ISSN: 2633-4828 Vol. 5 No.4, December, 2023

International Journal of Applied Engineering & Technology

# ENERGY OPTIMISATION SCHEDULING FOR CLOUD COMPUTING DATA CENTERE USING SHARING WITH LIVE MIGRATION

**Jayant D. Sawarkar and Dr Manoj Eknath Patil**
Department of Computer Science and Engineering, Dr. A. P. J. Abdul Kalam University, Indore 452010
jdsawarkar@yahoo.com and mepatil@gmail.com

## ABSTRACT
*Since cloud computing offers an appropriate platform for data centres, it has become a potent tool for data storage. According to recent studies, one of the main problems with cloud computing systems is their energy usage. In order to meet performance standards, minimise power usage, and optimise resource utilisation, we should reduce energy consumption. In this research, a novel algorithm based on the energy optimisation technique known as Sharing with Live Migration (SLM) is introduced, which may be used to assign resources in a cloud computing environment. Using a unique algorithm that learns and predicts task similarity, we employed the Cloud-Sim toolkit to control the usage of virtual machines (VMs) in this scheduler. Each task is then assigned to an appropriate VM. Conversely, SLM uses a migration procedure to meet the hosted applications' Quality of Services (QoS) requirements. The method performs better, saves electricity, and requires less processing time, according to the experimental results. As such, the SLM algorithm outperforms a modified state-of-the-art solution for a comparable situation in terms of virtual machine efficiency and resource utilisation.*

*Keywords: cloud computing, scheduling algorithm, green computing, energy optimization, virtualization.*

## INTRODUCTION
When it comes to on-demand processing and storage, cloud computing is the best option because it allows users to save, retrieve, and share any kind of data on the cloud. High power consumption is one of the major issues that cloud computing data centres face, despite their many advantages. In this regard, striking a balance between delivering performance and minimising energy consumption is a common challenge for cloud service providers. Furthermore, by 2020, data centres in the US are expected to consume roughly 73 billion kilowatt-hours of energy, according to current trends [1]. Power costs can quickly surpass hardware costs by a significant margin if power consumption keeps rising [2]. The cooling system, network, and servers in a cloud data centre use 90% of the electricity available [3].

High efficiency cloud computing data centres must be created in order to ensure a future of green computing and satisfy industrial demands, as this sector is growing into a significant consumer of energy and resources. The purpose of scheduling algorithms is to distribute work in a way that maximises the use of system resources [4]. There is a need for more creative and efficient scheduling models because despite the large number of models that have been proposed and thoroughly investigated, the majority of them are not appropriate for the modern cloud computing environment [5].

In order for a scheduler to properly categorise and evaluate computer resources and shorten task execution times, task scheduling—a crucial component of cloud computing—currently concentrates on all computing resources [6]. Enhancing the effectiveness of the scheduling system in a cloud setting can boost the server's performance and related resources, as well as the efficiency of the procedures that manage proficiency [7].

Task scheduling can be divided into three primary categories: heuristic, hybrid, and energy-efficient algorithms [8]. First, there are two categories of heuristic task scheduling: static and dynamic power usage. Static algorithms operate under the assumption that every task is independent and arrives at the same time. Numerous task-scheduling techniques, including evolutionary algorithms, min–min and max–min, minimum execution and minimum completion times, etc., are classified as static scheduling techniques. Dynamic scheduling strategies, including K-Percent and suffrage scheduling, rely on the system machine's current state of time and presume that the jobs' arrival times are dynamic.

The second category of algorithms relates to job scheduling that is energy-efficient. One of the key factors influencing a cloud data center's energy consumption is task scheduling. Cutting-edge green task-scheduling algorithms are created and optimised to save energy usage by figuring out the best processing speed to finish every work before a deadline. Last but not least, hybrid scheduling reduces execution time by combining several scheduling factors into a single scheduling strategy. The majority of these algorithms are either brand-new or created by fusing many outdated predesigned schedulers together.

This is how the rest of the paper is structured. Related works are shown in Section 2. The Sharing with Live Migration (SLM) paradigm is developed in Section 3. In Section 4, the SLM scheduler algorithm is suggested. The analysis and simulation are shown in Section 5. The experimental results are presented in Section 6. Section 7 concludes with discussion of conclusions and next steps.

## 2. RELATED WORK

Numerous articles discuss how to schedule and assign tasks in a cloud environment, which has been beneficial in lowering overall energy use.

An energy-efficient workflow task-scheduling method (DEWTS) enabled by Dynamic Voltage and Frequency Scaling (DVFS) was proposed by the authors [9]. By splitting up the concurrent jobs into workflows, the suggested approach balances the scheduling performance and then runs the workflows at the right time to minimise power usage. According to evaluation tests, the algorithm was successful in reducing power usage by 46.5% when doing jobs in parallel.

A green, energy efficient scheduler, the priority algorithm from [10] effectively applies the DVFS technique to assign appropriate resources to jobs in accordance with the requirements of workloads under the Service Level Agreement (SLA). Compared to the dynamic voltage scaling strategy, this method lowered server power consumption by 23% and enhanced resource utilisation [11]. In order to lower energy usage for data-intensive systems by lowering the SLA rate, an offline scheduling technique was put out in [12]. To lower cloud energy usage, the suggested scheduling approach enhanced server utilisation and reduced network traffic.

The authors[13] presented an energy-aware scheduling technique for batch and online mode workloads that integrates DVFS in multicore CPUs. For batch mode applications, the scheduler reduced energy consumption by 27%, and for online mode applications, it reduced energy consumption by 70%. The highest performance power virtual machine (VM) under the deadline limitation was chosen first when scheduling the virtual machines using the new virtual machine scheduler that was suggested in [14]. This scheduling approach reduced energy consumption by up to 20% while increasing processing capacity by 8%. Using an energy-aware load balancer in the cloud, Pavithra and Ranjana [15] proposed an energy-efficient resource-provisioning system with dynamic virtual machine deployment. The method that is being given reduces the cloud computing environment's cost, power consumption, and execution time to achieve quantifiable gains in resource utilisation.

By utilising a consolidation approach and optimising the virtual machine (VM) allocation, the suggested scheduling scheme in [16] made use of the available resources and reduced energy consumption. In comparison to two benchmarks, DVFS and an Energy-aware Scheduling algorithm utilising the Workload-aware Consolidation Technique (ESWCT), the results demonstrated an improvement in energy consumption. The effective server-first scheduling that was suggested in [17] made use of the response time in relation to the number of active servers in order to lower data centre energy usage. Approximately 70 times as much energy was saved as compared to the technique based on random selection.

While the aforementioned task-scheduling algorithms took into account one or two task scheduling criteria, the algorithm must give optimal performance with the lowest possible cost and power consumption while also being fair to users when providing services. The necessity for dynamic resource scheduling techniques was emphasised in Intel's Cloud Computing 2015 Vision. These techniques include using a good task scheduler that can lower an

application's processing time as well as the power consumption of its used resources, as well as shutting down idle servers [18, 19].

This work focuses on building a novel scheduler to minimise the total processing and communication costs for a group of activities aiming at reducing the overall energy consumption, taking into account the constraints of the previous studies.

## 3. THE SLM MODEL, OR PROPOSED SHARING WITH LIVE MIGRATION

Improving cloud computing's energy efficiency brought forth a demand for research studies and the best possible solutions. In this study, we introduce a unique algorithm for a cloud computing environment that might distribute resources according to SLM, or energy-saving optimisation techniques. To minimise bandwidth utilisation and transmission time needed for file transactions between various resources and optimise energy consumption in a data centre, the SLM model leverages two primary task characteristics: task migration and the similarity among the number of requests users deliver to the cloud.

### 3.1. MODEL OF WORKFLOW

The model is based on the idea that there are a set of tasks, and that tasks can be performed concurrently on a given available resource file if they will execute the same kind of operation on that source file while adhering to certain constraints.

The following is the setting in which cloud computing task scheduling takes place:

Inputs: The set of m resources, designated by R = [R1, R2,..., Rj,..., Rm], is expected to handle n independent tasks, represented by the set T = [T1, T2,..., Ti,..., Tn], i = 1, 2,..., n, j = 1, 2,..., m.

Results: Effective work scheduling to appropriate resources and timeliness is the result. Goals: Reducing makespan to achieve energy-efficient scheduling and enhancing the data center's energy efficiency.

Tasks are sent to the data centre by its cloud users, and the scheduler manages the distribution of these requests to the relevant resources and maintains track of task dependencies. This work presents a novel SLM scheduler that seeks to process several tasks in the cloud simultaneously in order to maximise makespan and overall energy consumption without compromising the performance of the application.

### 3.2. MODEL OF ENERGY

$$P(u) = k* P_{max} + (1 - k)* P_{max} * u \qquad (1)$$

The power model used in the SLM database, as specified in [23], where $P_{max}$ is the maximum power consumed when the host is fully utilised, k is the proportion of power consumed by the host when idle, and u is the CPU utilisation.

The integral of the power consumption function for a given time period (2) is used to determine the total energy consumed by a single host. This is represented as u(t).

$$E = \int_t \quad P(u(t)) \qquad (2)$$

When all tasks are completed, VMs and hosts are terminated, and the total energy used by the hosts to process all of the tasks at the data centre for a specified amount of time is what determines the data center's energy consumption. The algorithm's output displays each host's energy consumption as well as the amount of processing time needed to complete one task in order to compute the host's energy usage [20].

### 3.3. THE MODEL OF EXECUTION

The suggested SLM scheduler method is covered in the steps that follow:

1. The data centre receives numerous jobs from users.
2. In the event that the required server is idle, the scheduler begins processing jobs.
3. Incoming tasks that require a file that is currently being used are queued.
4. The scheduler goes through a task checkup process for every job that is queued for every queue. In case any of the tasks need to perform a similar type of process on the available resource, it will grant them permission to start concurrently with the current task without waiting in the queue if the designated host is not overloaded.
5. If the primary server is too busy, move workloads to the replica server that isn't being used.
6. Transfer duties from the replica server, which is underloaded, to the main server and put it back into sleep mode.
7. Use queuing for tasks with the shortest wait times in the event that resources are not available.

## 4. SLM SCHEDULER ALGORITHM

The two primary properties that the SLM algorithm leverages are the similarity of task characteristics and live migration. Similarity is the ability to process a set of jobs concurrently based on their kinds. Because the SLM scheduler uses four different job types i. e. read, write, download, and upload, it categorises the tasks according to their kind and the required source file. For better bandwidth utilisation, two processes that are reading the same file, for instance, will be scheduled concurrently. However, conflicts prevent processes like "write" and "write" or "upload" and "upload" from processing concurrently.

Live migration can also reduce the amount of physical computers that are in use by transferring jobs to idle machines (consolidation) and transferring processes from overloaded to underloaded servers (load balancing) [21]. Generally speaking, a cloud provider will take the threshold into account in order to enable auto-scaling. The aggregate value determined by using the current performance metric values is defined by the threshold parameter, which also determines the point at which resource auto-scaling is initiated. Pre-agreed Service Level Agreements (SLAs) based on [22] determine the top and lower auto-scaling thresholds, which are set at 80% and 30%, respectively.

Since 80% is the top utilisation criterion, the overload migration begins if the load is greater than that amount. Migration occurs in the SLM model between two replica servers, and the server replica configuration is used to generate the SLM data centre model. According to the replica configuration, each server is duplicated and used to serve as an updated replica of the master server. The cost of migrating the VMs was determined using the following equations in the SLM model [23].

$$T_{mj} = \frac{Memory_j}{Bandwidth_j} \qquad (3)$$

$$U_{d_j} = 0.1 * \int_{t_0}^{t_0+T_{m_j}} U_j(t)dt \qquad (4)$$

Where $U_j$(t) is the CPU utilisation by $VM_j$, $Memory_j$ is the amount of memory utilised by $VM_j$, $Bandwidth_j$ is the available network bandwidth, and $t_0$ is the time when the migration starts. $T_{mj}$ is the time taken to finish the migration.

As stated in Algorithm 1, one read/download counter and one write/upload lock referred to as the read/download lock and write/upload lock, respectively must be defined and utilised in order to apply the SLM algorithm. While

writing or uploading, the write/upload lock is activated. Once completion of the procedure, the lock will be turned back to off. As long as it is on, this lock will prohibit any other process from starting.

Since many read or download tasks can be processed in parallel but do not always start and end at the same time, a write or upload operation on a resource can only begin if no other process is using it. This is why the read/download counter uses a single counter to count the read and download operations. Every time a read or download procedure begins, the counter will be increased, and it will decrease after the process is complete. We utilise this counter because the write or upload procedure cannot begin until this number is zero.

**Algorithm 1:** The Sharing with Live Migration (SLM) algorithm

Create different types of task

Submit task to broker for execution

For each task from task queue

For each VM from VM list

Submit task to the VM

If VM.host.utilization > 80

Migrate VMs until host utilization is <80

Else

If VM.host.utilization < 30

Migrate to another host and hibernate current host

If the read/download counter=0 & write/upload lock is in off mode for the current task required

resource file

Submit task for execution

If the task type is read/download

Increment the read/download counter

Else

Set write/upload lock on

Else

If read/download counter 6=0

If task type is read/download

Submit task for execution, increment counter

Else

Submit task to waiting queue

Else

Submit task to the waiting queue

After execution of task

If task has used file for read or download

Decrement read/download count

Else

Set write/upload off

## 5. ANALYSIS AND SIMULATION OF SLM

We used the Cloud-Sim toolbox to simulate the overall energy usage in the SLM's recommended approach. Next, we contrasted the outcomes of the SLM simulation with the data centre layout found in [24].

### 5.1. ASSUMPTIONS

The cloud data centre offers six physical servers, each of which has ten virtual machines. All of the hosts and virtual machines have the same set of specifications, including processing power, storage capacity, and bandwidth. Eight GB of RAM, two TB of storage, four CPUs with a 10,000 MIPS capacity, and sixteen DVS-enabled processors make up each host in the data centre. Since the data centre has six hosts, we applied the replica method to half of the hosts, meaning that three hosts have nine different sources and the other half have an exact copy of the nine files, as shown in Figure 1. Each host has three different source files that the cloud users require.

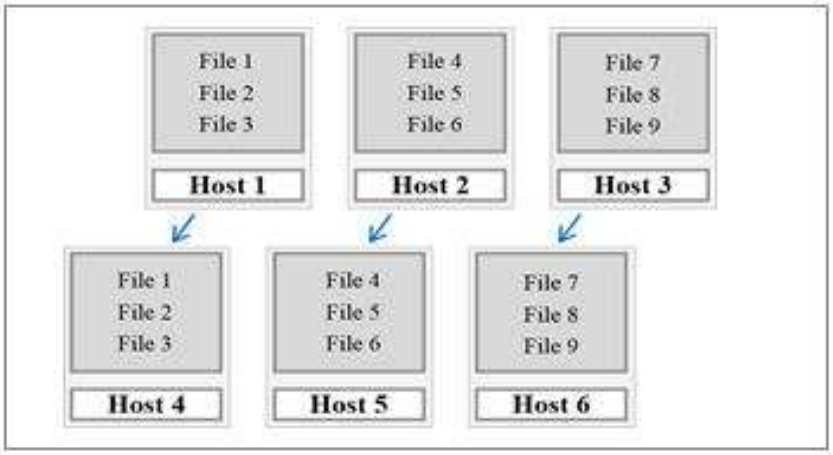


**Figure 1:** Identical hosts.

### 5.2. ENVIRONMENT FOR SIMULATION

We chose Cloud-Sim as the platform for simulation because it allows us to model and simulate cloud computing systems, offers a variety of application environments, and supports the system and behaviour modelling of cloud systems, including resource provisioning policies, virtual machines, and data centres [29].

The simulation's objective is to assess the algorithm's effectiveness in a typical data centre setting. Six hosts, sixty virtual machines (VMs), 500 tasks that required four different processing kinds (reading, downloading, writing, or uploading), and nine source files that were stored on the hosts and processed by the various tasks were all included in the simulation of the datacenter.

### 5.3. CASE STUDY OF SIMULATION

Users are connected to the cloud via a broker entity. A job is first submitted by the user to its broker, who subsequently schedules the work in accordance with a scheduling policy. The broker dynamically obtains a list of available resources and details about the hosts' locations before scheduling the task. 500 jobs are generated in the simulated environment and given to a scheduler who controls and prioritises their entrance. The simulation environment had been set up and executed following the next steps in the scenario:

[1] Task generation: Assign reading, writing, uploading, and downloading assignments with 10, 000, 40,000, 80,000, and 200,000 task durations. Create an arbitrary number of tasks and configure their kind, size, and quantity.

[2] Primary admission control: When there are sufficient free processing elements (PEs), the data centre broker assigns jobs to a class known as "cloudlet" to handle transferring them to the appropriate virtual machine (VM) without breaking any concurrency control rules.

[3] Analysis and ranking tailored to applications: Each task is described in the file execution information, together with the file number, task type, file host number, and file VM number. Tasks are assigned by the SLM scheduling strategy to the first underloaded virtual machine (VM) with the necessary file. The term "cloudlet" designates the kind of job process, each with a unique code, such as reading, downloading, writing, or uploading. Only'read' or 'download' task types are permitted for simultaneous processing by the SLM scheduler; 'write' or 'upload' task types are not permitted because of the overwriting issue. Concurrency control will therefore be provided by the SLM scheduler.

[4] Control over QoS and SLA: To identify changes in QoS and SLA conditions and stop any violations of such agreements, the suggested SLM simulation model included a continuous observation class.

## 6. RESULTS OF EXPERIMENT

Based on the outcomes of the experiments, we examine our algorithm's performance in this part. To validate the performance of the suggested SLM scheduler even further, we developed a simulation that we called SLM. Our suggested SLM scheduler's reaction time performance was evaluated by benchmarking it against the fundamental job scenario [30]. The fundamental work model is a greedy data centre that uses a FIFO (First In First Out) queuing system and no concurrency control to assign jobs to hosts with enough capacity to handle them. This prevents two tasks from accessing the same file at the same time. The waiting and response times will therefore gradually cause the makespan to grow with large loads.

The task scheduling methods have been evaluated based on two metrics: energy usage and makespan. The latter refers to the total time required to allocate all tasks to the virtual machines (VMs) that are available. As a result, the SLM model will offer task admission control for each and every incoming task in addition to offering migration-enabled parallel processing. As demonstrated in Table 1, the suggested method can considerably lower the system's makespan and energy usage when compared to the basic work algorithm.

**Table 1:** The SLM and the basic work simulation energy and makespan results.

| No of Tasks | **Basic Work Model** | | **Energy Consumption in the SLM** | |
|---|---|---|---|---|
| | Energy consumption (Kwh) | Makespan (s) | Energy consumption (Kwh) | Makespan (s) |
| 100 | 0.17 | 1102 | 0.09 | 460 |
| 200 | 0.23 | 1598 | 0.12 | 735 |
| 300 | 0.31 | 2245 | 0.15 | 1278 |
| 400 | 0.35 | 2320 | 0.20 | 1368 |
| 500 | 0.42 | 2605 | 0.29 | 1964 |

Table 1's simulation results for the SLM model demonstrate the progress made in lowering energy usage. In a baseline work setting, a batch of 100 jobs in the cloud data centre would require 0.17 kWh and 1102 s of makespan to complete; in contrast, the SLM model would only require 0.09 kWh and 460 s of makespan. The enhancements are shown in Figures 2 and 3.

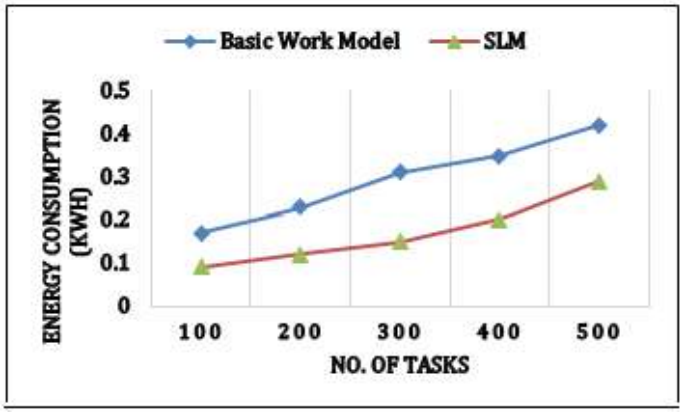


**Figure 2:** Comparison of the energy consumption of the SLM and the basic work model.


**Figure 3:** Comparison of the energy consumption of the SLM and the basic work model.

## 7. CONCLUSIONS

The reduction of expenses and computer infrastructure is the ultimate aim of energy-efficient scheduling in cloud computing. In order to lower a data center's overall energy usage, this article introduced a revolutionary scheduling method called SLM, which combines sharing and live migration principles. A series of simulated experiments showed that the approach results in a burden that is more evenly distributed across all machines. The SLM algorithm performs optimally overall and has a lower processing time than a basic task. The findings demonstrate good performance in terms of makespan and energy consumption. We anticipate that the suggested algorithm will be used by real-world cloud platforms, with the goal of minimising network load and lowering cloud data centre energy expenses.

## REFERENCES

[1] Singh, S.; Sharma, P.K.; Moon, S.Y. EH-GC: An Efficient and Secure Architecture of Energy Harvesting Green Cloud Infrastructure. *Sustainability* 2017, *9*, 673.

[2] Xu, D.; Wang, K. Stochastic Modeling and Analysis with Energy Optimization for Wireless Sensor Networks.*Int. J. Distrib. Sens. Netw.* 2014, *5*, 1–5.

[3] Patel, S.; Makwana, R.M. Optimized Energy Efficient Virtual Machine Placement Algorithm and Techniques for Cloud Data Centers. *J. Comput. Sci.* 2016, *12*, 1–7.

[4] Awad, A.I.; El-Hefnawy, N.A.; Abdelkader, H.M. Enhanced Particle Swarm Optimization for Task Scheduling in Cloud Computing Environments. *Procedia Comput. Sci.* 2015, *65*, 920–929.

[5] Mahmood, A.; Khan, S.A.; Bahlool, R.A. Hard Real-Time Task Scheduling in Cloud Computing Using an

Adaptive Genetic Algorithm. *Computers* 2017, *6*, 15.

[6] Ma, T.; Tang, M.; Shen, W.; Jin, Y. Improved FIFO Scheduling Algorithm Based on Fuzzy Clustering in Cloud Computing. *Information* 2017, *5*, 25.

[7] Yadav, A.K.; Rathod, S.B. Study of Scheduling Techniques in Cloud Computing Environment. *Int. J. Comput. Trends Technol.* 2015, *29*, 69–73.

[8] Yadav, A.K.; Mandoria, H.L. Study of Task Scheduling Algorithms in the Cloud Computing Environment: A Review. *Int. J. Comput. Sci. Inf. Technol.* 2017, *8*, 462–468.

[9] Tang, Z.; Qi, L.; Cheng, Z.; Li, K.; Khan, S. An Energy-Efficient Task Scheduling Algorithm in DVFS-enabled Cloud Environment. *J. Grid Comput.* 2015, *14*, 55–74.

[10] Wu, C.; Chang, R.; Chan, H. A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. *Future Gener. Comput. Syst.* 2014, *37*, 141–147.

[11] Ali, S.A.; Islamia, J.M. A Relative Study of Task Scheduling Algorithms in Cloud Computing Environment. In Proceedings of the 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), Noida, India, 14–17 December 2016.

[12] Zhao, Q.; Xiong, C.; Yu, C.; Zhang, C.; Zhao, X. A new energy-aware task scheduling method for data-intensive applications in the cloud. *J. Netw. Comput. Appl.* 2016, *59*, 14–27.

[13] Lin, C.; Syu, Y.; Chang, C.; Wu, J.; Liu, P.; Cheng, P.; Hsu, W. Energy-efficient task scheduling for multi-core platforms with per-core DVFS. *J. Parallel Distrib. Comput.* 2015, *86*, 71–81.

[14] Ding, Y.; Qin, X.; Liu, L.; Wang, T. Energy efficient scheduling of virtual machines in cloud with deadline constraint. *Future Gener. Comput. Syst.* 2015, *50*, 62–74.

[15] Pavithra, B.; Ranjana, R. Energy efficient resource provisioning with dynamic VM placement using energy aware load balancer in cloud. In Proceedings of the 2016 International Conference on Information Communication and Embedded Systems (ICICES), Chennai, India, 25–26 February 2016.

[16] AlIsmail, S.M.; Kurdi, H.A. Green algorithm to reduce the energy consumption in cloud computing data centers. In Proceedings of the 2016 SAI Computing Conference (SAI), London, UK, 13–15 July 2016; pp. 557–561.

[17] Ziqian, D.; Liu, N.; Rojas-Cessa, R. Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers. *J. Cloud Comput. Adv. Syst. Appl.* 2015, *4*, 5

[18] Intel's Cloud Computing 2015 Vision. Available online: http://www.intel.com/content/www/us/en/ cloud-computing/cloudcomputing-intel-cloud-2015-vision.html (accessed on 25 July 2018).

[19] Ismaila, L.; Fardoun, A. EATS: Energy-Aware Tasks Scheduling in Cloud Computing Systems. In Proceedings of the 6th International Conference on Sustainable Energy Information Technology (SEIT 2016), Madrid, Spain, 23–26 May 2016; pp. 870–877.

[20] Tian, W.; Xu, M.; Chen, A.; Li, G.; Wang, X.; Chen, Y. Open-Source Simulators for Cloud Computing: Comparative Study and Challenging Issues. *Simul. Model. Pract. Theory* 2015, *1*, 239–254.

[21] Kaur, P.; Rani, A. Virtual Machine Migration in Cloud Computing. *Int. J. Grid Distrib. Comput.* 2015, *8*,

[22] 337–342.

[23] Beloglazov, A. Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing. Ph.D. Thesis, The University of Melbourne, Melbourne, Australia, February 2013.

[24] Beloglazov, A.; Buyya, R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers. *Concurr. Comput. Pract. Exp.* 2012, *24*, 1397–1420.

[25] Shu, W.; Wang, W.; Wang, Y. A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing. *EURASIP J. Wirel. Commun. Netw.* 2014, *2014*, 64. Han, G.; Que, W.; Jia, G.; Shu, L. An Efficient Virtual Machine Consolidation Scheme for Multimedia Cloud Computing. *Sensors* 2016, *16*, 246.