Received: April 30, 2017

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Energy Aware Resource Management and Job Scheduling in Cloud Datacenter

Shyamala Loganathan[1]*, Renuka Devi Saravanan[1], Saswati Mukherjee[2]

[1]*Vellore Institute of Technology University Chennai, India*
[2]*Anna University, India*
* Corresponding author's Email: shyamalal@vit.ac.in

Abstract: A cloud system uses virtualization technology to provide cloud resources (e.g. CPU, memory) to users in form of virtual machines. Job requests are assigned on these VMs for execution. Efficient job assignment on VMs will reduce the number of hosts used. Hence, it is essential to achieve energy optimization in cloud computing environments. Therefore, in this paper, a job scheduling mechanism is proposed to assign job to a VM of the existing active hosts itself by considering job classification and preemption. So that minimizing the number of host used in allocation intern reduces the energy consumption in the Cloud datacenter. In our proposed job scheduling algorithm, categorizing the job in to three different types and assigned based on preemption policy with the earliest available time of the resource (VM) which is attached to a host. Thereby, we reduce the energy consumption by making less number of hosts in the active state and increase the utilization of active host. Finally, we conduct simulations using CloudSim and compare our algorithm with other existing methods. Significant energy savings can be obtained depending on system loads. Energy saving is about 2% to 46% with respect to the non-energy aware algorithm, 1% to 7% than the energy aware algorithms.

Keywords: Energy saving, Job allocation, Scheduling, Job types, Advanced reservation.

1. Introduction

Cloud computing is an internet based distributed computing technology. It is becoming adoptable technique by its dynamic scalability and usage of virtualized resources for many of the organizations. Thus, it represents a new paradigm for the dynamic provisioning of computing services, typically supported by state-of-the-art datacenters [1]. More recently, energy-efficient resource management in cloud system has attracted the attention of both the research community as well as the industry. Since Cloud requires that the provider should ensure the satisfaction of QoS (e.g. performance, resource availability on time, etc.) of its users, the problem of energy efficiency in cloud becomes a challenge in trade-off between performance and energy consumption. A well-known example is that Google datacenters consume more energy as would be required by a small city [2]. If nothing is done, the energy costs to run these datacenters may even overtake the huge initial installation costs, which in itself is prohibitive. In this regard, cloud datacenters consume more electricity since in many cases these are permanently switched on even when they are not used due to the expected usage. These servers, while not doing any tangible work, still consume about 70% of its peak power [3]. To reduce wastage of energy in a cloud environment, perhaps more effort needs to be spared for this waste of idle power.

A large part of the existing research has been directed towards finding suitable solutions that would contribute to the reduction of excessive energy consumption. Many research efforts, such as Intel's Cloud Computing 2015 Vision [4] draw the attention of the researchers towards the need for exploring avenues to improve power efficiency of data centers using dynamic resource scheduling approaches. We have attempted to minimize the energy consumption by allocating jobs to the VMs of existing active host thereby making unused idle host to get turned off. The challenge here is to be able to identify the servers as idle or near idle so that

these can become suitable candidates for turning off. A key consideration that needs to be addressed is how to assign an appropriate job to a suitable VM. To this end, we need to understand the type of the request being handled. Based on the user's need, a job request may be a time (deadline) sensitive or not. Hence we have made use of this time dependency as a key to classify the job request [5] in to three different categories. These are:

- Advance Reservation jobs (AR): Resources are reserved in advance. They must be made available at the specified time.
- Immediate jobs (IM): When a client submits a request, based on the resource availability, either the required resources are provisioned immediately, or the request is rejected.
- Best effort jobs (BE): These jobs are kept in a queue and resources are provided only when it is available without affecting the execution of the other two types of jobs. It can be batch jobs also.

Preemption in job scheduling increases the completion time of a submitted job. Hence a deadline sensitive request cannot be preempted. However, the same is not applicable for non-deadline sensitive requests and hence they are preemptible. We propose to exploit this and queue such jobs to be scheduled later when the resources are free or idle, thereby resulting in lesser number of idle servers in the system. Thus we exploit a combination of job request classification based on time along with preemption in job scheduling to achieve performance improvement as well as energy saving for our research.

Our contributions in the proposed system are as follows:

- First, we propose that hosts can exist in three different states.
- Second, we propose preemption aware scheduling using job classification to achieve performance improvement such as success rate and CPU utilization.
- We have developed an algorithm for job mapping to a VM of a host in a datacenter, with an aim of maximizing the contradictory requirements of resource utilization and energy saving.

The remainder of this paper is organized as follows. Section 2 provides a brief related work in. Next, Section 3 presents our energy model and the assumptions made in the system model while 4 give the problem formulation with our proposed algorithm. Section 5 provides the methodology and simulation models employed to evaluate our algorithm and test its efficacy. The evaluation results are analyzed in this section along with a comparison of our work with two existing works. Section 6 brings the rear with Conclusion where we summarize our key contributions.

2. Related work

Over the last few years, energy efficient resource management has extensively been studied. Many of these studies have employed VM consolidation for energy conservation. The authors in [6] applied limited look ahead control in order to maximize the datacenter profit via energy consumption minimization in work based on VM consolidation. The controller decides the number of physical and virtual machines to be allocated for each request. However, they have not considered how preemption can affect energy consumption where requests have preemptive priority. Authors in [7] consolidate servers using modified best-fit decreasing (MBFD) algorithm in their scheduling. They sort the hosts based on the host utilization and migrate the VM with double threshold method. Authors in [8] argues that VM migration may help to achieve successfully various resource management needs such as load balancing, power management, fault tolerance, and system maintenance. But the migration itself involves practical difficulties such as transfer delay, performance degradation etc which gives rise to lot of overhead and cost. Authors in [9] considered dynamic voltage and frequency scaling (DVFS) and deadline constraint of a job for scheduling. Optimal performance–power ratio of each host is calculated and deadline constraint jobs are given to those VM of a host. Finally consolidation is used for reduce energy where migration is used. This method is not very well suitable for a datacenter which consists of heterogeneous systems.

As part of scheduling algorithms, Selvarani [10] proposed an improved cost-based scheduling algorithm for making efficient mapping of jobs to the available resources by grouping them based on capacity in cloud. Jiayin Li [11] proposed a feedback pre-emptible task scheduling algorithm to generate scheduling with the shortest average execution time of jobs. In [12], Yang presented V-heuristics such as V-MCT for job allocation, which allocates every job in an arbitrary order of minimum completion time of the virtualized resource. In this, the completion time of the executing jobs is considered, but not the assigned jobs in the queue. Antony Thomas [13] presented a credit job scheduling in cloud computing by using user priority and task length. As part of job request types, algorithms proposed in [14]

discussed advance reservation and non-preemptive task scheduling in a grid environment. Kaushik et al. proposed a flexible reservation window scheme [15]. But it does not address the issue of low resource utilization by considering only advance reservation requests. In our proposed method, we consolidate VM based on types of requests, availability of a host and avoiding VM migration.

3. Proposed system model

The following are assumptions of the proposed model:

- Host (server) can be in one of the three states viz., Active (running) state A, idle state idle and standby state S. The energy consumption at each state is different. The active state is a high-energy state in which the hosts process users' requests and consume a lot of energy. Standby state consumes power only about 10% of the running state. The idle state is a state between the working state and the standby state in which a host consumes power about 70% of active state [16]. The system ensures that a minimum number of nodes in idle state remain within the threshold MinNum in order to respond quickly as well as to avoid the delay in the transition from standby to idle state.
- To start the system, 50% of the hosts are idle state. The other 50% hosts are in Standby state. Hosts are scaled up from Standby to Idle and Active based on the incoming requests.
- Jobs are classified into three types as advanced reservation, immediate and best effort [5] where advanced reservation and immediate can preempt the best effort jobs.
- The best effort jobs are backfilled in a queue and scheduled when the resources are free or underutilized.
- State transitions of a host are taken care of by the administrator by monitoring host utilization.
- Idle time threshold is a threshold value based on how long a system can be in idle state set by the administrator which is denoted as I_{th}. The values of I_{th}, MinNum are set by the administrator of the datacenter. These values are changed based on the incoming requests and resource usage through resource monitoring depending upon the workload of a datacenter.

Notations used in this system model are described in Table 1.

Table 1. Notations used in the system.

Notation	Description
{A},{idle},{S}	Sets of nodes in states active, idle and standby respectively.
CU_{avg}	Average CPU utilization of hosts in active state
I_{th}	Idle time threshold of a host in idle state.
H_t	't'-th possible host (server) from a set of {A}and {idle} in a datacenter.
r_{kt}	K^{th} VM in t^{th} host.
JQ	Job queue maintained in the datacenter.
RQ_{kt}	Job assigned queue on a k^{th} VM of t^{th} host.
$X(H_t) \rightarrow Y(H_t)$	Server H_t changes state from X toY.

The power consumption of the tth host during a time period $T = (t_A + t_{idle} + t_S)$ can be expressed as

$$E(H_t) = P_A t_A + P_{idle} t_{idle} + P_S t_S \quad (1)$$

where t_A, t_{idle} , t_S are the time and P_A , P_{idle}, P_S are the power consumptions per unit time of a host in states active, idle, and standby respectively.

The following are the state transitions in the system.

States: {S, Idle, A}
Actions: {Up, Down, Activate, Deactivate}

Up: When $|\{Idle\}| < MinNum \;\&\&\; CUavg > 90\%$
$\forall H_t \in \{S\}$, till $|\{Idle\}| >= MinNum$,

$S(H_t) \dashrightarrow idle(H_t)$

Down: If $\{Idle\} \neq \emptyset \;\&\&\; idle\_time(H \in \{Idle\}) > I_{th,,}$

$idle(H_t) \dashrightarrow S(H_t)$

Activate: The following transition happens automatically in the scenario described as on job arrival, scheduler finds a suitable node for scheduling the incoming job. When a job is scheduled on $H_t \in \{Idle\}$,

$idle(H_t) \dashrightarrow A(H_t)$

Deactivate: Hosts transit automatically if the following condition is satisfied:
If $RQ_{kt} == 0 \text{ for } H_t \in \{A\} \;\&\&\; JQ == \emptyset$,

$A(H_t) \dashrightarrow idle(H_t)$

4. System model

In Cloud, the end user's service requests are considered as job and each job is assigned to a virtual machine VM. The hosts are the homogeneous physical machines (or servers) that contain the computational power where the VMs run. In the proposed Cloud model shown in Fig.1, a datacenter consists of m hosts interconnected properly with the CMS (Control Management System). Based on the number of cores in a VM request, VM has different sizes as small, medium, Large, Xlarge. Each host consists of n number of VMs with different sizes, attached with the host. Job requests are assigned to these VMs and can execute in parallel on different cores of a host with different finish time. Each VM in a host maintains a request queue RQ_k, where jobs assigned by the CMS to that VM are queued.

The proposed CMS is a centralized server controlling all the hosts present in the datacenter deployed in the web portal for job submission. Clients submit their jobs to the CMS. The CMS maintains an incoming job queue JQ where all the submitted jobs to be scheduled are queued and CMS is further responsible for scheduling classifies the queued jobs based on the policy as advanced reservation, best effort and immediate. Incoming jobs to the VMs of different hosts by finding the availability of the resource (VM) and schedules based on the proposed preemption policy. The proposed scheduling process chooses the host based on the VM availability as per the algorithm in section 4.1. The main components of the proposed CMS are described below.

A. Client Request Handler

Client request handler presents a GUI for job submission. It receives the incoming requests and sends to the job classifier to identify the job type.

B. Job Classifier

Generally a job request consists of a tuple <Num_core, a_Ram, a_D, BW, Exe_time, St_time, End_time> as their requirement given by the user, St_time is start time and End_time is completion time Job classifier component in CMS helps to classify the incoming requests as any one of the following type.

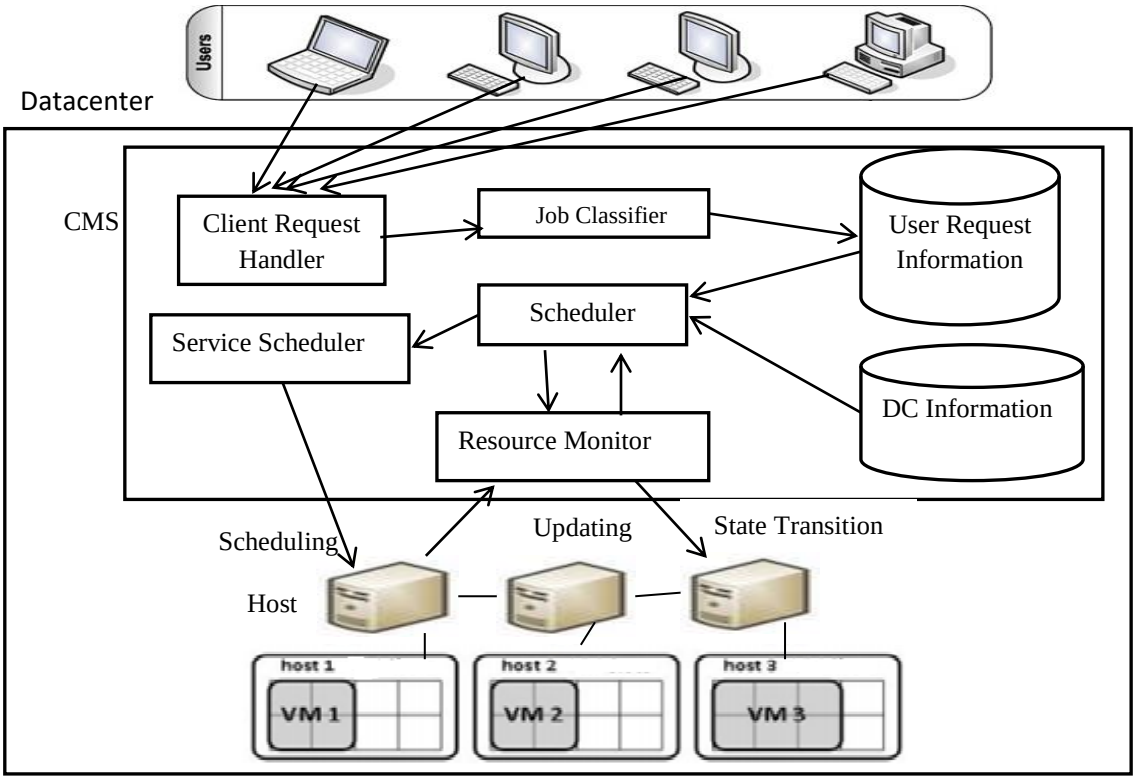

Figure.1 System Architecture

Depending on the kind of information submitted by the user during request, a job is identified as described below:

- Advance Reservation - Job request = *<Num_core, a_Ram, a_D, BW, Exe_time, St_time, End_time>*
- Best Effort -Job request = *<Num_core, a_Ram, a_D , BW, Exe_time, Nil, Nil>*
- Immediate –Job request = *<Num_core, a_Ram, a_D , BW, Exe_time, St_time,Nil>*
- Each job is labeled by the CMS and queued in job list JQ to be scheduled.

C. Resource Monitor

This component monitors the hosts and gathers any necessary information. Hosts change states as discussed earlier. During the running of the cloud, Resource Monitoring component monitors the required parameters and handles the necessary state changes of a host. It also maintains a resource list of active (running) hosts and updates it whenever state changes occur. The list contains the details of each host, details of all the VMs in the hosts and the Available Time (AT) of VMs. It calculates AT of a VM and updates the list whenever a new job is assigned. It updates AT as in Eq. (2), (5) as,

If free VM available in a host,

$$AT(r_{kt}) = CT(H_t) \quad (2)$$

where $AT(r_{kt})$ denotes the available time of a k^{th} VM in t^{th} host and $CT(H_t)$ is the current time of the t^{th} host. If none of the free VM of any host is available, we need to calculate the next available time of a VM for the incoming job j_i by considering its job queue RQ_K.

Let $j_{rkt(exec)}$ represent a job in exeution on a k^{th} VM r_k on t^{th} host. Let ST be the starttime, ET be the execution time and FT be the finishtime of this job. Let Job $j_{rkt(x)}$ represent the job assigned in the X^{th} position of RQ_k. Finish time FT of an executing job is given as in Eq.(3):

$$FT(j_{rkt(exec)}) = ST(j_{rkt(exec)}) + ET(j_{rkt(exec)}) \quad (3)$$

ST of job in the queue is given as in Eq.(4) as

$$ST(j_{rkt(1)}) = FT(j_{rkt(exec)}) + \mathcal{E} \quad (4)$$

Where Ɛ is a small slack value added for delay for next job to start. Therefore,

$$AT(r_{kt}) = \{max(FT(j(r_{kt(x)}))\}$$

$$\forall\, j(r_{kt(x)}) \in RQ_k \quad (5)$$

$AT(r_{kt})$ is the time in which VM r_{kt} is available for next incoming job. This value is updated in the resource list from the host whenever any new job is assigned in the VM's RQk. Resource monitoring also maintains a separate list of AR jobs (ARlist) with assigned VM list of each host. ARlist is used by the scheduler during scheduling to check whether any AR request is already assigned in one of the VM-s. Apart from this, for the incoming job request, resource monitoring creates a list called Available_VMlist containing all VM-s matching the requirements of the job along with AT, the availability information.

Algorithm 1: Creation of Available_VMlist
Input: Resource list, Requested VMsize, AR list.
Output: List of matching VMs with AT.

1. For t: Host $Ht \in D \ \forall$ t =1 to m
   Do For k: VM $rk \in Ht \ \forall$ k=1 to t do
   a. Find VMsize = Requested VMsize from the resource list.
   b. Find the AT using Eq. 2, 5.
   c. Add the VM which has not assigned any future AR request to available_VMlist.
   End for
   End for
2. Return matching available_VMlist.

D. Scheduler

Scheduler component in CMS identifies a suitable VM for an incoming request and assigns the job for execution. This is done by obtaining a matching VM list from the resource monitoring component and applying the proposed algorithm to assign the job on a VM r_{kt}. From the VM list the corresponding host is identified and scheduled to execute the job.

E. Service Scheduler

The service scheduler dispatches the job request to the corresponding host for execution.

F. Databases

The information about the incoming requests in JQ is stored in the user request information database. DC resource information database, on the other hand, maintains the resource list, ARlist, and the available_VM list, which gets updated whenever new jobs are assigned to a VM on a host.

4.1 Job scheduling in datacenter

The objective of the scheduler is to minimize the number of servers to save energy. To this end, we propose Energy Aware VM Available Time (EAVMAT) scheduling algorithm. When an AR or IM request comes, CMS will first check the resource availability in one of the active hosts from the list it holds and allocates the request to the hosts. If no free availability exists then checks for the earliest available host and assigns to it. If none of the possibility exists then pre-empts the request to allocate in the existing host itself without switching on the new host. If more workload comes and existing active hosts are not enough then the off state hosts are bring into on state. Since AR/IM jobs can preempt BE request the only scenario where an AR/IM job is rejected is when resources are reserved by other AR jobs at the required time and not enough resources left for the current job in any active host and idle hosts.

Algorithm 2**:** Energy Aware VM Available Time (EAVMAT) scheduling algorithm.

Input: Incoming job j in a list JQ, resource list, available_VMlist.
Output: Job allocation to a host.

1. For Each incoming job $j_i \in JQ \; \forall$ i= 1 to J
2. IF type== BE request THEN
3. IF free resource is available in active host at the requested time then allocate the request.
4. ELSE IF find the H_t from active or idle host with minimum EAT which is not assigned any AR request
5. Allocate the request.
6. ELSE put in the backfill queue.
7. IF type== IM request THEN
8. IF free resource is available in a active host at the requested time then allocate the request.
9. ELSE IF find the H_t from active or idle with minimum EAT which is not assigned any AR request
10. IF available $(EAT(V_{kt})==ST(j_i))$ THEN
11. Allocate the request.
12. ELSE call preemption();
13. IF type== AR request THEN
14. Pick first host from the list.
15. IF available resource is $ST(j_i) \; !\leq \; ST(j_{ARassigned})_t \; !\leq \; FT(j_i)$ && $(ST(j_i) \; !\leq \; FT(j_{ARassigned})_k \; !\leq T(j_i))$ THEN
16. Allocate the request.
17. ELSE call preemption();
18. ELSE reject the request.
19. Update the job list RQ
20. End while

Preemption()

21. Get all BE job in the host for the time interval T and check for flag status 1.
22. For(s=1 to number of BE jobs in a host
23. IF type==IM request THEN
24. IF $(EAT(V_{kt})==ST(j_i))$ THEN
25. Preempt the current BE request and schedule the incoming request on H_t
26. ELSE type==AR request
27. IF $(ST(j_{assigned})_t==ST(j_i)) \; \forall \; j_{assigned} \in RQ_t$ THEN
28. Preempt the current BE request and schedule the incoming request on H_t
29. End for.

5. Experiments and results

In this section we present an evaluation for our algorithm in terms of energy benefits and performance. In order to evaluate our algorithm in a large scale set up and calculate energy efficiency thereof, we expanded the CloudSim toolkit [17] to simulate Cloud architecture and perform our experiments.

5.1 Simulation set up

The implementation has been accomplished by modifying the original source code of the simulator that was written in the Java. We considered the workload as [18] having AR and BE mode requests. An additional set of IM jobs are interleaved in between to generate the mix of the three modes (AR, IM, BE) of requests randomly. From that we employed 3 sets of 1000 jobs and evaluated. Energy parameters are taken from [19] which depicts the real time values. Values assigned in simulation are given in Table 2.These parameters are kept constant at these values between different runs while comparing the results. We compared the result with four existing algorithms to see the scheduling performance and energy saving. Since the adopted method of V-MCT [12] and credit Scheduling [13] which are non-energy aware

Table 2. Values assigned in simulation.

Specification	Value
Number of host	50
Number of VMs	100 (30 small, 30 medium, 20 large, 20 X- large)
Number of requests	100-1000
Energy at active state -HP Xeon	315(Watts)
Idle state	259.5(Watts)
Stand by state	18(Watts)

existing algorithm are relatively similar to our proposed work. We considered these two algorithms for comparing our scheduling performance. To check the energy efficacy, MBFD scheduling [7] and DVFS [9] energy algorithm are considered, which exploits host utilization and migration of VM for energy saving.

5.2 Performance metrics

Various performance metrics were taken into consideration in order to measure and evaluate the proposed scheduling algorithms. These metrics include makespan, success rate, throughput, CPU utilization, energy consumption and energy saving.

Success rate:

The success rate is the ratio of number of jobs executed successfully to the total number of jobs submitted.

Makespan:

The makespan represent the maximum finishing time among all received jobs per unit time. This parameter shows the quality of job assignment to resources from the execution time perspective.

CPU utilization:

It is a ratio between the used capacity of CPU to the total CPU capacity of a host of a given time.

Energy Consumption:

Summation of the energy consumption of hosts in on state at a time.

5.3 Result analysis

We have conducted the simulation three times with the randomly generated workload and the results are obtained. These results are plotted as graphs and analyzed. The number of AR job and average duration of AR job highly influences the scheduling decision which, in true, affects the successful execution of the submitted requests [20]. In order to find the percentage of AR job in our workload, we conducted an experiment where the percentage of AR request varied to observe the effect of different percentage of AR jobs in a workload. We have taken a total of 100 requests, which contains a mix of three types of requests (AR, IM, BE) and the success percentage of these sets is plotted as shown. We find that the success rate drastically reduced for IM request when more AR requests are present in the workload due to the unavailability of the resource. Hence we consider the number of AR jobs in our workload submitted list is 20% for further evaluation of other metrics. It can be observed from Fig. 2, in that our algorithm has high success rate for AR request. We attained the guaranteed service for the AR request by preemption, which is our goal.

However, since the other two algorithms didn't consider the job classification, we analyzed metrics commonly used for the evaluation of scheduling algorithms. To find out the number of jobs executed successfully (throughput) we increased the number of incoming jobs as multiples of 100 and calculated the values for other metrics to evaluate our proposed algorithm.

Success rate of the algorithms are plotted in Fig. 3, where our proposed algorithm shows better results than other two algorithms due to pre-emption nature of the proposed algorithm. When AR and IM request needs to be scheduled, the BE request is preempted and kept in the backfill queue and hence the same host can accommodate more requests than the other two algorithms.

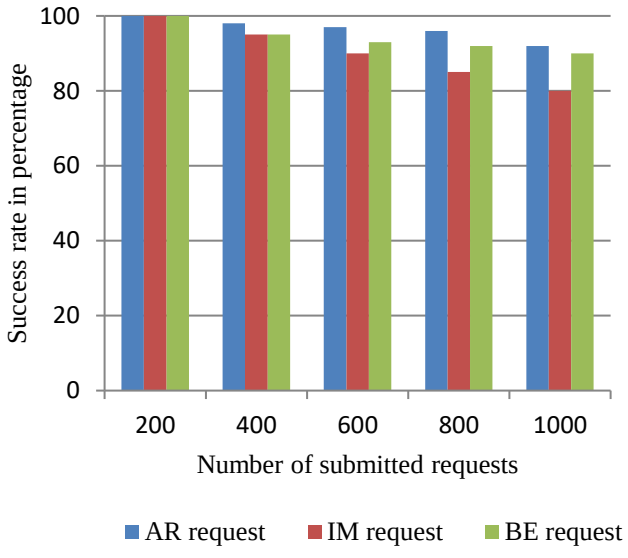

Figure.2 Success rate of proposed algorithm

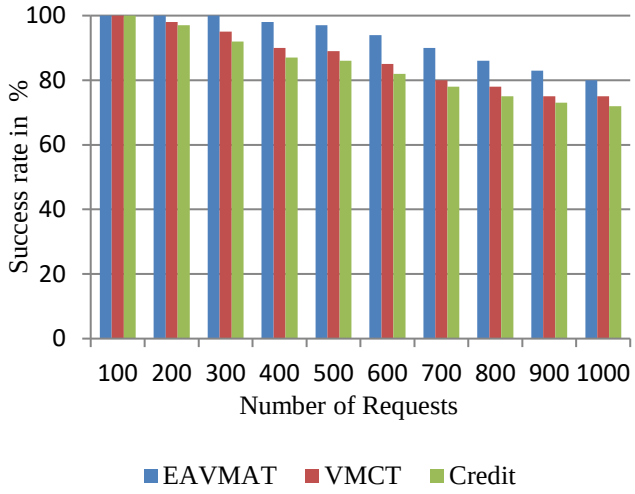

Figure.3 Success rate of scheduling algorithms

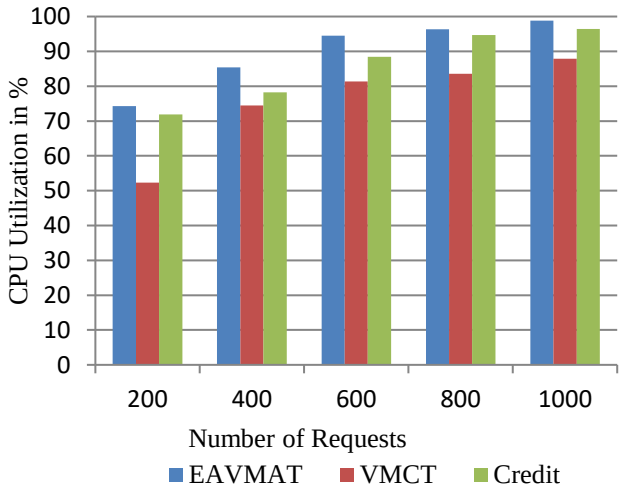

Figure.4 Comparison of utilization in percentage

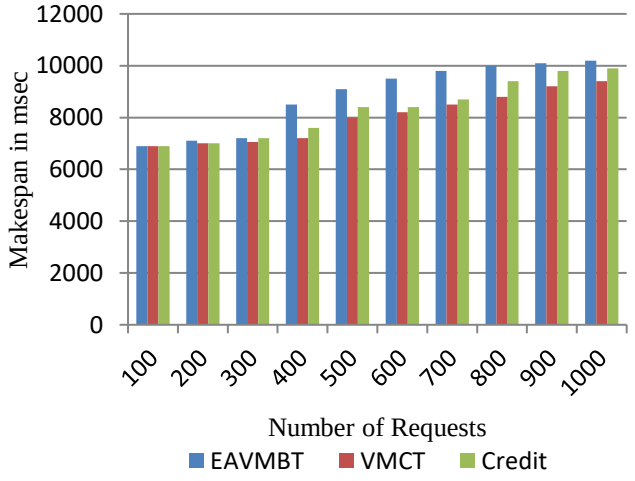

Figure.5 Comparison of makespan

In order to achieve better CPU utilization, we have considered the earliest available time of the resource to accommodate more number of requests on a same host, which is reflected in Fig. 4. Though VMCT employs minimum completion time, VMCT performs worse than others in utilization, because it takes the completion time of VM but not of same host. Therefore it schedules the requests across all the nodes arbitrarily reducing the CPU utilization of the hosts.

On the other hand, since credit scheduling uses user priority and task length to allocate a VM, it ensures the utilization by taking VM of finishing Host. However, they schedule based only on the lower task length of the host, thereby not fulfilling the basic requirements of assigning jobs to hosts. Therefore we conclude, EAVMAT has achieved the highest success rate and utilization in all cases compared to the other algorithms. This is mainly due to the fact that EAVMAT attempts to select the most suitable host that can rapidly respond and execute the given job.

Generally, the cost for preemption is an increased makespan. From Fig. 5, we notice that for EAVMAT, the makespan is higher than the other two algorithms when the number of jobs increases. Because, when more requests are submitted, BE request gets preempted and backfilled. Therefore finish time increases, which results in increase in total completion time. But it does not affect the performance, since BE jobs are non-deadline sensitive requests. VMCT algorithm performed better in makespan for more requests compared to the proposed algorithm since it does not consider preemption of any request, and arbitrarily chooses the finishing VM to assign the job. But it delays the other jobs to execute if any BE job is assigned. Hence, the success rate and throughput decreases which results in more failed job requests.

From the results we observe that the credit algorithm performs the worst among all algorithms considered with respect to makespan, success rate and utilization. This is because the credit algorithm attempts to pick a host from a computed value based on user priority and task length. For each job it computes a priority vector where the less priority jobs are preempted. This leads to the starvation and failure of jobs having less priority. Also, computing priority and accessibility lists take time, which further contribute to the increased makespan. Proposed EAVMAT algorithm shows improvement over the other two algorithms in terms of success rate and resource utilization since it takes the advantage of preemption and earliest available resources to achieve better results.

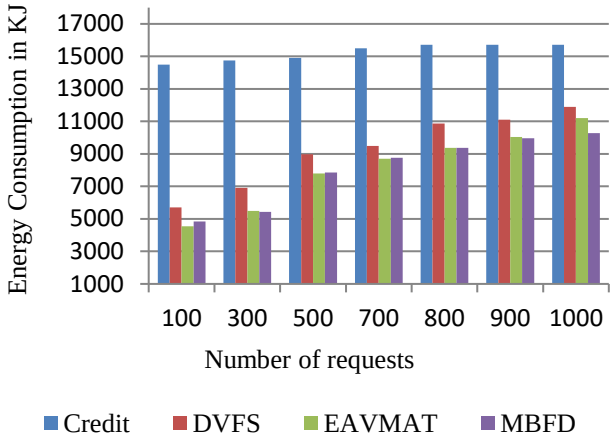

Figure.6 Comparison of energy consumption

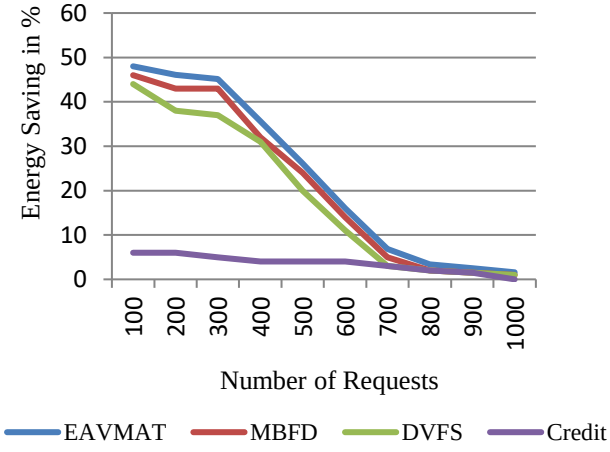

Figure.7 Comparison of energy saving

Energy metrics related results are shown in Fig. 6, 7. In order to show the energy efficacy of the proposed algorithm, 50 hosts are considered and the numbers of requests are gradually changed. The performance is compared with two energy aware existing algorithms and one non energy aware algorithm. Credit algorithm is non-energy aware algorithm.

In energy consumption, our proposed algorithm consumes lesser than DVFS and similar to MBFD algorithm. The advantage of our algorithm is that it does not take into consideration of VM migration, which is definitely a performance tradeoff. DVFS considers optimal frequency of each host and calculates a performance–power ratio of each host to allocate VM. Employing this in a highly heterogeneous environment is time consuming process. We achieved a similar energy saving like MBFD without migration, more than DVFS without consolidation which involves migration by considering job type, preemption and earliest available time of a host. It can be observed that the energy saving is about 2% to 46% with respect to credit algorithm which is a significant improvement over the non-energy aware algorithm, 1% to 7% than the energy aware algorithms.

6. Conclusion

In this paper, we presented an energy aware job scheduling algorithm in a cloud environment, EAVMAT. This paper explored the problem of job to VM mapping in cloud providers' datacenter. Our original contribution is that the proposed scheduling algorithm minimizes energy consumption and maximizes the resource utilization. We exploited job preemption as a way to reduce energy consumption in datacenters, where some requests have preemptive priority over the others. Significant energy savings can be obtained depending on system loads. At low load the gain is 46% which is significant. However, although at high loads the savings is less, it still remain sufficiently valuable. From the results we could conclude that our algorithm performs better in other metrics such as makespan, throughput, and success rate as well. Further investigation could be in the direction of the utility of this algorithm in other cloud scenario such as in a federated cloud environment including deadline sensitive request type also.

References

[1] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud Computing: State-Of-The-Art and Research Challenges", *Journal of internet services and applications*, Vol.1, No.1, pp.7-18, 2010.

[2] J. Leverich and C. Kozyrakis, "On The Energy (In) Efficiency of Hadoop Clusters", *ACM SIGOPS Operating Systems Review*, Vol.44, No.1, pp.61-65, 2010.

[3] W. Lang and J.M. Patel, "Energy Management for Mapreduce Clusters", *Proceedings of the VLDB Endowment*, Vol. 3, No.1-2, pp.129-139, 2010.

[4] K.X. Miao and J. He, "Cloud Computing and Open Datacenters", *Intel®Technology Journal*, Vol.16, No.4, 2012.

[5] S. Loganathan and S. Mukherjee, "Differentiated Policy Based Job Scheduling with Queue Model and Advanced Reservation Technique in a Private Cloud Environment", In: *Proc. of International Conf. On Grid and Pervasive Computing*, pp. 32-39, 2013.

[6] D. Kusic, N. Kandasamy, and G. Jiang, "Combined Power and Performance Management of Virtualized Computing Environments Serving Session-Based

Workloads", *IEEE Transactions on network and service management*, Vol.8, No.3, pp.245-258, 2011.

[7] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing", *Future Generation Computer Systems*, Vol.28, No.5, pp.755-768, 2012.

[8] R.W. Ahmad, A. Gani, S.H.A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, "A Survey on Virtual Machine Migration and Server Consolidation Frameworks for Cloud Data Centers", *Journal of Network and Computer Applications*, Vol.52, pp.11-25, 2015.

[9] Y. Ding, X. Qin, L. Liu, and T. Wang, "Energy Efficient Scheduling of Virtual Machines in Cloud with Deadline Constraint", *Future Generation Computer Systems*, Vol.50, pp.62-74, 2015.

[10] S. Selvarani and G.S. Sadhasivam, "Improved Cost-Based Algorithm for Task Scheduling in Cloud Computing", In: *Proc. of International Conf. On Computational intelligence and computing research*, pp. 1-5, 2010.

[11] J.Li, M. Qiu, J. Niu, W. Gao, Z. Zong, and X.Qin, "Feedback Dynamic Algorithms for Preemptable Job Scheduling in Cloud Systems", In: *Proc. of International Conf. On Web Intelligence and Intelligent Agent Technology*, Vol.1, pp. 561-564, 2010.

[12] Y. Yang, Y. Zhou, Z. Sun, and H. Cruickshank, "Heuristic Scheduling Algorithms For Allocation Of Virtualized Network And Computing Resources", *Journal of Software Engineering and Applications*, Vol.6, No.1, pp.1-13, 2013.

[13] A. Thomas, G. Krishnalal, and V.J. Raj, "Credit Based Scheduling Algorithm in Cloud Computing Environment", *Procedia Computer Science*, Vol.46, pp.913-920, 2015.

[14] W. Smith, I. Foster, and V. Taylor, "Schedulling with Advanced Reservations", In: *Proc. of International Conf. On CCGrid*, pp. 127 – 132, 2000.

[15] N.R. Kaushik, S.M. Figueira, and S.A. Chiappari, "Flexible Time-Windows for Advance Reservation Scheduling", In: *Proc. of International Conf. On Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 218-22, 2006.

[16] D.Meisner, B.T. Gold, and T.F. Wenisch, "Powernap: Eliminating Server Idle Power", *ACM Sigplan Notices*, Vol.44. No.3, pp. 205-216, 2009.

[17] R.N.Calheiros, R.Ranjan, A. Beloglazov, C.A De Rose, and R. Buyya, "Cloudsim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms", *Software: Practice and experience*, Vol. 41, No.1, pp.23-50, 2011.

[18] B. Sotomayor Basilio, *Provisioning Computational Resources Using Virtual Machines And Lease*, Report, University of Chicago, 2010.

[19] S. Long, Y. Zhao, and W. Chen, "A Three-Phase Energy-Saving Strategy for Cloud Storage Systems", *Journal of Systems and Software*, Vol. 87, pp.38-47, 2014.M.A. Salehi, P.R Krishna, K.S. Deepak and R. Buyya, "Preemption-Aware Energy Management in Virtualized Data Centers", In: *Proc. of International Conf. on CLOUD*, pp. 844-85, 2012.