

ENERGY EFFICIENT SCHEDULING OF REAL-TIME TASKS FOR CLOUD APPLICATION USING DVFS

A Dissertation submitted in partial fulfillment of the requirement for the

Award of degree of

MASTER OF TECHNOLOGY

IN

INFORMATION SYSTEMS

Submitted By

RAKESH Kr. MAURYA

(2K14/ISY/13)

Under the esteemed guidance of

Dr. N.S. RAGHAVA

Associate Professor



Department of Computer Science & Engineering

Delhi Technological University

Bawana Road, Delhi-110042

2014-2016

CERTIFICATE

This is to certify that the thesis entitled “ **Energy Efficient Scheduling of Real-Time Tasks for Cloud Application using DVFS**” submitted by **Rakesh Kr. Maurya(2K14/ISY/13)** to the Delhi Technological University, Delhi for the award of the degree of **Master of Technology** is a bona-fide record of research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Place: DTU, Delhi

Date: _____

Dr. N.S. RAGHAVA

Associate Professor

Department of Electronics & Communication Engineering
Delhi Technological University, Delhi

ACKNOWLEDGEMENT

I take the opportunity to express my sincere gratitude to my project supervisor **Dr. N.S. Raghava**, *Associate Professor, Department of Electronics & Communication Engineering, Delhi Technological University*, for providing valuable guidance, support and constant encouragement throughout the time. It is my pleasure to record my sincere thanks to him for his constructive criticism and insight without which the project would not have shaped as it has.

I would like to express my gratitude and thanks to **Dr. O.P Verma** (*Head of Dept. CSE*) for giving me such an opportunity to work on the project.

I humbly extend my words of gratitude to other faculty members & staff of Computer Science & Engineering Dept., DTU for providing their valuable help and time whenever it was required.

Rakesh Kr. Maurya

Roll No.: 2K14/ISY/13

M.Tech (Information System)

Dept. of Computer Science & Engineering

Delhi Technological University

ABSTRACT

Cloud computing is a modern technology which contains a network of systems that form a cloud. Energy conservation is one of the major concern in cloud computing. Large amount of energy is wasted by the computers and other devices and the carbon dioxide (CO₂) gas is released into the atmosphere polluting the environment. Green computing is an emerging technology which focuses on preserving the environment by reducing various kinds of pollutions. Pollutions include excessive emission of greenhouse gas, disposal of e-waste and so on leading to greenhouse effect. So pollution needs to be reduced by lowering the energy usage. By doing this, utilization of resources should be increased and with less usage of energy, maximum resource utilization should be possible.

To accomplish this purpose, many green task scheduling algorithms are used so that the energy consumption can be minimized in servers of cloud data centers. The growth of energy consumption has been explosive in current data centers, super computers, and public cloud systems. Reports estimate the energy consumptions of these data centers to be between 1.1% and 1.5% of the worldwide electricity consumption. This explosion has led to greater advocacy of green computing, and many efforts and works focus on the task scheduling in order to reduce energy dissipation. While most researchers have focused on reducing energy consumption of cloud data centers via server consolidation, we propose an approach for reducing the power required to execute real-time tasks. Energy efficient task scheduling is an effective technique to decrease the energy consumption in the Cloud Computing Systems. It exploits intelligent scheduling combined with the Dynamic Voltage and Frequency Scaling (DVFS) capability of modern CPU processors to keep the CPU operating at the minimized frequency level (and consequently minimizing power consumption) that enables the application to complete before a user-defined deadline.

In this thesis we proposed scheduling algorithm is to dynamically scale the frequency of CPUs assigned to a virtual machine in such a way that tasks sequentially executed in the VM complete before their deadlines. In a cloud environment, hardware is shared among VMs. It means that the energy consumption of a host is not determined by a single VM, but by the combined state of all the VMs. Furthermore, when scheduling a task, the physical location of

the VM can be taken into consideration, in such a way that requests are preferentially submitted to VMs whose host is already in use (i.e., other VMs in the same host are already executing tasks). This enables the system to consolidate the load whenever possible. This in turns helps in reducing the total energy consumption of the infrastructure, as it enables unused hosts to be suspended or kept in low power states. Experiments demonstrate that our approach reduces energy consumption as we promising.

Table of Contents

Title	Page no.
CERTIFICATE	ii
ACKNOWLEDGEMENT	iii
ABSTRACT	iv
LIST OF FIGURES	ix
1. Introduction	1
1.1 Research Background	1
1.2 Energy Efficient Cloud Computing	2
1.3 Research Problem and Objective	5
1.4 Motivation	5
2. Cloud Computing	8
2.1 Introduction	8
2.2 Cloud Components	8
2.2.1 Clients	9
2.2.2 Data Centers	9
2.2.3 Distributed Servers	9
2.3 Cloud Evolution	9
2.4 Cloud Computing Architecture	10
2.5 Cloud Service Model	11
2.5.1 Software as a Service	12
2.5.2 Platform as a Service	12
2.5.3 Infrastructure as a Service	13
2.5.4 Other Services	13
2.6 Cloud Deployment Model	14
2.6.1 Public Cloud	14
2.6.2 Private Cloud	14

2.6.2.1 On-Premise Private Cloud	14
2.6.2.2 Externally Hosted Private Cloud	15
2.6.3 Hybrid Cloud	15
2.6.4 Community Cloud	15
2.7 Virtualization and Cloud Computing	16
2.7.1 Key Characteristics of Virtualization	16
2.7.1.1 Increased Security	17
2.7.1.2 Managed Execution	17
2.7.1.3 Portability	17
2.7.2 Virtualization Techniques	17
2.7.2.1 Full Virtualization	17
2.7.2.2 Para Virtualization	18
2.8 Issues in Cloud Computing	19
2.8.1 Server Consolidation	19
2.8.2 Energy Consumptions	19
2.8.3 Security of Data	19
2.8.4 VM Migration	20
3. Survey of Energy-Efficient Data Center and Cloud Computing System	21
3.1 Introduction	21
3.2 Power and Energy Models	23
3.2.1 Static and Dynamic Power Consumption	24
3.2.2 Sources of Power Consumption	25
3.2.3 Modeling Power Consumption	26
3.3 State of Art in Energy-Efficient Computing System	28
3.3.1 Hardware Level	29
3.3.1.1 Dynamic Component Deactivation (DCD)	29
3.3.1.2 Dynamic Performance Scaling (DPS)	29
3.3.2 Operating System Level	30
3.3.3 Virtualization Level	30
3.3.4 Data Center Level	31

3.4 Conclusion	31
4. Related Works	33
5. Proposed Methodology	37
5.1 System and Application Model	37
5.2 Energy Model	39
5.3 Methodology Used	40
5.3.1 Dynamic Optimization	42
5.3.2 Static Optimization	43
5.3 Proposed Approach	44
6. Experimental Setup and Results	49
6.1 Used Technologies	49
6.1.1 Java	49
6.1.2 CloudSim	50
6.2 Implementation Details	51
6.3 Experimental Results	53
6.3.1 Output of MMF-DVFS	53
6.3.2 Output of Proposed Approach	58
7. Conclusion and Future Work	64
7.1 Conclusion	64
7.2 Future Work	64
References	65

LIST OF FIGURES

Fig. No	Title	Pg. No
1.1	The worldwide data center energy consumption 2000-2010	1
1.2	The high-level system view	3
1.3	Execution of task running at different frequency	6
2.1	Components of Cloud	8
2.2	Cloud Computing Architecture	11
2.3	Cloud Service Models	13
2.4	Private, Public and Hybrid Cloud Deployment	15
2.5	Google Trends showing the searches on Cloud computing Vs Virtualization	16
2.6	Full Virtualization	18
2.7	Para Virtualization	18
3.1	Energy consumption at different levels in computing systems	22
3.2	Power consumption by server components	25
3.3	Relation between power consumption and CPU utilization of server	27
3.4	A high level taxonomy of power and energy management	28
5.1	System and application model of proposed approach	37
5.2	Energy-frequency curve for single task	40
6.1	Layered architecture of CloudSim	50
6.2	Flow of program execution in CloudSim	52
6.3	Screenshot 1 of 4 of MMF-DVFS	54
6.4	Screenshot 2 of 4 of MMF-DVFS	55
6.5	Screenshot 3 of 4 of MMF-DVFS	56
6.6	Screenshot 4 of 4 of MMF-DVFS	57
6.7	Screenshot 1 of 4 of Proposed Approach	58
6.8	Screenshot 2 of 4 of Proposed Approach	59
6.9	Screenshot 3 of 4 of Proposed Approach	60
6.10	Screenshot 4 of 4 of Proposed Approach	61
6.11	Comparison between various energy efficient method	63

1.1 Research Background

CLOUD computing has revolutionized the Information and Communication Technology (ICT) industry by enabling on-demand provisioning of elastic computing resources. The cloud computing is based on the concept of dynamic provisioning, which is applied to services, computing capability, storage, networking, and information technology infrastructure to meet user requirements. The resources are made available for the users through the Internet and ordered on a pay-as-use basis from different Cloud computing vendors. An organization can either outsource its computational needs to the Cloud avoiding high up-front investments in a private computing infrastructure and consequent costs of maintenance and upgrades, or build a private Cloud data center to improve the resource management and provisioning processes.

The proliferation of Cloud computing has resulted in the establishment of large-scale data centers around the world containing thousands of compute nodes. However, Cloud data centers consume huge amounts of electrical energy resulting in high operating costs and carbon dioxide (CO₂) emissions to the environment. As shown in Figure 1.1, energy consumption by data centers worldwide has risen by 56% from 2005 to 2010, and in 2010 is accounted to be between 1.1% and 1.5% of the total electricity use [26].

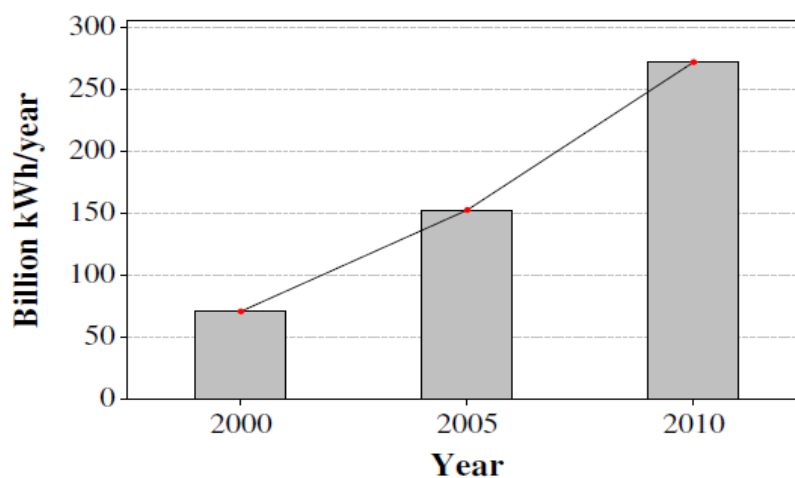


Figure 1.1: The worldwide data center energy consumption 2000-2010 [26]

Furthermore, carbon dioxide emissions of the ICT industry are currently estimated to be 2% of the global emissions, which is equivalent to the emissions of the aviation industry and significantly contributes to the greenhouse effect. Energy consumption in data centers will continue to grow rapidly unless advanced energy efficient resource management solutions are developed and applied.

To address the problem of high energy use, it is necessary to eliminate inefficiencies and waste in the way electricity is delivered to computing resources, and in the way these resources are utilized to serve application workloads. This can be done by improving both the physical infrastructure of data centers, and the resource allocation and management algorithms. Recent advancement in the data center design has resulted in a significant increase of the infrastructure efficiency.

1.2 Energy Efficient Cloud Computing

Cloud computing infrastructures are designed to support the accessibility and deployment of various services oriented applications by the users. Cloud computing services are made available through the server farms or data centers. To meet the growing demand for computations and large volume of data, the cloud computing environments provides high performance servers and high speed mass storage devices. These resources are the major source of the power consumption in data centers along with air conditioning and cooling equipment [20]. Moreover the energy consumption in the cloud is proportional to the resource utilization and data centers are almost the world's highest consumers of electricity. Due to the high energy consumption by data centers, it requires efficient technology to design green data center . On the other hand, Cloud data center can reduce the the total energy consumed through task consolidation and server consolidation using the virtualization by workloads can share the same server and unused servers can be switched off. The total computing power of the Cloud data center is the sum of the computing power of the individual physical machine.

An ideal way of addressing the energy inefficiency problem is implementing an energy proportional computing system, where energy consumption of the computing resources is proportional to the application workload [21]. This approach is partially implemented through the widely adopted Dynamic Voltage and Frequency Scaling (DVFS) technique. DVFS allows the dynamic adjustment of the voltage and frequency of the CPU based on the current

resource demand. As a result, current desktop and server CPUs can consume less than 30% of their peak power in low-activity modes, leading to dynamic power ranges of more than 70% [21].

One method to improve the utilization of resources and reduce energy consumption, which has been shown to be efficient is dynamic consolidation of Virtual Machines (VMs) enabled by the virtualization technology. Virtualization allows Cloud providers to create multiple VM instances on a single physical server, thus improving the utilization of resources and increasing the Return On Investment (ROI). The reduction in energy consumption can be achieved by switching idle nodes to low-power modes (i.e. sleep, hibernation), thus eliminating the idle power consumption as shown in Figure 1.2.

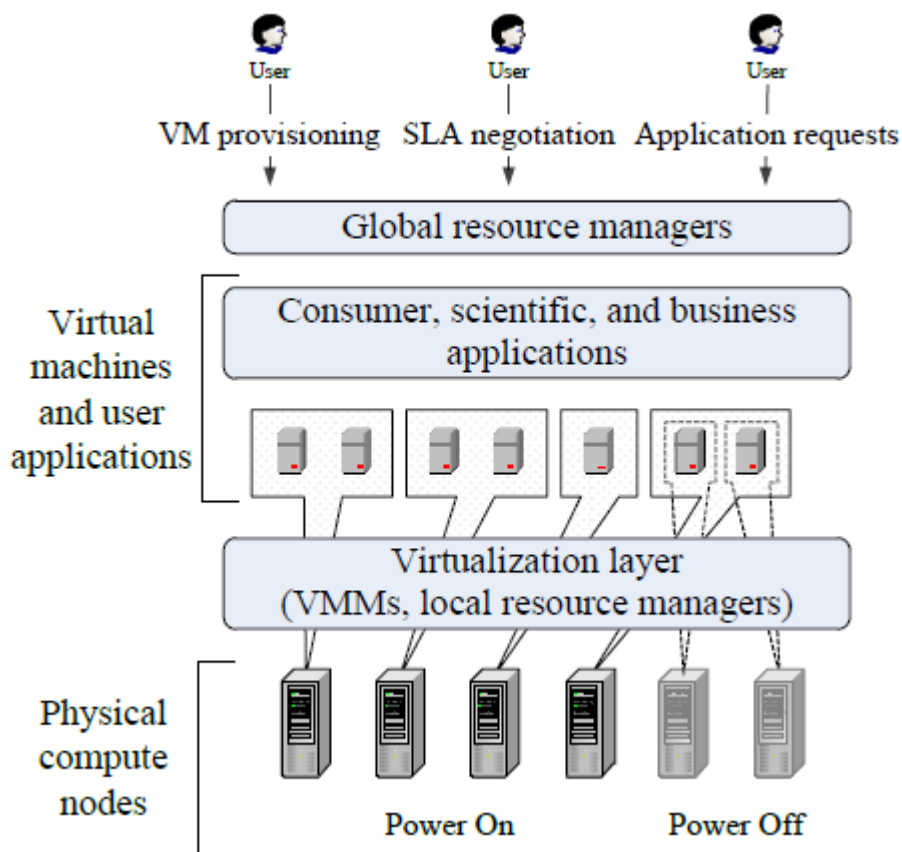


Figure 1.2: The high-level system view

Another capability provided by virtualization is live migration, which is the ability to transfer a VM between physical servers (referred to as hosts, or nodes) with a close to zero downtime.

Using live migration VMs can be dynamically consolidated to leverage fine-grained fluctuations in the workload and keep the number of active physical servers at the minimum at all times. Dynamic VM consolidation consists of two basic processes: migrating VMs from underutilized hosts to minimize the number of active hosts; and offloading VMs from hosts when those become overloaded to avoid performance degradation experienced by the VMs, which could lead to a violation of the QoS requirements. Idle hosts are automatically switched to a low-power mode to eliminate the static power and reduce the overall energy consumption by the system. When required, hosts are reactivated to accommodate new VMs or VMs being migrated.

However, dynamic VM consolidation in Clouds is not trivial since modern service applications often experience highly variable workloads causing dynamic resource usage patterns. Therefore, unconstrained VM consolidation may lead to performance degradation when an application encounters an increasing demand resulting in an unexpected rise of the resource usage. If the resource requirements of an application are not fulfilled, the application may face increased response times, time-outs or failures. Ensuring QoS defined via Service Level Agreements (SLAs) established between Cloud providers and their customers is essential for Cloud computing environments. Therefore, Cloud providers have to deal with the energy-performance trade-off – minimizing energy consumption, while meeting QoS requirements.

Cloud uses virtualization technology in data centers to allocate resources for the services as per need. Cloud gives three levels of access to the customers: SaaS, PaaS, and IaaS. The task originated by the customer can differ greatly from customer to the customer. Entities in the Cloud are autonomous and self-interested; however, they are willing to share their resources and services to achieve their individual and collective goals. In such an open environment, the scheduling decision is a challenge given the decentralized nature of the environment. Each entity has specific requirements and objectives that need to achieve. Server consolidation is allowing the multiple servers running on a single physical server simultaneously to minimize the energy consumed in a data center . Running the multiple servers on a single physical server are realized through virtual machine concept. The task consolidation also knows as server/workload consolidation problem. Task consolidation problem addressed in this thesis is to assign n task to a set of r resources in cloud computing environment. This energy efficient resource allocation maintains the utilization of all computing resources and distributes virtual machines in a way that the energy consumption

can minimize. The goal of these algorithms is to maintain availability to compute nodes while reducing the total energy consumed by the cloud infrastructure.

1.3 Research Problem and Objective

An emerging technology called Cloud computing can increase the utilization of ICT equipment. Therefore job scheduler is needed by cloud datacenter to manage the resources for executing jobs. In this research, we propose a scheduling algorithm for the cloud datacentre to execute real time tasks with a dynamic voltage frequency scaling technique.

The research in energy-efficient cloud computing focuses on the problems of VM consolidation and selection of energy sources for data centers. In this thesis, we consider a different approach, orthogonal to these studies, which aim at reducing the power required to execute real-time tasks for cloud application. An urgent application is a High Performance Computing application whose execution needs to complete before a user-defined deadline because of its utilization in sensitive contexts, such as disaster management and healthcare.

Our objective is to minimize the power consumption for above stated problem. In our approach we create VM in consolidated manner so that we to use minimal resources for executing task to get better utilisation of resources. After that we find the frequency assignment for each VM in a host so that deadline should not violate.

Our scheduling algorithm can efficiently increase resource utilization; hence, it can decrease the energy consumption for executing jobs. We define a deadline-constrained application as an application that needs to have its execution completed before a user-defined soft deadline. Experimental result shows that our scheme can reduce more energy consumption for the same job than other schemes do. Performance of executing the jobs is not sacrificed in our scheme.

1.4 Motivation

Energy efficiency is increasingly important for cloud computing, because the increased usage of cloud computing infrastructure, together with increasing energy costs. There is a need to reduce the greenhouse gas emissions call for the energy efficient technologies that decrease the overall energy consumption of computation, storage and communication equipment. Optimum utilization of energy is increasingly important in data centers. The power

dissipation of the physical servers is the root cause of power consumption, which leads to the power consumption of the cooling systems. Many efforts have been made to make data centers more energy efficient. One of these is to minimize the total power consumption of these servers in a data center, through minimizing the operating frequency of CPU.

We are encouraged by the power consumption behaviour of the recent multi core processors. If we observe carefully then we can see the relation between power consumption and number of thread used. We have decided to consider a simple and popular power model as given below:

$$P = C + i\delta \tag{1.1}$$

Where P = total power consumption of the processor, C = base power, i =number of core and δ = core power.

Energy efficient scheduling techniques uses the DVFS to design their algorithms and uses the power model $P \propto f^3$ where P is power consumption of the processor and f is frequency at which threads are running. In figure 1.3 it can be seen that energy consumption is more when executing with f_{max} frequency than that of $\frac{f_{max}}{2}$.

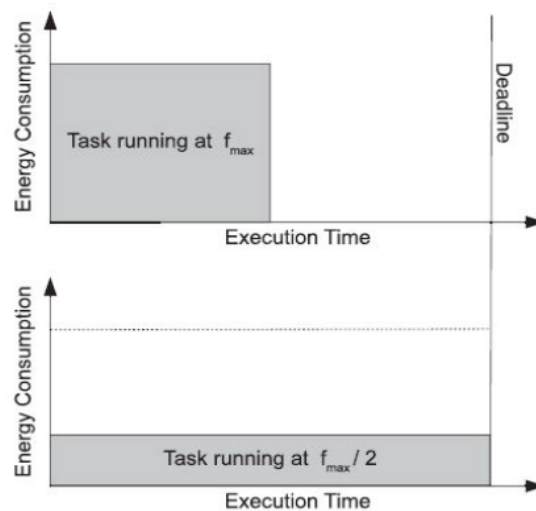


Figure 1.3: This shows execution of task running at different frequency

It can be easily observed that running at lower frequency minimizes the energy, for real time tasks meeting the deadline is sufficient. Hence we are developing an energy efficient

scheduling for real time tasks using DVFS and our main goal is to find the optimal MIPS assignment for each task/cloudlet.

2.1 Introduction

This chapter summarizes the brief overview of cloud computing, cloud component, architecture, cloud service model, deployment model, virtualization and issues in cloud computing. Previous decades have reinforced the idea of processing information at larger scale, in a more efficient manner. With the advent of cloud computing headache of storing, processing and accessing data through internet at larger scale disappeared now. From a very long time researchers are trying to provide utilities as a service to its users. Users may demand software, platform or hardware from a provider through an internet and charged on the usage basis.

2.2 Cloud Components

There are some components as topological aspect of cloud: clients or users, the datacenter and the distributed servers. These components integrate the cloud as one unit.

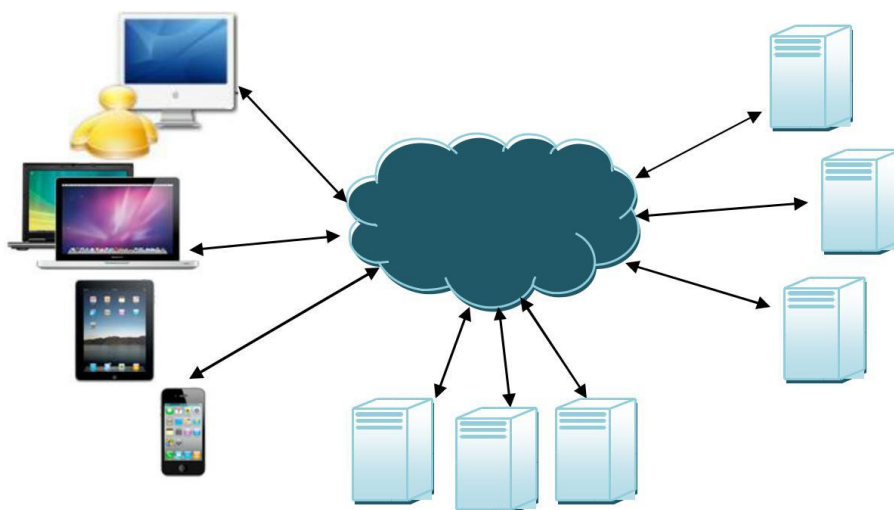


Figure 2.1: Components of Cloud

In Figure 2.1, these components are shown. Each component has specific role in delivering services asked by the cloud users. These components communicate over the entire network, as per the requirements and configuration. These components are discussed below in detail.

2.2.1 Clients

Clients are generally, the computers which are used in our day-to-day work. But it may be a laptop, a PDA, a mobile phone or a tablet computer. These all devices should be able to access the cloud computing interface via internet. With these devices end users are able to communicate to the cloud interface and afterwards can use the service or applications as per their choice.

2.2.2 Datacenters

A datacenter is one of the core elements of cloud computing systems. A datacenter consists of several nodes occupying a large room. These datacenters are configured by the cloud service providers. Those configurations are based on various factors like cloud deployment model and the cloud service model.

2.2.3 Distributed Servers

Distributed servers, sometimes called as nodes, need not be deployed at the same location but may be distributed geographically as per the convenience of the service provider. From users prospect these servers seems to work together, without knowing the actual location of the server. Distributed servers increase the fault tolerance of the network.

2.3 Cloud Evolution

The evolution of the cloud goes phase by phase that include the Grid computing, distributed computing. Cloud computing is used first in 1950s, the time during which large-scale mainframes were available in the business industry. The hardware used by the mainframe was installed in a big room and all users are accessing the mainframe through terminals.

Later in the year 1970, the IBM launches OS having a number of virtual machines at a single machine. The Virtual machine OS has taken the application of 1950s that is of sharing the access to a mainframe to a higher level by considering a number of virtual machines providing different accessible machines at a single physical machine.

Idea of cloud computing was first showed by J.C.R Licklider and John McCarthy in 1969. The vision behind this is that everyone goes interconnected and thus able to access data through anywhere.

The data are stored in a data center (a centralized infrastructure) which is a vast data storage space. The processing of the request or data performed through servers thus availability and security of the data will be addressed. The service provider and the clients has an agreement for the usage known as SLA (Service level Agreement). Then in 1999, salesforce.com put this idea to an application. Then in 2002, a Cloud based services of web launched by amazon. It provides on demand services to the subscribed users. There are many proposed definitions of the Cloud computing due to its growing popularity defining its characteristics. Some of the definitions given by many well-known scientists and organizations are:

- Raj Kumar Buyya defines the Cloud computing in terms of its utility to end user as A Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers [22].
- National Institute of Standards and Technology (NIST) defines Cloud computing as follows: Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This Cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models [23].

2.4 Cloud Computing Architecture

Cloud computing supports any IT service that can be consumed as a utility and delivered through a network, most likely the Internet. Such characterization includes quite different aspects: infrastructure, development platforms, application and services. It is possible to organize all the concrete realizations of cloud computing into a layered view covering the entire stack from hardware appliances to software systems.

The whole architecture can be understood by studying separately each layer in the architecture. These layers are confined to work for specific tasks. The architecture covers cloud deployment models and all the service modes. Figure 2.2 shows the layered architecture of cloud computing.

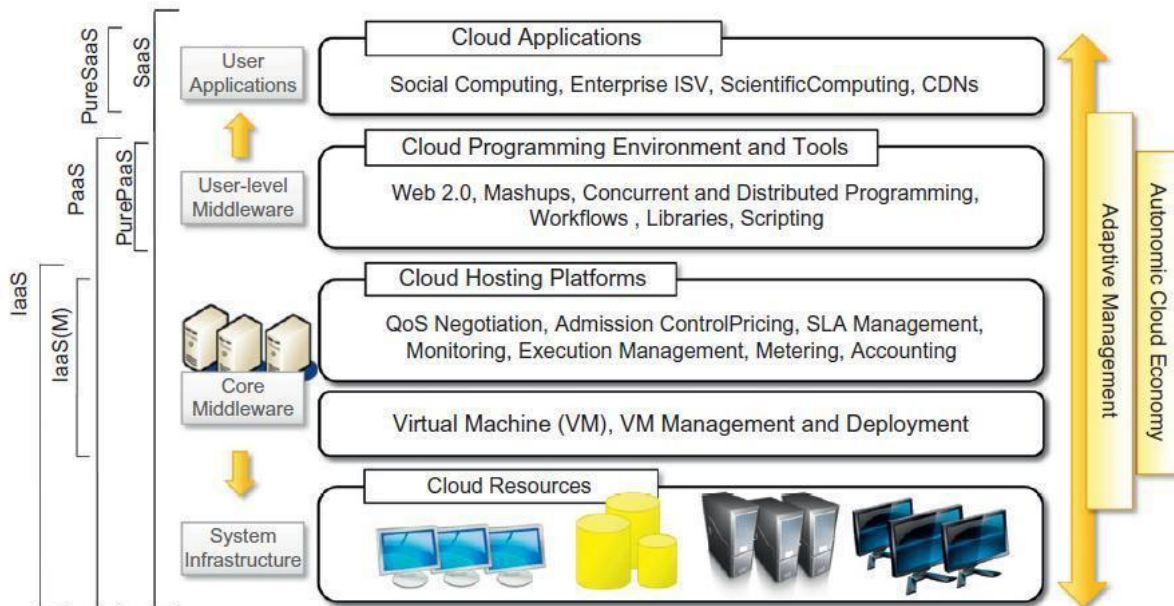


Figure 2.2: A typical cloud computing architecture

2.5 Cloud Service Models

Once a cloud is developed and ready to be used by its consumers, it has to be decided that how to use the services offered by the cloud. The most common way in which it can be used is by categorizing all the services according to the commonly used business model. In cloud computing there are basically three delivery mechanisms through which a service can be delivered to the cloud users: software, platform and infrastructure. There are number of services offered by the cloud, but the primary service models which are deployed over the cloud are discussed below:

2.5.1 Software as a Service

Software-as-a-Service (SaaS) is a software delivery model. In this, applications can be accessed through the Internet like a Web-based service. This delivery model facilitates the cloud users with wide variety of applications, for example social networking applications and customer relationship management applications. Users can make use of these applications by registering on the cloud service provider's website and then simply passing on the tasks to cloud service provider for completion. In this model users should not worry about the hardware and software configurations, it is the responsibility of the third party, whom they are registered with. In this scenario, customers neither need install anything on their premises nor have to pay considerable up-front costs to purchase the software and the required licenses. They simply access the application website, enter their credentials and billing details, and can instantly use the application, which, in most of the cases, can be further customized for their needs. On the provider side, the specific details and features of each customer's application are maintained in the infrastructure and made available on demand. Examples of some popular SaaS applications are: Facebook, Google Docs, NetSuit and Microsoft online.

2.5.2 Platform as a Service

Platform as a Service (PaaS) provides facility to users to develop their own application using programming languages, libraries, services, and tools. Therefore, it can be said that the environment is integrated with various components in order to give users more number of offerings. Some PaaS providers provide a generalized development environment, while some only provides hosting-level services such as on-demand scalability and security. Management of applications and the underlying infrastructure configured by the users is done by the cloud service provider in PaaS whereas in SaaS users have to manage their applications by themselves. In PaaS, users have full control over the deployed applications and their hosting environment configuration. PaaS constitute the middleware above which applications are built by the users. Some popular examples of PaaS are Windows Azure, Engine Yard and Google App Engine.

2.5.3 Infrastructure as a Service

Infrastructure- and Hardware-as-a-Service (IaaS/HaaS) offerings facilitates the user to customize the infrastructure as per their requirements. Users may demand from a single server to entire infrastructure, network devices, load balancers, databases and web servers. These services are most popular in the cloud market as these services prove to be a good option for the organizations who wants their own infrastructure with less maintenance cost. IaaS can service users with its capabilities of processing huge amount of data over a huge network, and other scalable computing resources. A user need not to worry about managing the cloud infrastructure, but has to manage the storage, operating systems, and deployed applications. Some popular examples of IaaS include Amazon, GoGrid and 3 Tera.

2.5.4 Other Services

Above discussed services are the core services which are categorized in accordance with the major market trends, user specifications and computing capabilities. Other services like security as a service, testing as a service are domain-specific services. These aims at providing particular fields of services to the users interested in that particular field for their applications.

Figure 2.3, shows the service models in cloud computing.

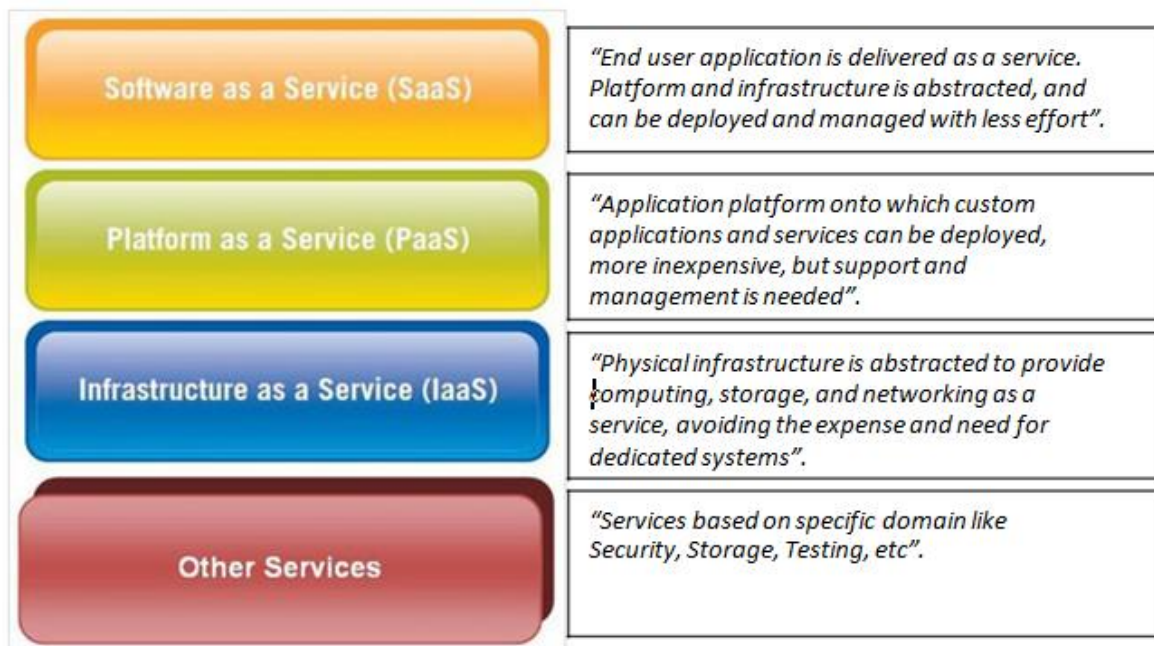


Figure 2.3: Cloud Service Models.

2.6 Cloud Deployment Models

There are number of cloud consumers who want infrastructure of different sizes and each of these infrastructures requires different kind of management. Also different user groups want different types of infrastructures based on the nature and services offered by the cloud. Clouds constitute the primary outcome of cloud computing. They are a type of parallel and distributed system harnessing physical and virtual computers presented as a unified computing resource. There are four most important cloud deployment models.

2.6.1 Public Cloud

Public cloud as the name suggests are developed to serve the requirements of general public. A cloud service provider serves the service requirement like if any storage is required by the user, or if a user wants to make use of certain applications on cloud platform through internet. A public cloud may be managed, operated and owned by a government, academic, or business organization. Examples of public clouds are Amazon's Elastic Compute Cloud (EC2), Google's AppEngine, IBM's Blue Cloud, Sun Cloud and Windows Azure Services Platform.

2.6.2 Private Cloud

In Private cloud, the cloud infrastructure is provisioned for private use. An organization can request to a third party or it may develop its own private cloud infrastructure. Big organizations like Google, Microsoft have their own cloud infrastructure, which is supposed to be more secure than the public clouds. An organization can serve many other users of it by its own private cloud for e.g., Google is serving its users by providing them services like Gmail, Drive, GoogleApp engine and many more. Depending upon the deployment of the cloud, there are two types of private cloud:

2.6.2.1 On-Premise Private Cloud

As the name suggests on-premise private clouds are hosted privately within its own datacenter. Therefore, these are also known as internal cloud. Benefits of this type of model includes security and standardization of processes, but still some issues related to size and scalability restricts a person from choosing this kind of model.

2.6.2.2 Externally Hosted Private Cloud

It is also known as external cloud. This type of model provides a special cloud environment with full privacy, as these are hosted externally with a cloud provider. Enterprises who don't want to share their physical resources in public cloud can be benefitted by using this model. HP CloudStart and eBay are two popular providers of private cloud deployments.

2.6.3 Hybrid Cloud

Hybrid cloud as the name suggests is a combination of two or more different types of cloud. It may be a combination of a private cloud, a public cloud or a community cloud. Cloud infrastructures of these clouds are entirely different. A cloud service provider needs to manage between the clouds to provide a required service to its users.

2.6.4 Community Cloud

The concept of community cloud is similar to grid computing. There are certain specific communities of users from different organizations. As a day to day example let us consider a private firm in which there are two main business domain which works separately from one another, in such cases a firm can create two community clouds for the separate working of the two domains. In such cases the mission, security requirements and the other policies for the two domains are entirely different. This type of cloud can be owned by firm itself or they can hire some third party for the purpose.

Figure 2.4 shows the cloud deployment model.

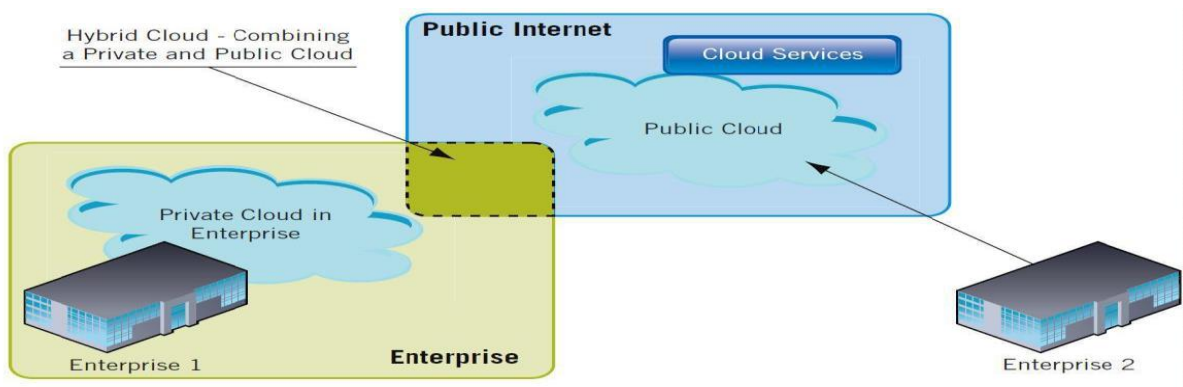


Figure 2.4: Private, Public and Hybrid Cloud Deployment.

2.7 Virtualization and Cloud Computing

Virtualization technology is one of the fundamental components of cloud computing. In the delivery of IaaS it plays a significant role, therefore it is sometimes also known as hardware virtualization. Virtualization abstracts the underlying resources and simplifies their use, isolates users from one another, and supports replication which, in turn, increases the elasticity of the system. Though the concept of virtualization comes earlier than the cloud computing but in a very short span of time cloud computing has gained much popularity. As shown in Figure 2.5, Google search index for cloud computing increased at much faster rate.

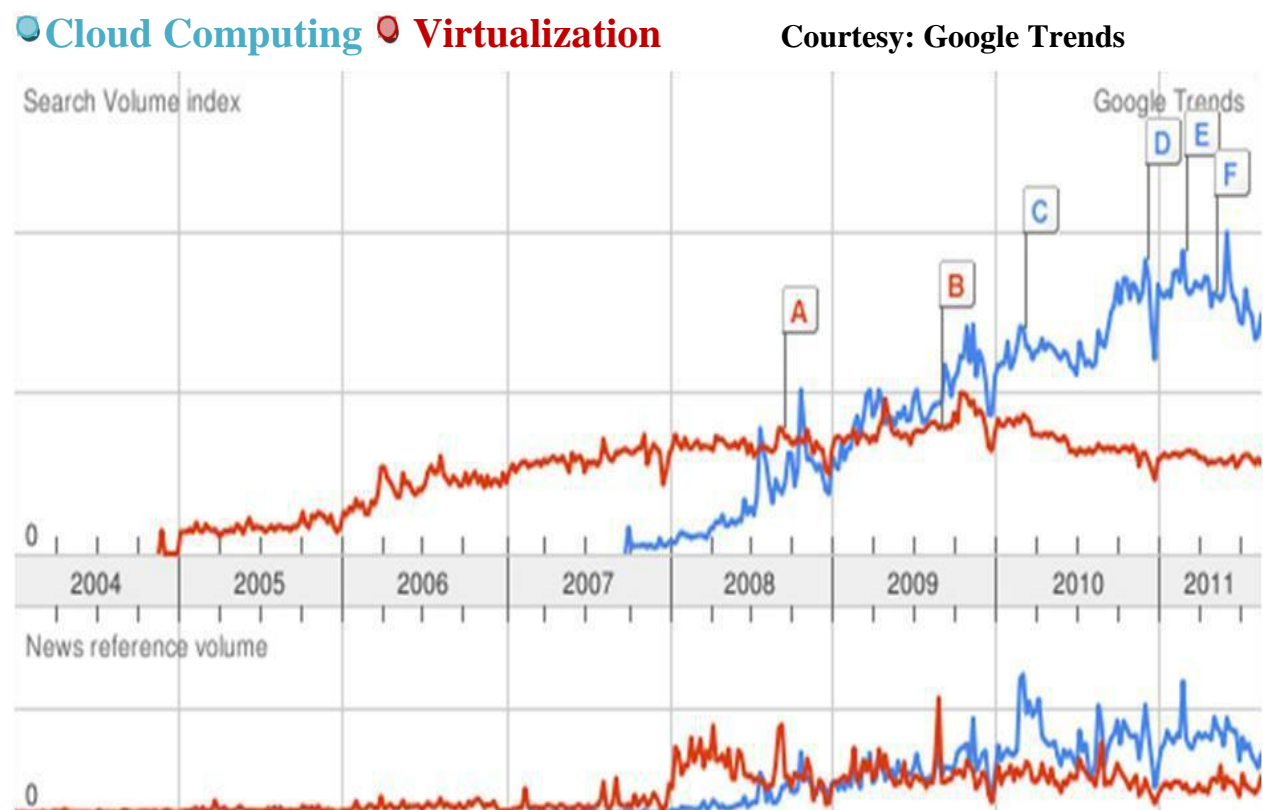


Figure.2.5: Google Trends showing the searches on Cloud computing Vs Virtualization

2.7.1 Key Characteristics of Virtualization

Virtualization technology owns some very important characteristics which makes this technology so popular. Let us discuss these characteristics in detail:

2.7.1.1 Increased Security

Introducing a new layer of virtualization in between the guest and the host, the level of security has increased. All operations of guest are generally performed on virtual machine. Therefore, the virtual machine manager controls and filters the activity of the guest, thereby preventing some harmful operations from being performed.

2.7.1.2 Managed Execution

A virtual machine manager is responsible for the proper management of all the tasks assigned by the guest. Each task is executed under the supervision of virtual machine manager. Sharing and aggregation among various guests is possible through virtualization.

2.7.1.3 Portability

Portability is applied differently in hardware and programming level virtualization. In hardware-level virtualization guest is packaged into a virtual image while in programming-level virtualization no recompilation is required while running different programs from various guests.

2.7.2 Virtualization Techniques

In cloud computing there are two types of virtualization techniques:

2.7.2.1 Full Virtualization

In this technique of virtualization the entire system is virtualized i.e., the whole operating system is directly placed upon the virtual machine. It looks like the entire system is running on the raw hardware. In figure 2.6 the concept of full virtualization is shown.

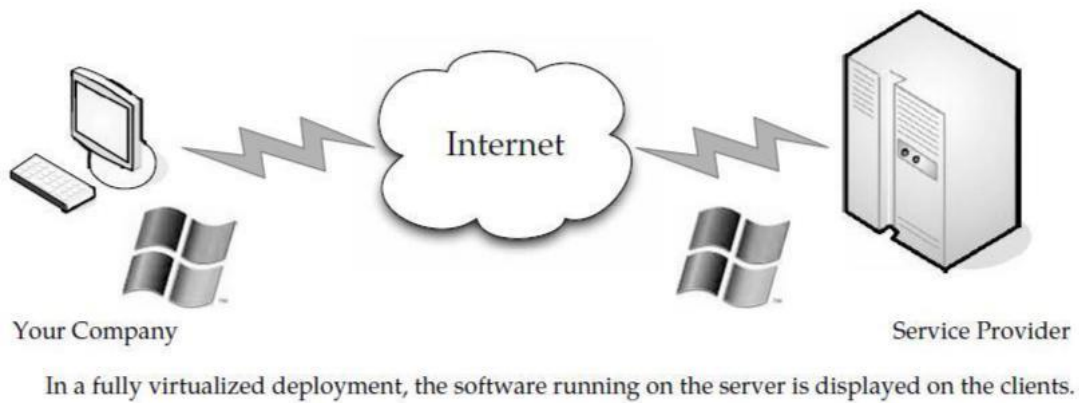


Figure.2.6: Full Virtualization

2.7.2.2 Para Virtualization

Para virtualization provides the ability to operate performance-critical tasks directly on the host. This technique is beneficial in case of disaster recovery, migration from one system to another system and capacity management. In figure 2.7 Para virtualization is shown.

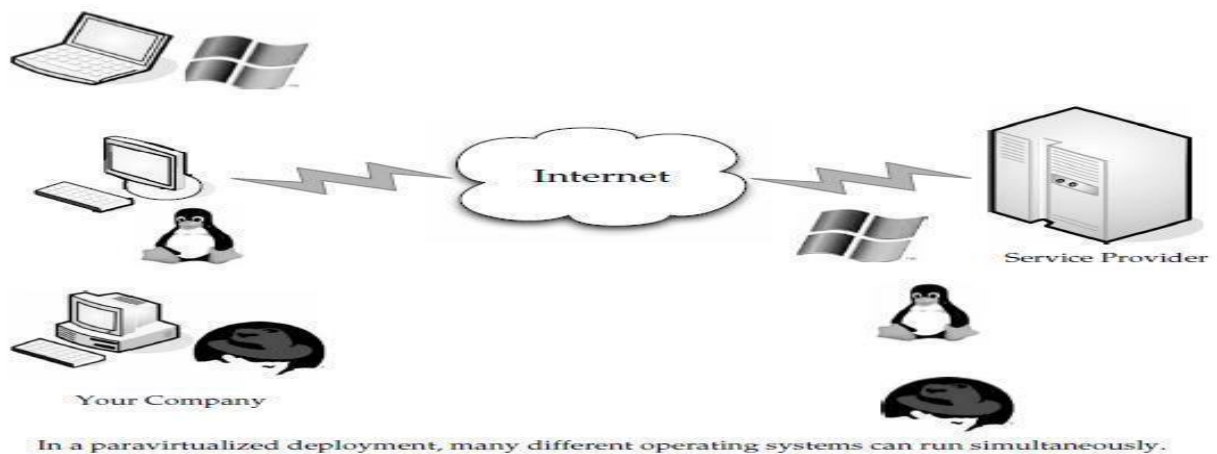


Figure.2.7: Para Virtualization

2.8 Issues in Cloud Computing

To reach the extent of CLOUD computing, major aspects have not yet been developed and in some cases not even researched. Many issues are still needs to be known.

2.8.1 Server Consolidation

Energy optimization is an important issue in cloud computing environments. To reduce this, an idea is to redeem the idle power going waste by underutilized servers. The fact is that a server even runs a very small workload, then also it consume over 50% of the peak power [24]. Thus the focus of conserving energy is to turn on as few servers as possible by consolidating the workload. This condition is referred to as server consolidation. It is a nice approach to better utilize the resources and to redeem the consumption of power.

2.8.2 Energy Consumption

To reduce the energy consumption in data centres is another issue in cloud computing. It has been found that that a server even runs a very small workload, and then also it consumes over 50% of the peak power. Energy consumption by Google 2,675,898 MWh in 2011 and it is increasing regularly. Thus there is a need to control the consumption of energy otherwise the cost of cloud computing will increase tremendously. The aim is to reduce the energy consumed in data centers by following the service level agreement. This issue is now started gaining importance. This issue can be addressed by a lot of approach. For example by selectively shutting down the unutilized server the power consumption can be greatly reduced. The utilization of resources can be enhanced by addressing the problem.

2.8.3 Security of Data

Security of data is a vital and important research issue in cloud computing. It hampers the expansion of the cloud. Usually, the services of cloud computing are delivered by a third party known as service provide, who owns the infrastructure. Even for a virtual private Cloud, the service provider can only specify the security setting remotely, without knowing whether it is fully implemented. It is hard to make the trust at each layer of the Cloud.

Firstly, the hardware layer must be trusted using hardware trusted platform module. Secondly, the virtualization platform must be trusted using secure virtual machine monitors. VM migration should only be allowed if both source and destination servers are trusted.

2.8.4 VM Migration

In a cloud environment having a more number of datacenters, virtual machines have to be migrated between physical machines located in any (same or different) datacenter in order to achieve better provisioning of resources. Migration of VMs which means to transfer a virtual machine from one to another physical machines helps in greatly reducing the energy consumption. The benefits for VM migration is it avoid the hotspots.

Survey of Energy-Efficient Data Center and Cloud Computing Systems

3.1 Introduction

The primary focus of designers of computing systems and the industry has been on the improvement of the system performance. According to this objective, the performance has been steadily growing driven by more efficient system design and increasing density of the components described by Moore's law. Although the performance per watt ratio has been constantly rising, the total power drawn by computing systems has hardly decreased. Oppositely, it has been increasing every year. The problem is even worse for large-scale compute infrastructures, such as clusters and data centers. It was estimated that energy consumption by IT infrastructures has risen by 56% from 2005 to 2010, and in 2010 accounted to be between 1.1% and 1.5% of the global electricity use [26]. Apart from high operating costs, this results in substantial carbon dioxide (CO₂) emissions, which were estimated to be 2% of the global emissions.

Energy consumption is not only determined by hardware efficiency, but also by the resource management system deployed on the infrastructure and the efficiency of applications running in the system. Energy efficiency impacts end-users in terms of resource usage costs, which are typically determined by the Total Cost of Ownership (TCO) incurred by the resource provider. Higher power consumption results not only in boosted electricity bills but also in additional requirements to the cooling system and power delivery infrastructure, i.e., Uninterruptible Power Supplies (UPS), Power Distribution Units (PDU), and so on.

Apart from the overwhelming operating costs and the total cost of acquisition (TCA), another rising concern is the environmental impact in terms of carbon dioxide (CO₂) emissions caused by high energy consumption. Therefore, the reduction of power and energy consumption has become a first-order objective in the design of modern computing systems.

The interdependence of different levels of computing systems in regard to energy consumption is shown in Figure 3.1

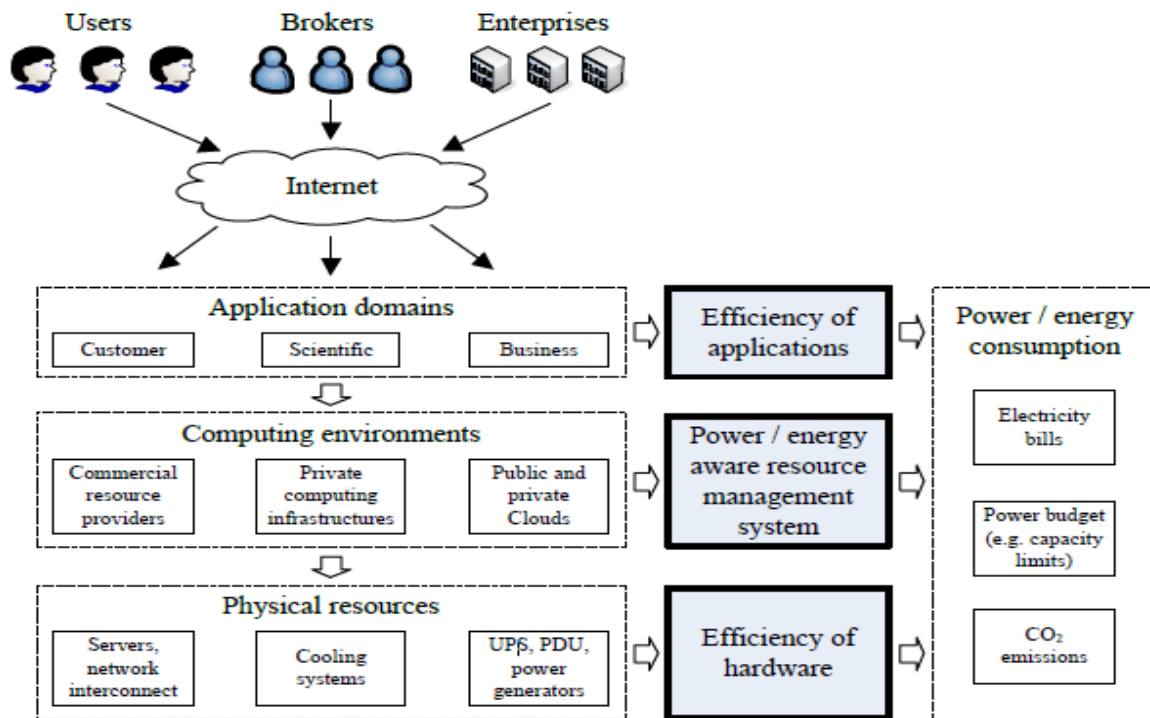


Figure 3.1: Energy consumption at different levels in computing systems

The root of energy efficient computing, or Green information technology practices can be traced back to 1992, at that time U.S. Environmental Protection Agency launched “Energy Star”, a voluntary labeling program which designed to identify and promote energy efficient products to reduce greenhouse gas emission. Computers and monitor was the first labeled product. At that time “green computing” was introduced to refer to energy efficient personal computers.

There are a number of industry initiatives aiming at the development of standardized methods and techniques for the reduction of energy consumption and carbon dioxide emissions in computing environments. They include Climate Savers Computing Initiative (CSCI), Green Computing Impact Organization, Inc. (GCIO), Green Electronics Council, The Green Grid, FIT4Green, ECO2Clouds, Eco4Cloud, International Professional Practice Partnership (IP3),

with the membership of large companies, such as AMD, Dell, HP, IBM, Intel, Microsoft, Sun Microsystems, and VMware.

This chapter discusses various ways of reducing power and energy consumptions in the computing systems, and also recent research that deals with the power and energy-efficiency at hardware and firmware, Operating System (OS), virtualization, and data center levels. Objectives of this chapter is to give an overview of recent research advancements in energy efficient computing, classify the approaches, discuss open research challenges, and position the current thesis within the research area.

3.2 Power and Energy Models

To understand power and energy management mechanisms, it is essential to clarify the terminology. Electric current is the flow of electric charge measured in amperes. Amperes define the amount of electric charge transferred by a circuit per second. Power and energy can be defined in terms of work that a system performs. Power is the rate at which the system performs the work, while energy is the total amount of work performed over a period of time. Power and energy are measured in watts (W) and watt-hour (Wh), respectively. Work is done at the rate of 1 W when 1 A is transferred through a potential difference of 1 V. A kilowatt-hour (kWh) is the amount of energy equivalent to a power of 1 kW (1000 W) being applied for one hour. Formally, power and energy can be defined as (3.1) and (3.2):

$$P = \frac{W}{T} \quad (3.1)$$

$$E = PT \quad (3.2)$$

Where P is power, T is a period of time, W is the total work performed during that period of time, and E is energy. The difference between power and energy is very important since a reduction of power consumption does not always reduce the consumed energy. For example, power consumption can be decreased by lowering the CPU performance. However, in this case, a program may take longer to complete its execution consuming the same amount of energy. On one hand, a reduction of peak power consumption results in decreased costs of the infrastructure provisioning, such as costs associated with capacities of UPS, PDU, power

generators, cooling system, and power distribution equipment. On the other hand, decreased energy consumption reduces the electricity bills.

Energy consumption can be reduced temporarily via Dynamic Power Management (DPM) techniques, or permanently applying Static Power Management (SPM). DPM utilizes the knowledge of the real-time resource usage and application workloads to optimize energy consumption. However, it does not necessarily decrease peak power consumption. In contrast, SPM prescribes the usage of highly efficient hardware components, such as CPUs, disk storage, network devices, UPS, and power supplies. These structural changes usually reduce both energy and peak power consumption.

3.2.1 Static and Dynamic Power Consumption

The major part of power consumption in complementary metal-oxide-semiconductor (CMOS) circuits comprises static and dynamic power. Static power consumption, or leakage power, is caused by leakage currents that are present in any active circuit, independently of clock rates and usage scenarios. This static power is mainly determined by the type of transistors and process technology. The reduction of static power requires improvements of the low-level system design.

Dynamic power consumption is created by circuit activity (i.e., transistor switches, changes of values in registers, etc.) and depends mainly on a specific usage scenario, clock rates, and I/O activity. The sources of dynamic power consumption are the short circuit current and switched capacitance. Short-circuit current causes only 10-15% of the total power consumption and so far no way has been found to reduce this value without compromising the performance. Switched capacitance is the primary source of dynamic power consumption; therefore, dynamic power consumption can be defined as (3.3) & (3.4):

$$P_d = aCV^2f \quad (3.3)$$

$$f \propto \frac{(v-v_t)^2}{v_t} \quad (3.4)$$

Where a is the switching activity, C is the physical capacitance, V is the supply voltage, v_t is circuit threshold voltage and f is the clock frequency. The values of switching activity and capacitance are determined by the low-level system design. The combined reduction of the

supply voltage and clock frequency lies in the roots of the widely adopted DPM technique called Dynamic Voltage and Frequency Scaling (DVFS). The main idea of this technique is to intentionally scale down the CPU performance, when it is not fully utilized, by decreasing the voltage and frequency of the CPU. In the ideal case, this should result in a cubic reduction of dynamic power consumption. DVFS is supported by most modern CPUs including mobile, desktop, and server systems.

3.2.2 Sources of Power Consumption

According to data provided by Intel Labs, the main part of power consumed by a server is accounted for the CPU, followed by the memory and losses due to the power supply inefficiency are shown in Figure 3.2. The data show that the CPU no longer dominates power consumption by a server. This resulted from the continuous improvements of the CPU power efficiency combined with power-saving techniques (e.g., DVFS) that enable active low-power modes. In these modes, a CPU consumes a fraction of the total power, while preserving the ability to execute programs. As a result, current desktop and server CPUs can consume less than 30% of their peak power in low-activity modes, leading to dynamic power ranges of more than 70% of the peak power [21].

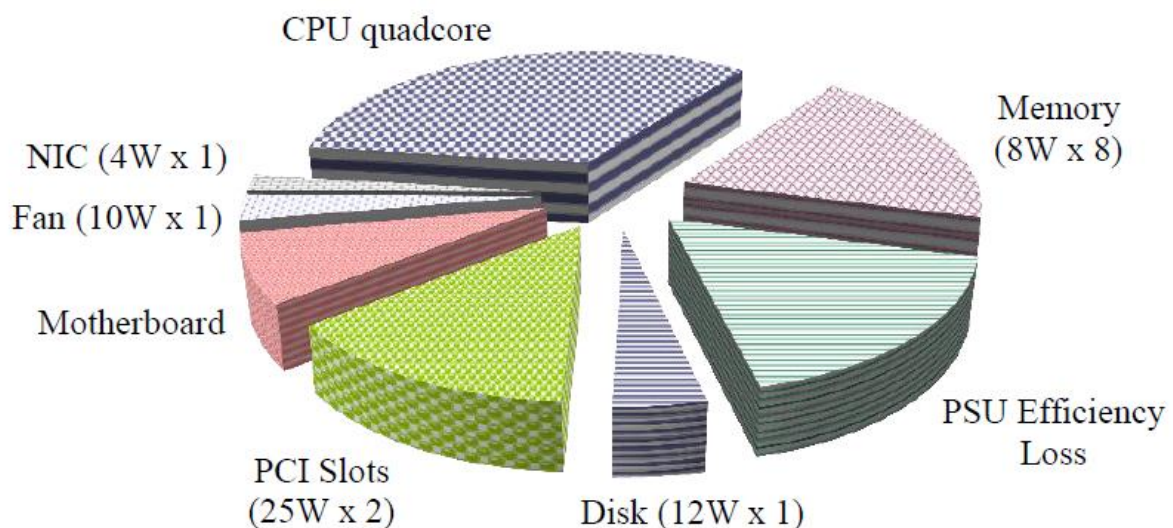


Figure 3.2: Power consumption by server components

In contrast, dynamic power ranges of all the other server components are much narrower: less than 50% for Dynamic Random Access Memory (DRAM), 25% for disk drives, 15% for network switches, and negligible for other components [27]. The reason is that only the CPU supports active low-power modes, whereas other components can only be completely or partially switched off. However, the performance overhead of a transition between the active and inactive modes is substantial. For example, a disk drive in the deep-sleep mode consumes almost no power, but a transition to the active mode incurs latency 1000 times higher than the regular access latency. Power inefficiency of the server components in the idle state leads to a narrow overall dynamic power range of 30%: even if a server is completely idle, it still consumes more than 70% of its peak power.

Another reason for the reduction of the fraction of power consumed by the CPU relatively to the whole system is the adoption of multi-core architectures. Multi-core processors are substantially more efficient than conventional single-core processors. For example, servers built with recent Quad-core Intel Xeon processor can deliver 1.8 teraflops at the peak performance, using less than 10 kW of power. To compare with, Pentium processors in 1998 would consume about 800 kW to achieve the same performance [28].

The adoption of multi-core CPUs along with the increasing use of virtualization and data intensive applications resulted in the growing amount of memory in servers. In contrast to the CPU, DRAM has a narrower dynamic power range, and power consumption by memory chips is increasing. Memory is packaged in dual in-line memory modules (DIMMs), and power consumption by these modules varies from 5 to 21 W per DIMM for the DDR3 and fully buffered DIMM (FB-DIMM) memory technologies [28]. Power consumption by a server with eight 1 GB DIMMs is about 80 W.

3.2.3 Modelling Power Consumption

To develop new policies for DPM and understand their impact, it is necessary to create a model of dynamic power consumption. Such a model should be able to predict the actual value of power consumption by a system based on some run-time system characteristics. One of the ways to accomplish this is to utilize power monitoring capabilities that are built into modern computer servers. These capabilities allow the monitoring of power consumption by a server in real-time and collecting accurate statistics of the power usage. Based on the data,

it is possible to derive a power consumption model for a particular system. However, this approach requires collecting data for every target system.

Fan et al. [27] found a strong relationship between the CPU utilization and total power consumption by a server. The idea behind the proposed model is that power consumption by a server grows linearly with the growth of the CPU utilization from the value of power consumption in the idle state up to the power consumed when the server is fully utilized. This relationship can be expressed as shown in (3.5).

$$P(u) = P_{idle} + (P_{busy} - P_{idle})u \quad (3.5)$$

Extensive experiments on thousand nodes under different types of workloads (Figure 3.3) showed that the derived models accurately predict power consumption by server systems with the error below 5% for the linear model and 1% for the empirical model. The calibration parameter r was set to 1.4 to obtain the presented results. These precise results can be explained by the fact that the CPU is the main power consumer in servers and, in contrast to the CPU, other system components (e.g., I/O, memory) have narrow dynamic power ranges or their activities correlate with the CPU activity. For example, current server processors can reduce power consumption up to 70% by switching to low-power-performance modes [21]. However, dynamic power ranges of other components are much narrower: < 50% for DRAM, 25% for disk drives, and 15% for network switches.

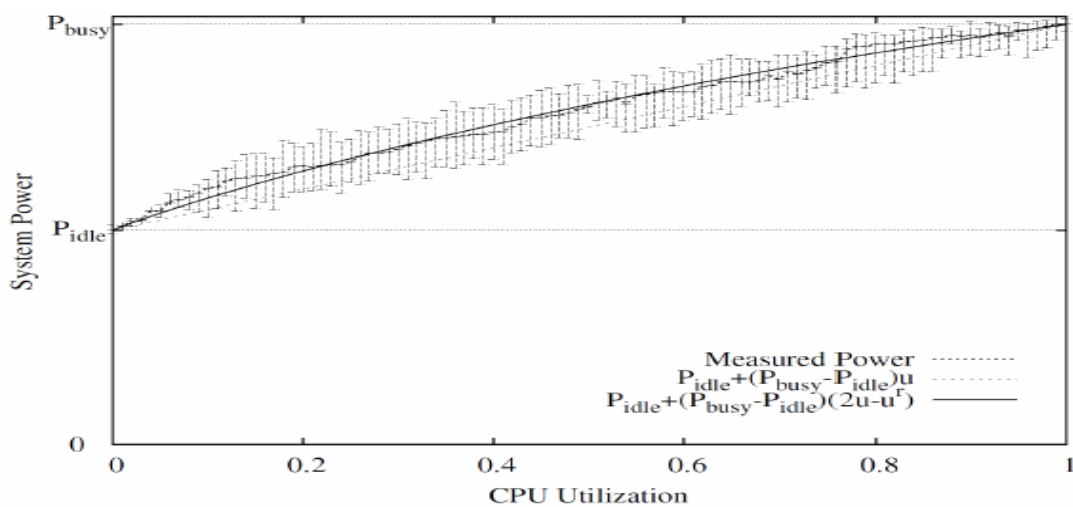


Figure 3.3: The relation between power consumption and CPU utilization of a server [27]

Dhiman et al. [29] found that although regression models based on just the CPU utilization are able to provide reasonable prediction accuracy for CPU-intensive workloads, they tend to be considerably inaccurate for prediction of power consumption caused by I/O and memory-intensive applications. The authors proposed a power modelling methodology based on Gaussian mixture models that predicts power consumption by a physical machine running multiple virtual machine (VM) instances. To perform predictions, in addition to the CPU utilization, the model relies on run-time workload characteristics such as the number of instructions per cycle (IPC) and the number of memory accesses per cycle (MPC).

3.3 The State of Art in Energy-Efficient Computing System

A large volume of research has been done in the area of power and energy-efficient resource management in computing systems. As power and energy management techniques are closely connected, in this chapter they are referred to as power management. As shown in Figure 3.4, at the high-level, power management techniques can be divided into static and dynamic.

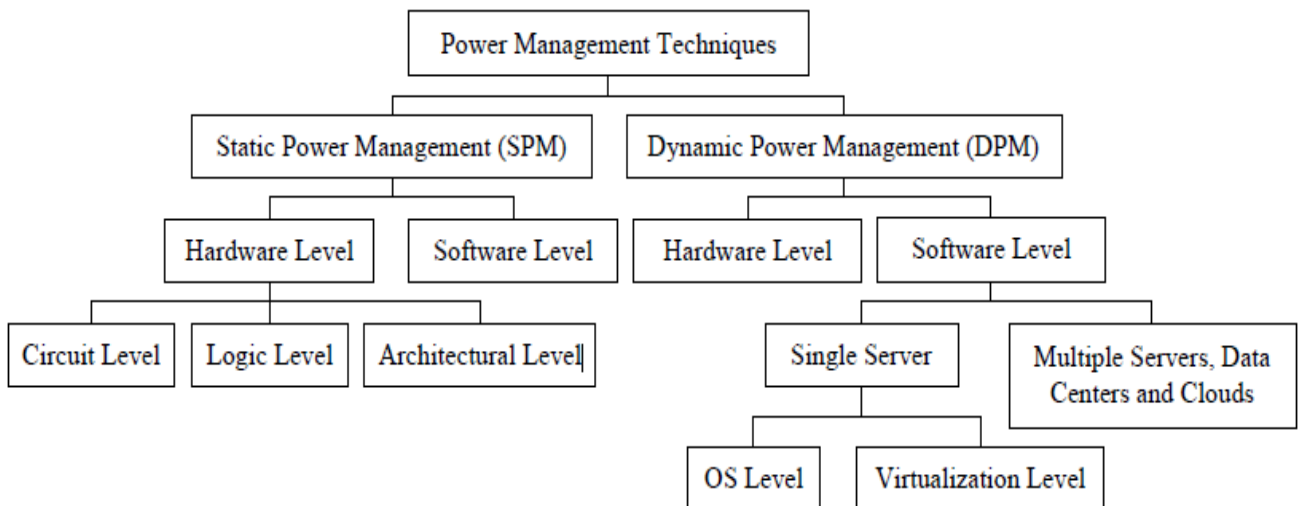


Figure 3.4: A high-level taxonomy of power and energy management

3.3.1 Hardware Level

DPM techniques applied at the hardware and firmware level can be broadly divided into two categories: Dynamic Component Deactivation (DCD) and Dynamic Performance Scaling (DPS).

3.3.1.1 Dynamic Component Deactivation (DCD)

Computer components that do not support performance scaling and can only be deactivated require techniques that leverage the workload variability and disable the components when they are idle. The problem is trivial in the case of a negligible power and performance overhead of transitions between power states. However, in reality such transitions lead to not only delays, which can degrade performance of the system, but also to additional power draw caused by the initialization of the components in transition from off to on.

3.3.1.2 Dynamic Performance Scaling (DPS)

DPS includes power management techniques that can be applied to computer components supporting dynamic adjustment of their performance proportionally to power consumption. In addition to complete deactivation, some components, such as the CPU, allow gradual reductions or increases of the clock frequency along with adjustments of the supply voltage. This approach is useful when the resource is not fully utilized. The widely adopted DVFS technique is an example of DPS. DVFS reduces the number of instructions a processor can issue in a given amount of time, thus reducing the performance.

Although the CPU frequency can be adjusted independently, frequency scaling by itself is rarely worthwhile as a way to conserve switching power. Most power savings come from dynamic voltage scaling in addition to frequency scaling because of the V^2 component and the fact that modern CPUs are strongly optimized for low voltage states.

3.3.2 Operating System Level

This section discusses research works that deal with power-efficient resource management in computing systems at the OS level. In particular, the proposed solutions are classified in regard to the following characteristics:

- **Application adaption** – whether the system requires modifications of the application level software to take advantage of power management.
- **System resources** – whether the system focuses on optimizing a single system resource, such as the CPU; or multiple system resources.
- **Target systems** – whether the approach is general and can be applied to an arbitrary system; or specializes on mobile devices or server hardware.
- **Power saving techniques** – DPS techniques, such as DVFS; DCD techniques; or just making the resources idle without explicit changes in hardware power states, which is referred to as resource throttling.

3.3.3 Virtualization Level

A technology that is able to improve the utilization of server resources, and thus, reduce power consumption, is virtualization of computing resources. Virtualization introduces an abstraction layer between an OS and hardware. Physical resources can be split into a number of logical slices called Virtual Machines (VMs). Each VM can accommodate an individual OS creating for the user a view of a dedicated physical resource and ensuring the performance and failure isolation between VMs sharing a single physical machine.

Virtualization allows one to create several VMs on a physical server; and therefore, reduce the amount of hardware in use and improve the utilization of resources. The concept originated with the IBM mainframe OSes of the 1960s, but was commercialized for x86-compatible computers only in the 1990s. Several commercial companies and open source projects now offer software packages to enable the transition to virtual computing.

The virtualization layer lies between the hardware and OS, and is implemented by a Virtual Machine Monitor (VMM). The VMM takes control over the resource multiplexing and manages the allocation of physical resources to the VMs. There are two ways in which a VMM can participate in power management:

- A VMM can act as a power-aware OS: monitor the overall system performance and appropriately apply DVFS or any DCD techniques to the system components.
- A VMM can leverage the power management policies applied by the guest OSes using the application-level knowledge, and map power management commands issued by the OSes of different VMs on actual changes in the hardware power state, or enforce system-wide power limits in a coordinated manner.

There are following three most popular virtualization technology:

- Xen Hypervisor
- VMware solutions
- Kernel-based Virtual Machine (KVM)

3.3.4 Data Center Level

This section discusses recent research efforts in the area of power management at the data center level. Although DVFS provides an efficient way of managing power consumption of the CPU, the overall dynamic power range of servers remains narrow. Even if a server is completely idle, it still consumes up to 70% of power. Eliminating static power consumption by servers is only possible by switching them off, or to a low-power mode, such as the sleep mode. These circumstances have led to the creation of various data center level solutions aimed at consolidating the workload to fewer physical servers and deactivating the idle servers, which improves the utilization of resources and reduces power / energy consumption.

3.4 Conclusions

In recent years, energy efficiency has emerged as one of the most important design requirements for modern computing systems, ranging from single servers to data centers and Clouds, as they continue to consume enormous amounts of electrical power. Apart from high operating costs incurred by computing resources, this leads to significant emissions of CO₂ into the environment, and these are growing rapidly in now a days.

It has been shown that intelligent management of computing resources can lead to a significant reduction of energy consumption by a system, while still meeting performance

requirements. One of the significant advancements that have facilitated the progress in managing compute servers is the implementation of the ability to dynamically adjust the voltage and frequency of the CPU (DVFS).

To counter the increasing energy consumption and greenhouse gases emissions caused by cloud data centers, recent studies focused on optimizing the utilization of hosts via dynamic consolidation (with or without reconfiguration) of virtual machines. In this model, VMs with low utilization are placed together on a single host, which enables the other hosts initially hosting the VMs to be shut down. Laszewski et al. [10] developed an algorithm for energy-efficient scheduling of VMs in hosts that are part of a virtualized cluster. These approaches see VMs as “black boxes” and thus cannot guarantee the deadlines of urgent applications within the VM.

The dynamic voltage and frequency scaling (DVFS) technique is commonly used to reduce the power consumption of electrical devices such as cell phones, PDAs, and PCs. The power consumption of an integrated circuit is proportional to the simple formula fcv^2 , with f the frequency, c the capacitance, and v the voltage. Thus, the supply voltage and work frequency profoundly affect the energy consumption of integrated circuits. The DVFS enables integrated circuits to run at different combinations of frequencies and voltages. Voltage supply can be increased or decreased depending upon circumstances. The DVFS can dynamically lower down the supply voltage and work frequency to reduce the energy consumption while the performance can satisfy the requirement of a job.

In [6], the Maximum and Minimum Frequencies DFVS (MMF-DVFS) algorithm is presented to reduce the energy consumption of processors. In this paper, authors model an optimal energy consumption formula. They use the linear combination of the minimum and maximum processor frequencies to approach the optimal energy consumption. They find two suitable time t_1 and t_2 for which task must be executed on maximum frequency and minimum frequency sequentially. However, this method can only be used in homogeneous computing systems.

MVFS-DVFS [8] algorithm explored the use of a linear combination of more than one voltage-frequency to full utilize slack time and reduce energy consumption on processors. They divide the total execution time in discrete time interval and for each time interval find

an optimal frequency level to execute the task. This approach is an extension of MMFDVFS algorithm.

Chia-Ming Wu and RS Chang [2] developed a green energy efficient scheduling algorithm for cloud datacenters. They consider resources acquired by VM as weight of VM and then allocate VM to the task according to the weight assigned to VM. If jobs not satisfied by the VM then choose next VM with next higher weight. They have also taken SLA level in consideration which should satisfied along with the task.

Li et al. [11] developed an energy-aware algorithm for scheduling tasks on heterogeneous clusters. However, the approach tries to minimize the execution time without enforcing deadline for applications. It also does not explore DVFS. Rather, it takes into account the different power consumption incurred by heterogeneous machines.

Chetsa et al. [12] developed a technique where characteristics of HPC applications are inferred at runtime and measures for energy savings are applied based on the characteristics of the application. Although the approach does not address the problem of deadline-constrained applications, it can be used in conjunction with our algorithm. This is because we focus on dynamic scaling the CPU frequency, whereas Chetsa et al. approach also applies energy-saving measures to other system components, such as network, hard disk, and memory.

The utilization of DVFS as a means to reduce energy consumption of applications has been explored in the last years in related areas. Ruan et al. [13] and Wang et al. [14] applied the technique in workflow applications on clusters, whereas Tian et al. [15] apply the approach for web servers. Our approach is designed for urgent, CPU-intensive BoT applications, where there are no execution dependencies between tasks, and works based on user-defined deadlines for application execution.

Gregor von Laszewski [14], state scheduling algorithm focus on scheduling virtual machines such that to calculate cluster to minimize power consumption via Dynamic Voltage frequency scaling (DVFS). Algorithm [14] is implemented in a simulator DVFS-enabled cluster by dynamically scaling supplied voltage. This is scenario operated for multiple voltages and frequencies, virtual machines are scheduled such that minimum voltage and frequency is required.

Eyerman and Eeckhout [16] and March et al. [17] propose techniques for fine grain control of frequency and voltage at the CPU level. Eyerman and Eeckhout [16] explores regulation of frequency to slow down the flow of CPU instructions through CPU units (such as buffers and functional units) in the event of cache misses that disrupt the flow of execution of operations being processed. March et al. [17] apply DVFS and task migrations to balance load among CPU cores, what in some circumstances enables the whole CPU to execute at a lower frequency/voltage. These approaches are complementary to ours, as they operate at the CPU level whereas our approach is applied at middleware/Operating System level. Thus, one could expect further reduction in energy consumption if our approach is applied in conjunction with Eyerman and Eeckhout's approach at CPU level.

Kim et al. [18] and Zhang et al. [19] developed heuristics for scheduling deadline-constrained BoT applications in clusters and heterogeneous distributed systems, respectively. Kim et al. [18] approach requires each execution node to have knowledge about the total consumption from the underlying hardware. Although this is a reasonable approach for clusters, this is not reasonable in cloud systems, where nodes are virtual machine that, because of security issues, cannot have access to information about energy consumption from other VMs in the same host. Zhang et al. [19] approach is also not tailored for cloud environments, where VMs share the same hardware and therefore their potential co-location affects the total consumption caused by the host. Therefore, a different approach, like ours, is necessary for cloud infrastructures.

Wang and G von Laszewski [14], states a scheduling algorithm in DVFS-enabled clusters is proposed for executing parallel tasks. The proposed algorithm finds the slack time for non-critical jobs without increasing the scheduling length. They also develop a green SLA-based mechanism to reduce energy consumption by increasing scheduling make-spans. In their work, they sacrifice the performance to reduce the power consumption or CO2 emission. They provide an acceptable extension of execution time to meet the green SLA.

Most of the author only considered the dynamic part of processor power which is basically depends on supplied voltage and frequency, but processor consumed constant static power as long as processor is switched on. We can't ignore static power consumed by power as stated

by Euseong Set, Jinkyu Jeong [5], Weixun Wang and Prabhat Mishra [9]. They stated that in the low frequency domain static power goes higher than the dynamic power consumption. So we must consider static power too while designing the algorithm for energy efficient scheduling. In my work, I considered both dynamic as well as static power consumption in the CPU.

5.1 System and Application Model

Our target system model is depicted in Figure 5.1. It is composed of a virtualized IaaS cloud data center that supports a PaaS layer that is available to multiple users for execution of CPU-intensive real-time applications. The data center is composed of a number of (potentially heterogeneous) physical servers (hosts). Virtual machines are executed in the servers, and they are managed by a virtual machine manager installed on each host.

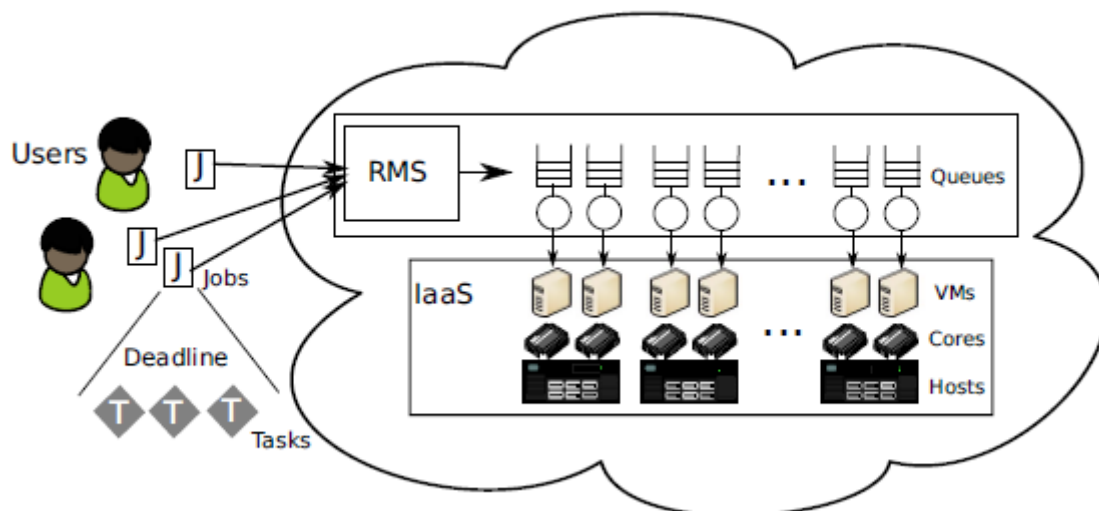


Figure 5.1: System and application models. A cloud data center containing VMs with dedicated CPU cores execute, on its PaaS layer, real-time CPU-intensive task. The CPU frequency of each core/VM can be independently adjusted in order to balance energy utilization and application deadlines.

VMs do not share CPU cores with other VMs in the same host. It means that the CPU core is exclusively available for the VM (a configuration that can be obtained in current virtual machine managers such as Xen). Each VM has exclusive use of a share of the host's memory.

Only one task is executed simultaneously on each VM, and if more than one task is scheduled to the same VM, the remaining tasks are kept in waiting queue.

The application model consists of CPU-intensive real time task submitted by users of the PaaS service. The job execution request for a job contains the following information:

- (i) The independent task t_1, t_2, \dots, t_m that compose the job.
- (ii) Estimated runtime e_i of each task t_i executed at the maximum frequency of the CPU.
- (iii) A deadline d_i for the job to complete.

Tasks are non preemptable, and the completion time of the job is computed as the completion time of the last task. It means that scheduling algorithm has to operate in such a way that all the tasks that compose the job are finished before the job deadline.

In the proposed model, jobs have “soft deadlines”, which means that violations on deadlines do not render the computation irrelevant, but reduces its value for the user. As loss of value of the computation is proportional to the amount of delay for job completion regarding its deadline, it is important that deadlines are not missed at all, or are missed by just a small margin. Deadlines missed by large margins result in the computation being useless for the user and therefore the energy spent on its computation can be considered wasted. Therefore the job deadline cannot be met, the request is rejected.

The objective of the cloud platform is to balance the energy consumption of the system and comply with application deadlines. This is achieved with the proper configuration of the frequency of VMs to speed up or slow down the CPU core allocated to each VM (and consequently reduce its energy consumption). We assume that the frequency and voltage of each core can be individually managed (a set up that is demonstrated [30] to provide more energy efficiency than per-chip DVFS). Formally, each host h_i of the data center has m cores, c_1, c_2, \dots, c_m . Therefore different hosts can have different number of cores. Each core c can have different CPU levels to run in term of frequency.

The power consumed by each host is calculated based on the individual contribution of each core of the host. Furthermore, we assume that hosts that are not in use (i.e., all its VMs are idle, and therefore their respective CPU cores are in the idle state) are suspended until one of

the VMs is necessary, so the host is restored to a full operational mode. Also, as the time for scaling the CPU frequency is at the scale of nanoseconds while application execution is minutes or hours, we assume as negligible the time taken by the CPU core to reach the desired frequency level. The total energy consumed by a host is a function of the frequency level of each of its cores.

5.2 Energy Model

We assume that cores of the processor support DVFS. For DVFS, each core has different voltage/frequency levels, and denoted as $\{(v_1, f_1), (v_2, f_2), \dots, (v_{max}, f_{max})\}$. Power consumption associated with each m-threaded processor is given as:

$$P = C + i\delta \quad (5.1)$$

Where P=total power consumption of the processor, C=base power, i=number of active thread and δ =thread power. Power of a single thread (δ) is given as:

$$\delta = K * f_i^3 \quad (5.2)$$

Power can be computed as static power (P_s) and dynamic power (P_d)

$$P_s = C \quad (5.3)$$

$$P_d = K * f^3 \quad (5.4)$$

Energy consumption of executing a task t_i at frequency f_j is defined as

$$E(t_i, f_j) = p_j * e_{ij} \quad (5.5)$$

Where e_{ij} is execution time of task t_i at frequency f_j .

Total energy consumption E can be represented as follows:

$$E = E_d + E_s \quad (5.6)$$

Where E_d is the dynamic energy consumption when the processor is executing tasks, E_s is the static energy consumption due to the base power(C) of the processor.

$$E_d = P_d(f_j) * e_{ij} \quad (5.7)$$

$$E_s = C * t_l \quad (5.8)$$

If e_i is the execution time of the task t_i at frequency f_{max} then execution time of t_i at frequency f_j is given as $e_{ij} = f_{max} * \frac{e_i}{f_j}$. And t_l is the maximum finish time among all tasks.

Hence Total energy consumption of a host having m core is given as:

$$E = K * f_{max} * \sum_{j=1}^m f_j^2 * e_i + C * \max_{\forall j \in m} \left\{ f_{max} * \frac{e_i}{f_j} \right\} \quad (5.9)$$

5.3 Methodology Used

Given a set of task T where each task is associated with execution time e_i and deadline d_i . Our aim is to find the frequency assignment for each thread in which task t_i is scheduled to complete its execution.

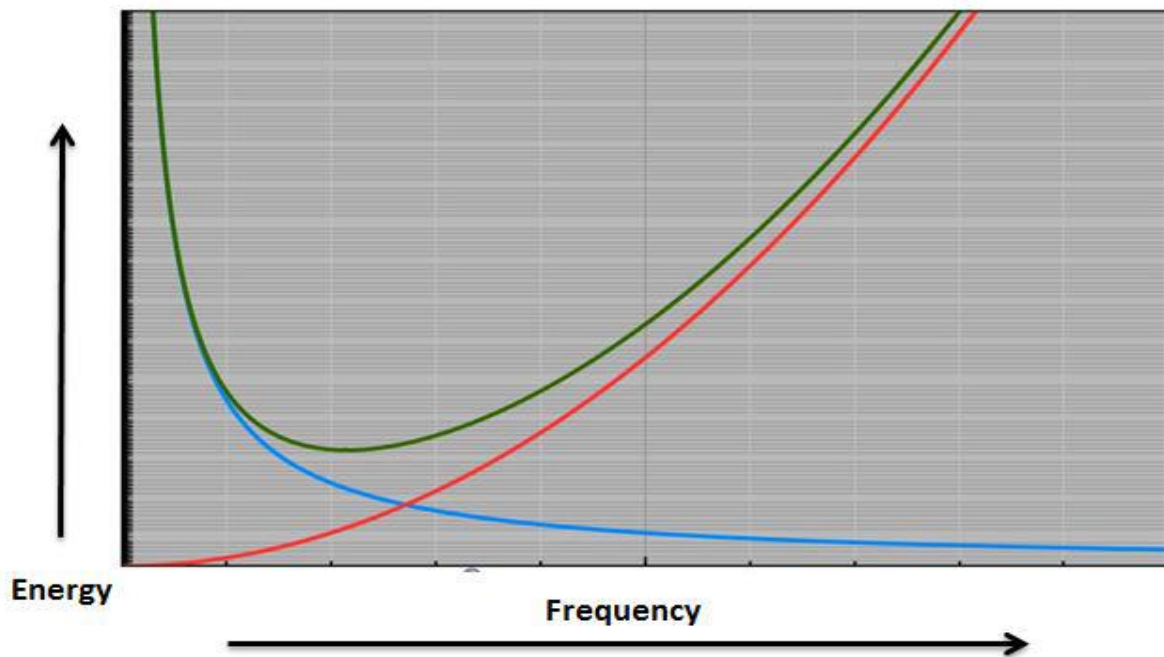


Figure 5.2: Energy-frequency curve for single task

We assume that the frequency assignment $\{f_1, f_2, \dots, f_m\}$ where overall energy is minimum must be exist before the point where both dynamic and static energy equal. In above graph (Figure 5.2) red line represents the dynamic energy, blue line represents the static energy and green line represents the overall energy.

To minimize the energy consumption of a data center, this problem can be reduce to the minimizing the energy consumption of each host in the data center. Our aim is to find the frequency $f_{ij} \in (f_{d_{ij}}, f_{max})$ for each core (at which task t_i is scheduled for execution) of m -core host for which overall energy is minimum.

Now overall minimum energy is reduced to each individual's host minimum energy which can be expressed as:

$$E = \min \left\{ K * f_{max} * \sum_{j=1}^m f_j^2 * e_i + C * \max_{\forall j \in m} \left\{ f_{max} * \frac{e_i}{f_j} \right\} \right\} \quad (5.10)$$

Now our task is to find the frequency assignment for core of a single processor only for which energy should be minimum. On the basis of above fact we proposed two strategies one is based on increasing dynamic energy and other is based on decreasing static energy.

Deadline frequency: minimum frequency required by the core to achieve the deadline of the task scheduled on it. Let a task scheduled on i^{th} core has execution time e_i at frequency f_{max} and deadline d_i then at frequency f_i execution time become e'_i and given as:

$$e'_i = \frac{e_i * f_{max}}{f_i} \leq d_i \quad (5.11)$$

$$f_{di} = \frac{e_i * f_{max}}{d_i} \quad (5.12)$$

Both the algorithm starts from initial frequency set which is deadline frequency of each thread. First strategy is based on minimizing static energy by increasing dynamic energy so named as Dynamic-Optimization and second strategy is based on minimizing longest finish time which is associated with static energy so named as Static-Optimization.

5.3.1 Dynamic-Optimization

In this strategy we are focused on dynamic energy. We start from initial frequency set $F[]$ which is deadline frequency (f_d) of each core. As we discussed our assumption earlier that minimum energy must be exist before the point where both dynamic energy and static energy are equal. Hence according to the above fact while static energy is more than the dynamic energy we should try to increase dynamic energy optimally. Increase optimally means increase the frequency of such core for which dynamic energy increases minimally i.e among all next available frequencies increase the frequency for which is minimum. Among all possible frequency set optimal frequency set is the set for which overall energy is minimum. Algorithm for this strategy is described below.

Algorithm 1: Dynamic Optimization

1. Generate initial frequency set: deadline frequency f_{di} for every core
 $F[1 \dots m] = \{f_{d1}, f_{d2}, \dots, f_{dm}\}$
2. Calculate static energy (Es) and dynamic energy(Ed):
 $Es = c * \max_{vi \in m} \left\{ f_{max} * \frac{e_i}{f_i} \right\}$ and $Ed = K * f_{max} * \sum_{i=1}^m f_i^2 * e_i$
3. Initialize min frequencies: $MIN - F[] = \{f_{d1}, f_{d2}, \dots, f_{dm}\}$ and overall minimum energy $MIN - E_{total} = Ed + Es$
4. Initialize a min priority queue 'Q' with the next frequency level (i.e $f_{di} + 1$) of each core. Queue priority is based on $f_i^2 * e_i$ (dynamic part).
5. **While** $Ed < Es$ and Q is not empty **do**
 - a. Increase frequency of core for which $f_i^2 * e_i$ is minimum i.e. $F[i] = f_i + 1$
 - b. Push the next frequency of this core into Q, i.e $F[i] + 1 \leq f_{max}$
 - c. Calculate static energy (Es) and dynamic energy(Ed) for the new frequency set
 - d. Update the $MIN - F[]$, $MIN - E_{total}$ for the new Ed, Es and $F[]$
6. **end While**
7. $MIN - E_{total}$ is the minimum energy and $MIN - F[]$ is frequency assignment corresponding to $MIN - E_{total}$

5.3.2 Static-Optimization

In this strategy we are focused on static energy. Our algorithm starts from initial frequency set which is deadline frequency of each thread. Here we should try to decrease the static energy while static energy is more than dynamic energy. Static energy is depends on the longest finish time in the processor so in this strategy our task is to minimize the longest finish time of the processor. Now At each step we increase the frequency of the thread for which current $\frac{e_i}{f_i}$ (finish time) is maximum. This strategy is repeated while we reach the point where dynamic energy equal to static energy $Ed \geq Es$ or f_{max} for that processor. Algorithm for this strategy is described below.

Algorithm 2: Static-Optimization

1. Generate initial frequency set: deadline frequency f_{di} for every core,
 $F[1 \dots m] = \{f_{d1}, f_{d2}, \dots, f_{dm}\}$
2. Calculate static energy (Es) and dynamic energy(Ed):
 $Es = c * \max_{\forall i \in m} \left\{ f_{max} * \frac{e_i}{f_i} \right\}$ and $Ed = K * f_{max} * \sum_{i=1}^m f_i^2 * e_i$
3. Initialize min frequencies: $MIN - F[] = \{f_{d1}, f_{d2}, \dots, f_{dm}\}$ and overall minimum energy $MIN - E_{total} = Ed + Es$
4. **While** $Ed < Es$ **do**
 - a. Increase frequency of core for which $\frac{e_i}{f_i}$ is maximum i.e. $F[i] = f_i + 1$ with condition $f_i + 1 \leq f_{max}$
 - b. Calculate static energy (Es) and dynamic energy(Ed) for the new frequency set.
 - c. Update the $MIN - F[]$, $MIN - E_{total}$ for the new Es, Ed and $F[]$
5. **end While**
6. $MIN - E_{total}$ is the minimum energy and $MIN - F[]$ is frequency assignment corresponding to $MIN - E_{total}$

5.4 Proposed Approach

In this strategy we proposed scheduling algorithm is to dynamically scale the frequency of CPUs assigned to a virtual machine in such a way that tasks sequentially executed in the VM complete before their deadlines.

In a cloud environment, hardware is shared among VMs. It means that the energy consumption of a host is not determined by a single VM, but by the combined state of all the VMs. Furthermore, when scheduling a task, the physical location of the VM can be taken into consideration, in such a way that requests are preferentially submitted to VMs whose host is already in use (i.e., other VMs in the same host are already executing tasks). This enables the system to consolidate the load whenever possible. This in turns helps in reducing the total energy consumption of the infrastructure, as it enables unused hosts to be suspended or kept in low power states. Figure 5.3 shows the flow diagram of the proposed approach and respective algorithm is given below.

Algorithm 3: DVFS-Aware Energy-Efficient Scheduling

Input & Initialization:

- (i) List of cloudlets $C, (t_1, t_2, \dots, t_r)$
- (ii) List of requested virtual machine $(VM_1, VM_2, \dots, VM_n)$
- (iii) List of Hosts (H_1, H_2, \dots, H_m)

1. Step 1. Proposed scheduler maintains:
 2. (i) VM_table (It will maintained list of host allocated to VM with their id)
 3. (ii) Each host maintain the PE_Count indicating the available processing
 4. elements (PE) on that host.
5. **Step 2. Allocate Host for VM**
6. **for each VM**
7. success \leftarrow false

8. **for each** Host

9. **if** PE_Count > 0 **then**

10. (i) Proposed Scheduler request VMM to create VM with

11. respective Host_id and VM_id and send acknowledgement.

12. (ii) Success \leftarrow true

13. (iii) Proposed Scheduler update the VM_table and PE_Count value.

14. (iv) **break** the inner loop.

15. **end if**

16. **end for**

17. **if** success=*false* **then**

18. **Return** failure

19. **end for**

 //After executing above all VM has assigned with resources/host or return failure.

20. **Step 3. Schedule cloudlets on VM**

21. (i).Schedule the cloudlets sequentially on VM with CloudletSchedulerSpaceShared policy.

22. (ii). Get the optimal frequency assignment for the VM with respect to task assigned on it. For getting frequency assignment, call the *Algorithm 1* and *Algorithm 2* and choose the best allocation.

23. (iii). According to the frequency assignments received from above adjust the MIPS of each VM (i.e. operating frequency of core).

24. **Step 4. Stop Simulation:**

25. (i) Stop the simulation process and release the resources assigned to VM.

26. (ii) Restore the PE_Count for each host.

27. **Step 5. Energy Calculation**

28. (i) Fetch the execution time for each VM from the output.

29. (ii) Calculate the energy consumption for each host with the help of equation (5.6), (5.7) and (5.8).

30. (iii) Total energy consumption is the sum of energy consumed at each host.

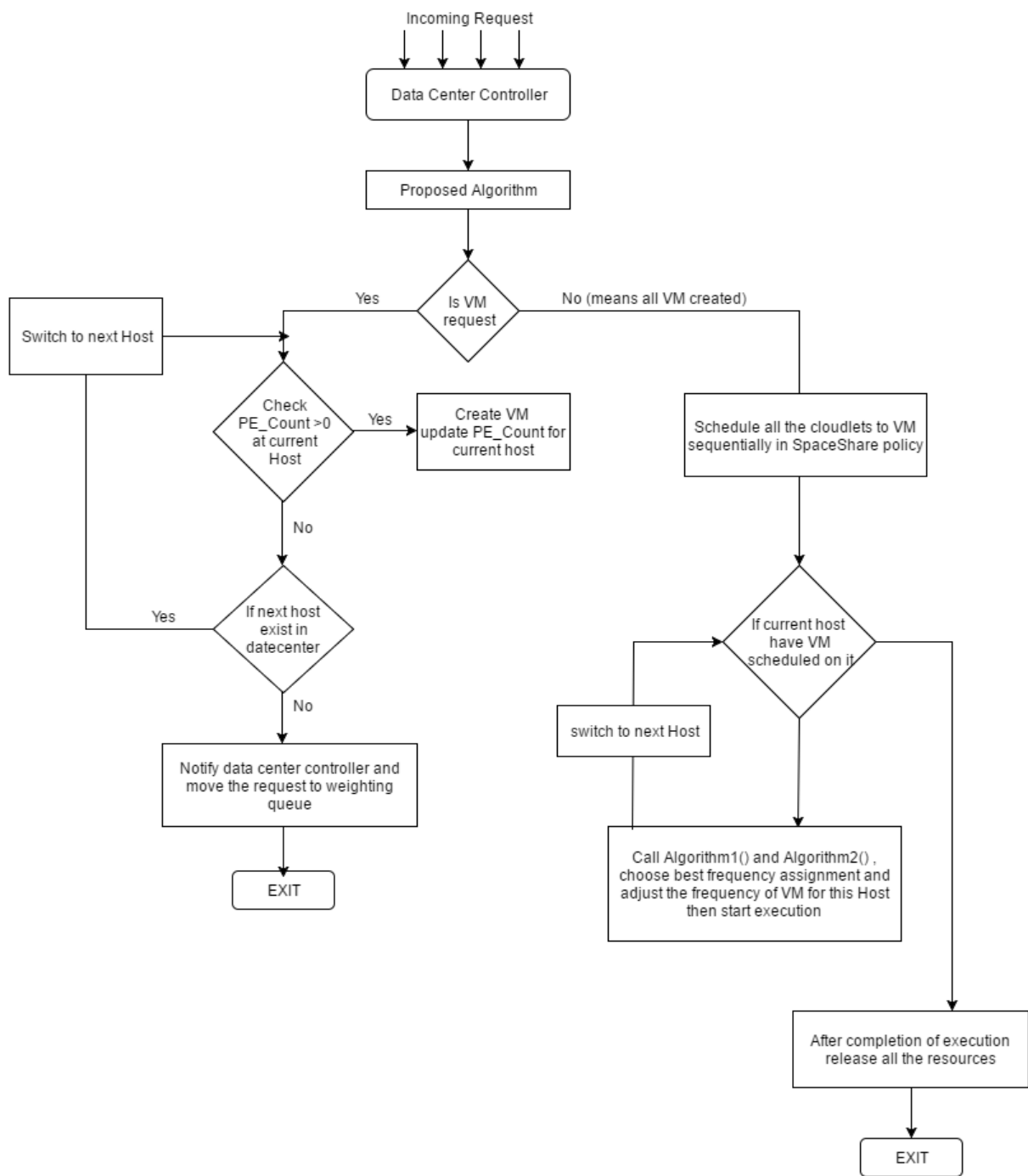


Figure 5.3: Flow diagram of the proposed algorithm

Proposed algorithm uses the optimal frequency level to execute the task so will consumes lesser energy as traditional method. This algorithm allocation and execution has summarised in four main steps which is described as follows:

In step 1 we are initializing the processing elements counts for each host and VM table. Since the work in thesis was done at host level, therefore as an input to the algorithm, all the details of cloudlets, virtual machines and hosts was provided.

Step 2 is basically the VM allocation policy in which host is assigned to requesting VM. In this physical location of the VM has been taken into consideration, in such a way that requests are preferentially submitted to VM whose host is already in use (i.e., other VMs in the same host are already executing tasks). This enables the system to consolidate the load whenever possible. This in turns helps in reducing the total energy consumption of the infrastructure, as it enables unused hosts to be suspended or kept in low power states. In our approach this has been accomplished by choosing the host sequentially for allocation host.

In the algorithm we iterate for each VM to allocate a host to it. During iteration we check if a host has free processing element then VM will be allocated to this host and if not then look for the next host and so on. This is done by taking a variable PE_Count which maintain the free processing elements for each host. If any host has free PE then scheduler request to create a VM on the respected host and update the VM_table.

In any case if VM request has arrived and no host has free processing elements the VM allocation fails and returns failure.

In the step three actual tasks are assigned onto the VM by using the CloudletSchedulerSpaceShared policy. After scheduling of cloudlets to VM it's time to assigned the frequency to VM for execution of task such that overall energy consumption is minimum. To get the frequency assignment for each VM for a host we call Algorithm 1 and Algorithm 2, and whichever give the best allocation of frequency that will be taken into consideration and then adjust the frequency of core to execute the task. Frequency should be chosen in such a manner that the entire task assigned to VM should be completed before the deadline. But we need not to worry about that because frequency assignment returned from algorithm 1 and 2 are aware of this deadline.

At the end simulation process is completed and all the resources assigned to VM should be released and variable should be restored to its original value. Now we can calculate the energy consumed in executing of given job by using the equation (5.6), (5.7) and (5.8) and simulation result. A detailed analysis on result will be given in next chapter.

Experimental Setup and Results

As discussed in previous chapter's energy efficiency is becoming the major goal in HPC. That's why a number of approaches have been given in this area which has their own assumption of system and application model. With the proposed methodology we consider both dynamic as well as static part of power consumption in HPC. Lowering the operating frequency not always save power it can lead to high static power in very low frequency domain. So in proposed method we find the frequency on which a balance is maintained between dynamic and static power consumption. This approach is more dynamic in nature if it found that static power is becoming larger than dynamic it increases the operating frequency to maintain a balance with subject to minimize energy consumption.

6.1 Used Technologies

This section presents a brief overview on the technologies used for the proposed work. The main technologies which have played an important role in implementing the proposed methodology are discussed below:

6.1.1 Java

Java has many attractive features which made it so popular among its users. These features help developers to deploy and develop many applications and services in Cloud computing environment. In cloud computing a platform is provided through which a user can interact with the cloud services and further using as per the needs. This platform must be accessible from any device whether it is a mobile, a laptop, a tablet, or any other device. Java's ability to be run on any platform makes this possible. Java's byte code made it more secure and portable for use. Many computational tasks performed in cloud network require smooth and quick networking services. Java has many networking features which are used in cloud computing network. Java's applets are one of the most popular features which have revolutionized the web and the internet technology. In implementing the proposed methodology java is used intensively. The core idea was implemented in java only.

6.1.2 CloudSim

A CloudSim [31] is just like other simulators which simulate certain scenarios as made by the users to test any policy or working of any other entity in a well-configured environment. For simulating the cloud computing scenarios we use CloudSim. CloudSim provides a vast number of choices for its user to test and model cloud computing system with different combination of provisioning policies and hardware resources. With the CloudSim one can model the cloud components with respect to system as well as behavior. These cloud components includes different datacenters, virtual machines (VMs), and various resource provisioning policies. CloudSim toolkit also offers federation of clouds i.e., internetworking of different types of clouds. Big organizations like HP Labs in U.S.A., are using CloudSim for their research and development work. The proposed load balancing algorithm is also tested using CloudSim toolkit. Figure 6.1 shows the layered architecture of CloudSim.

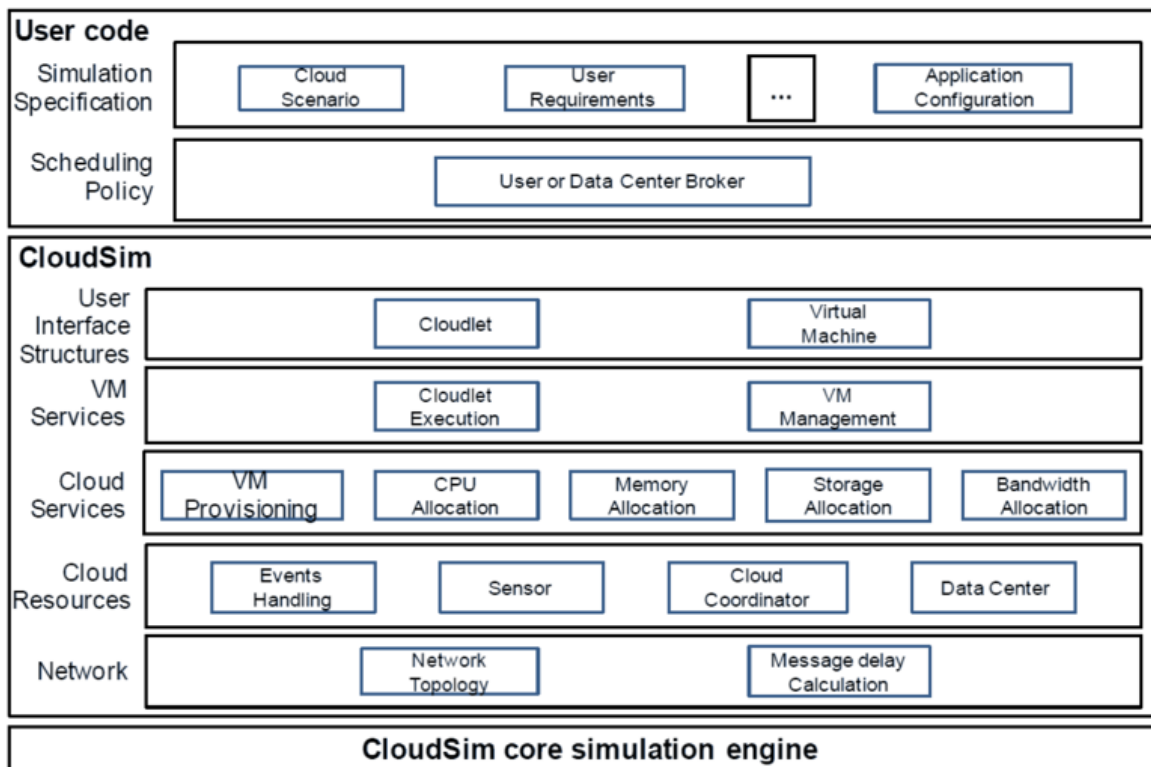


Figure 6.1: Layered architecture of CloudSim

6.2 Implementation Details

Implementation is done using the above technologies. For the real experiment constant parameter K and C used in power equation in previous chapter for static and dynamic power (equation (5.3), (5.4)) is taken from a actual CPU, **Intel Core Duo** for which $K = 2.213 \times 10^{-27}$ and $C = 35.6$ [32].

CloudSim is used to simulate the experiment. In order to provide the simulation framework more real environment certain assumptions regarding the cloud parameter has been taken. We assumed that processor used for simulation is DVFS enabled and operating frequency can be adjusted according to requirement. Machine characteristics are listed in table 6.1:

Parameters	Value
Data Centers	1 (can be increase)
Number of Hosts	5 (can be increase)
Processing Elements in each Host	4 (can be heterogeneous)
Virtual Machine	20 (only one VM per PE)
Cloudlet length	Flexible (input submitted by user)
Cloudlets Scheduler	CloudletSchedulerSpaceShared()
Virtual Machine RAM	2 GB
Virtual Machine MIPS	Adjustable (supporting DVFS), maximum MIPS=1000
Virtual Available Bandwidth	1000 Mbps
VM PEs requirement	1
VMM (Virtual Machine Monitor)	Xen
VM Scheduler	VmSchedulerTimeShared()
Host/Server RAM	8 GB

Host Storage	1000000 MB
Host Bandwidth	10000 Mbps
VM Allocation Policy	VmAllocationPolicyMySol() (allocation policy is given by proposed method)

Table 6.1: System Specification for simulation

In this simulation of the proposed algorithm is carried out and further comparison is done with the traditional approach and MMF-DVFS algorithm [6]. The algorithm is written in java and is run on Eclipse IDE using CloudSim toolkit. The program is implemented according to the flow of execution in CloudSim as shown in Figure 6.2.

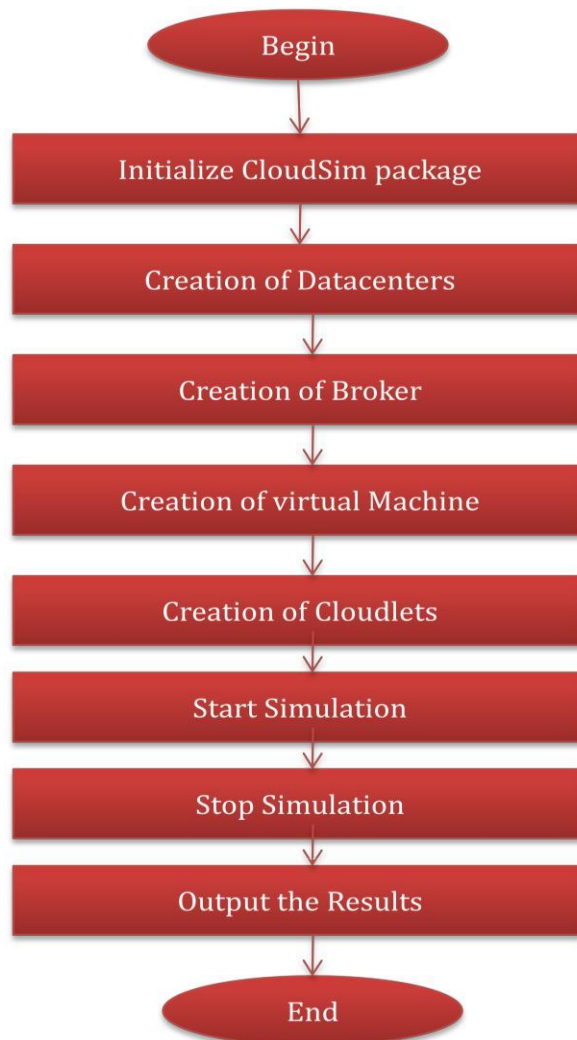


Figure 6.2: Flow of program execution in CloudSim

6.3 Experimental Results

We are taking the real time input from the user in a file which consist cloudlet size and deadline for each task. We schedule these cloudlets onto VM and execute them on suitable frequency which results lowest CPU energy consumption and fulfilled the deadline of task. For this many experiments has been performed and results are compared with the traditional approach, MMF-DVFS algorithm and proposed scheduling algorithm. All the experiments have performed on CloudSim toolkit.

For the present experimentation work a most realistic scenario has been considered. However, a proposed algorithm can be tested for a large number of scenarios, but in this thesis, for the sake of simplicity and understanding a general scenario in a cloud computing environment has been assumed. In this scenario, one datacenter is created consisting 4 hosts each are quad core. Data center broker requests for VM to perform his job and host will be assigned to VM in consolidated manner to save energy. Although the main code is written in such a way that a user can take any number of datacenters, hosts, virtual machines and cloudlets.

6.3.1 Output of MMF-DVFS

We implemented MMF-DVFS [6] to compare the results with proposed algorithm. Same input set with deadline constraint is given to both the approaches so that we can compare their results. With respect to cloudlet size and deadline, both approaches assign the appropriate frequency to processor core of each VM to execute in their manner and calculate the energy consumption at each host. Overall energy consumption is sum of energy consumption at each individual host with the assumption that energy consumed by idle host is zero.

So, for comparing result by both the approaches we first executed the given input set by MMF-DVFS. Output screenshot of MMF-DVFS is given below (Figure 6.3 – Figure 6.6):

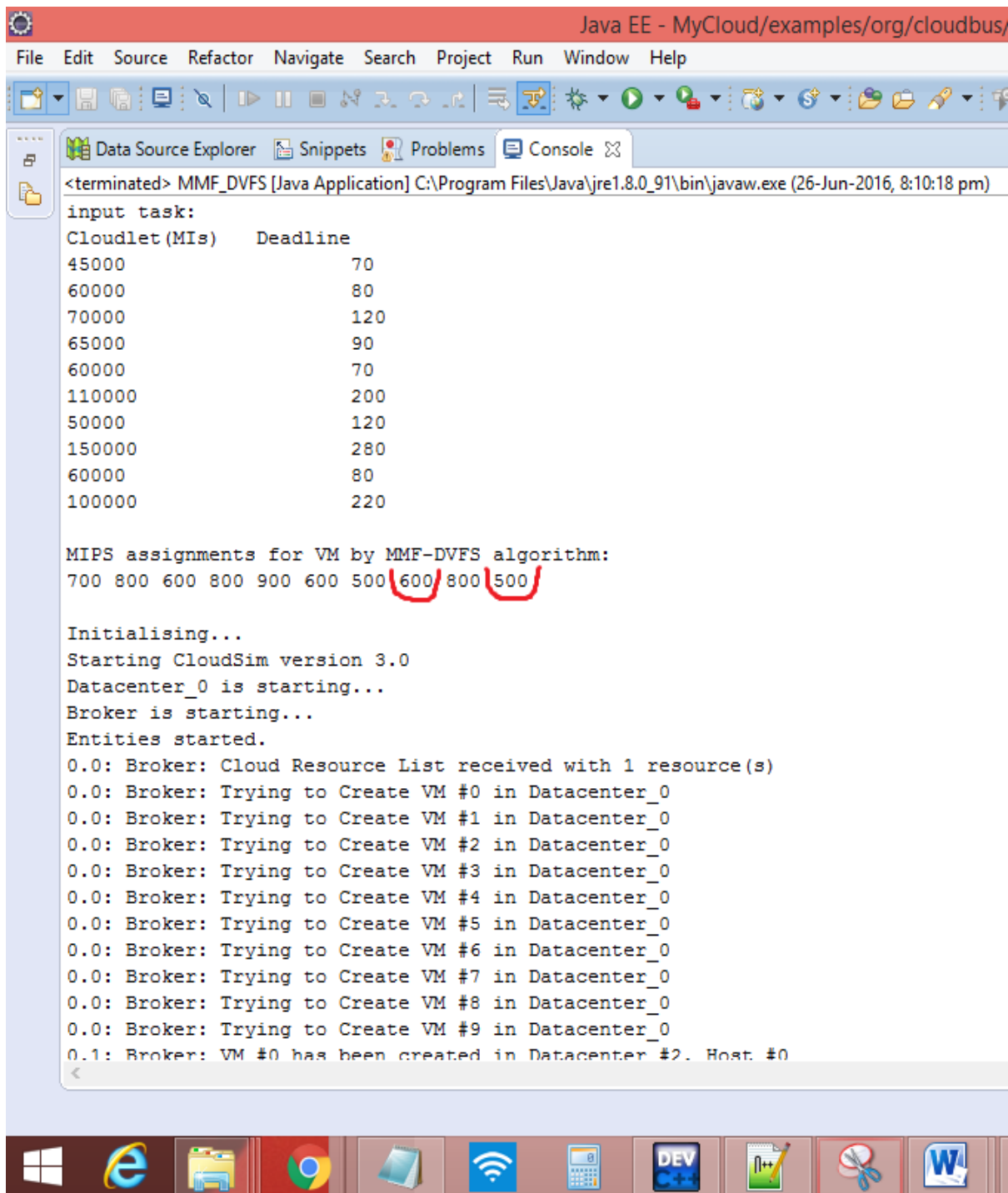


Figure 6.3: Screenshot 1 of 4 of MMF-DVFS

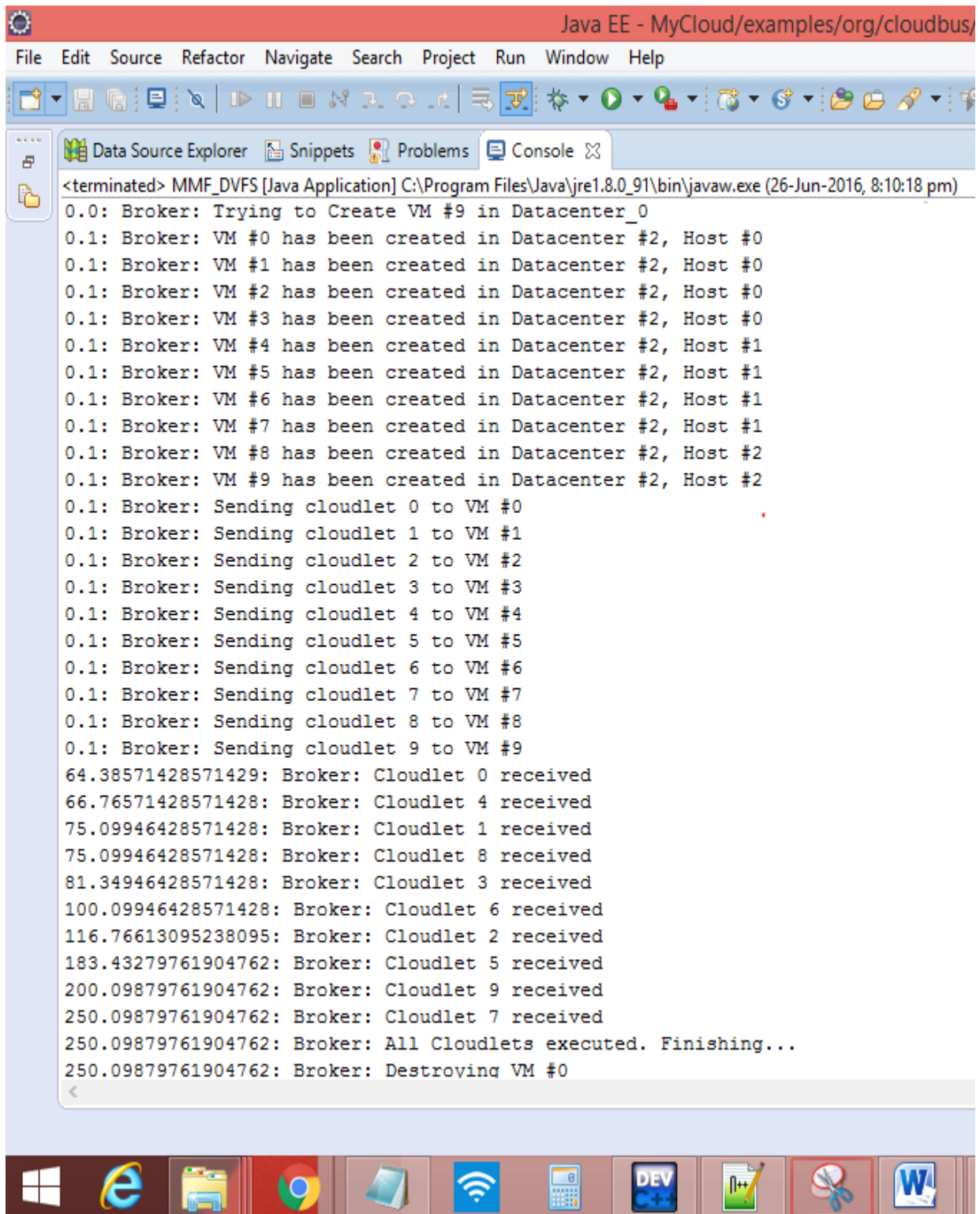


Figure 6.4: Screenshot 2 of 4 of MMF-DVFS

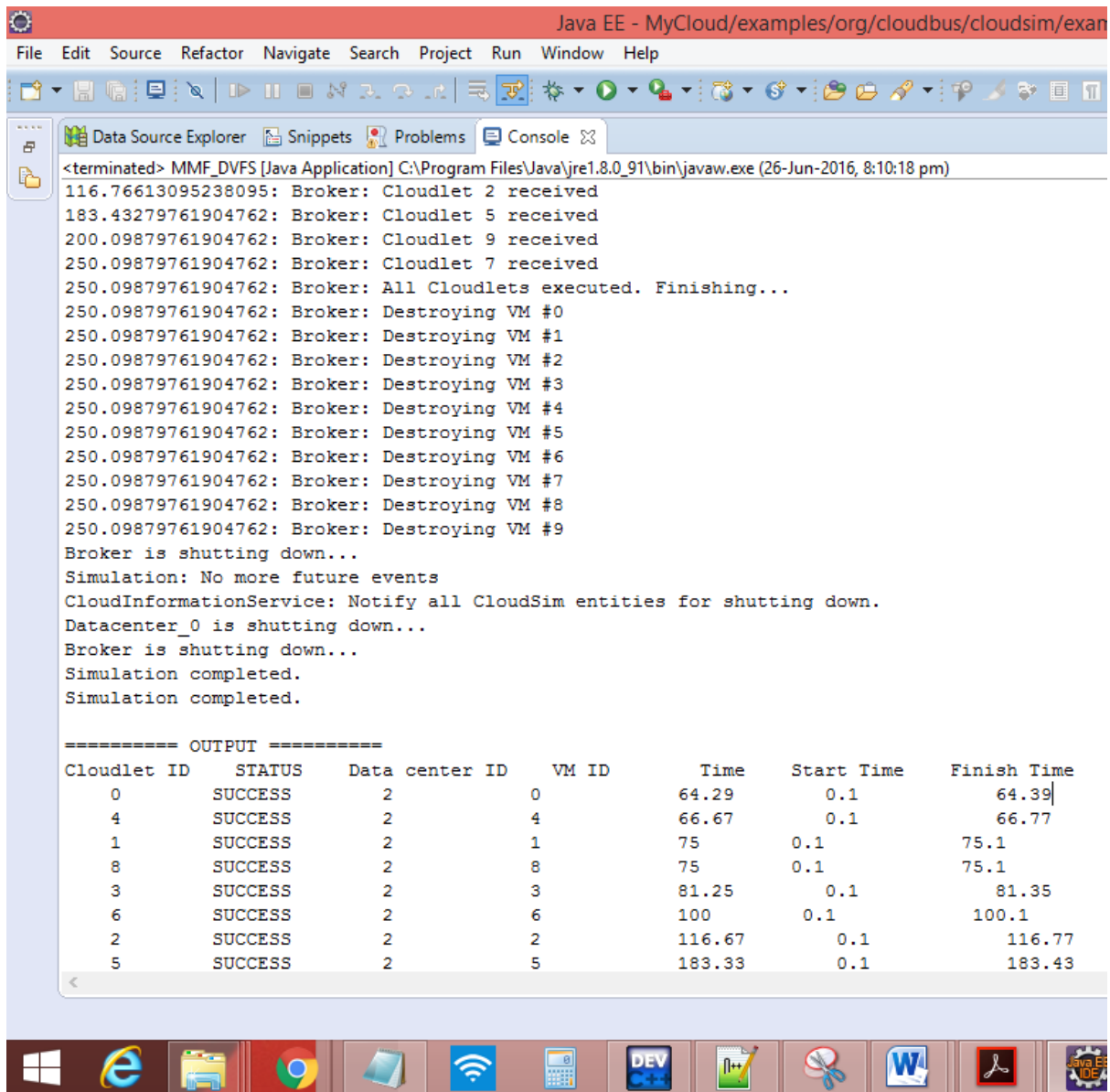


Figure 6.5: Screenshot 3 of 4 of MMF-DVFS

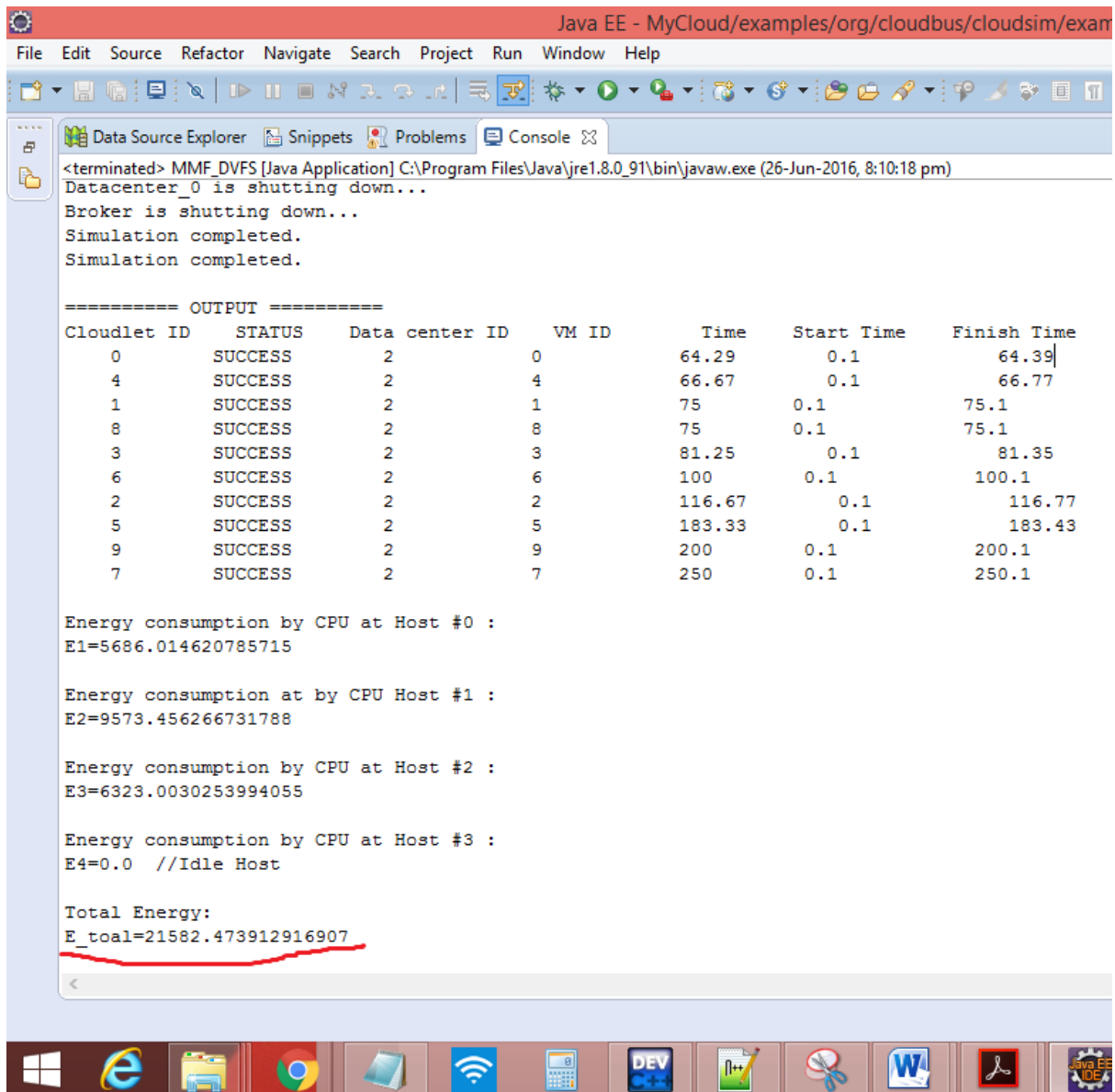


Figure 6.6: Screenshot 4 of 4 of MMF-DVFS

Total energy consumption by MMF-DVFS:

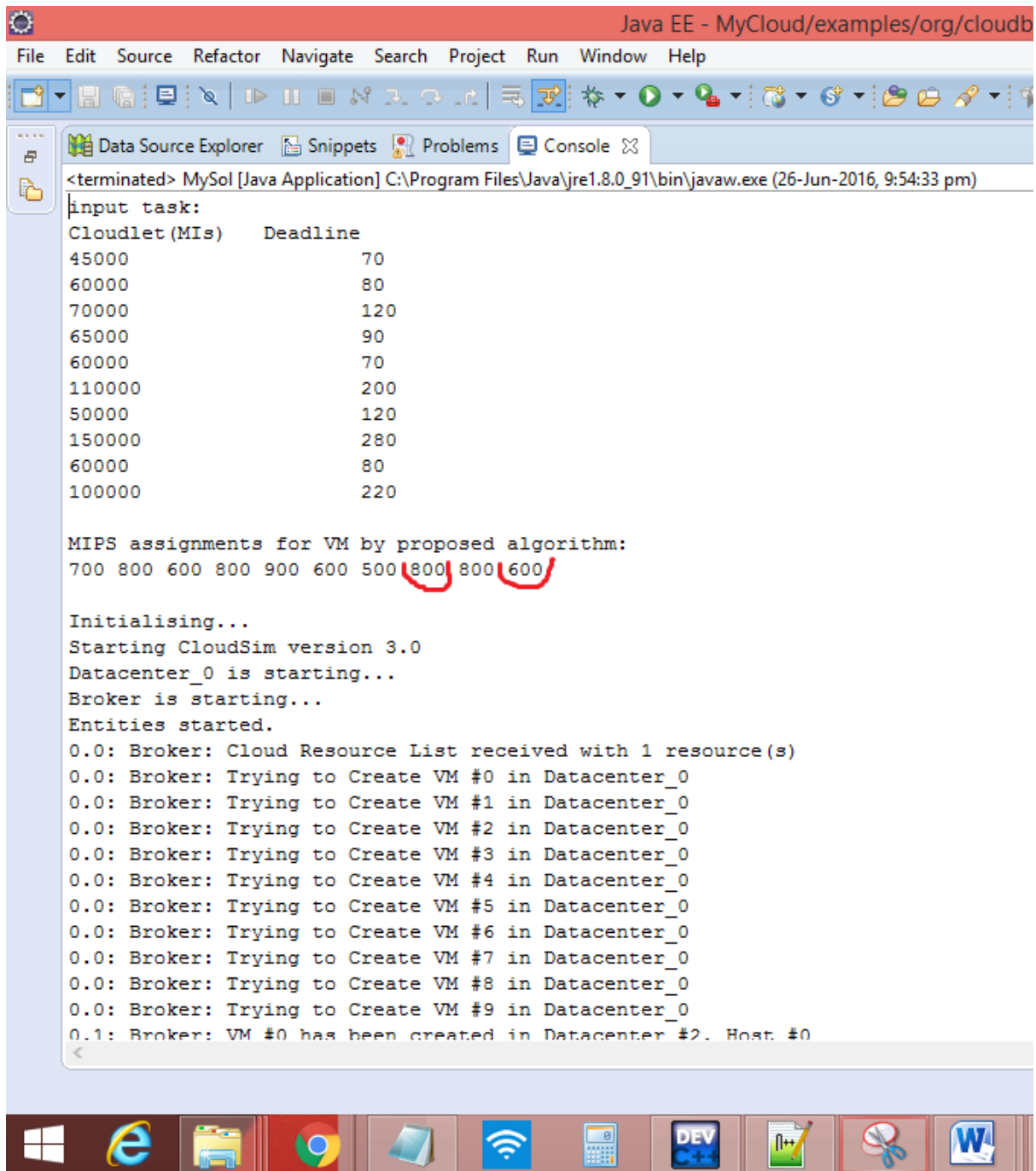
Total Energy = Energy_host0+ Energy_host1+ Energy_host2+ Energy_host3

$$=5686 + 9573 + 6323 + 0$$

$$=21582 \text{ joule} \quad (6.1)$$

6.3.2 Output of Proposed Approach

The output screenshot of the above stated scenario with same input job by the proposed approach is given below (Figure 6.7 – Figure 6.10):



```
<terminated> MySol [Java Application] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (26-Jun-2016, 9:54:33 pm)
input task:
Cloudlet (MIs)   Deadline
45000            70
60000            80
70000            120
65000            90
60000            70
110000           200
50000            120
150000           280
60000            80
100000           220

MIPS assignments for VM by proposed algorithm:
700 800 600 800 900 600 500 800 800 600

Initialising...
Starting CloudSim version 3.0
Datacenter_0 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
0.0: Broker: Trying to Create VM #0 in Datacenter_0
0.0: Broker: Trying to Create VM #1 in Datacenter_0
0.0: Broker: Trying to Create VM #2 in Datacenter_0
0.0: Broker: Trying to Create VM #3 in Datacenter_0
0.0: Broker: Trying to Create VM #4 in Datacenter_0
0.0: Broker: Trying to Create VM #5 in Datacenter_0
0.0: Broker: Trying to Create VM #6 in Datacenter_0
0.0: Broker: Trying to Create VM #7 in Datacenter_0
0.0: Broker: Trying to Create VM #8 in Datacenter_0
0.0: Broker: Trying to Create VM #9 in Datacenter_0
0.1: Broker: VM #0 has been created in Datacenter #2. Host: #0
```

Figure 6.7: Screenshot 1 of 4 of “Proposed Approach”

In Figure 6.7, we can see that MIPS assignment for VM #7 and VM #9 has been changed (Underlined by Red color) from MMF-DVFS approach. Previous MIPS assignment for VM #7 & VM#9 by MMF-DVFS was 600 & 500 respectively. And now with our proposed approach, MIPS assignment for VM#7 & VM#9 is 800 & 600 respectively. *This will execute the task faster and consequently decrease static energy with ensuring deadline of task.*

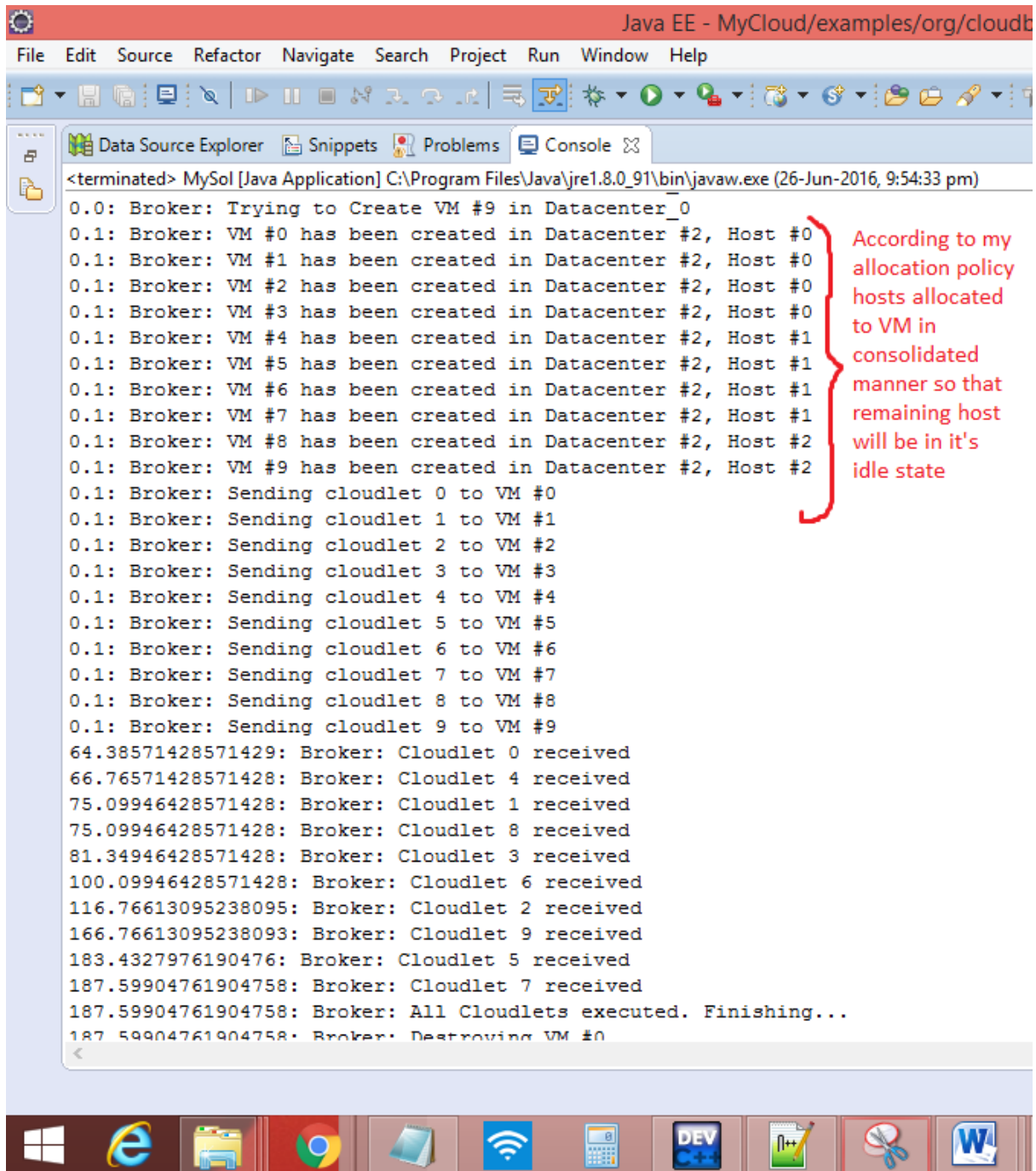


Figure 6.8: Screenshot 2 of 4 of "Proposed Approach"

In the Figure 6.8, we can see the allocation of host to VM. In our approach we ensuring the efficient utilization of resources, for this we allocate host to VM in consolidated manner so that we can maximize the host utilization and keep remaining host in its low power state or idle state. For our example, *we allocate the resources to all requesting VM by utilizing 3 hosts only and host 4 is remaining in its idle state to save the energy.*

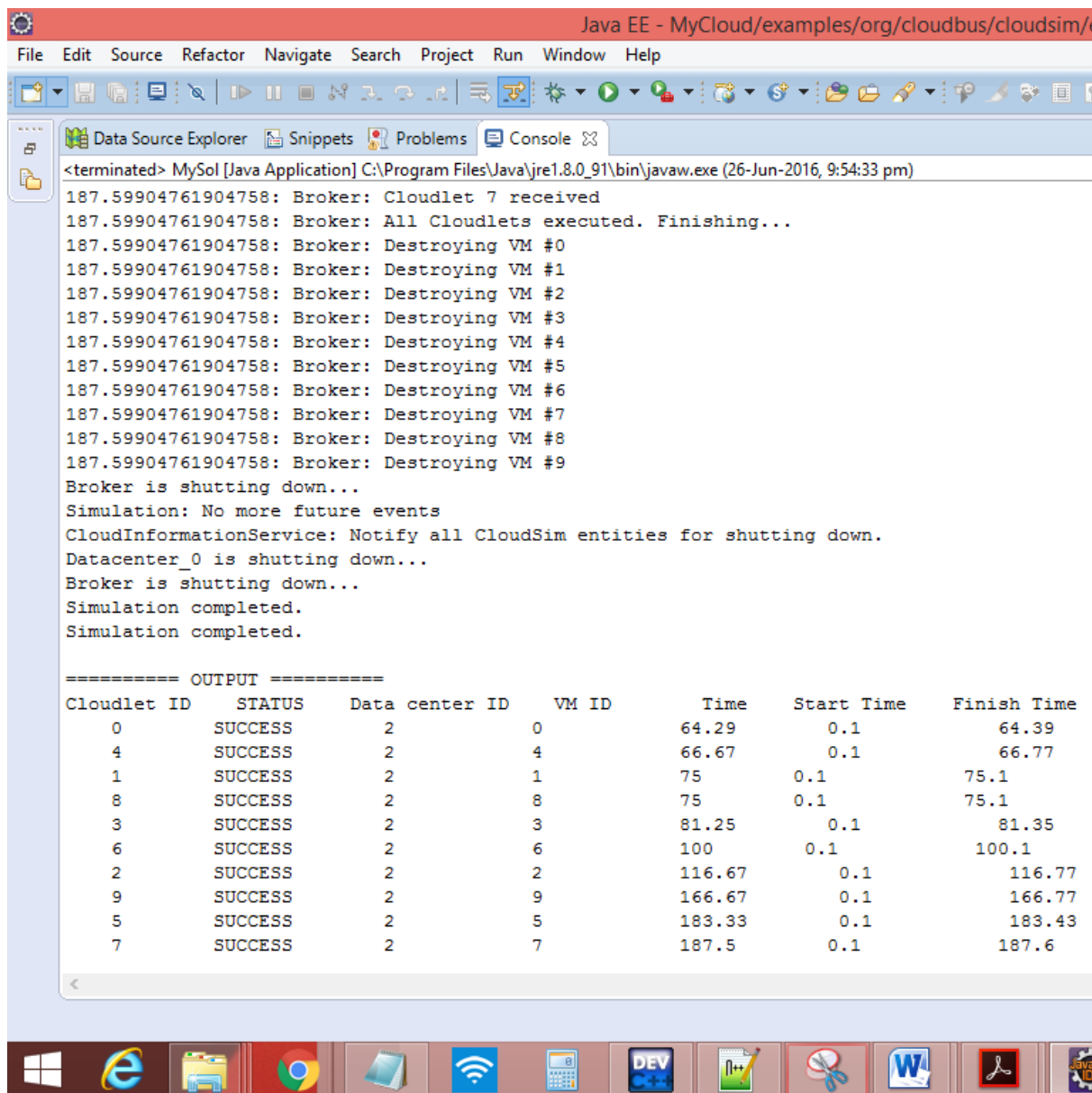


Figure 6.9: Screenshot 3 of 4 of “Proposed Approach”

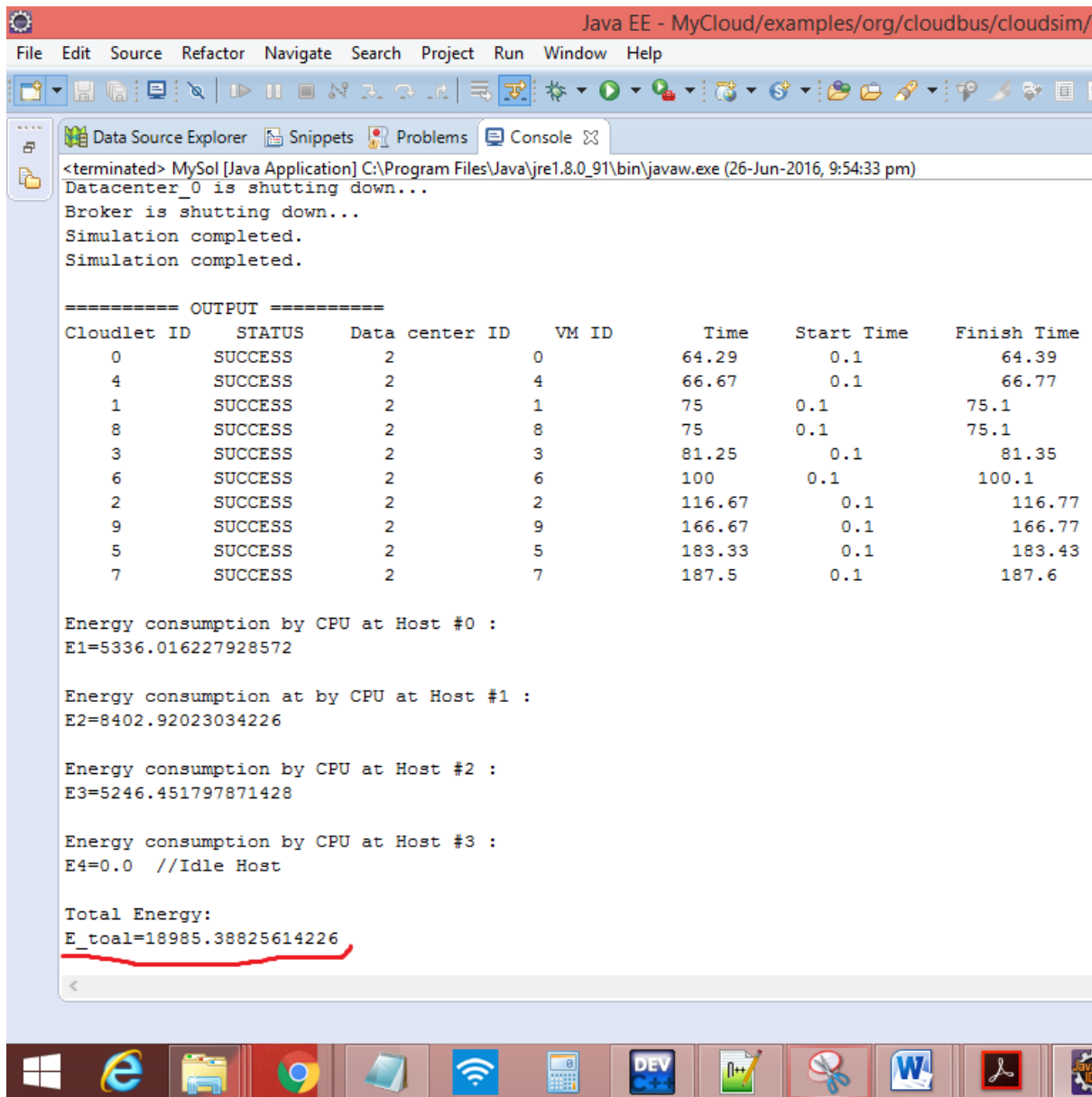


Figure 6.10: Screenshot 4 of 4 of “Proposed Approach”

Total energy consumption by Proposed Approach:

Total Energy = Energy_host0+ Energy_host1+ Energy_host2+ Energy_host3

$$=5336.02 + 8402.92 + 5246.45 + 0$$

$$=18985 \text{ joule} \quad (6.2)$$

From above screenshot we can easily observed and compare the result of both approaches. Since MMF-DVFS [6] doesn't aware about the static power consumption of processor so they stated that to reduce the operating frequency as much as possible with the constraint to complete execution of task before deadline. But with this approach results higher energy consumption in very low frequency domain (when deadline of a task is too large) that results the less dynamic power but at the same time more static power, so overall energy consumption increased.

In the proposed approach we lower the frequency to decrease the dynamic power but as our algorithm is aware about the static power so it can increase the operating frequency of the core when static power becoming higher than the dynamic power. In either case task deadline should be accomplished. To watch the proof of this behavior of our approach we can observe the screenshot both approach. Consider the figure 6.3 and figure 6.7 and go through the red-marked MIPS assignment of both approaches. Our approach increases the MIPS assignment of VM #7 from 600 to 800 and of VM #9 from 500 to 600. This optimization will execute the task in comparatively lesser time that lead to decrease the static power and overall energy consumption is minimum.

Now coming to the final result as we can see in figure 6.6 the final energy consumed in MMF-DVFS is 21582 joule (6.1) and in figure 6.10 in our proposed approach final energy is 18985 joule (6.2) which is lesser than previous. For above stated scenario of job our results save 12% energy as compared to MMF-DVFS.

Above example is taken for the sake of simplicity so that we can understand the experiments and analysis easily. But for proof of our approach we have taken various example with higher number of VMs and hosts and perform the experiments by two existing approach and our proposed approach which results is displayed in graph given in Figure 6.11.

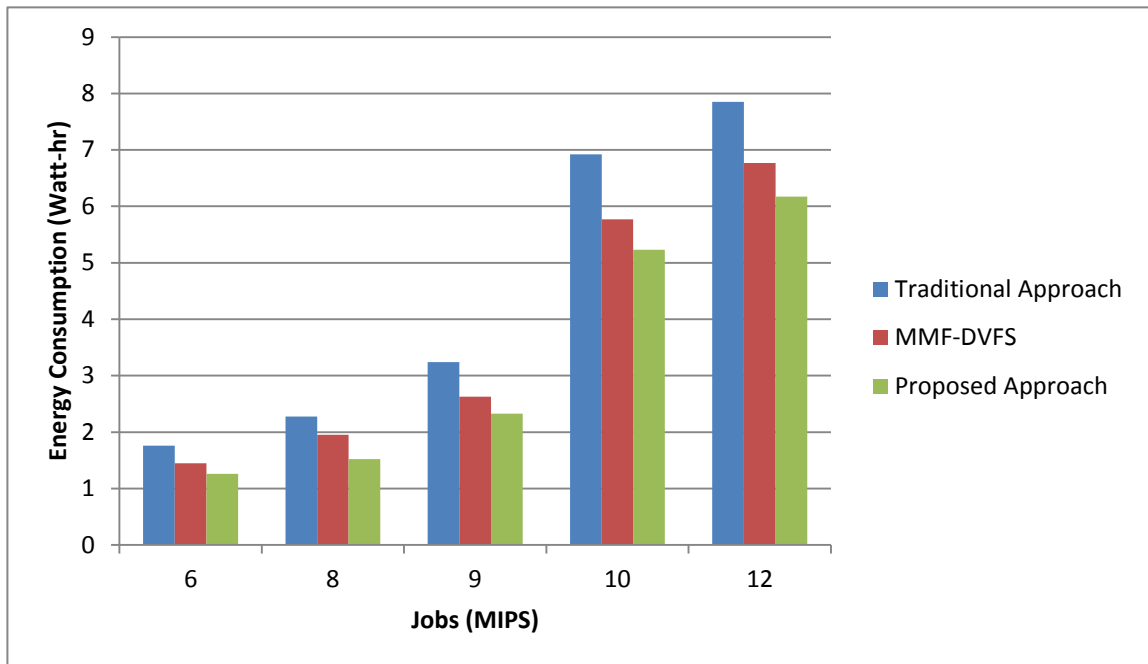


Figure 6.11: Comparison between the Energy-Consumption over various methods

We performed experiments for various jobs by various methods and energy consumption on those can be seen in above graph (Figure 6.11). We can see that energy consumption by proposed approach is less as compared to traditional approach and MMF-DVFS.

7.1 Conclusion

As the size of cloud data centers increases, the need for more energy-efficient use of such infrastructures becomes critical to enable sustainable utilization of such platforms. In this thesis, we targeted the problem of energy-efficient execution of real-time task in clouds. In our method allocation of host for VM is done in an consolidated manner which leads to high utilization of resources and keep the remaining resources in idle or low power state. Our method can satisfy the minimum resource requirement of a job and prevent the excess use of resources. Hence, we can increase the resource utilization. We proposed a energy aware scheduling algorithm that applies DVFS to enable deadlines for execution of real-time tasks to be met with reduced energy expenditure. Whilst existing approaches focus only on dynamic power, they do not address the issue of static power so they can't give minimized total energy. Our approach is able to significantly reduce energy consumption of the cloud while not incurring any impact on the Quality of Service offered to users. In the proposed approach we lower the frequency to decrease the dynamic power but as our algorithm is aware about the static power so it can increase the operating frequency of the core when static power becoming higher than the dynamic power. In either case task deadline should be accomplished.

7.2 Future Work

As future work, we will improve our algorithm to support other types of applications, such as workflows and MapReduce. As in this thesis we only consider the CPU energy but in practice other factor also responsible for energy consumption such as RAM, Hard Disk, Cache, IO operation. So as a future work we can also work on these factor to reduce overall energy of cloud datacentre. This requires consideration on the energy consumption of network and storage equipments during the scheduling process. Another interesting research direction is the investigation of an energy-efficient algorithm in the context of hybrid cloud scenarios.

REFERENCES:

- [1] Hideaki Kimura, Mitsuhsa Sato, Yoshihiko Hotta, Taisuke Boku, Daisuke Takahashi, “Empirical study on Reducing Energy of Parallel Programs using Slack Reclamation by DVFS in a Power-scalable High Performance Cluster”, 2006 IEEE.
- [2] Chia-Ming Wu*, Ruay-Shiung Chang, Hsin-Yu Chan,” A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters”, 2014 Elsevier.
- [3] Hyekseong Kweon, Younggu Do, Jaejeong Lee, Byoungchul Ahn,” Efficient Power-Aware Scheduling Algorithm in Real Time System”, 2007 IEEE.
- [4] Zhuo Tang · Ling Qi · Zhenzhen Cheng · Kenli Li · Samee U. Khan · Keqin Li,” An Energy Efficient Task Scheduling Algorithm in DVFS-enabled Cloud Environment”, 2015 Springer
- [5] Euseong Seo, Jinkyu Jeong, Seonyeong Park, and Joonwon Lee,” Energy Efficient Scheduling of Real-Time Tasks on Multicore Processors”, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 19, NO. 11, NOVEMBER 2008.
- [6] Nikzad Babaii Rizvandi, Javid Taheri,Albert Y. Zomaya, Young Choon Lee, “Linear Combination of DVFS enabled Processor Frequencies to modify Energy-Aware Scheduling Algorithms”,”MMF-DVFS”, 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing.
- [7] Dawei Li and Jie Wu,” Energy-Aware Scheduling for Aperiodic Tasks on Multi-core Processors”, IEEE 2014 43rd International Conference on Parallel Processing.
- [8] Nikzad Babaii Rizvandi, Javid Taheri,Albert Y. Zomaya,” Some Observations on Optimal Frequency Selection in DVFS-based Energy Consumption Minimization”,”MVFS-DVFS”.

- [9] Weixun Wang, Student member IEEE and Prabhat Mishar, Senior member IEEE, "System-Wide Leakage-Aware Energy Minimization Using Dynamic Voltage Scaling and Cache Reconfiguration in Multitasking Systems", *IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS*, VOL. 20, NO. 5, MAY 2012.
- [10] G. von Laszewski, L. Wang, A. J. Younge, and X. He, "Poweraware scheduling of virtual machines in DVFS-enabled clusters," in *Proceedings of the IEEE International Conference on Cluster Computing (CLUSTER)*, 2009.
- [11] Y. Li, Y. Liu, and D. Qian, "An energy-aware heuristic scheduling algorithm for heterogeneous clusters," in *Proceedings of the 15th International Conference on Parallel and Distributed Systems (ICPADS)*, 2009.
- [12] G. L. T. Chetsa, L. Lefevre, J.-M. Pierson, P. Stolf, and G. D. Costa, "A runtime framework for energy efficient HPC systems without a priori knowledge of applications," in *Proceedings of the 18th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, 2012.
- [13] X. Ruan, X. Qin, Z. Zong, K. Bellam, and M. Nijim, "An energy-efficient scheduling algorithm using dynamic voltage scaling for parallel applications on clusters," in *Proceedings of the 16th International Conference on Computer Communications and Networks (ICCCN)*, 2007.
- [14] L. Wang, G. von Laszewski, J. Dayal, and F. Wang, "Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS," in *Proceedings of the 10th International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, 2010.
- [15] Y. Tian, C. Lin, Z. Chen, J. Wan, and X. Peng, "Performance evaluation and dynamic optimization of speed scaling on web servers in cloud computing," *Tsinghua Science and Technology*, vol. 18, no. 3, pp. 298–307, 2013.
- [16] S. Eyerman and L. Eeckhout, "Fine-grained DVFS using on-chip regulators," *ACM Transactions on Architecture and Code Optimization*, vol. 8, no. 1, pp. 1:1–1:24, 2011.

- [17] J. L. March, J. Sahuquillo, S. Petit, H. Hassan, and J. Duato, “Poweraware scheduling with effective task migration for real-time multicore embedded systems,” *Concurrency and Computation: Practice and Experience*, vol. 25, no. 14, pp. 1987–2001, 2013.
- [18] K. H. Kim, W. Y. Lee, J. Kim, and R. Buyya, “SLA-based scheduling of bag-of-tasks applications on power-aware cluster systems,” *IEICE Transactions on Information and Systems*, vol. E93-D, no. 12, pp. 3194–3201, 2010.
- [19] L. M. Zhang, K. Li, D. C.-T. Lo, and Y. Zhang, “Energy-efficient task scheduling algorithms on heterogeneous computers with continuous and discrete speeds,” *Sustainable Computing: Informatics and Systems*, vol. 3, no. 2, pp. 109–118, 2013.
- [20] R. Buyya, A. Beloglazov, and J. Abawajy, “Energy-efficient management of data center resources for Cloud computing: A vision, architectural elements, and open challenges,” in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, 2010.
- [21] L. A. Barroso and U. Holzle, “The case for energy-proportional computing,” *Computer*, vol. 40, no. 12, pp. 33–37, 2007.
- [22] R. Buyya, C. Yeo, and S. Venugopal, “Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities,” in *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC-OB, IEEE CS Press. Los Alamitos, CA. USA)*, 2008.
- [23] P. Mell, and T. Grance, “The NIST Definition of Cloud computing,” National Institute of Standards and Technology, 2009.
- [24] Amazon ec2. <http://aws.amazon.com/ec2/>.
- [25] Amazon s3. <http://aws.amazon.com/s3/>.
- [26] “Growth in data center electricity use 2005 to 2010,” Analytics Press, Tech. Rep., 2011.

- [27] X. Fan, W. D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA), 2007, pp. 13–23.
- [28] L. Minas and B. Ellison, Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers. Intel Press, 2009.
- [29] G. Dhiman, K. Mihic, and T. Rosing, "A system for online power prediction in virtualized environments using gaussian mixture models," in Proceedings of the 47th Annual ACM/IEEE Design Automation Conference (DAC), 2010, pp. 807–812.
- [30] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in Proceedings of the 14th International Symposium on High Performance Computer Architecture (HPCA), 2008.
- [31] R. Buyya, R. Ranjan, and R. N. Calheiros, "Modeling And Simulation Of Scalable Cloud Computing Environments And The Cloudsim Toolkit: Challenges And Opportunities," Proc. Of The 7th High Performance Computing and Simulation Conference (HPCS 09), IEEE Computer Society, June 2009.
- [32] Yujian Zhang, Yun Wang and Cheng Hu, "CloudFreq: Elastic Energy-Efficient Bag-of-Tasks Scheduling in DVFS-enabled cloud", 2015 IEEE 21st International Conference on Parallel and Distributed Systems