A Novel Approach for Energy Management in Cloud using Load Balancing Mechanism

Yuvraj Arora,
School of Computer Science and Engineering
Lovely Professional University
Phagwara, India
akaran320@gmail.com

Shivam Tiwari,
School of Computer Science and Engineering
Lovely Professional University
Phagwara, India
tiwarishivam96624@gmail.com

Jyotish Kumar Ram,
School of Computer Science and Engineering
Lovely Professional University
Phagwara, India
jyotish.jhon@gmail.com

Moin Hasan
School of Computer Science and Engineering
Lovely Professional University
Phagwara, India
moin.25676@lpu.co.in

Abstract

Cloud computing is the virtual platform that consists of virtual desktops, software platforms, applications, data storage, and servers that can be accessed over the internet. In Cloud Computing, the consumption of immense amounts of energy is one thing that we are concerned about. Computational services are being provided to the users and a lot of heat is being generated by the hardware, so cooling is also necessary. A lot of energy is being used by the hardware. Because of the consumption of a large amount of energy, more amount of carbon prints are generated and released into the environment. As carbon emission is more by the data centers, it may lead to global warming and other climatic changes which are bad for the environment. Thus, the amount of energy consumption has to be reduced which will increase the sustainability and productivity of the system. So here we are mainly using an algorithm that works on the principle of the Load Balancing Mechanism.

Keywords—cloud computing; green computing; load balancing; sustainability; task scheduling; vm migration.

1. Introduction

Cloud Computing like infrastructure was first developed in 1960 by Joseph Carl Robnett Licklider to interact with people and data from anywhere at any time along with his work on ARPANET. He is the one who laid the foundation of the cloud computing that revolutionized today. In 1983, CompuServe was the primary company that provides its users with a touch amount of disk space that might be used to accumulate any files they prefer to upload bit like a cloud [1]. Cloud computing is the virtual cloud that consists of virtual desktops, software platforms, applications, data storage, and servers that can be accessed over the internet according to the requirement. The infrastructure of cloud computing is hosted at a far-off data center and managed by a cloud services provider. The cloud services provider provides these resources available to the user over the internet [2].

Virtualization facilitates cloud service providers to form maximum use of their data center resources. Data centers are located in remote areas where high-capacity servers and storage systems help in the deployment of Cloud applications efficiently. The demand for cloud-based services is increasing day by day so enormous data centers were established to fulfill the required demand which increases in consumption of a high amount of electricity. Energy-efficient models are needed for data centers to reduce operational costs while maintaining vital Quality of Service [3]. Examples- sustainable use of resources (put idle physical servers on sleep mode), data center area must have a proper cooling mechanism. On the other side, the flexibility of resources of cloud computing is the most important thing that every user wants these days. As increasingly more cell gadgets have become taken into consideration as a predominant intake factor for faraway customers in mainstream business, energy management has been a bottleneck for the correct functioning of services at the customer's end [4].

Cloud services are software, infrastructure, and platforms programs that are provided by cloud service providers and made available to users or customers through the internet. There are three main types of Services: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Each service facilitates the flow of user or customer data from front-end clients through the internet, to the cloud service provider's systems. IaaS (Infrastructure as a Service) means the cloud services provider manages and maintains the infrastructure (the servers, network, virtualization, and data storage) online for their users. The user or customers has to access their

service through an API or dashboard [5]. The user or customer uses the dashboard just like his own operating system, and applications, while the cloud services provider is responsible for the maintenance, outages, repairs, and hardware issues of infrastructure (hardware, networking, hard drives, data storage, and servers). Examples of IaaS are Azure Virtual, Amazon EC2, and Google Compute. PaaS (Platform as a Service) means the hardware and software development tools are provided cloud service provider and the user can use these resources for the development of software applications [6]. For the software developers and programmers, PaaS gives its users a shared cloud platform that can be managed by multiple users at a time for application development and management without they have to build and maintain the infrastructure. Examples of PaaS are AWS Beanstalk, IBM Cloud, and SAP Cloud Platform. SaaS (Software as a Service) is a service that provides a software application that the cloud service provider manages for its users. Typically, SaaS applications are websites or mobile applications that users can access these service via the internet [7]. The service providers take care of software updates, bug fixes, and other general software maintenance for the user, and the user or customer can use the cloud applications via a dashboard or API. In SaaS users are not required the need to have a software application installed locally on each user's computer, allowing multiple users to access the software at the same time. Examples of SaaS are Dynamics 365, Trello, Hub Spot, Bit.ai, and Slack.

Data centers are the back-end hub of large-scale computing resources (hardware/software) in the cloud infrastructure. These resources in the data centers are provisioned to consumers as subscription-based services. The number of the consumer has been growing exponentially for cloud services, and to satisfy the growing demand for the cloud services, the data centers, and their resource capacities are also being increasing day by day. For example, Google has installed about 9, 00,000 servers in its different data centers worldwide to fulfill the service demands of its vast consumer base. Large-scale data centers consume a high amount of energy which has a significant impact on the working budget of data centers and the environment [8]. Carbon dioxide emitting an excessive amount from the high energy consumption in data centers is also a reason for global warming. According to an estimate, a high-performance server (with 300W power consumption) emits about 1300 kg of CO2 in a year. Hence, a data center with a considerable number of similar servers installed can produce about 80–116 metric megatons of CO2 each year [9].

The reason for the high consumption of energy in the data centers is because of its huge number of computing resources and other infrastructure like the cooling system as the hardware release heat during computation, low utilization of the resources is also a crucial factor in its high energy consumption. The average utilization of the computing resources in the data centers has been reported as about 40% by Tomas and Johan [10]. In the idle state, a server can consume approximately 70% of the energy [11]. The consumption of energy by resources when no output is produced (resources are not utilized) has more severe economic effects. Primary techniques like virtualization and resource consolidation are being used in the data centers to increase resource utilization. In virtualization, a physical host (server) is divided into multiple virtual machines (VMs) which are scheduled independently to multiple consumers in the cloud. It increases the machine utilization and therefore energy efficiency per unit of work performed increases. Resource consolidation through VM migration further increases resource utilization and minimizes energy consumption. The operational VMs are migrated across the physical hosts for resource consolidation in a data center. The unused physical machines (hosts) are put on hibernate/sleep mode to improve energy efficiency [12].

In this paper, we propose an algorithm for Scheduling the virtual machines and migration of virtual machines on physical machines that is sustainable use of resources for green cloud computing. Our algorithm assigns a VM from one physical machine to another physical machine that has idle MIPS to operate multiple VM at the same time and put the physical machine with no VM in sleep mode which gives the least value of an increase in energy consumption, provided by all the active and idle physical machines.

The rest of the paper is organized as follows: Next section discusses the proposed work which includes the cloud and data center model, physical machine model, virtual machine model, task model, the algorithm proposed, and its working explanation Section 3 is for experimentation and results. This section contains the details about where we carried out our experiment and its outcome. Concluding remarks and references are given in the last section.

2. Proposed Work

In the presented load balancing mechanism model, the virtualized data center resources (hosts or physical machines) are scheduled to execute the independent and dynamically arriving tasks. VMs are deployed over the physical hosts as per tasks load and completion deadline. During task scheduling, the first approach is to find a schedule such that a minimum

number of physical machines are used for the deployment of virtual machines to save energy. If no MIPS is found on the physical machines after already deployed VMs, the second approach is to deploy a new VM as per the MIPS requirement on the next physical machines. While deploying a new VM firstly already operational host is considered. A New host is started if the already operational hosts do not have enough MIPS to execute the task. The task is rejected if no schedule is found to meet the task requirements.

Notation	Definition
MIPS	Millions instruction per second
PM	Physical machines
VM	Virtual machines
C	cloud
pm_{ij}^{b}	working state of physical machine
pm_{ij}^{c}	Operation mode of physical machine
$VM_{ak}^{performance}$	Total performance of VM on host a_k
$CPU_{k}^{performance}$	Total performance of CPU on host k
$core_{uk}^{performance}$	Total performance of core.
T	Set of independent tasks
t_l^a	arrival time
sl_i	Slack time of task t_i
t_l^n	size (number of instructions measured in MI)
t_l^d	Deadline time
st_l^s	start time
t_l^e	execution time
t_l^f	expected finish time
PM_i	i no of physical machine in data center.
pm_{ij}^{a}	Processing power of physical machine
VM_{ij}	No of virtual machines deployed on physical machines.
$mt(vm_{pk})$	migration time
$ct(vm_{jk})$	creation time of new VM
$st(pm_k)$	start-up time of the host
$ct(vm_{jk})$	creation time
spt_{ejk}	suspension time
pm_i^b	buffer power of physical machine
pm_i^{lp}	Left power in a physical machine after scheduling.
vm_j^d	power of deployed VM

2.1 Cloud and Data Center Model

A Cloud environment is considered as a set C of n regionally distributed data centers and each data center is represented by the set of physical Machines PM in it [13].

$$C = \{PM, PM_2, \dots\dots\dots, PM_n\} \quad (1)$$

A set of physical machines in a data center is represented as

$$PM_i = \{pm_{i1}, pm_{i2}, \dots\dots\dots, pm_{im}\}\ (1 \le i \le n) \quad (2)$$

2.2 Physical machine Model

Commercially, each data center in the cloud consists of more than 5,000 physical machines and are characterized by their processing power (pm_{ij}^{a}) expressed in million instructions per second (MIPS), working state(pm_{ij}^{b}), and operation mode(pm_{ij}^{c}) [14] .

$$pm_{ij} = (pm_{ij}^{a}, pm_{ij}^{b}, pm_{ij}^{c}) \quad (3)$$

The working state of a physical machine pm_{ij}^{b} is represented in the binary form as,

$$pm_{ij}^{b} = \begin{cases} 0 & if\ the\ host\ is\ in\ a\ failed\ state. \\ 1 & if\ the\ host\ is\ in\ a\ working\ state. \end{cases} \quad (4)$$

The operation mode of a host pm_{ij}^{c} is represented in the binary form as,

$$pm_{ij}^{c} = \begin{cases} 0 & if\ the\ host\ is\ in\ sleep\ mode. \\ 1 & if\ the\ host\ is\ in\ active\ mode. \end{cases} \quad (5)$$

2.3 Virtual Machine Model

Virtualization is the way to provide sharable access to physical machines. It is the main key of cloud computing, without virtualization we can't think of cloud computing. Sharing in the virtual machine are of types time shared and space shared. Virtual machines can be configured on-demand or may-be preconfigured [15].

q Number of VMs deployed on host pm_{ij} are denoted by the set VM_{ij}.

$$VM_{ij} = \{vm_{ij1}, vm_{ij2}, \dots\dots\dots, vm_{ijq}\}$$

$$(1 \le i \le m,\ 1 \le j \le n) \quad (6)$$

Performance of a VM VM_{ak} on physical machine pm_k is observed as

$$VM_{ak}^{performance} = \sum_{u=1}^{\omega} core_{uk}^{performance} \quad (7)$$

Where ω is the number of cores included in the VM. The total CPU performance of a host is observed as

$$CPU_k^{performance} = \sum_{a=1}^{n} VM_{ak}^{performance} + \sum_{b=1}^{m} core_{bk}^{performance} \quad (8)$$

Where n is the number of deployed VMs on a host PM_k and m is the unused CPU core [16].

2.4 Task model

A finite set of tasks $T = \{t_1, t_2, t_3, \dots, t_q\}$ are there to be executed may be independent or dependent, Dependent tasks depend on workflows, ensembles, etc. A task may be Priorities on the differences (delay-sensitive, delay insensitive, etc.) [17]

A task t_l submitted for execution is modelled as

$$t_l = (t_l^a, t_l^n, t_l^d, t_l^s, t_l^e, t_l^f) \quad (9)$$

Where; t_l^a is the arrival time, t_l^n is the size (number of instructions measured in MI), t_l^d is the deadline, st_l^s is the start time, t_l^e is the execution time, and t_l^f is the expected finish time.

If t_l is allocated to VM vm_{ijk} for execution, then the *execution time* and the *expected finish* time are computed as

Execution time, $$t_l^e = \frac{t_l^n}{vm_{ijk}^q} \quad (10)$$

Expected finish time, $$t_l^f = st_l^s + t_l^e \quad (11)$$

The start time of the task t_i on VM vm_j at host pm_k (i.e.,st_{ijk}^s) can be calculated in five different ways.

• If execution of the task can be started after a currently executing task t_e, then the start time of the task t_i is the finish time of the currently executing a task, i.e.,

$$st_{ijk}^s = ft_{ejk} \quad (12)$$

• If a new VM is to be deployed to schedule the task, then its start time is the addition of its arrival time and creation time of the new VM, i.e.,

$$st_{ijk}^s = t_i^a + ct(vm_{jk}) \quad (13)$$

• If an executing task may be temporarily suspended based on its slack time to accommodate a new task that can complete the execution during the slack time, then the start time of the new task is the addition of its arrival time and suspension time of the currently executing task, i.e.,

$$st_{ijk}^s = t_i^a + spt_{ejk} \quad (14)$$

• If a VM is migrated to another host and a new VM is deployed on the host (from which VM is migrated) to schedule the task, then the start time of the task is the addition of its arrival time, migration time of the VM and creation time of new VM, i.e.,

$$st_{ijk}^s = t_i^a + \sum_{p=1}^{|p|} mt(vm_{pk}) + ct(vm_{jk}) \quad (15)$$

• If a new host is started to schedule the task, then its start time is the addition of its arrival time, start-up time of the host, and creation time of a VM, i.e., [18]

$$st_{ijk}^s = t_i^a + st(pm_k) + st(vm_{jk}) \quad (16)$$

3. Proposed Algorithms

3.1Algorithm 1: Task scheduling

```
# In[ ]:
while Task1:
    for Each Physical Server in Datacenter:
        if VM not NULL:
            for each VM in PhysicalServer:
                If VM[mips]>=Task[MI]:
                    Executing Task.n
                    if Task1 is not empty:
                        While Task.K in Task1:
```

```
10.                     if VM[mips]>=Task[MI]:
11.                         Executing Task.j
12.                         Execution of Task.j
13.                      Task.j Execution Complete

14.                         Task.j(pop)
15.                     else:
16.                         Task.k+=1
17.                     END if
18.                 END if
19.                 Execution Task.n
20.                 Execution of Task.n
21.                 Task.n Execution Complete
22.                 Task.n(pop)
23.             else:
24.                 CreateVM(Task.n[Mips],OS)
25. #VM created according to Task Requirement

26.                 MIGres.append(VM)
27.             end if
28.         END for
29.     else:
30.         CreateVM(Task.n[Mips],OS)
31.         Migres.append(VM)
32.     END if
33. END for
34. else:
35.     Shift to Next Physical Server
36. end While
```

In this pseudocode (Algorithm 1), step1 and step2 N number of the task will be inputted by the user and these inputted tasks came to the data center for their execution. Step 3. The algorithm looks for the available physical machine in a data center for the deployment of a virtual machine. Step 4 to step 25. VM will be deployed on physical machines according to the MIPS of the task required and MIPS available on the physical machines. Step 26 to step 36. If MIPS of the VM is greater than the available MIPS of the physical machine, the algorithm looks for a new physical machine. All this process will carry out until execution of all the tasks.

3.2 Algorithm 2: VM Migration

```
1.  # In[ ]:
2.  for each VirtualMachine In MIGres-1:
3.     k1=0
4.     #n=current Virtual Machine
5.     if MIGres[n] is not NULL:
6.
    SpareAdded=PhysicalServer[n].SpareResour
    ces
7.
    resSpare=PhysicalServerEnergyUnused[n]-
    SpareAdded
8.         for Each VirtualMachine+1.j in MIGres:

9.             while k1 in range MIGres:
10.                if MIGres[j][k1]<=resSpare and k1
    in MIGres:
11.                    Migrate VM of PhysicalServer j
    to N PhysicalServer
12.
    MIGres[mit].append(MIGres[j].pop(k1)
13.                else:
14.                    k1+=1
15.                END if
16.        END for
17.    END if
18. END for
19. for each checkVM in MIGres:
20.    if MIGres[check] is NONE:
21.       PhysicalServer[check] is in sleep MODE
22.        ANSWER.append(check)
23.    END if
24. END for
```

In this pseudocode (Algorithm 2), step1 to step5. Algorithm look for the deployed virtual machine that is currently executing the task assigned to that VM. Step6 to step10. Searching of a virtual machine will proceed based on of MIPS (buffer MIPS+MIPS left after deployment VM) available on the physical machine. Step 11 to step 18. VM will migrate if the MIPS of a virtual machine is smaller than or equal to the MIPS available on the physical machine. Step19 to step21. After the migration of VM, the physical machine with no virtual machine will go to sleep mode.

3.3 Explanation

1. Here we have task $T = \{t_1, t_2, \dots, \dots, \dots, t_n\}$.
2. We have 100 physical machines of 2000 MIPS each.
   Physical machines are $pm_1, pm_2, \dots, \dots, pm_{100}$ of 2000 MIPS.

3. Firstly, the scheduling algorithm creates virtual machines on physical machines according to the power needed to complete the task. Since we have 2000MIPS in each physical machine, so N number of the virtual machine was created whose summation value will not exceed 2000MIPS in a physical machine.

$$no.of\ Virtual\ machines\ created\ is,$$

$$N = \sum_{i=0}^{n} \quad MIPS\ of\ vm_n \leq 2000 \qquad (17)$$

4. After the utilization of maximum power of a physical machine by creating virtual machines which are executing the tasks, the algorithm looks for other physical machines for the execution of the next task.
5. The above process will go on until each task gets its virtual machine for execution in physical machines.
6. Now here load balancing mechanism algorithm come into play. Because each physical machine has some buffer power remaining with them and a small amount of power left after the creation of virtual machines. This algorithm checks if any virtual machine running on the physical machine is less than or equal to the summation of buffer power and power left after the creation of a VM.

   VM will be created in a physical machine if

   $$\sum pm_i^b + pm_i^{lp} \leq vm_j^d \qquad (18)$$

   Where pm_i^b is buffer power of a physical machine i, pm_i^{lp} is leftover power in a physical machine i after deployment of VM while scheduling, and vm_j^d is the power of deployed VM on another physical machine j.

7. If the algorithm finds the virtual machine of power less than or equal to the summation of buffer power and remaining power after scheduling the virtual machine on the physical machine. It will migrate the virtual machine to the physical machine whose overall power (buffer power + remaining power after creation on virtual machines) hasn't been used.
8. This process will continue and due to this mechanism, some of the physical machines utilize their maximum capacity and some of the physical machines become free from execution and goes to sleep mode.

4. Experimentation and Results

In our experiment, we used the Load Balancing Mechanism approach for Energy Management in Cloud.

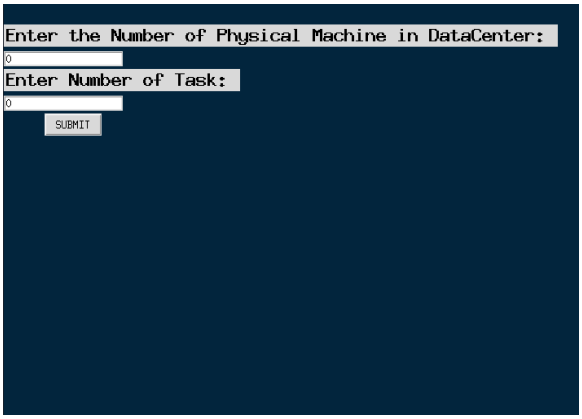
Load Balancing Mechanism is scheduling the systems using the Algorithm that we have designed in Cloud Computing that efficiently allocates tasks to the resources to balance the system load that leading to a reduction in execution cost with faster convergence.

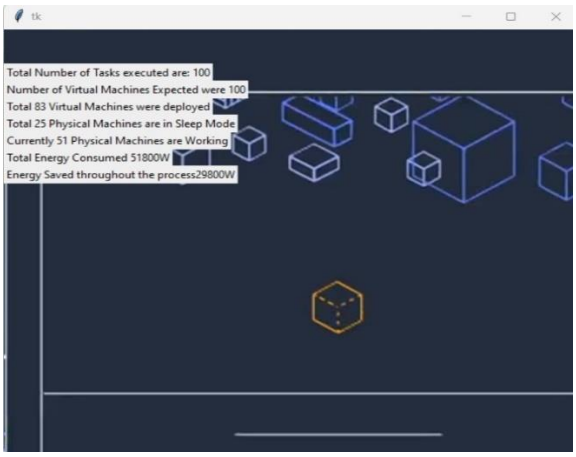
Experimentation

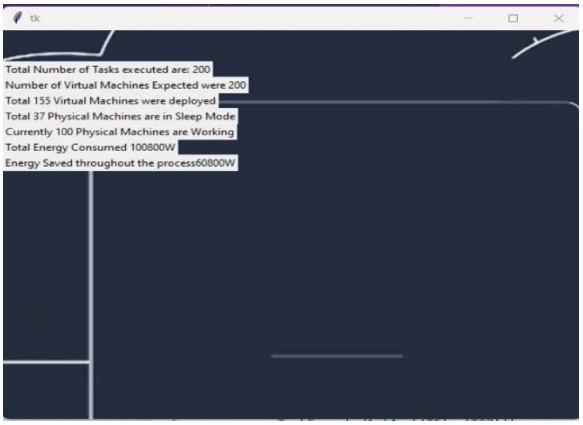
Our experiment was carried out in python programming language. We framed the code there to create data centers, physical machines, virtual machines, tasks, physical servers, and algorithms. And we designed a simple GUI for input and output purposes.

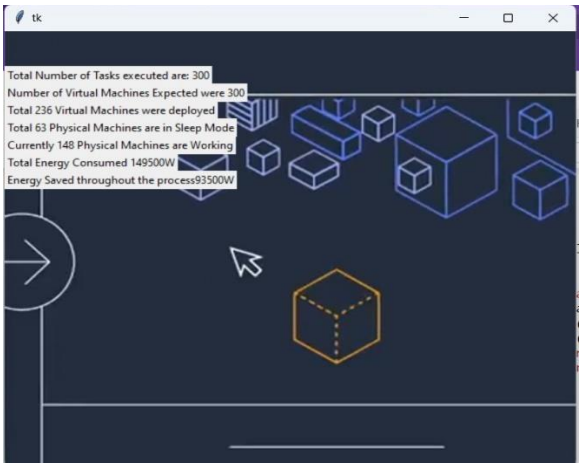
Results

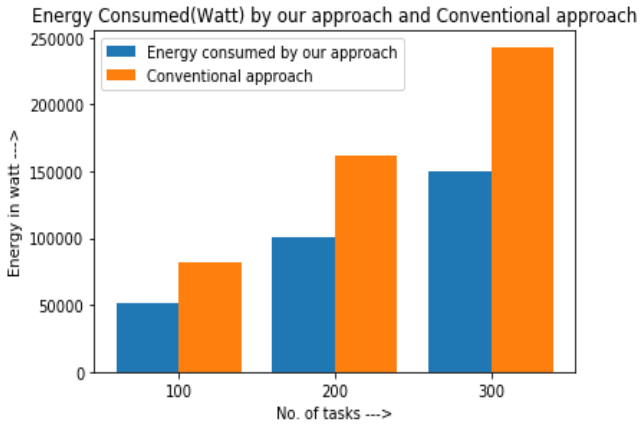
In our experiment, we took 100 tasks for first time 200 tasks for second time, and 300 tasks for the third time for execution. After analysis of all the execution of tasks, we can say that our approach has sufficient potential to save the energy in the data center with the increase in the tasks. The Graphs represented below gives an idea that our approach is one of the ways to reduce energy consumption in the data centers.



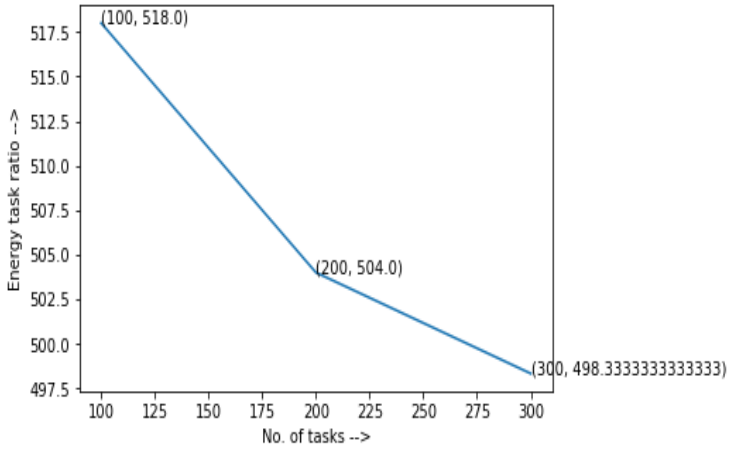








The above bar graph is the comparison of our experimental approach and conventional approach for energy consumed for task execution.



The above line graph is the graph generated from the energy task ratio and the number of tasks executed. We can see, with the increase in the number of task execution, energy task ratio decreasing as it's a declining line graph

4. Conclusion

In this paper, we have investigated the need for power consumption and energy efficiency in the cloud computing model. It has been shown that there are systems which not doing tasks that they are capable of but are done by another system at the same time, are responsible for a high amount of power dissipation in the cloud. So we pointed out that problem and proposed an algorithm for sustainable use of

resources. The possible way of designing an energy efficiency model has also been studied. Finally, we have shown the future research direction and the continuity of this work for next-level implementation.

References

[1] "What is cloud computing? A beginner's guide | Microsoft Azure." https://azure.microsoft.com/en-in/overview/what-is-cloud-computing/ (accessed Mar. 05, 2022).

[2] "Cloud computing | IBM." https://www.ibm.com/cloud/learn/cloud-computing-gbl (accessed Mar. 07, 2022).

[3] "NIST CLOUD COMPUTING STANDARDS ROADMAP".

[4] "Why Worry About Energy?," 2015.

[5] "Data Centers - Google." https://www.google.com/about/datacenters/ (accessed Apr. 02, 2022).

[6] M. Hasan and M. S. Goraya, "Resource Efficient Fault-Tolerant Computing Service Framework in Cloud."

[7] A. Sheth, S. Bhosale, and H. Kadam, "Emerging Advancement and Challenges in Science, Technology and Management " 23 rd & 24 th April, 2021 CONTEMPORARY RESEARCH IN INDIA."

[8] A. Banerjee, P. Agrawal, and N. C. h. S. N. Iyengar, "Energy Efficiency Model for Cloud Computing," *International Journal of Energy, Information and Communications*, vol. 4, no. 6, pp. 29–42, Dec. 2013, doi: 10.14257/ijeic.2013.4.6.04.

[9] M. Hasan and M. S. Goraya, "Flexible fault tolerance in cloud through replicated cooperative resource group," *Computer Communications*, vol. 145, pp. 176–192, Sep. 2019, doi: 10.1016/j.comcom.2019.06.005.

[10] X. Zhu, L. T. Yang, H. Chen, J. Wang, S. Yin, and X. Liu, "Real-Time Tasks Oriented Energy-Aware Scheduling in Virtualized Clouds," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 168–180, Apr. 2014, doi: 10.1109/tcc.2014.2310452.

[11] C. Cheng, J. Li, and Y. Wang, "An Energy-Saving Task Scheduling Strategy Based on Vacation Queuing Theory in Cloud Computing," 2015.

[12] H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin, and J. Wu, "Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment," *Journal of Systems and Software*, vol. 99, pp. 20–35, Jan. 2015, doi: 10.1016/j.jss.2014.08.065.

[13] M. Hasan and M. A. Siddique, "A Research-Oriented Mathematical Model for Cloud Simulations," in *Proceedings of the 5th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud), I-SMAC 2021*, 2021, pp. 875–878. doi: 10.1109/I-SMAC52330.2021.9640902.

[14] N. Malik, M. Sardaraz, M. Tahir, B. Shah, G. Ali, and F. Moreira, "Energy-efficient load balancing algorithm for workflow scheduling in cloud data centers using queuing and thresholds," *Applied Sciences (Switzerland)*, vol. 11, no. 13, Jul. 2021, doi: 10.3390/app11135849.

[15] N. Garg and M. S. Goraya, "Task Deadline-Aware Energy-Efficient Scheduling Model for a Virtualized Cloud," *Arabian Journal for Science and Engineering*, vol. 43, no. 2, pp. 829–841, Feb. 2018, doi: 10.1007/s13369-017-2779-5.

[16] A. Berl *et al.*, "Energy-efficient cloud computing," *Computer Journal*, vol. 53, no. 7, pp. 1045–1051, Sep. 2010, doi: 10.1093/COMJNL/BXP080.

[17] "Energy Efficiency in Cloud Computing - GeeksforGeeks." https://www.geeksforgeeks.org/energy-efficiency-in-cloud-computing/ (accessed Apr. 26, 2022).

[18] A. M. Sampaio and J. G. Barbosa, "Towards high-available and energy-efficient virtual computing environments in the cloud," *Future Generation Computer Systems*, vol. 40, pp. 30–43, 2014, doi: 10.1016/j.future.2014.06.008.