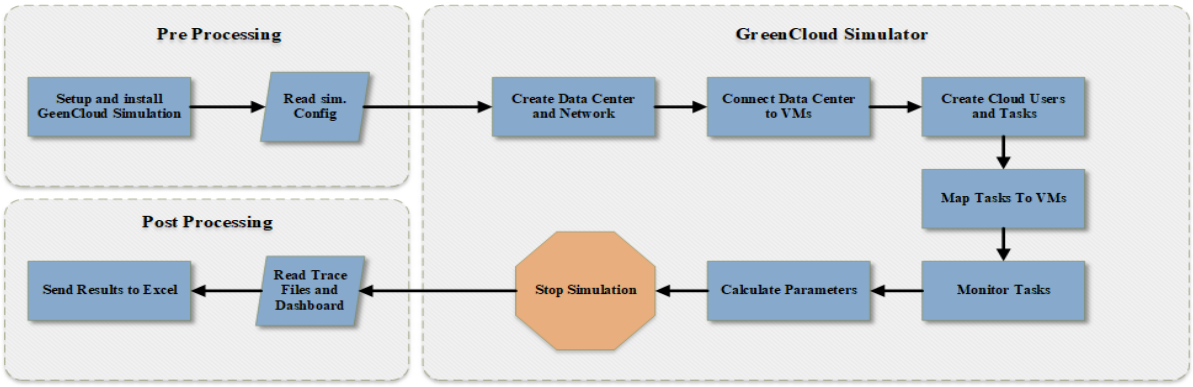

Figure 4 Simulation Steps

The main configuration for the GreenCloud simulator used in this work is provided in Table 1. The table lists the components of the proposed system and their specifications.

Table 1 GreenCloud Configuration

Parameters	Value
DC type	Three–tier topology
No. of core switches	2
No. of aggregation switches	4
No. of access switches	8
No. of servers	1,440
Access links	1 Gb
Aggregation links	1 Gb
Core links	10 Gb
DC load	0.1, 0.2, 0.3, …, 0.9, 1.0
Simulation time	60 min
Power management in server	DVFS and DNS
Task size	8,500 bit
Task deadline	20 s
Task type	High-performance computing

5. DC DESIGN MODEL

Some commonly adopted network architectures in DCs include multi-tier architectures (i.e., two-tier (2T), three-tier (3T), and 3T high-speed (3Ths) architectures [18]). 3T is the most popular architecture in large-scale DCs. In such architecture, the core layer connects the DC to the Internet backbone, the aggregation layer provides diverse functions (e.g., content switching, secure socket layer, and firewalls), and the access layer connects the internal data servers that are arranged in a rack–blade assembly. Multiple links are present from one tier to another. These links, along with multiple internal servers, ensure availability and fault tolerance in the DC but at the cost of generating redundancy.

Server farms in current DCs include over 100,000 hosts, in which 70% of all communication activities are performed internally [19]. The most frequently applied DC architecture is the 3T architecture. The three layers of the DC architecture, namely, the core, aggregation, and access networks, are presented in Figure 5 [20].

The 3T DC topology selected for the simulations includes 1,440 servers, which are set into 16 racks (i.e., 90 servers per rack). The racks are linked using two cores, four aggregations, and eight access switches. The network links that connect the aggregation switches to the core have a data rate of 10 Gb/s. The links that connect the aggregation and access switches, along

with the access links that connect computing servers to the top-of-rack switches, have a data rate of 1 Gb/s. The propagation delay of all the links is fixed at 3.3 µs. Table 1 summarizes the simulation setup parameters [21].

6. SYSTEM PARAMETERS

The criterion for evaluating the virtualized environment is introduced in this section. Two types of parameters are used: input and output. Input parameters configure the system, whereas output parameters measure system performance.

6.1. Input Parameters

The following input parameters are fed to the simulator before it starts.

- **Number of DCs**: Given that we are focusing on a VM in DC and the consumed energy in DC, only one DC is assumed to be present.

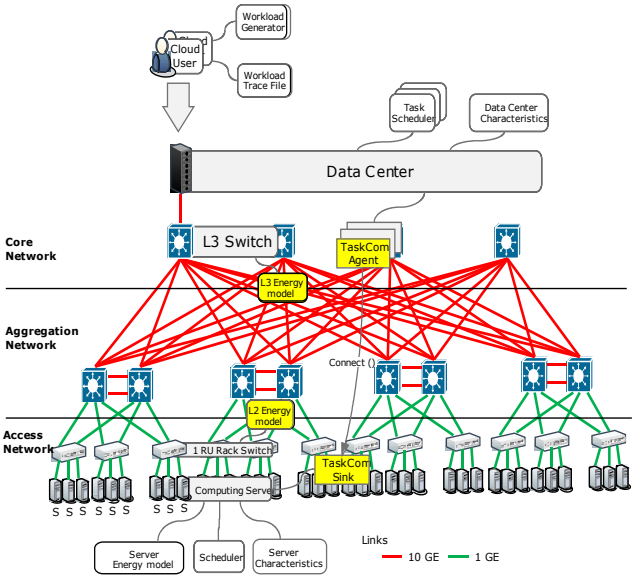

Figure 5 GreenCloud Simulator: A 3T Architecture [17]

- **Number of VMs in DC and their specifications:** The number of VMs in DC that are dedicated to finishing all the submitted tasks and the specifications of these VMs.
- **Number of tasks submitted and their specifications:** A set of tasks is generated and submitted to the DC, and each task has a deadline and size. The scheduler should handle tasks according to their specifications.
- **Scheduling algorithm:** The manner in which tasks are mapped onto VMs affects the simulation results. In each experiment, the algorithm that maps the tasks onto VMs is presented as an input parameter. In this research, we adopt two task scheduling algorithms, namely, the green scheduler algorithm and PSSA.

6.2. Output Parameters

Several performance metrics are used to test and evaluate the proposed models. These parameters determine system efficiency according to the input parameters. The output parameters are described as follows:

- **Makespan:** The maximum completion time of all the received tasks per unit time. This parameter indicates the quality of job assignment to resources in terms of execution time. This parameter can be written formally as Equation 1.

$$Makespan = max\{FT_j | \forall j \in l\} \quad , \qquad (1)$$

where FT_j denotes the completion time of task j that belongs to task list l.

- **Throughput:** The number of executed tasks is calculated to study the efficiency of meeting task deadlines. This parameter is calculated using Equation 2.

$$Throughput\ l = \sum_{j \in l} X_j, \qquad (2)$$

where X_j is

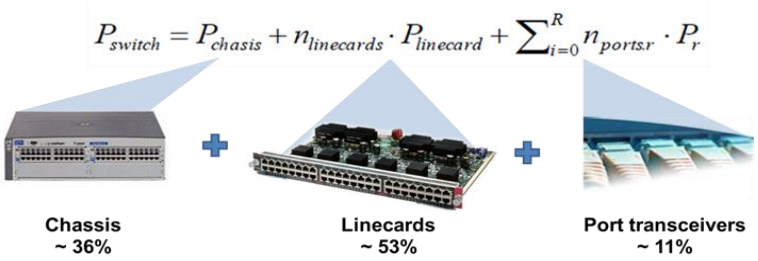

Figure 6 Detailed Components of Switch Energy Consumption

$$X_j = \begin{cases} 1, & task\ j\ has\ finished\ execution \\ 0, & otherwise \end{cases}$$

- **Task failure:** The number of task failures indicates the number of tasks that fail to meet their deadlines, as shown in Equation 4.

$$F = \sum_{j \in l} 1 - X_j, \qquad (4)$$

where F is the number of is failed tasks; and X_j is the decision variable that indicates the task's completion time, which is provided in Equation 3.

- **DC and server loads:** The DC load represents the percentage of computing resources that are allocated for incoming tasks with respect to DC capacity. This load should be between 0% and 100%. A load close to 0 indicates an idle DC, whereas a load equal to 100% denotes a saturated DC [17]. To calculate DC and server loads, let S be the set of M servers in DC, where $S = \{S_1, S_2, \dots, S_M\}$. Each server s_i has nominal million instructions per second (MIPS; N), which denotes the maximum computing capability of the server at the maximum frequency. Server load C corresponds to the current load of the server in MIPS. Equation 5 indicates the load for each server si, which is equal to the ratio of the current server load to the maximum computing capability.

$$load\ (s_i) = \frac{C(s_i)}{N(s_i)} \qquad (5)$$

- The DC load measured using Equation 6 is equal to the average load of all its hosts.

$$load\ (DC) = \frac{\sum_{\forall s \in S} \frac{C(s_i)}{N(s_i)}}{M} \qquad (6)$$

- **DC energy consumption**: The total energy consumption in DC represents the sum of the energy consumed by the servers and switches [17]. In this research, we focus on the energy of servers and network switches. Hence, the power consumption of an average server can be expressed using Equation 7.

$$P = P_{fixed} + P_f . f^3 \quad , \qquad (7)$$

where P_{fixed} accounts for the portion of the consumed power that does not scale with the operating frequency f, and Pf is the frequency-dependent CPU power consumption.

The power consumption of a switch can be expressed using Equation 8.

$$P_{switch} = P_{chassis} + n_{linecards} + P_{linecard} + \sum_{i=0}^{R} n_{ports,r} + P_r \quad (8)$$

Figure 6 shows the detailed components of switch energy consumption, where P_{switch} is the total power consumed for the switch, $P_{chassis}$ is the consumed power for the switch's chassis (hardware), n is the number of line cards in the switch, $P_{linecards}$ is the power consumed using any active switch line card, and P_r is the power consumed by a port (transceiver) that runs at bit rate r.

7. RESULTS

To demonstrate the impact of server virtualization on the server's energy consumption, two experiments were conducted to measure the following parameters: server's energy consumption, DC load, and makespan. That is, PSSA is used under the GreenCloud simulator to compare two simulation scenarios (with and without server virtualization).

7.1. Makespan

Figure 7 represents the total time required to finish all tasks in minutes for each scheduling algorithm, the experiment was carried out 10 times, each time with a different DC load starting from 10% to 100%. The results reveled that random and round robin schedulers showed the worst finishing time from 10% to 60%. Meanwhile, the green scheduler required more time to finish all tasks at load 80%–100%. The power-saver scheduler showed the best execution time among all schedulers at all simulation loads.

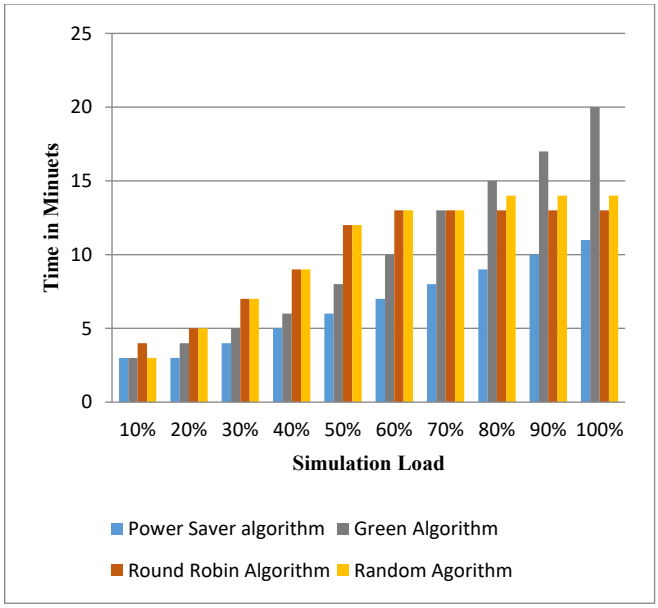

Figure 7 Simulation Time at Different Simulation Loads

7.2. DC Load

Figure 8 represents the DC load after implementing different simulation scenarios with varying loads starting from 10% to 100%. The figure shows a continuous load difference between the four scheduling algorithms. The power-saver scheduler shows an improvement in load from 10% to 70%. Meanwhile, at 80% to 100%, the random scheduler showed remarkable DC load.

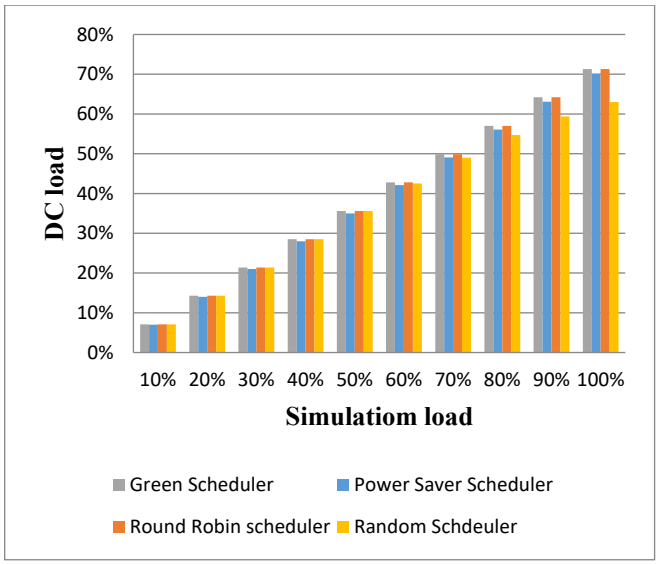

Figure 8 DC Load at Different Simulation Loads

7.3. DC energy consumption

Figure 9 represents the consumed energy of servers at different DC loads for the four scheduling algorithms (i.e., our proposed scheduling algorithm, green scheduling, round robin, and random algorithms). The experiment was carried out 10 times, each time with a different DC load starting from 10% to 100%. When the DC load increases, the total energy consumed by the servers increases for all four schedulers. However, the power-saver scheduler demonstrated a better energy saving mechanism than other scheduling algorithms. Here, a higher DC load implies greater energy consumption.

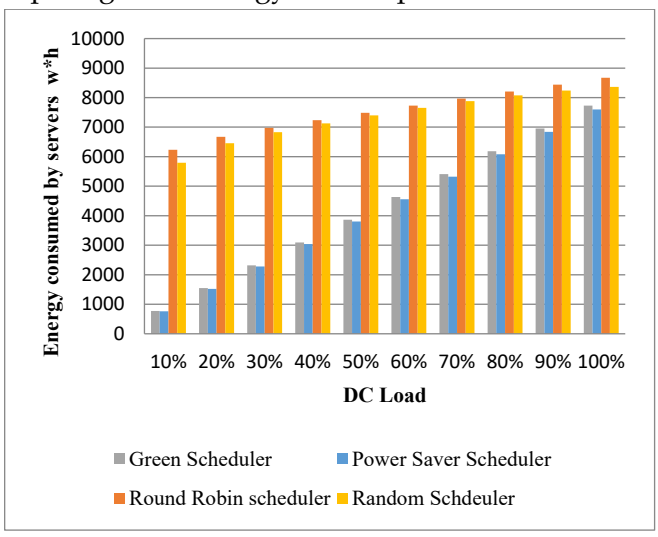

Figure 9Energy consumption at DC Loads

8. DISCUSSION

The green and power-saver schedulers perform better in terms of power saving, as revealed in Figure 9. The reason behind this phenomenon is that the round robin and random schedulers have used all 1440 servers to finish all DC tasks, whereas the green and power-saver schedulers at 30% load have only used one-third of all 1440 servers to finish the DC tasks and to keep the remaining servers in DNS mode.

The results also showed that the power-saver scheduler performs better than the green scheduler in terms of utilization. Hence, the total consumed energy using the power-saver scheduler is less than that of the green scheduler (Figure 13). The reason is that the green scheduler loops in all machines until it can schedule the task. Therefore, if all machines are occupied and cannot schedule the task, then the task will be marked as failed. Meanwhile, the power-saver scheduler tries to schedule on the powered-on machines until it is scheduled. However, if all machines that are turned on are occupied, then the system will turn on another machine and schedule the task in it. If no more machines can be turned on, then the system will give another try, and if all the machines are still occupied, then the task is marked as failed.

9. CONCLUSIONS

In this study, we propose a new task scheduling algorithm called power-saver scheduling algorithm based on DVFS and DNS. The proposed scheduling algorithm was compared with the best existing energy-efficient scheduling algorithm (i.e., green scheduler), round robin algorithm, and random algorithm. The comparison was conducted using GreenCloud simulator with a focus on energy consumption, server load, DC load, and task finishing time. The simulation results revealed that the proposed algorithm performed better than all schedulers in all aspects (i.e., energy consumption, server load, DC load, and task finishing time). The results also demonstrated that the proposed algorithm did not compromise the DC performance, and all tasks were fully completed as they achieved the SLA. According to these results, the scheduling algorithm based on DVFS and DNS can offer enhanced performance in terms of power consumption and server load while maintaining the SLA.

REFERENCES

[1] A. Beloglazov, R. Buyya, Y. C. Lee and A. Y. Zomaya, "A taxonomy and survey of energy-efficient data centers and cloud computing systems," Advances in Computers, vol. 82, pp. 47-111, 2011. doi: https://doi.org/10.1016/B978-0-12-385512-1.00003-7

[2] R. Buyya, R. Ranjan and R. N. Calheiros, "InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services," in Algorithms and Architectures for Parallel Processing, Berlin: Springer, 2010, pp. 13-31. doi: https://doi.org/10.1007/978-3-642-13119-6_2

[3] L. M. Vaquero, L. Rodero-Merino, J. Caceres and M. Lindner, "A break in the clouds: towards a cloud definition," ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, pp. 50-55, 2009. doi: https://doi.org/10.1145/1496091.1496100

[4] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," NIST Special Publication 800-145, Sep. 2011. doi: https://doi.org/10.6028/NIST.SP.800-145

[5] I. Foster, Y. Zhao, I. Raicu and S. Lu, "Cloud computing and grid computing 360-degree compared," in Proc. GCE'08, 2008, pp. 1-10. doi: https://doi.org/10.1109/GCE.2008.4738445

[6] M. Armbrust et al., "A view of cloud computing," Commun. ACM, vol. 53, no. 4, pp. 50-58, 2010. doi: https://doi.org/10.1145/1721654.1721672

[7] K. Hwang, G. Fox and J. J. Dongarra, *Distributed and Cloud Computing: From Parallel Processing to the Internet of Things*, 1st ed. Amsterdam, Netherlands: Morgan Kaufmann, 2012. [Online]. Available: https://www.elsevier.com/books/distributed-and-cloud-computing/hwang/978-0-12-385880-1

[8] J. Hamilton, "Internet-scale service infrastructure efficiency," in Proc. LADIS'08, 2008. [Online]. Available: https://mvdirona.com/jrh/TalksAndPapers/JamesHamilton_LADIS2008.pdf

[9] I. Foster, C. Kesselman and S. Tuecke, "The anatomy of the grid: Enabling scalable virtual organizations," International Journal of High Performance Computing Applications, vol. 15, no. 3, pp. 200-222, 2001. doi: https://doi.org/10.1177/109434200101500302

[10] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," Future Generation Computer Systems, vol. 25, no. 6, pp. 599-616, 2009. doi: https://doi.org/10.1016/j.future.2008.12.001

[11] Y. Jadeja and K. Modi, "Cloud computing - concepts, architecture and challenges," in Proc. ICCCNT'12, Coimbatore, 2012, pp. 1-5. doi: https://doi.org/10.1109/ICCCNT.2012.6158533

[12] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang and Q. Li, "Comparison of several cloud computing platforms," in Proc. ISISE'09, pp. 23-27. doi: https://doi.org/10.1109/ISISE.2009.58

[13] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, vol. 41, no. 1, pp. 23-50, 2011. doi: https://doi.org/10.1002/spe.995

[14] D. Ghosh, R. Sharman, H. Raghav Rao and S. Upadhyaya, "Self-healing systems - survey and synthesis," Decision Support Systems, vol. 42, no. 4, pp. 2164-2185, 2007. doi: https://doi.org/10.1016/j.dss.2006.05.009

[15] T. Wood, P. Shenoy, A. Venkataramani and M. Yousif, "Black-box and gray-box strategies for virtual machine migration," in Proc. NSDI'07, 2007. [Online]. Available: https://www.usenix.org/legacy/events/nsdi07/tech/full_papers/wood/wood.pdf

[16] R. Buyya, C. Vecchiola and S. T. Selvi, *Mastering Cloud Computing: Foundations and Applications Programming*, 1st ed. Cambridge, MA, USA: Morgan Kaufmann, 2013. [Online]. Available: https://www.elsevier.com/books/mastering-cloud-computing/buyya/978-0-12-411454-8

[17] D. Kliazovich, P. Bouvry and S. U. Khan, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," Journal of Supercomputing, vol. 62, no. 3, pp. 1263-1283, 2012. doi: https://doi.org/10.1007/s11227-010-0504-1

[18] N. Farrington and A. Andreyev, "Facebook's data center network architecture," in Proc. ONS'13, 2013. [Online]. Available: https://www.opencompute.org/files/ONS2013-facebook-network-infra.pdf

[19] D. Abts, M. R. Marty, P. M. Wells, P. Klausler and H. Liu, "Energy proportional datacenter networks," in Proc. ISCA'10, pp. 338–347. doi: https://doi.org/10.1145/1815961.1816004

[20] S. R. Alam et al., "Early evaluation of IBM BlueGene/P," in Proc. SC'07, pp. 1-12. doi: https://doi.org/10.1145/1362622.1362708

[21] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee and N. McKeown, "ElasticTree: Saving energy in data center networks," in Proc. NSDI'10, pp. 249-264. [Online]. Available: https://www.usenix.org/legacy/event/nsdi10/tech/full_papers/heller.pdf