computers

MDPI

Article

An Efficient Energy-Aware Tasks Scheduling with Deadline-Constrained in Cloud Computing [†]

Said BEN ALLA [1,*], Hicham BEN ALLA [1], Abdellah TOUHAFI [2] and Abdellah EZZATI [1]

1 LAVETE laboratory, Mathematics and Computer Science Department, Science and Technical Faculty Hassan 1 University, Settat 26000, Morocco; h.benalla@uhp.ac.ma (H.B.A.); abdellah.ezzati@uhp.ac.ma (A.E.)
2 Department of Electronics and Informatics (ETRO), Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium; abdellah.touhafi@vub.ac.be
* Correspondence: said.benalla@uhp.ac.ma

† This paper is an extended version of the conference paper: BEN ALLA, H.; BEN ALLA, S.; TOUHAFI, A.; EZZATI, A.: Deadline and Energy Aware Task Scheduling in Cloud Computing. Presented at the 4th International Conference on Cloud Computing Technologies and Applications (CloudTech 2018), Brussels, Belgium, 26–28 November 2018.

Received: 26 March 2019; Accepted: 3 June 2019; Published: 10 June 2019

Abstract: Nowadays, Cloud Computing (CC) has emerged as a new paradigm for hosting and delivering services over the Internet. However, the wider deployment of Cloud and the rapid increase in the capacity, as well as the size of data centers, induces a tremendous rise in electricity consumption, escalating data center ownership costs and increasing carbon footprints. This expanding scale of data centers has made energy consumption an imperative issue. Besides, users' requirements regarding execution time, deadline, QoS have become more sophisticated and demanding. These requirements often conflict with the objectives of cloud providers, especially in a high-stress environment in which the tasks have very critical deadlines. To address these issues, this paper proposes an efficient Energy-Aware Tasks Scheduling with Deadline-constrained in Cloud Computing (EATSD). The main goal of the proposed solution is to reduce the energy consumption of the cloud resources, consider different users' priorities and optimize the makespan under the deadlines constraints. Further, the proposed algorithm has been simulated using the CloudSim simulator. The experimental results validate that the proposed approach can effectively achieve good performance by minimizing the makespan, reducing energy consumption and improving resource utilization while meeting deadline constraints.

Keywords: Cloud Computing; priority; energy consumption; deadline; task scheduling; dynamic queues

1. Introduction

Cloud Computing (CC) has recently emerged as a new paradigm for hosting and delivering services over the Internet. The CC environment has appeared with the rapid development of processing as well as storage technologies and the success of the Internet. In this model, the infrastructure and software applications are provided as general utilities that can be managed, leased and released by users through the internet from anywhere and anytime. Cloud Computing offers customized, scalable and QoS guaranteed computing environments for users with easy and pervasive access [1]. The users have only to use the cloud without making an investment in computing resources or the cost of maintaining or administering the network, systems and databases. Therefore, the cloud users pay only for the computing resources and services they have used. These advantages make Cloud Computing a compelling paradigm for servicing computing needs for users [2]. Moreover, the massive resources offered and the flexibility associated with the size variability of the Cloud brings great benefits. However, it brings new challenges, in which tasks scheduling and resources allocation are major problems.

The wider deployment and use of cloud and virtualization technologies have led to the establishment of large scale data centers that provide cloud services. This can result in a huge increase in the amount of energy consumed. Minimizing energy consumption is an increasingly important challenge for data centers in the Cloud [3]. Therefore, many techniques have been proposed in order to reduce energy consumption such as load balancing [4]. This technique aims at dispensing the load between all available resources (Virtual Machines (VMs)). This can effectively help improve the utilization of resources and minimize energy consumption [5]. Besides, the virtualization concept and consolidation (VM consolidation; server consolidation) contribute also to energy awareness. VM consolidation saves energy using the live migration which aims at packing VMs into the minimum number of Hosts and switching the idle Hosts to a power-saving mode [6]. On the other hand, server consolidation is considered also as an efficient method towards energy conservation in data centers by consolidating many VMs residing on multiple under-utilized servers onto a single server [7]. In addition, energy consumption and resource utilization are strongly coupled [8]. This means that the resources with a low utilization rate still consume an unacceptable amount of energy compared to the energy consumption of fully utilized or sufficiently loaded Cloud Computing. Moreover, deadline guarantee is an important QoS requirement for tasks especially the critical one, because Cloud users often want to finish their applications within a certain deadline with minimum cost [9]. This creates a huge pressure for Cloud providers to design an efficient deadline guarantee service for the execution of deadline sensitive tasks while taking into consideration other issues such as the energy consumption, variety of resources, profits, etc. Thus, providing an efficient deadline and energy-aware service requires carefully determining an optimal schedule of the received tasks to the available resources taking into account various tasks' requirements and resources' properties.

The open and dynamic characteristics of a Cloud Computing environment are easy to cause some issues like availability and allocation of physical resources, low utilization of resources, user priority management, excessive energy consumption, etc. Thus, we have designed our approach not only to tackle the problems at hand, but also to take into account the dynamic characteristics of Cloud Computing which makes our solution more robust, flexible and more practical in real-world deployments. Hence, the main objective of this study is to propose a new optimization algorithm that is energy-efficient and deadline-aware. Therefore, the solution should be generic enough in order to significantly improve the system's performance and make it aware of the key optimization factors such as dynamic/flexible scheduling, the requirements between Cloud service components, the perspectives of Cloud users and providers. Thus, our solution aims at minimizing the makespan and energy consumption while meeting users' deadline and achieving efficient resource utilization with maximum profit and high-performance computing.

The rest of this paper is organized as follows. First, we present related works. Next, the proposed work is described. Then, we present the experimental setup and simulation results of our solution. Finally, the conclusion is given at the end.

2. Related Work

Energy consumption aware tasks scheduling is an important issue in Cloud Computing and has attracted great attention. Many studies have been proposed to solve this issue and achieve maximum utilization of resources on the basis of users' requirements and the Cloud provider. Therefore, some single objective can influence the tasks scheduling process such as cost, QoS, energy consumption, waiting time, deadline and makespan.

The authors in paper [10], introduce an energy-efficient task scheduling in a cloud environment. The proposed solution uses the Clonal Selection Algorithm (CSA) to optimize energy consumption and makespan. The proposed work introduces a multi-objective CSA based optimization algorithm which aims at solving the problem of task scheduling under the computing environment, where the number of tasks and the datacenter changes dynamically. The results illustrate that the proposed work achieves good performances and outperforms the maximum applications scheduling algorithm and random scheduling in terms of processing time, energy and the failed tasks.

In Reference [11], Shojafar et al. have proposed a genetic-based scheduler to decrease energy and makespan. The proposed work has two phases for scheduling. The first phase is task prioritization and the second is processor assignment. Authors use three prioritization methods for prioritizing the tasks and produce optimized initial chromosomes and assign the tasks to processors which are an energy-aware model.

Similarly, the authors in Reference [12] proposed an energy-efficient tasks scheduling algorithm in a heterogeneous environment to address the demerits associated with task consolidation and scheduling. The proposed algorithm considers the completion time and total utilization of resources. Additionally, it follows a normalization procedure to make a scheduling decision.

Yousri et al. [13] proposed a mechanism which addresses both load-balancing and temperature-awareness. The proposed algorithm selects a physical machine to host a virtual machine based on the user requirements, the load on the hosts and the temperature of the hosts while maintaining the quality of the service. Therefore, VMs are allocated to physical hosts with respect to the temperature and CPU utilization of processors in a way that the target host will not be overloaded or over-heated.

Authors in Reference [14] went further in this direction by formulating an energy efficiency problem to improve the resource utilization for high system throughput. The proposed work refers to multiple-procedure heuristic workflow scheduling and consolidation strategy which aims at maximizing the resource utilization and minimizing the power consumption. Thus, various techniques have been utilized such as dynamic voltage and frequency scaling DVFS with task module migration for workload balance and task consolidation for VM overhead reduction.

In Reference [15], Zhao et al. proposed an energy and deadline aware tasks scheduling. Firstly, the modeling of tasks and datasets as a binary tree based on a data correlation clustering algorithm is introduced. The aim is to reduce the amount of global data transmission, and as a consequence, reduce of SLA violation rate. Secondly, another method called the Tree-to-Tree tasks scheduling approach based on the calculation of Task Requirement Degree (TRD) is proposed. The goal is to improve energy efficiency by minimizing the number of active machines, decreasing the global time consumption on data transmission, and optimizing the utilization of its computing resources and network bandwidth.

Most of the works discussed above deal with energy consumption, but there are still some limits. Some researchers improved tasks scheduling by taking energy consumption and profit as objectives function while others address the problem in hand without considering other important performance metrics such as the makespan and load balancing. Indeed, a resources allocation strategy that takes into account resource utilization should lead to better energy efficiency. However, without a flexible and optimal resources allocation policy, scheduling the task with the focus on only one objective such as energy consumption can lead to degrading the overall performance. Although the deadline is very important, very few works have considered this constraint. Another important aspect which was not always considered when moving VMs is the energy cost of migration. This cost should be taken into account before making decisions as migration brings additional power consumption and its cost in terms of energy is not negligible. Hence, in this paper, we propose an efficient solution to address the energy consumption challenge and the above-mentioned issues in Cloud data centers.

3. Proposed Work

3.1. Problem Formulation

The dynamic and uncertain nature of the Cloud Computing environment makes the tasks scheduling problem hard to solve. Therefore, a good tasks scheduling approach should be designed and implemented in the Cloud broker to not only satisfy the QoS constraints imposed by Cloud users, but also perform good load balancing among virtual machines in order to improve the resource utilization and maximize the profit earned by the Cloud provider. However, an efficient scheduling algorithm does not depend only on the tasks set received from users, but on the resources reserved by providers for processing these tasks as well. In this situation, many issues arise regarding the variety of the tasks which require a different Quality of Service (QoS) and various resource types which may exist either in homogeneous or in heterogeneous environments. Moreover, although the cloud is usually said to offer unlimited resources and can be practically huge, the available resources still have a limit, especially in private Clouds and free Cloud services. Therefore, due to a large number of users, which increases every day and the huge amount of requests sent at the same time to the cloud, the management and execution of all these tasks at the same time require a large number of resources. Therefore, with the growing demand for Cloud Computing, energy consumption has drawn enormous attention in the research community. This is due to the number of carbon footprints generated from the information and communication technology resources such as servers, network and storage [16]. Therefore, the foremost goal is to minimize energy consumption without compromising users' requests. The energy consumed by these requests strongly depends on CPU utilization and resource utilization. This means that the energy consumption will be high when CPUs are not properly utilized because idle power is not effectively used. Sometimes, the energy consumption becomes high due to the heavy demand of the resources, and this may reduce the performance especially if the rented resources are not used efficiently [17].

The objective function used in our work intends to minimize the makespan and energy consumption. Moreover, other metrics can be considered as measures of effectiveness such as load balancing, resource utilization and scalability. Therefore, our goal is to find a compromise between better execution time, lower energy, better resources utilization and good load balancing, because solving the problem in hand by only minimizing the energy consumption does not necessarily means that it can provide optimum scheduling in terms of other metrics such as the execution time, makespan or the resources utilization, etc. Therefore, based on these key performance parameters, the main objectives of the present work is

- To identify intelligently and proactively the best VMs for allocating the tasks with the consideration of energy and deadline constraints.
- To guarantee that the tasks received from the users are handled with a high quality of service (QoS) and without Service Level Agreement (SLA) violation (e.g., improve users satisfaction by meeting their requirements).
- To ensure that the resources available are used to have better effects in order to improve the overall performance in terms of energy and resource utilization.
- To balance a load of all resources using a dynamic priority scheduling based on the tasks features and resources capabilities while taking care of the fact that no PM/VM will get overloaded.
- To maximize resource utilization and minimize the number of PMs used to host the VMs.

Our contribution in this dimension aims at achieving energy reduction for cloud providers and payment saving for cloud users, and at the same time, without introducing VM migration overhead and without compromising deadline guarantees for user tasks. Thus, the goal of our technique is to define a multi-objective function for optimal scheduling and allocation of resources by not only reducing the two-dimensional aspects such as makespan and energy under deadline constraints, but also achieving a good load balancing and high utilization optimization without overloading VMs/Hosts and degrading the performance. In order to design our model, we consider that N tasks (i.e., $T_1, T_2, \ldots, T_N$) are received from cloud users to be scheduled to M VMs (i.e., $VM_1, VM_2, \ldots, VM_M$). We assume that the tasks are non-preemptive in nature, which means that will be executed only once in a particular VM. We assume also that the VMs are heterogeneous (i.e., VMs have different computing capabilities) [18].

The main objective of tasks scheduling is to search the schedule x (mapping of tasks to resources) that minimizes the makespan and energy consumption. A mathematical model for the multi-objective tasks scheduling problem can be expressed as follows:

$$\text{Minimize y = F (x) =Min \{Makespan, Energy\}} \tag{1}$$

3.1.1. Makespan Model

$$\begin{gathered}\text{Minimize Makespan} = \\ Min\Big\{max\Big\{\varphi_{vm_1}, \varphi_{vm_2}, \ldots, \varphi_{vm_j}, \ldots\ldots\ldots, \varphi_{vm_m}\Big\}\Big\}\end{gathered} \tag{2}$$

$$\varphi_{vm_j} = \sum_{i=0}^{n} \delta_j(task_i) \tag{3}$$

$$\delta_j(task_i) = \frac{L(task_i)}{npe_j \times Vmips_j} \tag{4}$$

$$\forall i = \{1, 2, \ldots, n\},\ j = \{1, 2, \ldots, m\}$$

where φ_{vm_j} is the total execution time of a set of tasks running on vm_j, $\delta_j(task_i)$ is the execution time of the task i on vm_j, n is the number of tasks and m is the number of VM, $L(task_i)$ is the length of a task in Million Instruction (MI), npe_j is the number of processing elements and $Vmips_j$ is the VM speeds in Million Instructions Per Second (MIPS).

3.1.2. Energy Model

$$\text{Minimize Energy} = Min\ E(x) \tag{5}$$

where *E(x)* is the total energy consumption which can be calculated by

$$\begin{gathered}E(x) = \sum EN_{ij} + E_0 \\ \text{subject to: } \delta_j(task_i) < T_{deadline}\end{gathered} \tag{6}$$

$$EN_{ij} = \mathrm{R}_j \times \delta_j(task_i) \tag{7}$$

where EN_{ij} is the energy consumption produced by the task $\mathrm{T_i}$ running on the virtual machine $\mathrm{V_j}$, E_0 is the power needed to operate a data center, R_j represents the energy consumption rate of the virtual machine, $T_{deadline}$ represents the maximal latency that cloud users can tolerate (e.g., users' time constraint).

From the user perspective, our method performs efficient tasks execution within the specified deadlines. These tasks are prioritized and classified in order to reduce the number of tasks violating their deadline and also to increase the performance of the system. From the provider perspective, energy efficiency is attained by reducing energy consumption. Additionally, a good load balancing strategy can ensure maximum utilization of resources. As a result, this helps to minimize energy-related costs and prevent system performance degradation. Therefore, the proposed work tackles the energy and deadline dimension based on our previous work [9] in which we have introduced a novel hybrid MCDM method called DEEL based on the Differential Evolution (DE) algorithm and the Multiple Criteria Decision Making (MCDM) method namely ELECTRE III. We propose a new method which intends to prioritize dynamically the tasks and schedule them to the best suitable resource selected using a hybrid meta-heuristic algorithm based on Fuzzy Logic and Particle Swarm Optimization algorithms (FLPSO) [19]. Therefore, the priority of each task is defined based on different parameters and different levels. The parameters considered for the priority assignment are the task length (size), waiting time, burst time and deadline. These metrics are arguably the most important metrics that can meet the user's requirements and improve overall system performances. The ELECTRE III method aims to generate a priority score for each task. After calculating all the scores, the tasks ranking is made and the tasks are sorted based on the score values in descending order. In our model, we consider different levels of priorities (Very High, High, Medium and Low) to address the task priority, the queue priority and the VM priority.

In Figure 1, independent tasks are submitted from n users. Next, these tasks are stored in the global waiting queue according to their arrival time. Then, to manage the tasks received, we apply a Dynamic classifier algorithm. This algorithm in general aims at managing the global queue by using the DE algorithm and ELECTRE III model in order to rank and prioritize the tasks. Next, tasks classification and deadline controller are applied to dispatch and classify the ranked tasks. Based on the DEEL model, the proposed algorithm seeks to find an optimal compromise between the defined parameters such as length, deadline, waiting time and burst time. These ranked tasks are dispatched into three level priorities (Low, Medium, and High). However, some tasks have a very critical deadline and they could miss their deadline. To solve this issue, the tasks which miss their deadline should be firstly and immediately scheduled. As a result, this strategy helps to not only prioritize the tasks dynamically, but also to guarantee no deadline violation when the worst-case scenario really happens. For this, all the tasks which have a critical deadline after the classification process are considered as the most urgent tasks and are assigned as a very high priority. Therefore, this group of tasks should be immediately scheduled to VMs group with a very high priority, and then dispatched among dynamic priority-queues into four levels, such as the very high, high, medium and low priority queues respectively. The tasks can be dispatched in priority levels using static or dynamic strategy. We define three thresholds α, β, δ for the tasks, queues and VMs prioritization. In this paper, these thresholds are determined by simulation. Therefore, after ranking the tasks, they are dispatched among four priority levels as follows:

- Low priority level task (Lpt) = $Task\ (i) \leq \alpha_T$
- Medium priority level task (Mpt) = $\alpha_T < Task\ (i) \leq \beta_T$
- High priority level task (Hpt) = $\beta_T < Task\ (i) \leq \delta_T$
- Very High priority level task (VHpt) = $Task\ (i) > \delta_T\ and\ EFT_{\ Task\ (i)} > DL_{\ Task\ (i)}$, where EFT is the expected finish time if the task i and DL is its deadline.

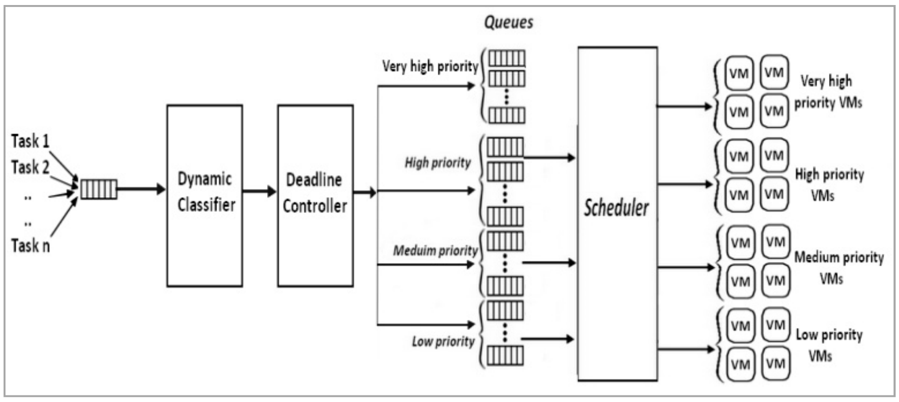

Figure 1. The Energy-Aware Tasks Scheduling with Deadline-constrained in Cloud Computing (EATSD) Scheduling Model.

Next, to prioritize the queues, we consider four different queue priority levels, described as follows:

- Low priority level queue (Lpq): each queue contains the tasks with Lpt.
- Medium priority level queue (Mpq): = each queue contains the tasks with Mpt.
- High priority level of queue (Hpq): = each queue contains the tasks with Hpt.

- Very High priority level of queue (VHpq): = each queue contains the tasks with VHpt.

Then, for the prioritization of the virtual machines, all available VMs are sorted in ascending order based on their processing power. The same method as the prioritization of tasks is used to divide all VMs into four priority levels based on three thresholds $\alpha_V, \beta_V, \delta_V$. Thus, we create four priority levels such as Low priority level VM (Lpv), Medium priority level VM (Mpv), High priority level VM (Hpv) and Very High priority level VM (VHpv). The pseudo-code of the proposed work is described in Algorithm 1.

Algorithm 1. pseudo-code of the proposed work

```
Create the list of tasks
Create the list of Hosts
Create the list of VMs
Put the tasks in the global queue
Calculate the priority for each task in {Ltask} using DE and ELECTRE III method
Get the final ranking score Pr(t)
Classification of tasks (tasks classifier)
Control the tasks' deadline constraint (Deadline Controller)
Define the group of the tasks (Low, Medium, High, Very High)
Calculate the capacity of VMs
Calculate the priority for each VM
Get the final ranking score Pr(v)
Classification of VMs
Define the group of VMs (Low, Medium, High, Very High)
Get List of all queues {LQueue} by DPQ Algorithm
Sort the LQueue in descending order according to Pr(q)
For lv_pr in level_priority
    Create {Lvms'} = {Lvms(lv_pr)}
    For queue Q(lv_pr) inLQueue do
        Call FLPSO algorithm
        Keep track of thebest solution of scheduling tasks among VMs
    End for
End for
```

Subsequently, the DPQ algorithm is applied in order to dispatch the tasks among dynamic queues taking into consideration the ranking order and the priority group of tasks. This algorithm starts to calculate the sum of the tasks length under the same priority level, until a threshold is reached, then makes the decision to create a new queue with the same priority level as tasks under dispatching, then dispatch the appropriate tasks to the corresponding queue, and restart the calculus again until it dispatches all the tasks stored in the global priority queue. Applying the DPQ will create dynamic queues with different priority levels on the basis of a decision threshold and dispatched task priority. Moreover, in order to achieve a good load balancing among different available resources, we need to classify and prioritize the VMs in the next step, taking into consideration the computing capabilities of each VM. Therefore, a priority order is assigned for each VM based on the obtained order (e.g., the higher the order number, the higher the priority). Next, VMs are classified into different priority group in a way to have the same number of tasks group. This classification uses the obtained thresholds in the tasks grouping step to determine all VMs which can be classified and grouped in the same group priority. Then for each group priority, all queues $Q_1, \ldots, Q_n$ are selected to search for the optimal mapping of tasks to the VMs. Next, a scheduling algorithm is applied based on the queue Q and VMGroup which refers to the VMs corresponding to the same group of selected queue. This process is repeated for the same queue by adding the next VMGroup i+1 to the initial VMGroup in order to increase the number of VMs. In this stage, the algorithm calculates the execution time of tasks and returns the maximum execution time among VMGroup. Then, it keeps track of the best solution for mapping the tasks to VMs. The pseudo-code of the VMs classification algorithm is presented in Algorithm 2.

Algorithm 2. pseudo-code of VMs classification algorithm

```
Initialize Best_solution, I, j;
Create the list of VMs and determine all required parameters
Calculate the capacity of all VM.
Sort the list of VMs in descending order.
Define the priority order of each VM
Group VMs based on the thresholds into the Priority Group PG (high, medium, low)/(Group 1,.., Group n)
For each PG do
    For each Qin LQueue(PG)
        VMGroup = VM_ PG
        BestSolution = CallScheduling_Algorithm (Q, VMGroup)
        Fori = 0 to k    do
            VMGroup += VM_PCi
            NewSolution = CallScheduling_Algorithm (Q, VMGroup)
            If (NewSolution > Best_Solution)    then
                Best_Solution = NewSolution
            End if
        End for
    End for
End for
```

4. Experimental Set-Up and Results

To evaluate our algorithms, the experimental simulation is carried out based on Cloudsim simulator [20]. This simulator supports both modeling and simulation for single and inter-networked clouds. CloudSim enables seamless modeling, experiments and simulation of the cloud computing systems and application provisioning environments. Moreover, many Cloud system components can be modeled such as VMs, provisioning of resource and data centers which can be designed with different configurations in order to handle the service tasks. These requests refer to the tasks which need to be allocated to VMs. CloudSim gives several functionalities such as generate a different workload, with different scenarios and perform strong and robust tests based on the custom configurations.

The simulation is done under the following conditions described in Table 1.

Table 1. The Resources Parameters.

Parameters	Values
Number of Datacenter	10
Number of hosts	2–6
Number of VMs	10
MIPS	1000–20,000
VM memory (RAM)	256–2048
Bandwidth	500–1000
Tasks size	100–300,000 MIPS

In the simulation experiments, we compare the proposed EATSD algorithm with the DEELDPQ-SAPSO [21], the Earliest Deadline First (EDF) and the First Come First Served (FCFS) algorithms. DEELDPQ-SAPSO algorithm is an algorithm based on the priority model and dynamic priority queues combined with a hybrid algorithm based on Simulated Annealing with Particle Swarm Optimization. EDF algorithm is a dynamic priority algorithm; the priorities are assigned based on the value of the relative deadline of the tasks. FCFS is the basic policy in the CloudSim Simulator. In this simulation, 100 independent experiments with independent tasks and different parameters settings were performed to evaluate the efficiency of the proposed algorithm. The performance evaluation is compared in terms of average makespan, energy consumption, resources utilization, Execution time and scalability performance.

4.1. Makespan, Energy Consumption and Resource Utilization Analysis

Figure 2 shows the comparative analysis of the makespan of both the algorithms. The makespan of EATSD is better when compared with the DEELDPQ-SAPSO, FCFS, EDF algorithms because the tasks and resources are prioritized into different levels. Thus, these tasks are submitted according to their priorities. When the number of tasks increases, the EATSD makespan becomes very slower than the other algorithm because our algorithm selects the resource on the basis of the task priority. This model has the potential to improve the convergence speed and optimization efficiency of the EATSD algorithm. The proposed algorithm shows a good ability to evaluate the obtained results and to find the best fitness value and come up with the best decision even if the number of tasks increases.

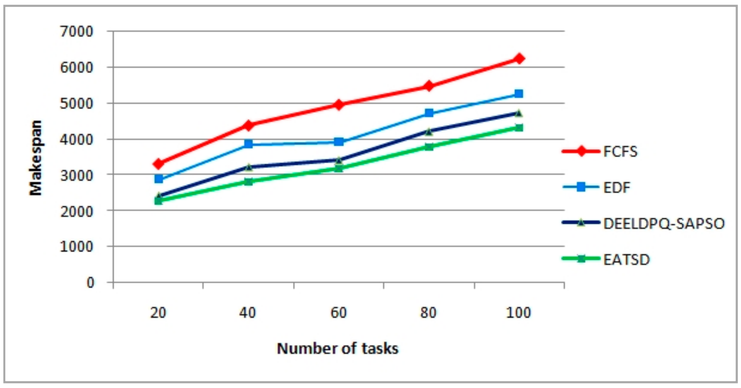

Figure 2. The average makespan with a different number of tasks.

Figure 3 shows the energy consumption comparison between the proposed EATSD algorithm, DEELDPQ-SAPSO, EDF and FCFS algorithms. The energy consumed by EATSD algorithm is considerably less than the DEELDPQ-SAPSO, EDF and FCFS algorithms. It can be observed that there is a significant difference among the compared algorithms and our proposed EATSD algorithm consumes less energy in different scenarios with different workloads. EDF and FCFS energy consumption increase rapidly when the number of tasks increases, while in the case of our algorithm it increases very slowly. The results also show the difference between EATSD and DEELDPQ-SAPSO algorithm, which do not take into account the energy consumption as an objective function. The performance of our EATSD algorithm can be explained by the priority mechanism incorporated in our model and the faster searching of our algorithm for an optimal solution which not only optimizes the processing time and the energy consumption, but also takes into account the resources utilization and the number of resources which can give a good result to the user's tasks.

Figure 4 shows the average resource utilization of the EATSD, DEELDPQ-SAPSO, EDF and FCFS algorithms. The resource utilization is an important metric in the scheduling task process and is beneficial to the cloud provider. The objective is to maximize and achieve a high utilization of resources. Additionally, it is used to keep the resources as busy as possible in order to maximize profit. From the results, it can be shown that the resource utilization of our EATSD algorithm is maintained at a high level, which means that it has the best resource utilization compared with the two other algorithms. An important improvement of resources utilization can be observed with the increase of the tasks number. This result means that EATSD algorithm can maintain a high occupancy rate of resources during the process of scheduling tasks.

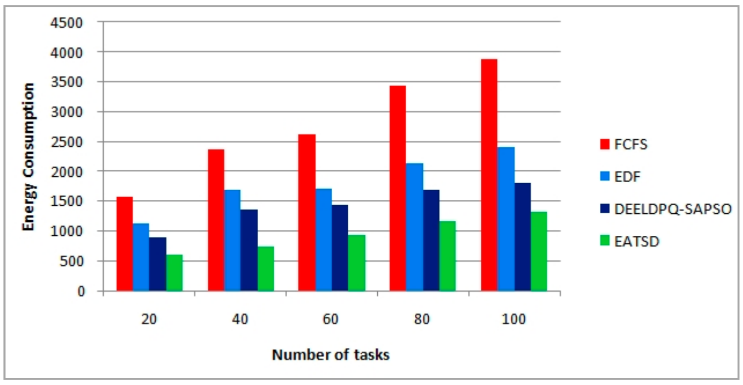

Figure 3. The average energy consumption (Watt) with a different number of tasks.

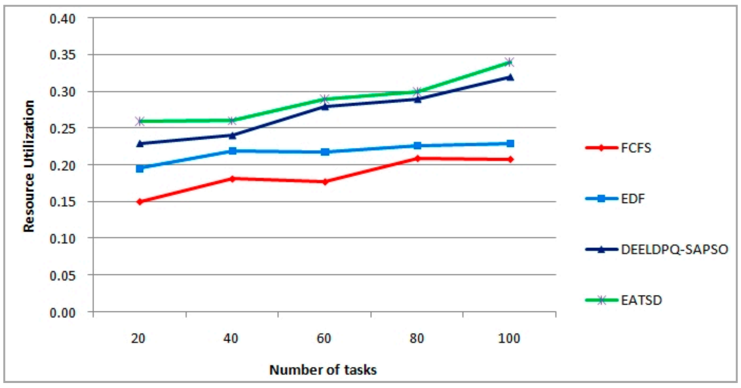

Figure 4. The average resource utilization with a different number of tasks.

Finally, we calculate the Performance Improvement Rate (PIR) percentage. PIR represents the percentage of performance improvement in performance metric (Makespan, Resources Utilization and Energy Consumption) for our proposed EATSD algorithm in comparison with other algorithms. It can be calculated using the following equation (Equation (8)):

$$PIR(\%) = \left(\frac{metric(\text{other algorithm}) - metric(\text{EATSD})}{metric(\text{other algorithm})}\right) \times 100 \tag{8}$$

The PIR results are tabulated in the table as follows:

Table 2 shows the percentage improvement rate achieved by EATSD algorithm over three other algorithms. Firstly, it shows that the proposed EATSD achieves 32.88%, 20.59% and 8,76% of makespan reduction over the FCFS, EDF and DEELDPQ-SAPSO algorithms respectively. Thus, we can see that EATSD has an obviously higher execution time than three other algorithms. Secondly, the results obtained in terms of resources utilization show that the proposed EATSD algorithm is better than the other algorithms and keeps the resources as busy as possible by using all possible resources capabilities. The results illustrated for resources utilization improvement indicate that our algorithm EATSD was able to produce 36.23%, 25.06% and 6.20% resources utilization improvement over the FCFS, EDF and DEELDPQ-SAPSO algorithms respectively. The main reason for the superiority of the proposed algorithm is that our solution takes into consideration the tasks and resources features. Moreover, the priority model incorporated into the prioritization and classification of the tasks, as well as the resources selection, result in a good scheduling decision. Finally, Table 2 indicates the percentage improvement rate on energy consumption of EATSD over the other algorithms. Our algorithm was able to accomplish 65.88%, 47.56% and 33.92% over the FCFS, EDF and DEELDPQ-SAPSO algorithms. The proposed algorithm shows great superiority in the performance improvement rate in terms of energy consumption.

Table 2. The Performance Improvement Rate (PIR) in terms of makespan, resources utilization and energy consumption.

		FCFS	EDF	DEELDPQ-SAPSO	EATSD
Makespan	Total average makespan	24,320.72	20,557.83	17,891.72	16,324.88
	PIR % over FCFS		15.47%	26.43%	32.88%
	PIR % over EDF			12.97%	20.59%
	PIR % over DEELDPQ-SAPSO				8.76%
Resources Utilization	Total average resources utilization	0.93	1.09	1.36	1.45
	PIR % over FCFS		14.91%	32.01%	36.23%
	PIR % over EDF			25.16%	25.06%
	PIR % over DEELDPQ-SAPSO				6.20%
Energy consumption	Total average Energy Consumption	13,881	9032	7167	4736
	PIR % over FCFS		34.93%	48.37%	65.88%
	PIR % over EDF			20.65%	47.56%
	PIR % over DEELDPQ-SAPSO				33.92%

In this simulation, 100 independent experiments were carried out in order to compare the different algorithms. The results and the table of performance improvement rate show that the computational cost of our EATSD Algorithm is lower than other algorithms especially for a larger number of tasks.

4.2. Execution Time and Scalability Analysis

The scalability is an important factor in the performance evaluation which helps to assess the influence of system size and problem size on the performance of an algorithm. Therefore, we have performed the scalability testing and analysis by using a parallel workload called DAS2 5-Cluster Grid to log from a Parallel Workloads Archive (PWA) [22]. This log contains 225,711 jobs from which 100–20,000 tasks are used in this simulation. To calculate the scalability value, we use the Isospeed-e scalability function as follows [23]:

$$\psi(C, C') = \frac{C' * W}{C * W'} \tag{9}$$

We assume that C is the initial execution time of the workload on VMs. C′ is the scaled execution time. W and W′ are the initial and increased workload size respectively. In this scenario, we compare our algorithm EATSD with DEELDPQ-SAPSO, EDF, FCFS and a hybrid algorithm called SA-PSO based on PSO and SA algorithms. SA-PSO is similar to DEELDPQ-SAPSO without the priority model and the dynamic priority queues. The simulation is carried out 20 times in a highly stressful environment and then the average performance is analyzed. Processing Element (PE) from 10–30 are allocated in each VM. The average execution time comparison of the four algorithms is shown in Tables 3 and 4 for Scenario 1 and 2 respectively. Moreover, based on the execution time comparison, the scalability performance is calculated as shown in Tables 5 and 6.

Table 3. The Average Execution Time Comparison (Scenario 1).

Workload	Number of PE	SA-PSO	FCFS	EDF	DEELDPQ-SAPSO	EATSD
1000	10	56	93	68	44	37
5000	15	150	270	192	114	88
8000	20	198	390	250	140	103
12000	25	260	511	322	173	121
20000	30	403	840	521	250	172

Table 4. The Average Execution Time Comparison (Scenario 2).

Workload	Number of PE	SA-PSO	FCFS	EDF	DEELDPQ-SAPSO	EATSD
1000	15	44.8	74.4	54.4	33	25.9
5000	20	120	216	153.6	85.5	61.6
8000	25	158.4	312	200	105	72.1
12000	30	208	408.8	257.6	129.75	84.7
20000	35	322.4	672	416.8	187.5	120.4

Table 5. The Computed Scalability of the Compared Algorithms (Scenario 1).

Scalability	Number of PE	SA-PSO	FCFS	EDF	DEELDPQ-SAPSO	EATSD
$\psi\ (C_{10}, C_{15})$	(10,15)	0.536	0.581	0.565	0.518	0.476
$\psi\ (C_{15}, C_{20})$	(15,20)	0.825	0.903	0.814	0.768	0.732
$\psi\ (C_{20}, C_{25})$	(20,25)	0.875	0.874	0.859	0.824	0.783
$\psi\ (C_{25}, C_{30})$	(25,30)	0.930	0.986	0.971	0.867	0.853

Table 6. The Computed Scalability of the Compared Algorithms (Scenario 2).

Scalability	Number of PE	SA-PSO	FCFS	EDF	DEELDPQ-SAPSO	EATSD
$\psi\ (C_{15}, C_{20})$	(15,20)	0.544	0.602	0.568	0.530	0.528
$\psi\ (C_{20}, C_{25})$	(20,25)	0.822	0.903	0.765	0.761	0.729
$\psi\ (C_{25}, C_{30})$	(25,30)	0.858	0.920	0.874	0.775	0.762
$\psi\ (C_{30}, C_{35})$	(30,35)	0.895	0.964	0.941	0.800	0.784

From the results tabulated in Tables 3 and 4, Figures 5 and 6, it can be observed that when the number of workloads increases, the average execution time increases. However, the EATSD algorithm has good performance in terms of average execution time compared with other algorithms. We have evaluated the compared algorithms with a different number of Processing Elements (PEs) (10–35) in order to assess the execution time performance and to satisfy a desired SLA execution time (say 100ms). From the results, it can be observed that with 1000 until 5000 workloads, our algorithm stills have a good satisfaction of the desired SLA while other algorithms violate the SLA. However, with 8000, 12,000 and 20,000 workloads, we see that our algorithm can reduce the violation of SLA compared with others which have higher violation rate.

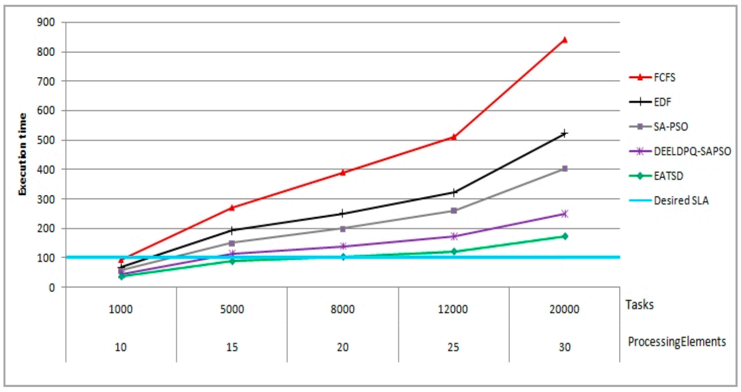

Figure 5. The Execution Time of the Compared Algorithms (Scenario1).

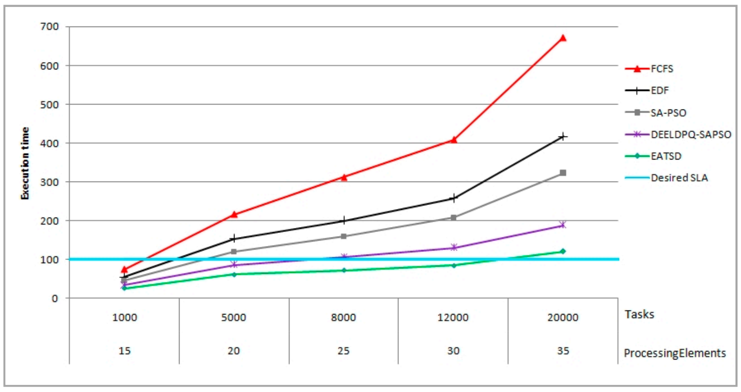

Figure 6. The Execution Time of the Compared Algorithms (Scenario2).

In scenario 2, we evaluate our EATSD algorithm compared with others algorithms using the same workloads number but with different processing elements in order to show the impact of an increasing number of PEs with the same number of resources. The results show clearly the great impact of increasing the number of PEs on mean execution time when the workloads are varied. The proposed algorithm EATSD can effectively satisfy the desired SLA until 16,000 workloads, while other algorithms violate the SLA before processing 5000 workloads. Moreover, based on this result, we have evaluated the scalability performance of our algorithm in order to not only study its ability to support parallel processing at different problem sizes, but also predict its performance in large-scale heterogeneous computing environments. The results tabulated in Tables 5 and 6, and Figures 7 and 8, show the computed scalability of the four algorithms. The workloads and processing elements allocated in VMs are increased in each test case in order to design a heterogeneous system. From the results, it can be observed that our algorithm EATSD can achieve better performance and return a better scalability value with an acceptable range of $0 < \psi < 1$. Thus, the results demonstrate the great ability of our algorithm to perform excellent scalability compared to other algorithms.

Through the above experimental results, it can be observed that EATSD algorithm can effectively meet the users and the providers' requirements in terms of makespan, energy consumption and resources utilization compared to the other algorithms. Moreover, the proposed algorithm achieves a good performance in different scenarios. Additionally, EATSD algorithm is well balanced on exploration and exploitation and has better stability and scalability. Thus, EATSD algorithm shows its effectiveness to decrease the makespan under the deadline constraint, improve the energy efficiency and achieve a good utilization of resources.

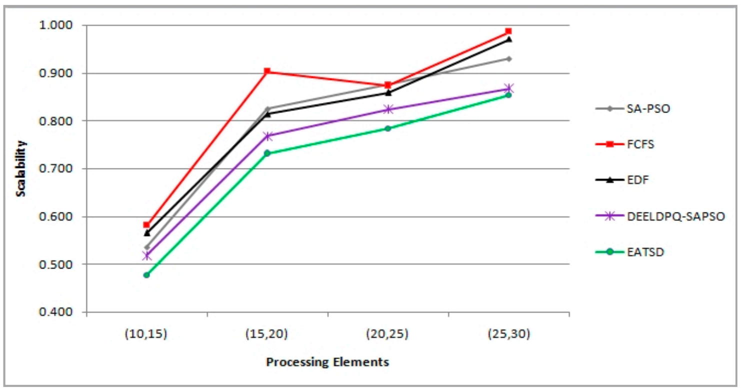

Figure 7. The Scalability of the Compared Algorithms (Scenario 1).

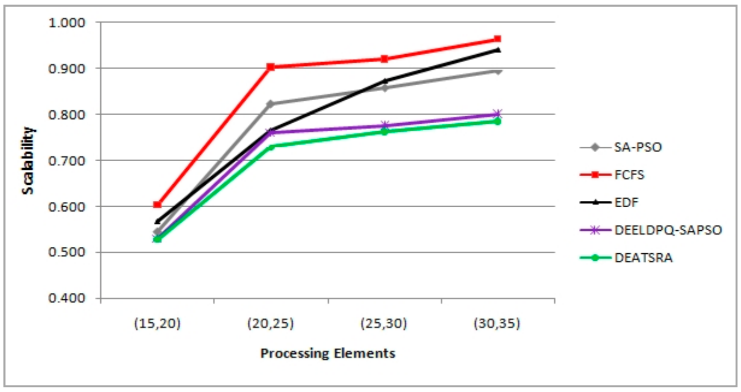

Figure 8. The Scalability of the Compared Algorithms (Scenario 2).

5. Conclusions

This paper presents an efficient algorithm to deal with the Energy Aware for Tasks Scheduling with Deadline constraint (EATSD). The proposed solution aims at minimizing the makespan and reducing energy consumption while meeting the deadline constraint. We consider the problem at hand by allocating the resources on the basis of their class as well as the class of received tasks. Next, each task is mapped to the appropriate VM so that the energy consumed to execute the task is reduced without overloading VMs /Hosts and degrading the performance. This novel EATSD algorithm yields a better schedule with minimum makespan and energy consumption compared to Earliest Deadline First (EDF), First Come First Served (FCFS) and DEELDPQ-SAPSO algorithms. The experimental results demonstrate that our EATSD algorithm outperforms other algorithms in terms of makespan, resources utilization and energy consumption. Further, it shows an excellent ability to execute a variety of tasks and performs very close to the optimum. Additionally, it can scale well with the problem size and achieve fast convergence times. Hence, the proposed algorithm can not only achieve good performances and satisfy the cloud providers, but also improve users' comprehensive QoS significantly.

Author Contributions: Conceptualization, S.B.A., H.B.A., A.T. and A.E.; methodology, S.B.A., H.B.A., A.T. and A.E.; software, S.B.A., H.B.A.; validation, S.B.A., H.B.A., A.T. and A.E.; formal analysis, S.B.A., H.B.A., A.T. and A.E.; investigation, S.B.A., H.B.A., A.T. and A.E.; resources, S.B.A., H.B.A., A.T. and A.E.; data curation, S.B.A., H.B.A. and A.E.; writing—original draft preparation, S.B.A., H.B.A., A.T. and A.E.; writing—review and editing, S.B.A., H.B.A., A.T. and A.E.; visualization, S.B.A.; supervision, A.T. and A.E.; project administration, S.B.A., A.T. and A.E.; funding acquisition, S.B.A., H.B.A., A.T. and A.E.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Shawish, A.; Salama, M. Cloud computing: Paradigms and technologies. In *Inter-Cooperative Collective Intelligence: Techniques and Applications*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 39–67.
2. Kim, W. Cloud Computing: Today and Tomorrow. *J. Object Technol.* **2009**, *8*, 65–72. [CrossRef]
3. Ismail, L.; Fardoun, A. EATS: Energy-Aware Tasks Scheduling in Cloud Computing Systems. *Procedia Comput. Sci.* **2016**, *83*, 870–877. [CrossRef]
4. Gao, Y.; Yu, L. Energy-aware Load Balancing in Heterogeneous Cloud Data Centers. In Proceedings of the 2017 International Conference on Management Engineering, Software Engineering and Service Sciences, Wuhan, China, 14–16 January 2017; pp. 80–84.
5. Mishra, S.K.; Sahoo, B.; Parida, P.P. Load balancing in cloud computing: A big picture. *J. King Saud Univ. Comput. Inf. Sci.* **2018**. [CrossRef]
6. Farahnakian, F.; Pahikkala, T.; Liljeberg, P.; Plosila, J.; Hieu, N.T.; Tenhunen, H. Energy-aware VM Consolidation in Cloud Data Centers Using Utilization Prediction Model. *IEEE Trans. Cloud Comput.* **2018**. [CrossRef]

7. Chen, F.; Grundy, J.; Schneider, J.-G.; Yang, Y.; He, Q. Automated Analysis of Performance and Energy Consumption for Cloud Applications. In Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering, Dublin, Ireland, 22–26 March 2014; pp. 39–50.
8. Valentini, G.L.; Khan, S.U.; Bouvry, P. Energy-Efficient Resource Utilization in Cloud Computing. In *Large Scale Network-Centric Distributed Systems*; John Wiley & Sons, Ltd.: Hoboken, NJ, USA, 2013; pp. 377–408.
9. Alla, H.B.; Alla, S.B.; Ezzati, A.; Touhafi, A. An Efficient Dynamic Priority-Queue Algorithm Based on AHP and PSO for Task Scheduling in Cloud Computing. In Proceedings of the 16th International Conference on Hybrid Intelligent Systems (HIS 2016), Marrakech, Morocco, 21–23 November 2016; pp. 134–143.
10. Jena, R.K. Energy Efficient Task Scheduling in Cloud Environment. *Energy Procedia* **2017**, *141*, 222–227. [CrossRef]
11. Shojafar, M.; Kardgar, M.; Hosseinabadi, A.A.R.; Shamshirband, S.; Abraham, A. TETS: A Genetic-Based Scheduler in Cloud Computing to Decrease Energy and Makespan. In Proceedings of the 15th International Conference HIS 2015 on Hybrid Intelligent Systems, Seoul, South Korea, 16–18 November 2015; pp. 103–115.
12. Panda, S.K.; Jana, P.K. An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems. *Clust. Comput.* **2018**. [CrossRef]
13. Mhedheb, Y.; Streit, A. Energy-efficient Task Scheduling in Data Centers. In Proceedings of the 6th International Conference on Cloud Computing and Services Science, Rome, Italy, 23–25 April 2016; Volume 1, pp. 273–282.
14. Khaleel, M.; Zhu, M.M. Energy-efficient Task Scheduling and Consolidation Algorithm for Workflow Jobs in Cloud. *Int. J. Comput. Sci. Eng.* **2016**, *13*, 268–284.
15. 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. [CrossRef]
16. Panda, S.K.; Jana, P.K. An Efficient Task Consolidation Algorithm for Cloud Computing Systems. In Proceedings of the 12th International Conference on Distributed Computing and Internet Technology, ICDCIT 2016, Bhubaneswar, India, 15–18 January 2016; pp. 61–74.
17. Verma, C.S.; Reddy, V.D.; Gangadharan, G.R.; Negi, A. Energy Efficient Virtual Machine Placement in Cloud Data Centers Using Modified Intelligent Water Drop Algorithm. In Proceedings of the 2017 13th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS), Jaipur, India, 4–7 December 2017; pp. 13–20.
18. Ben, H.; Alla, S.; Alla, B.; Touhafi, A.; Ezzati, A. A novel task scheduling approach based on dynamic queues and hybrid meta-heuristic algorithms for cloud computing environment. *Clust. Comput.* **2018**, *21*, 1797–1820.
19. Alla, H.B.; Alla, S.B.; Ezzati, A.; Mouhsen, A. A Novel Architecture with Dynamic Queues Based on Fuzzy Logic and Particle Swarm Optimization Algorithm for Task Scheduling in Cloud Computing. In Proceedings of the International Symposium on Ubiquitous Networking, Casablanca, Morocco, 30 May–1 June 2016; pp. 205–217.
20. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; de Rose, C.A.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **2011**, *41*, 23–50. [CrossRef]
21. Alla, H.B.; Alla, S.B.; Ezzati, A. A Priority Based Task Scheduling in Cloud Computing Using a Hybrid MCDM Model. In Proceedings of the Third International Symposium on Ubiquitous Networking, UNet 2017, Casablanca, Morocco, 9–12 May 2017; pp. 235–246.
22. Parallel Workloads Archive. Available online: http://www.cs.huji.ac.il/labs/parallel/workload/ (accessed on 18 January 2019).
23. Chen, Y.; Sun, X.-H.; Wu, M. Algorithm-system scalability of heterogeneous computing. *J. Parallel Distrib. Comput.* **2008**, *68*, 1403–1412. [CrossRef]