IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 24, No. 6, December, 2019, pp. 583-590
Journal homepage: http://iieta.org/journals/isi

An Energy and Deadline Aware Scheduling Using Greedy Algorithm for Cloud Computing

Pradeep Venuthurumilli[1*], Sridhar Mandapati[2]

[1] Department of CSE, Acharya Nagarujuna University, Guntur 522510, India
[2] Department of CSE, RVR & JC Engineering College, Chowdavaram 522019, India

Corresponding Author Email: venuthurumillipradeep@stmarysgroup.com

https://doi.org/10.18280/isi.240604

Received: 20 August 2019
Accepted: 13 October 2019

Keywords:
cloud computing, scheduling, energy efficiency, cloud service provider (CSP), first come first served (FCFS) scheduling, Min-Min scheduling and greedy algorithm

ABSTRACT

Cloud computing has been providing various services to different users by means of an aid of large and scalable virtualized resources on the internet. Owing to all the recent and inventive developments that are found in the field, there are several scheduling algorithms which were developed in a cloud computing environment with the intention of decreasing the services given in cloud computing. For a very enormous gauge, assorted and the multi-user atmosphere in the cloud scheme where maximization of profit for that of the Cloud Service Provider (CSP) has been the primary objective. For the purpose of this work, the inclusive optimization problem in the operation of the cloud system by means of lowering the cost of procedure and by maximizing the efficiency of energy. At the same time, it satisfies the deadlines that are definite in Service Level Agreements (SLA) that has been addressed from a CSP perspective. The work proposes a Greedy algorithm for the environment of the cloud and this is compared to the scheduling of a First Come First Served (FCFS) and the Min-Min scheduling procedure. This system exploits the tasks and their heterogeneity and also the resources using a scheduler unit that schedules and allocates the tasks which are deadline-constrained which is delimited to the nodes that are energy conscious. After this, the CSP capitalizes on the parallelisms of data for every user workload and also effectively manages all collective user requests and also apply the custom optimization that creates a cost of global energy and a cloud platform which is dead-line aware. The results of the experiment prove that this proposed Greedy algorithm which achieves a performance which is better (a guarantee ratio, utilization of resources and energy saving) compared to the FCFS and the Min-Min scheduling algorithm.

1. INTRODUCTION

The cloud has been defined to be a kind of equivalent and a scattered scheme that is an assortment of the intersected virtualized workstations provisioned enthusiastically. The computers have been offered to be a resource of a single computer based on facility level promise which was established or even signed amongst the facility supplier and consumer. Cloud estimates the shared structure which can attach some large system pools providing resources of complexity and storage through the internet to users. There are three characteristics the describe cloud computing: 1) Unrestricted estimating possessions such as applications, information stowage and computing power that are accessible based on demand enabling a higher degree of extensibility and agility to meet the needs of business 2) There are no commitments that are long-term: the resources of computing are available immediately and are used for as lengthy as necessary and then are emeritus since they have been acquired either on a point-on-point and a tiny- to-tiny manner 3) The wage-as-we-go-cost erection: and this is owing to the fact that there are not any long-term promises in which the resources of cloud estimates and their costs are dependent on the usage [1].

Virtualization has a crucial role to play in the delivery of resources to consumers in an efficient manner within the environment of the cloud. It may be done in several ways such as the virtualization of storage, memory or the server. For the purpose of achieving this efficiently, the Virtual Machine (VM) had been designed. The VM is a very logical instance of the PC system that controls in a parallel way to this system. The demand made by the user for accessing a physical source in the cloud atmosphere was acknowledged by a VM and allocated to the user on the base of a policy which is well-suited or the constraints that have been specified. For this purpose, there are VM schedulers being used and are employed for dynamically conveying this VM to the users for performing certain specific operations [2].

This denotes the utilization of resources that is improved and the load balancing of systems that is managed. This is the way that all slaves will equitably segment the capacity and the amenities that are demanded by consumers. Every cloud environment will then incorporate a VM policy to efficiently use the resources. This VM scheduling is crucial to maintaining its Quality of Service (QoS) and the SLA which is specified by the provider of cloud service at the time a customer prefers to take cloud services. The process of VM scheduling in the cloud may be generalized into a total of three stages which are: The discovery of resources and filtering – where the broker in a data center finds out all resources that are present within the complex structure and also accumulates the related eminence data. The possessions selection – here the target resource is chosen on the basis of assured constraints of both process and facilities. This will be the significant phase. The chore proposal – here the thread will be acquiesced to the designated sources.

In order to realize the cloud computing potential, the Cloud

Service Providers (CSPs) need to make sure they can be flexible in terms of their service supply for meeting the needs of consumers. Until now, there has been a high performance that is the main reason in the deployments of data centers. A data center that is average tends to consume energy that can be consumed 25,000 households. With the increase in the energy costs and with the dwindling availability, there is not a need to bring about a shift in the emphasis from the optimization of resource management of data center to the energy effectiveness of untainted performance by optimization along with maintaining a service level performance that is high [3].

Task Scheduling is crucial area of exploration that is intended at representing all appropriate professions or chores by revenues of seeing cost restrictions, QoS, boundaries and so on. Without making use of a scheduling solution that is effective, the time taken for task accomplishment and increase in workflow with cloud resources not being utilized fully, brings down the scalability and availability of the cloud systems. The cloud environment scheduling strategy is now a critical issue which distresses the presentation and also has an influence on the cloud users and their cost issue [4].

There are many challenges in scheduling that arise from either a multitenant, elastic, pay-as-you-go and on-demand resource models. On being compared to the other schemes like grids, the cloud offers more control in terms of resources and quantity. Such a flexibility along with resource abundance can create a need for the strategy of resource provisioning working with its scheduling algorithm. The heuristic which decides the number and type of VMs to be used also has to be considered. One more challenge that needs to be addressed by means of scheduling algorithms was the pricing model that is utility-based [5].

The schedulers will also have to identify a new trade-off between the performance, the cost of avoidance of payment and the non-functional needs along with the prices that are potentially prohibitive.

Finally, all the algorithms will have to be aware of the cloud platforms and their dynamic nature along with the uncertainties involved in them owing to the variation of performance that is observed in the resources like the VM CPUs, the storage systems and the network links. Additionally, the providers do not may any guarantee on the period occupied for the provision or the de-provision VMs and these values tend to be variable and also very unpredictable. The schedulers will have to be aware of the variability and will have to recover from any unexpected delay to achieve their objectives. There are numerous different kinds of development procedures which are: The First Come First Serve (FCFS) scheduling, the Shortest Job First (SJF) scheduling, the Round Robin (RR) scheduling, the Priority scheduling Min-Min scheduling and the Max-Min development procedure [6].

These energy-conscious provisioning of resources and their scheduling algorithms tend to help in bringing down the demand for energy but at the same time need support from all effective and underlying technology or the stage for seriatim them proficiently. Cloud computing is the most recent buzz in the industry and it has been observed as a very helpful platform that achieves efficiency of energy. This type of virtualized support is agile and useful in terms of service provisioning by the data centers of the cloud which lower the demand for energy. Consciousness of vitality that is protracted by means of association and virtualization (the VM association; the slave alliance and the task association) which was expedited by means of virtualization can make Cloud estimating approach that is desirable in realizing effectiveness of vitality. Even though the consumers do not demand more VMs that are comparable, or programmed their assignment, they incline to gain controller of the enactment of their assignment by means of deadline specification in their SLAs.

For the purpose of this work, a greed scheduling algorithm is proposed where the scheduling is based on choosing the locally optimum resource to the task the other advantages of greedy scheduling are simple and ease of implementation. The proposed algorithm is compared with the FCFS, and the Min-Min scheduling algorithm. The rest of the investigation has been organized thus. The Section 2 has discussed all related work found in the literature. All the different methods used have been discussed in Section 3. Section 4 has discussed the experimental results and the conclusion has been made in Section 5.

2. RELATED WORK

Marahatta et al. [7] had developed another dynamic and new task assignment with a scheduling scheme which was called the Energy-aware Fault-Tolerant Dynamic Scheduling scheme (EFDTS), for co-ordinating and optimizing the utilization of resources using a mechanism which was fault accepting. In the obligation of a chore, there is a scheme of task classification which has been developed for partitioning all the errands into various courses and allocating them to the VMs that are utmost appropriate based on courses for reducing mean response time at the time of keeping consumption of energy in mind. Repetition is employed for fault acceptance for minimalizing the proportion of chore denunciation that is affected by the disappointment of a mechanism and its interruption. There is a mechanism of elastic resource provisioning that has been intended in a situation of fault acceptance for improving the exploitation of resources and the effectiveness of energy. Also, there was a migration policy that has been developed which simultaneously improves the utilization of resources along with energy efficiency.

Stavrinides and Karatza [8] had made a proposal of an energy-aware heuristic that was used for the preparation of the applications of real-time workflow in the environment of the cloud. This approach makes use of the per-core Dynamic Voltage and Frequency Scaling (DVFS) which is based on the underlying multi-core processors that are heterogeneous to fill all the schedule gaps. The aim was to provide energy efficiency and timeliness by means of trading off the precision of results and keeps the average result and the precision of the jobs completed at a level which is acceptable. The scheduling heuristic that is proposed has been compared to the other two baseline policies. The experiments of simulation have shown that the approach has outperformed all other policies that provide some promising results.

Chen et al. [9] had developed another scheduling architecture that is a novel transforming the problem of dynamic development into many fixed programmes. The authors then proposed another Energy-Efficient Reactive Scheduling algorithm (ERECT), for scheduling all real period errands with calculating possessions in the virtualized exhausts. This procedure ERECT considers the variability of all real period errands and their multitudes. Additionally, while accumulating and then obliterating VMs, an optimum operational frequency and vitality efficiency for the assorted multitudes have been demoralized for achieving preservation

of energy. The results of the experiment proved that the ERECT outperforms all the existing algorithms for guaranteeing the deadlines of the tasks (up to about 14.06%) and an energy saving (up to about 9.81%).

Zhang et al. [10] had projected another a new procedure of heuristic task scheduling known as the Energy and Deadline Aware with Non-Migration Scheduling (EDA-NMS) exploiting the task deadlines and their looseness that postpone the task execution with loose deadlines for avoiding the waking up of the new Physical Machines (PMs). While defining the instant types of the VM, the EDA-NMS chooses all instant types which are sufficient for guaranteeing a task deadline and for bringing down the cost of user payment. The results of all these algorithms show that it achieved energy efficiency without any introduction of a VM migration overhead and any compromise on the guarantees of deadlines.

Garg and Goraya [11] had presented another model for task deadline cognizant energy proficient development used in a virtualized haze. The independent and the deadline conscious errands have been programmed by revenues of virtualizing physical multitudes found in an information center. The programming model in its first illustration also realizes and energy effectiveness by means of effecting an extreme assignment originate in the operating stage for the multitude. The second illustration is by means of vitality equivalent in its idle stage of that of the multitude. There is the extreme vitality that is protected by means of organizing a core-level granularity for occurrence clambering and dynamic energy. The results show that this model for scheduling outperforms the currently existing model owing to the performance and their parameters of its guarantee ratio, resource utilization, consumption of energy for each task and total consumption of energy.

Narwal and Dhingra [12] had proposed another algorithm of multi-objective task scheduling considering various attributes in the environment of the cloud. This algorithm has three different parameters which are the total cost of processing, the total time of processing and the average waiting time. The primary objective of the work was to enhance the performance and also evaluate the performance with the FCFS, the Shortest Job First (SJF) and also the previously implemented multi-objective algorithm of task scheduling.

Rehman et al. [13] had proposed another effective environment that was based on the fog and cloud helping in the management of energy of resources. This switches the information of the groups of structures and every cluster has many different apartments. There are six fogs that have been considered for this situation and each group will have one vapour. The Micro Grids (MG) are made available close to the constructions and are reachable by fog and there are manifold procedures that are recycled for the purpose of consignment harmonizing. The Min-Min algorithm has been proposed for this scenario to manage these resources efficiently. During the time of completion of the task, the time is initially considered and the possessions are distributed to errands that have a slightest interval for execution. The outcomes have been compared to the Round Robin (RR) procedure that has been recycled for the purpose of load balancing. The outcomes of imitation prove that applying this algorithm helps in reducing the cost as being compared to the RR.

Zhang et al. [14] had focused on energy saving for the VM selections in the environment of cloud computing. The work also examines the influences that effect drive with the VM algorithms that were based on the Greedy algorithms and the method of dynamic programming. The experiments are conducted with the Cloud Sim and the results have proved that this algorithm can bring down the consumption of vitality and at the same time satisfy the constraints of the Service Level Agreement (SLA).

Sarvabhatla et al. [15] had made a presentation of a model used for scheduling tasks for the cloud data core and formulates the scheduling of VM tasks to be a problem of integer programming and an objective of bringing down the consumption of energy. It further proves that usage of greedy task schedulers confines the constraint of SLA and brings down the active servers. It also conducts some extensive experiments on the Cloud Sim tool having some typical algorithms of task scheduling. The results of the experiment proved that this scheme performed better compared to the other algorithms.

3. METHODOLOGY

In the case of a cloud environment, the task scheduling will be accomplished by a CSP. The primary objective of the CSP was to choose and assent the desires of the workflow by means of the admission control policy after which the right amount of VMs are allocated for every workload request. Lastly, the schedule of the accepted requests that meet the deadlines of the SLA and the drop requests are made at the same time bringing down the cost of global energy. For the purpose of this section, there is the FCFS scheduling, the Greedy algorithm and the Min-Min scheduling have been discussed.

3.1 Cloud environment

Testing a cloud environment has been described as below in this work [16].

User Workload Requests: It makes use of the Directed Acyclic Graphs (DAGs) for modelling the requests of user workload. The whole workload has been represented to be a collection of the N disjoints DAGS: $\{G_1(V_1, E_1), G_2(V_2, E_2), \ldots, G_N(V_N, E_N)\}$. Every DAG $G_a(1 \leq a \leq N)$ will epitomise the workload request and every vertex $T_i^a(1 \leq i \leq |V_a|)$ in the G_a will embody a task. Without any loss of generality, it has been assumed that every assignment request will be an application belonging to another user. So, for this work, the application will be equivalent to the workload request of the user.

The edge from T_i^a to T_j^a in the G_a indicates the T_j^a is reliant on that of the yield of T_i^a. The actual load of a brink $W_{i,j}^a$ will epitomize the actual quantity of information to be conceded from the chore of the predecessor (T_i^a) to that of the task of the successor (T_j^a). An example for the illustration of the N application group is shown in Figure 1.

Task Model: for the purpose of this work, a set $T = \{t_1, t_2, \ldots\}$ of the independent tasks arriving dynamically was considered. The task t_i which was submitted by the user was modelled by the collection of some constraints $t_i = \{a_i, l_i, d_i, f_i\}$, wherein the a_i, l_i, d_i and f_i denote the time of arrival, the task size or length, the target and the appearance period of the job t_i [17]. If rt_{jk} indicates the organized period of the VM v_{jk} at the multitude h_k and st_{ijk} indicates the actual flinch period of the chore t_i on the VM v_{jk}. Owed to the heterogeneity of the capabilities of the dispensation of the

CPU and the VMs, let et_{jk} indicate the time taken for execution of the task t_i on the VM v_{jk} as in (1).

$$et_{ijk} = \frac{l_i}{c(v_{jk})} \tag{1}$$

It has been expected that the ft_{ijk} indicates the actual texture period of the task t_i on the VM v_{jk}, which is determined as per (2):

$$ft_{ijk} = sft_{ijk} + et_{ijk} \tag{2}$$

Additionally, the x_{ijk} has been employed for reflecting the mapping of tasks to that of the VMs at various hosts within the virtualized cloud in which x_{ijk} indicates "1" in case the task t_i has been assigned to the VM v_{jk} at the multitude and then is "0", else.

Now the texture period will be recycled for determining if the time restriction of the task may be guaranteed as in (3):

$$x_{ijk} = \begin{cases} 0, & if\ ft_{ijk} > d_i, \\ 1\ or\ 0, & if\ ft_{ijk} \le d_i, \end{cases} \tag{3}$$

The Cloud Platform Model: there are a set of M servers in the cloud: {D_1, D_2... D_M}, and this has been modelled as a graph that is aimless with M vertices that represent a new server. Every edge's weight (D_x, D_y), $B_{x,y}$, will represent the capacity of communication which is between that of D_x and D_y.

There are many servers forming a slave grange having native networks. The slave granges will be able to communicate with one another by means of great speed stations. The actual space among that of the slaves and the bandwidths of the channels are reflected in $B_{x,y}$ values. By default, a $B_{x,y} = \infty$, which means, the tasks executing in one server will not have any overheads in terms of communication. Further, it assumes a path to exist between two of the servers by means of a straight connection or multi stages. The path of the multi-paths will get distracted as the concerning brink having a $B_{x,y}$ value that was low.

The Virtual Machine Configurations: at the time of operation every server D_x was associated with a new integer array Q^x of the K members: $\{Q_1^x, Q_2^x, \ldots, Q_K^x\}$, wherein the Q_g^x indicates that the Q_g^x no of the category g VMs (VM_g) have been presented on the D_x. Q^x that is active as it can alteration over period owed to a VM slit depressed and reconfiguration that is originated by a CSP. This denotes the fact that a Q^x configuration during t as Q^x (t). Every slave Dx will have a fixed set of resources which is the CPU and the memory. The configuration of the VM for the D_x will abide by the total resources and the ∀ t will have the $\sum_{g=1}^{K} Q_g^x(t)\, R_{CPU}^g \le C_{CPU}^x$ and the $\sum_{g=1}^{K} Q_g^x(t)\, R_{MEM}^g \le C_{MEM}^x$.

Consumption of Energy: The actual influence depletion of the D_x at a period t has a fixed influence depletion $P_{static}^x(t)$ and an active influence depletion $P_{dynamic}^x(t)$. These are associated using a rate of utilization of the D_x at the time t: $Util_x$ (t). It further evaluates the $Util_x$ (t) by means of taking the requirements of the CPU of the VMs which are hosted shown in Q^x (t), and they show no difference between VMs that run and are idle as the CPU activities in the background will be required even in the indolent periods as in (4).

$$Util_x(t) = \frac{\sum_{g=1}^{K} Q_g^x(t).R_{CPU}^g}{R_{CPU}^x} \times 100\% \tag{4}$$

$P_{static}^x(t)$ denotes a constant when the $Util_x$ (t) > 0, 0 else. The actual association amongst the $P_{dynamic}^x(t)$ and the $Util_x$ (t) will be very complex.

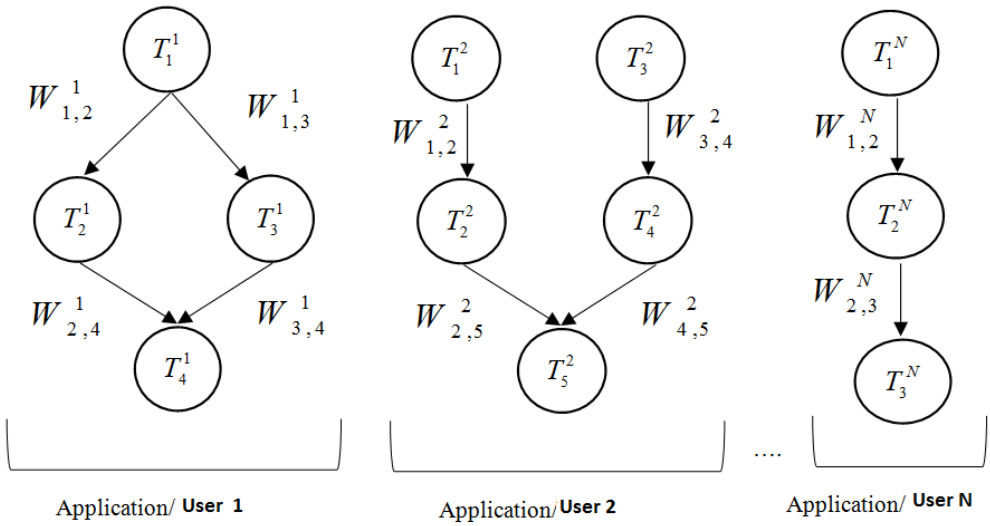

Figure 1. Application model

3.2 First come first served (FCFS) scheduling

The FCFS development was for the purpose of parallel processing and the targets will have to come link which is designated for the work established. A Sim toolkit that spines the FCFS proposal for scheduling for the errands of interior development. The distribution of the app quantified VMs to the Multitudes in a data interior based on the cloud which is the work of the element based on the VM. The policy of default had been adopted using a VM stipulated policy which distributes the VM in the FCFS method [18].

If in case the FCFS has a required resource that is

unavailable, the system will wait for its availability and the algorithm gives the resource in small parts of places the request to wait in the queue to check if the subsequent request may be serviced. This will follow another dynamic allocation towards the constraint of deadlines or the constraints of the cost that depend on it its current usage. This will further proceed an allocation of data with requests that are based on the classification where the request will fit in. In the case of the deadline constraint, in case there is a request found below its verge assessment, it will be overhauled directly and in case it is above the value it will be taken to be an allocation based on cost constraint. Considering the constraint allocation in the subsequent requests, there is a request which provides the cost efficiency that is initially allocated.

This FCFS is advantageous [19] owing to its simplicity and its minimization of the average time it takes to execute the process. It is probably one of the simplest algorithms as it has to allocate a CPU to finalize the order of the process. It also has to assume that the queue has to be managed as per the First In First Out (FIFO) method that means the job that is the first is processed first without one of the additional predilections.

3.3 Min-Min scheduling algorithm

The Min-Min procedure will prefer the assigning of reduced errands to the faster resources for running in order to ensure completion time is at its minimum. The Min-Min will calculate the minimum time taken for completion for every task that has been assigned to all related resources. This means the Min-Min will choose the minimum value two times and its description is as below [20]:

1) Calculation of the minimum completion time taken for every task assigned to all related resources.

2) Choosing a minimum value identified from among the minimum time of completion.

3) Completion of task scheduling and updating all the related variables.

4) Repeating all the above steps until such time the tasks have been assigned.

The Min-Min ensures the total time of completion of tasks to be at its minimum. There is, conversely a shortage where the Min-Min leads to resources that are fast which have a very heavy load beside with some slow possessions that have a light load. This means the Min-Min will result in a lower rate of utilization compared to the whole system [21].

3.4 Proposed greedy algorithm

In the case of a greedy algorithm, the data items found in sequence, where each takes the time and it's deeded the "best" based on convinced criterion without any regard for choices made in upcoming. The greedy procedure will influence a resolution by revenues of creation additional categorization of selections and everyone will seem to be the finest at that argument. The greedy algorithm will begin with an empty set and will then add items to the same set in the categorization until such time the set represents another solution to the problem. The components in the per iteration are as below [22]:

- A process of selection which chooses the subsequent item for being added to the set. This selection has been performed in accordance with a greedy criterion which can satisfy any local optimal consideration.
- A new feasibility check is made to determine whether this new set is feasible and to check it this can be completed in a way in which a solution to this instance can be reached.
- There is also a solution check to determine whether this new set contains a solution to this instance.

For the purpose of a new set of trades or the VMs, the Greedy-based procedure will be dependent on the scheme which is locally optimal in allocating resources. This is the actual reason as to why this is known as the Greedy procedure. Another general procedure for the Greedy-Based one may be outlined as below [23].

1) The consumers will acquiesce their trades in the pre-processing unit to that of the original trade and will from two diverse trades that are made by the classifiers.
2) Based on the job type, there are two lists created: one was for the time type job (the time type list) and one for the bwtype jobs (the bwtype List).
3) Enter another set of the VMs called thef vmList.
4) On the basis of the number of CPUs and the expectation time of the various time type jobs, it has ascended the VMs and then these time type jobs will be ascended.
5) Based on the real bandwidth of that of the VMs, expectation of the bwjobs, is inclined by the VMs and the bwtype jobs.
6) By using a local optimal algorithm, these are bundled in two tables to its local optimal VM.
7) Lastly, this is calculated with the Justice Evaluation Function (JEF) from that of the expected and the actual values, for judging the user fairness.

The JEF will be (5):

$$JEF = \Theta(AR / JR) \tag{5}$$

where, it denotes a constant ($0 < \Theta \leq 1$). The AR indicates the definite recompense that is attained by the trade and the JR indicates its expectation recompense at the period the trade is predictable. At the time the JEF is zero, the objectivity is achieved. The others are not fair and the role played by this purpose was to evaluator the actual consequence of the provision of possessions to see if it was fair. Even before the function is evaluated, it has to normalize jobs and the virtual machines. As soon as this job scheduling is completed, the subsequent step was to standardize tasks and the resources in accordance with the different QoS time or bandwidth.

4. RESULTS AND DISCUSSION

In this section, the FCFS, Min-Min scheduling and Greedy methods are used Experiments are carried out using varying number of tasks (5000 to 20000). The guarantee ratio, energy savings compared to FCFS and resource utilization as shown in Tables 1 to 3 and Figures 2 to 4.

Table 1. Guarantee ratio % for greedy

Number of Tasks	FCFS	Min-Min Scheduling	Greedy
5000	78	79	80
7500	81	82	84
10000	80	81	82
12500	76	77	78
15000	78	79	80
17500	80	81	82
20000	82	83	85

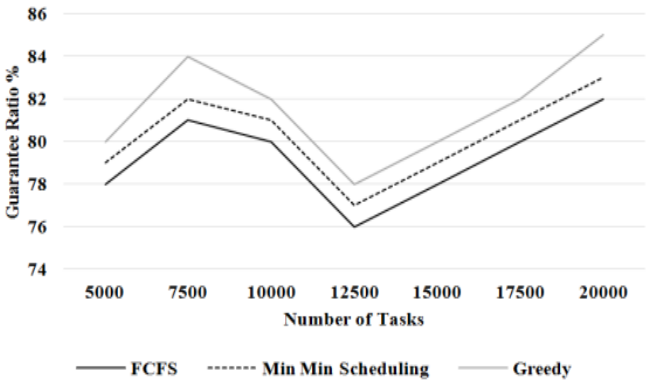

Figure 2. Guarantee ratio % for greedy

From the Figure 2, it can be observed that the Greedy has higher guarantee ratio by 2.53% than FCFS & 1.25% than min-min scheduling for 5000 number of tasks. The Greedy has higher guarantee ratio by 3.63% than FCFS & 2.4% than min-min scheduling for 7500 number of tasks. The Greedy has higher guarantee ratio by 2.46% than FCFS & 1.22% than min-min scheduling for 12500 numbers of tasks. The Greedy has higher guarantee ratio by 3.59% than FCFS & 2.38 than min-min scheduling for 20000 numbers of tasks. As the proposed greedy algorithm aims to choose locally optimal pairing of task and resource, the guarantee ratio increases.

Table 2. Energy savings % compared to FCFS

Number of Tasks	Min-Min Scheduling	Greedy
5000	2.6	2.7
7500	3.2	3.3
10000	2.9	3
12500	2.8	2.9
15000	3.2	3.3
17500	2.7	2.8
20000	2.4	2.5

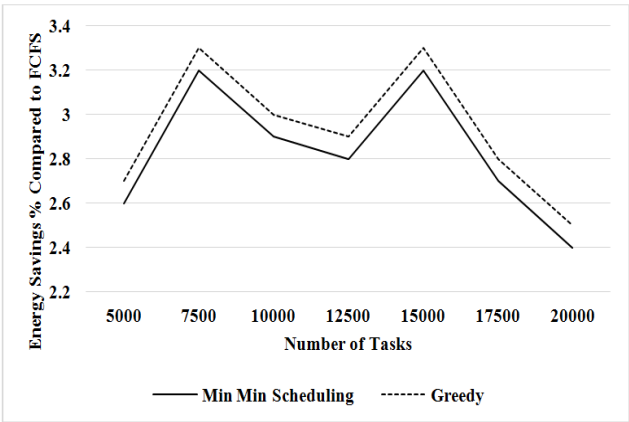

Figure 3. Energy savings % compared to FCFS

From the Figure 3, it can be observed that the Greedy has higher energy savings compared to FCFS by 3.77% than min-min scheduling for 5000 number of tasks. The greedy has higher energy savings compared to FCFS by 3.07% than min-min scheduling for 7500 number of tasks. The Greedy has higher energy savings compared to FCFS by 3.38% than min-min scheduling for 12500 numbers of tasks. The Greedy has higher energy savings compared to FCFS by 3.07% than min-min scheduling for 15000 numbers if tasks. The Greedy has higher energy savings compared to FCFS by 3.63% than min-min scheduling for 17500 numbers of tasks. The Greedy has higher energy savings compared to FCFS by 4.08% than min-min scheduling for 20000 number of tasks.

Table 3. Resource utilization % for greedy

Number of Tasks	FCFS	Min-Min Scheduling	Greedy
5000	55	56	57
7500	61	62	63
10000	70	71	72
12500	69	70	71
15000	71	72	73
17500	72	73	74
20000	70	71	72

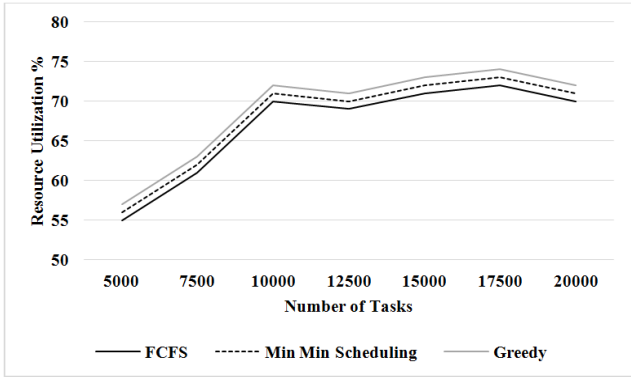

Figure 4. Resource utilization % for greedy

From the Figure 4, it can be observed that the Greedy has higher resource utilization by 3.57% than FCFS & 1.76% than min-min scheduling for 5000 number of tasks. The Greedy has higher resource utilization by 3.22% than FCFS & 1.6% than min-min scheduling for 7500 number of tasks. The Greedy has higher resource utilization by 2.81% than FCFS & 1.39% than min-min scheduling for 10000 number of tasks. The Greedy has higher resource utilization by 2.85% than FCFS & 1.41% than min-min scheduling for 12500 number of tasks. The Greedy has higher resource utilization by 2.77% than FCFS & 1.37% than min-min scheduling for 17500 number of tasks. The Greedy has higher resource utilization by 2.81% than FCFS & 1.39% than min-min scheduling for 20000 number of tasks.

5. CONCLUSION

The Virtual Machine (VM) will be its basic deployment and the management unit found in cloud computing. The users rent the VMs from CSPs for constructing their own applications and services. For the purpose of this work, the issue of global operation and its optimization in cloud computing from a CSP perspective is considered. The main goal was to provide the CSP a versatile scheduling with an optimization framework aiming to maximize the efficiency of energy in meeting user deadlines. For the purpose of this work, the FCFS scheduling and its jobs have been queued in the order of arrival and have been assigned to resources as soon as they are available. The Min-Min scheduling algorithm was the execution of the smallest of jobs that is made and the larger job has to wait. The Greedy algorithm is well suited for the heterogeneous environments of cloud resources that have a dynamic

behaviour and has been connected to the process scheduler by means of a simple and heterogeneous environment in cloud communication. The Greedy approach for the optimized profit is an ideal approach used for determining job scheduling problems. The results prove that a Greedy has a higher ratio of guarantee by about 2.53% and 1.25% for the 5000 number of tasks, by about 3.63% and 2.4% for the 7500 number of tasks, by about 2.46% and 1.22% for the 10000 number of tasks, by about 2.59% and 1.29% for the 12500 number of tasks, by about 2.53% and 1.25% for the 15000 number of tasks, by about 2.46% and 1.22% for the 17500 number of tasks and by about 3.59% and 2.38% for the 20000 number of tasks on being compared to the FCFS and the min-min scheduling. In future the more work can be done in improving the framework so as to gain maximum profit in terms of consumption. In addition, improvement of the proposed algorithm with respect to computational complexity needs to be explored.

REFERENCES

[1] Alkhashai, H.M., Omara, F.A. (2016). An enhanced task scheduling algorithm on cloud computing environment. International Journal of Grid and Distributed Computing, 9(7): 91-100. http://dx.doi.org/10.14257/ijgdc.2016.9.7.10

[2] Himthani, P., Saxena, A., Manoria, M. (2015). Comparative analysis of VM scheduling algorithms in cloud environment. International Journal of Computer Applications, 120(6): 1-6. http://doi.org/10.5120/21228-3964

[3] Beloglazov, A., Abawajy, J., Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. Future Generation Computer Systems, 28(5): 755-768. http://doi.org/10.1016/j.future.2011.04.017

[4] Masdari, M., Salehi, F., Jalali, M., Bidaki, M. (2017). A survey of PSO-based scheduling algorithms in cloud computing. Journal of Network and Systems Management, 25(1): 122-158.

[5] Rodriguez, M.A., Buyya, R. (2017). A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments. Concurrency and Computation: Practice and Experience, 29(8): e4041. http://doi.org/10.1002/cpe.4041

[6] Kaur, T., Chana, I. (2016). Energy aware scheduling of deadline-constrained tasks in cloud computing. Cluster Computing, 19(2): 679-698. http://doi.org/10.1007/s10586-016-0566-9

[7] Marahatta, A., Wang, Y., Zhang, F., Sangaiah, A.K., Tyagi, S.K.S., Liu, Z. (2018). Energy-aware fault-tolerant dynamic task scheduling scheme for virtualized cloud data centers. Mobile Networks and Applications, 24(3): 1063-1077. http://doi.org/10.1007/s11036-018-1062-7

[8] Stavrinides, G.L., Karatza, H.D. (2018). Energy-aware scheduling of real-time workflow applications in clouds utilizing DVFS and approximate computations. In 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), Barcelona, Spain, pp. 33-40. http://doi.org/10.1109/FiCloud.2018.00013

[9] Chen, H., Liu, G., Yin, S., Liu, X., Qiu, D. (2018). Erect: energy-efficient reactive scheduling for real-time tasks in heterogeneous virtualized clouds. Journal of Computational Science, 28: 416-425. http://doi.org/10.1016/j.jocs.2017.03.017

[10] Zhang, Y., Cheng, X., Chen, L., Shen, H. (2018). Energy-efficient tasks scheduling heuristics with multi-constraints in virtualized clouds. Journal of Grid Computing, 16(3): 459-475. http://doi.org/10.1007/s10723-018-9426-6

[11] Garg, N., Goraya, M.S. (2018). Task deadline-aware energy-efficient scheduling model for a virtualized cloud. Arabian Journal for Science and Engineering, 43(2): 829-841. http://doi.org/10.1007/s13369-017-2779-5

[12] Narwal, A., Dhingra, S. (2017, October). Enhanced task scheduling algorithm using multi-objective function for cloud computing framework. In International Conference on Next Generation Computing Technologies, Springer, Singapore, pp. 110-121. https://doi.org/10.1007/978-981-10-8657-1_9

[13] Rehman, S., Javaid, N., Rasheed, S., Hassan, K., Zafar, F., Naeem, M. (2018). Min-Min scheduling algorithm for efficient resource distribution using cloud and fog in smart buildings. In International Conference on Broadband and Wireless Computing, Communication and Applications, Springer, Cham, pp. 15-27. https://doi.org/10.1007/978-3-030-02613-4_2

[14] Zhang, K., Wu, T., Chen, S., Cai, L., Peng, C. (2017). A new energy efficient VM scheduling algorithm for cloud computing based on dynamic programming. In Cyber Security and Cloud Computing (CSCloud), 2017 IEEE 4th International Conference on, New York, NY, USA, pp. 249-254. https://doi.org/10.1109/CSCloud.2017.46

[15] Sarvabhatla, M., Konda, S., Vorugunti, C.S., Babu, M.N. (2017). A dynamic and energy efficient greedy scheduling algorithm for cloud data centers. In 2017 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM), Bangalore, India, pp. 47-52. https://doi.org/10.1109/CCEM.2017.9

[16] Gao, Y., Wang, Y., Gupta, S.K., Pedram, M. (2013). An energy and deadline aware resource provisioning, scheduling and optimization framework for cloud systems. In Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, Montreal, QC, Canada, p. 31. https://doi.org/10.1109/CODES-ISSS.2013.6659018

[17] Zhu, X., Yang, L.T., Chen, H., Wang, J., Yin, S., Liu, X. (2014). Real-time tasks oriented energy-aware scheduling in virtualized clouds. IEEE Transactions on Cloud Computing, 2(2): 168-180. https://doi.org/10.1109/TCC.2014.2310452

[18] Agarwal, D., Jain, S. (2014). Efficient optimal algorithm of task scheduling in cloud computing environment. arXiv preprint. International Journal of Computer Trends and Technology (IJCTT), 9(7): 344-349. https://doi.org/10.14445/22312803/IJCTT-V9P163

[19] Kaur, R., Kinger, S. (2014). Analysis of job scheduling algorithms in cloud computing. International Journal of Computer Trends and Technology (IJCTT), 9(7): 379-386. https://doi.org/10.14445/22312803/IJCTT-V29P113

[20] Zhou, Z., Zhigang, H. (2014). Task scheduling algorithm based on greedy strategy in cloud computing. The Open Cybernetics & Systemics Journal, 8(1): 111-114. https://doi.org/10.2174/1874110X01408010111

[21] Li, J., Qiu, M., Ming, Z., Quan, G., Qin, X., Gu, Z. (2012).

Online optimization for scheduling preemptable tasks on IaaS cloud systems. Journal of Parallel and Distributed Computing, 72(5): 666-677. https://doi.org/10.1016/j.jpdc.2012.02.002

[22] Baghshahi, S.S., Jabbehdari, S., Adabi, S. (2014). Virtual machines migration based on greedy algorithm in cloud computing. International Journal of Computer Applications, 96(12): 32-35. https://doi.org/10.5120/16849-6709

[23] Li, J., Feng, L., Fang, S. (2014). A greedy-based job scheduling algorithm in cloud computing. JSW, 9(4): 921-925.