Project Report

on

Energy Aware Resource Management and Job Scheduling in Cloud Datacenter.

Submitted to

Shri Ramdeobaba College of Engineering & Management, Nagpur

(An Autonomous Institute Affiliated to Rashtrasant Tukdoji Maharaj Nagpur University)

for partial fulfillment of the degree in

Bachelor of Engineering
(Information Technology)
Sixth Semester

by

MEHVASH KHAN **TEJAS BHUTADA**

MUKESH KUMAR **REETIK KUMAR**

Under the Guidance of

Dr. P. J. Assudani

Department of Information Technology

Shri Ramdeobaba College of Engineering & Management,

Nagpur-440013

2021-22

CERTIFICATE

This is to certify that the Project Report on

"ENERGY AWARE RESOURCE MANAGEMENT AND JOB SCHEDULING IN CLOUD DATACENTER"

is a bonafide work and it is submitted to

Shri Ramdeobaba College of Engineering & Management, Nagpur

(An Autonomous Institute Affiliated to Rashtrasant Tukdoji Maharaj Nagpur University)

by

MEHVASH KHAN **TEJAS BHUTADA**

MUKESH KUMAR **REETIK KUMAR**

For partial fulfillment of the degree in

Bachelor of Engineering in Information Technology,

Sixth Semester

during the academic year 2021-22

under the guidance of

Dr. P. J. Assudani
Assistant Professor, RCOEM, Nagpur

Dr. P. D. Adane
Head, Department of Information Technology
RCOEM, Nagpur

Dr. R. S. Pande
Principal
RCOEM, Nagpur

RCOEM
Shri Ramdeobaba College of Engineering and Management, Nagpur

Department of Information Technology

Shri Ramdeobaba College of Engineering & Management,

Nagpur-13

2021-22

ACKNOWLEDGEMENTS

WE EXPRESS OUR GRATITUDE TO PROF. PURUSHOTTAM J ASSUDANI (ASSISTANT PROFESSOR, DEPARTMENT OF INFORMATION TECHNOLOGY, RCOEM) FOR HIS CONSTANT FEEDBACK, TIMELESS SUPPORT AND GUIDING US THROUGHOUT THE PROJECT.

WE WOULD ALSO LIKE TO THANK THE DEPARTMENT OF INFORMATION TECHNOLOGY, RCOEM FOR PROVIDING US THE OPPORTUNITY TO WORK ON THE PROJECT.

Name of the Projectees
Mehvash Khan
Tejas Bhutada
Mukesh Kumar
Retik Kumar

CONTENTS

CHAPTER 5
CONCLUSION AND FUTURE WORK

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.

LIST OF FIGURES

LIST OF TABLES

CHAPTER-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. 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.

1.1 CLOUD COMPUTING

Cloud computing is a general term for anything that involves delivering hosted services over the internet. These services are divided into three main categories or types of cloud computing: infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS).

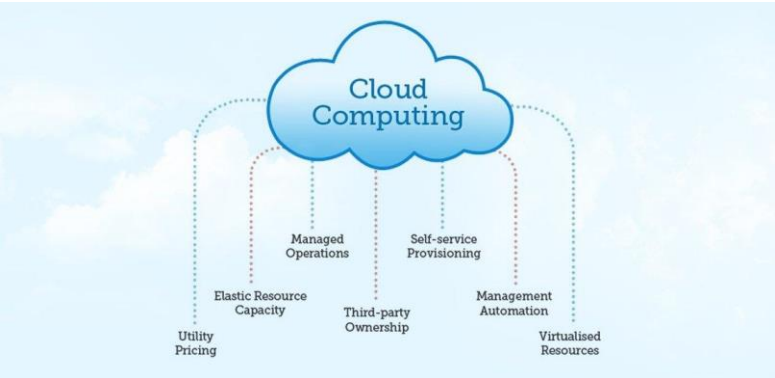

Figure 1.1 Features of Cloud Computing

1.1.1 WHAT IS CLOUD COMPUTING?

A cloud can be private or public. A public cloud sells services to anyone on the Internet. A private cloud is a proprietary network or a data center that supplies hosted services to a limited number of people, with certain access and permission settings, private or public, the goal of cloud computing is to provide easy, scalable access to computing resources and IT services. Cloud infrastructure involves the hardware and software components required for proper implementation of a cloud computing model.

Cloud Computing can also be thought of as a utility computing or on Demand computing. The name cloud computing was inspired by the cloud symbol that's often used to represent the internet in flowcharts and diagrams. Cloud computing works by enablinG devices to access data and cloud applications over the internet from remote physical servers, databases, and computers.

1.1.2 APPLICATIONS OF CLOUD COMPUTING

Online Data Storage

Organizations have a lot of data to store and with time and size of this data increases. This data can be in any format like text, image, audio, or video. Now in order to store and maintain this huge amount of data, organizations are no longer needed to set physical storage systems.

Backup and Recovery

Cloud service providers offer a lot of options for data recovery. They offer various recovery plans at different costs. Companies can decide which plan they need based on their requirements

Testing and Development

After the development of a product, testing plays a major role in finalizing it for deployment. Before the final delivery, a product needs to be tested properly. It must be tested on different machines with different infrastructures because the end-user of that product can be anywhere.

Big Data analysis

Big Data analysis involves dealing with huge amounts of data having sizes from terabytes to zettabytes (known as big data). Now for any traditional database management system, it is very difficult to maintain this amount of data. Cloud Computing allows us to store large data sets that include structured, and unstructured data, from different sources, and in different sizes from terabytes to zettabytes

1.1.3 HOW DOES CLOUD STORAGE WORK?

Cloud storage involves at least one data server that a user connects to via the internet. The user sends files manually or in an automated fashion over the Internet to the data server which forwards the information to multiple servers. The stored data is then accessible through a web- based interface.

Cloud storage systems involve vast numbers of data servers to ensure the availability. That way, if one server requires maintenance or fails, the user can rest assured that the data has been replicated elsewhere to ensure availability. While the data in a public cloud is replicated in different physical locations for fault tolerance and disaster recovery purposes, the primary or local location tends to be nearer physically to the company's facility using it so the data can be processed faster and at lower costs then, say choosing a primary location halfway around the globe.

Cloud storage management trends continue to unfold with more companies extending out to the cloud. Public clouds are managed by public cloud service providers. Their infrastructure and services include Servers, Storage, Networking, Data center operations.

1.2 CLOUDSIM

1.2.1 WHAT IS CLOUDSIM FRAMEWORK?

CloudSim is a simulation toolkit that supports the modeling and simulation of the Core functionality of cloud, like job/task queue, processing of events, creation of cloud entities (datacenter, datacenter brokers, etc), communication between different entities, implementation of broker policies, etc.

This toolkit allows to:

- Test application services in a repeatable and controllable environment.
- Tune the system bottlenecks before deploying apps in an actual cloud.

Experiment with different workload mix and resource performance scenarios on simulated infrastructure for developing and testing adaptive application provisioning techniques.

Core features of CloudSim are:

The Support of modeling and simulation of large-scale computing environment as federated cloud data centers, virtualized server hosts, with customizable policies for provisioning host resources to virtual machines and energy-aware computational resources

It is a self-contained platform for modeling cloud's service brokers, provisioning, and allocation policies.

It supports the simulation of network connections among simulated system elements. Support for simulation of federated cloud environment, that inter-networks resources from both private and public domains.

Availability of a virtualization engine that aids in the creation and management of multiple independent and co-hosted virtual services on a data center node.

Flexibility to switch between space shared and time-shared allocation of processing cores to virtualized services.

1.2.2 HOW CLOUDSIM WORKS?

As it is already mentioned that the cloudsim allows to model and simulate the cloud system components, therefore, to support its function different set of classes has been developed by its developers like:

To simulating the regions and datacenters the class named "Datacenter.java" is available in org.cloudbus.cloudsim package.

To simulate the workloads for cloud, the class named as "Cloudlet.java" is available in org.cloudbus.cloudsim package.

To simulate the load balancing and policy-related implementation the classes named "DatacenterBroker.java", "CloudletScheduler.java",

"VmAllocationPolicy.java", etc are available under org.cloudbus.cloudsim package.

Now because all the different simulated hardware models are required to communicate with each other to share the simulation work updates for this cloudsim has implemented a discrete event simulation engine that keeps track of all the task assignments among the different simulated cloud components.

1.2.3 APPLICATIONS OF CLOUDSIM

- Load Balancing of resources and tasks
- Task scheduling and its migrations
- Optimizing the Virtual machine allocation and placement policies
- Energy-aware Consolidations or Migrations of virtual machines
- Optimizing schemes for Network latencies for various cloud scenarios

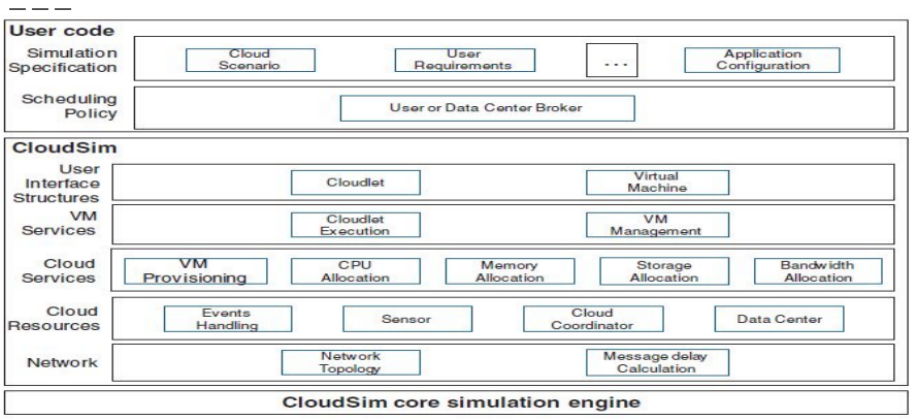

Figure 1.2.1 Layered Architecture of Cloudsim

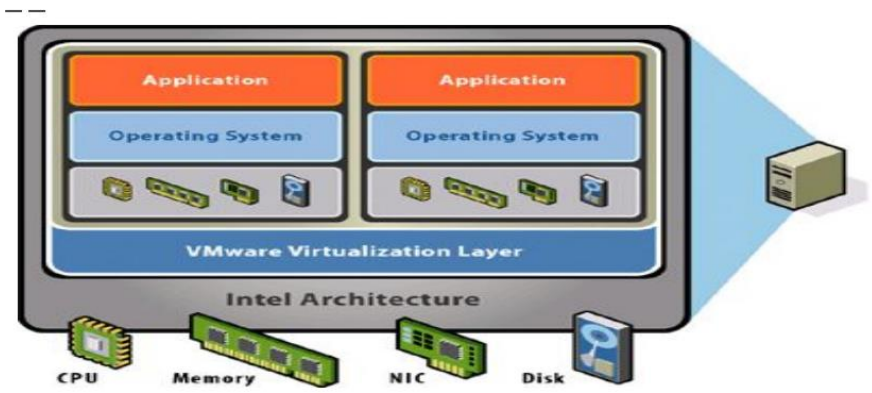

Figure 1.2.2 Components of Virtualized Infrastructure

1.3 DOCKER

1.3.1 WHAT IS DOCKER

Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. With Docker, you can manage your infrastructure in the same ways you manage your applications. By taking advantage of Docker's methodologies for shipping, testing, and deploying code quickly, you can significantly reduce the delay between writing code and running it in production.

Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow you to run many containers simultaneously on a given host. Containers are lightweight and contain everything needed to run the application, so you do not need to rely on what is currently installed on the host. You can easily share containers while you work and be sure that everyone you share with gets the same container that works in the same way.

Docker provides tooling and a platform to manage the lifecycle of your containers:

- Develop your application and its supporting components using containers.
- The container becomes the unit for distributing and testing your application
- When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.

1.3.2 DOCKER ARCHITECTURE

Docker uses a client-server architecture. The Docker *client* talks to the Docker *daemon*, which does the heavy lifting of building, running, and Distributing your Docker containers. The Docker client and daemon *can* run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API, over UNIX sockets or a network interface. Another Docker client is Docker.

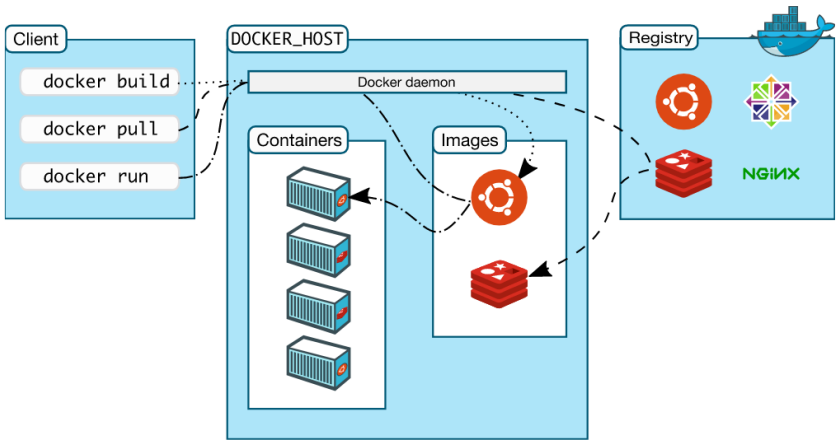

Figure 1.3 Architecture of Docker

1.3.3 VIRTUALISATION USING DOCKER TOOL

An *image* is a read-only template with instructions for creating a Docker container. Often, an image is *based on* another image, with some additional customization. For example, you may build an image which is based on the ubuntu image, but installs the Apache web server and your application, as well as the configuration details needed to make your application run.

A container is a runnable instance of an image. You can create, start, stop, move, or delete a container using the Docker API or CLI. You can connect a container to one or more networks, attach storage to it, or even create a new image based on its current state.

By default, a container is relatively well isolated from other containers and its host machine. You can control how isolated a container's network, storage, or other underlying subsystems are from other containers or from the host machine.

1. If you do not have the ubuntu image locally, Docker pulls it from your configured registry, as though you had run docker pull ubuntu manually.
2. Docker creates a new container, as though you had run a docker container create command manually.
3. Docker allocates a read-write filesystem to the container, as its final layer. This allows a running container to create or modify files and directories in its local filesystem.

4. Docker creates a network interface to connect the container to the default network, since you did not specify any networking options. This includes assigning an IP address to the container. By default, containers can connect to external networks using the host machine's network connection.

5. Docker starts the container and executes `/bin/bash`. Because the container is running interactively and attached to your terminal (due to the `-i` and `-t` flags), you can provide input using your keyboard while the output is logged to your terminal.

6. When you type `exit` to terminate the `/bin/bash` command, the container stops but is not removed. You can start it again or remove it.

CHAPTER-2

LITERATURE REVIEW

Over the last few years, energy efficient resource management has extensively been studied. Many of these studies have employed VM consolidation for energy conservation. Authors mentioned in the literature review have proposed various ideologies and innovations to minimize the energy consumption and regulate between the jobs in the datacenters.

2.1 RELATED WORK

The authors in [6] applied limited look ahead control 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 reducing 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.

CHAPTER-3
METHODOLOGY

In this chapter, the explained system model briefly deals with consolidating the VM based on types of requests, availability of the a host and avoiding VM migration to maximize resource utilization even for the jobs which do not belong to advance reservation category. The hosts have been revised to be in three different states to regulate and minimize energy consumption. The System Model Architecture is a detailed representation of how the concept is implemented With varying components to compile a resultant Job Scheduler.

3.1 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. 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 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 Ith. The values of Ith, 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.

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.

Table 3.1 Notations used in the system.

3.2 SYSTEM MODEL ARCHITECTURE

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.3.2, 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 RQk, 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

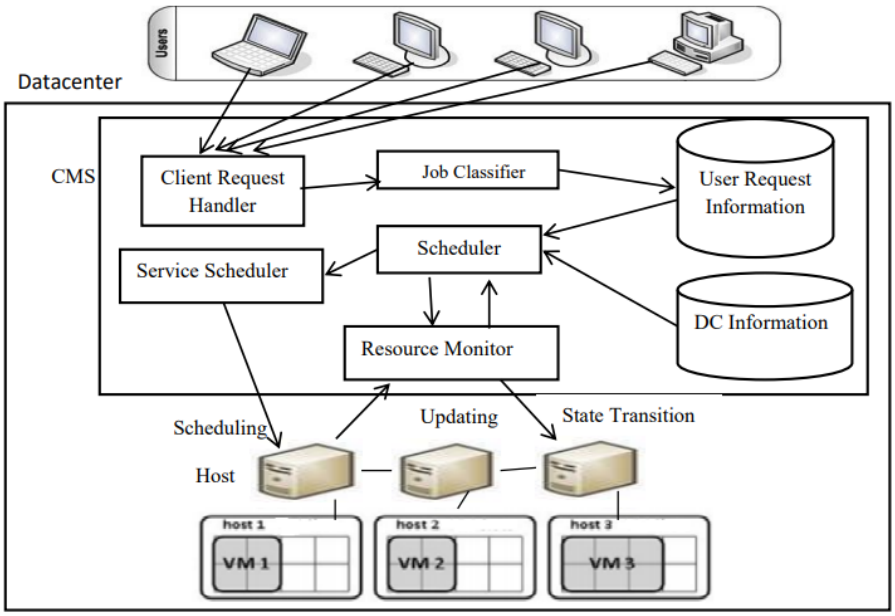

Figure 3.2 System Model Architecture

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.

3.3 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.

3.4 ALGORITHMS

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 D t =1 to m

   Do For k: VM rk Ht 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.

Algorithm 1 is aimed to produce the list of available Virtual Machines so that they can be used further for allocation to the deserving job(Best Effort, Immediate). It takes the input of all the VMsize as requested based on jobs and the available resources along with the VMs assigned to Jobs of Advance Reservation so that they may not be included for available VMs for jobs such as Best Effort and Immediate type.

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 ji ∈ JQ ∀ 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 Ht 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 Ht from active or idle with minimum EAT which is not assigned any AR request.

10. IF available (EAT(Vkt)==ST(ji)) 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(ji) !≤ ST(jARassigned)t !≤ FT(ji)

&& (ST(ji) !≤ FT(jARassigned)k !≤T(ji)) 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(Vkt)==ST(ji)) THEN

25. Preempt the current BE request and schedule the incoming request on Ht

26. ELSE type==AR request

27. IF (ST(jassigned)t==ST(ji) jassigned RQt THEN

28. Preempt the current BE request and schedule the incoming request on Ht

29. End for.

The objective of Algorithm 2 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.

CHAPTER-4

RESULTS

This chapter shows the graphical analysis of the simulation conducted. The EAVMAT algorithm shows improvement over the previous algorithms proposed by the other authors in terms of success and resource utilization since it takes the advantage of preemption and earliest available resources to achieve better results

4.1 RESULTS

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.

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)

Table 4.1 Values assigned in simulation.

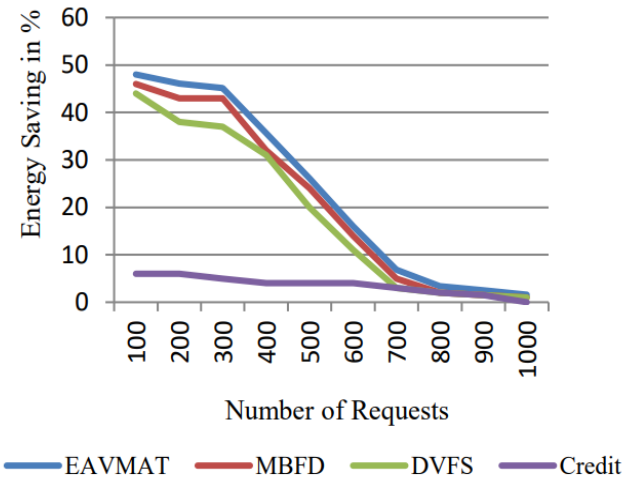

Figure 4.2 Success Rate of Scheduling Algorithm.

Figure 4.2 represents the growth of Energy Aware VM Available Time faster than those of previously proposed**.** With the increase of the number of Jobs, Energy is still being efficiently saved for upto 50%.

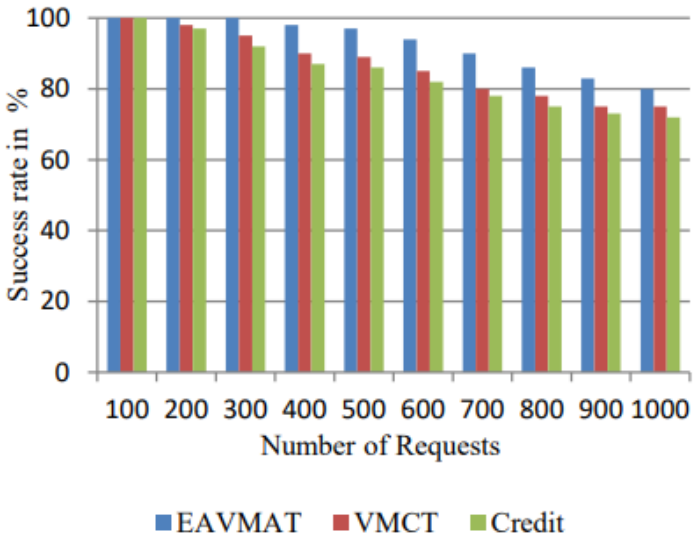

Figure 4.3 Comparison of energy saving.

Figure 4.3 represents the success rate of the EVMAT algorithm as the requests increase. The EVMAT Algorithm manages to be efficient from the rest of the algorithms.

CHAPTER-5

CONCLUSIONS AND FUTURE WORK

We presented an energy aware job scheduling algorithm in a cloud environment, EAVMAT. 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.

5.1 CONCLUSIONS

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.

5.2 FUTURE WORK

1. Create Simulation setup using CloudSim Silmulator.
2. Aim to construct the Algorithm 1 and 2 using CloudSim Simulator
3. Simulation test to be performed to check the proposed energy efficiency.

REFERENCES

1. [6] D. Kusic, N. Kandasamy, and G. Jiang, “Combined Power and Performance Management
2. of Virtualized Computing Environments Serving Session-Based Workloads”, IEEE
3. Transactions on network and service management, Vol.8, No.3, pp.245-258, 2011.
4. [7] A. Beloglazov, J. Abawajy, and R. Buyya, “Energy-Aware Resource Allocation Heuristics
5. for Efficient Management of Data Centers for Cloud Computing”, Future Generation Computer
6. Systems, Vol.28, No.5, pp.755-768, 2012.
7. [8] R.W. Ahmad, A. Gani, S.H.A. Hamid, M. Shiraz, A. Yousafzai, and F. Xia, “A Survey on
8. Virtual Machine Migration and Server Consolidation Frameworks for Cloud Data Centers”,
9. Journal of Network and Computer Applications, Vol.52, pp.11-25, 2015.
10. [9] Y. Ding, X. Qin, L. Liu, and T. Wang, “Energy Efficient Scheduling of Virtual Machines
11. in Cloud with Deadline Constraint”, Future Generation Computer Systems, Vol.50, pp.62-74,2015.
12. [10]S. Selvarani and G.S. Sadhasivam, “Improved Cost-Based Algorithm for Task Scheduling
13. in Cloud Computing”, In: Proc. of International Conf. On Computational intelligence and
14. computing research, pp. 1-5, 2010.
15. [11]J.Li, M. Qiu, J. Niu, W. Gao, Z. Zong, and X.Qin, “Feedback Dynamic Algorithms for
16. Preemptable Job Scheduling in Cloud Systems”, In: Proc. of International Conf. On Web
17. Intelligence and Intelligent Agent Technology, Vol.1, pp. 561-564, 2010.
18. [12] Y. Yang, Y. Zhou, Z. Sun, and H. Cruickshank, “Heuristic Scheduling Algorithms For
19. Allocation Of Virtualized Network And Computing Resources”, Journal of Software
20. Engineering and Applications, Vol.6, No.1, pp.1-13, 2013.
21. [13]A. Thomas, G. Krishnalal, and V.J. Raj, “Credit Based Scheduling Algorithm in Cloud
22. Computing Environment”, Procedia Computer Science, Vol.46, pp.913-920, 2015.

23. [14]W. Smith, I. Foster, and V. Tay
24. lor, “Schedulling with Advanced Reservations”, In: Proc.
25. of International Conf. On CCGrid, pp. 127 – 132, 2000.
26. [15]N.R. Kaushik, S.M. Figueira, and S.A. Chiappari, “Flexible Time-Windows for Advance
27. Reservation Scheduling”, In: Proc. of International Conf. On Modeling, Analysis, and
28. Simulation of Computer and Telecommunication Systems, pp. 218-22, 2006.
29. [16] D.Meisner, B.T. Gold, and T.F. Wenisch, “Powernap: Eliminating Server Idle Power”,
30. ACM Sigplan Notices, Vol.44. No.3, pp. 205- 216, 2009.
31. https://www.researchgate.net/publication/319403592_Energy_Aware_Resource_Management_and_Job_Scheduling_in_Cloud_Datacenter?enrichId=rgreq-5214e2d51b158d57f176fcc85fb227a2 Research Paper on Energy Aware Resoource Management and Job Scheduling in Cloud Datacenter
32. https://docs.docker.com/get-started/overview/ docker tool documentation.
33. http://www.cloudbus.org/cloudsim/ Cloudsim tutorials and documentation.
34. https://www.enterprisestorageforum.com/cloud/cloud-storage/ How cloud storage works.