

# Power Modeling and Resource Optimization in Virtualized Environments

*A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy*

*by*

Humaira Abdul Salam

*in the Faculty of*

\* Department of Electrical, Electronic and Telecommunications Engineering, and Naval Architecture (DITEN)  
University of Genova (UniGe)

*and in collaboration with*

†Institute of Communication Networks (ComNets)  
Hamburg University of Technology (TUHH)

*on*

November 15, 2019

Supervisors: Prof. Franco Davoli\*  
Prof. Raffaele Bolla\*

Co-supervisor: Prof. Dr.-Ing. Andreas Timm-Giel†

**DITEN**



**TUHH**  
Hamburg University of Technology

**ComNets**  
Institute of Communication Networks



# Declaration

I hereby declare that the work in this thesis was composed and originated by myself and has not been submitted for another degree or diploma at any university or other institute of tertiary education.

I certify that all information sources and literature used are indicated in the text and a list of references is given in the bibliography.

Genoa, Nov 15, 2019



Humaira Abdul Salam



# Acknowledgement

I would like to express my sincere gratitude to my Ph.D. supervisors for their continuous support, patience, motivation and knowledge. Their guidance helped me throughout my research and during the writing of this thesis. The insightful comments, encouragement, and sometime critical questions incited me to widen my research from different perspectives.

Besides my supervisors, a special thanks to my husband for his immense love and support during all the good and bad times. He was always around, at times when I thought that it is impossible to continue, he helped me to keep perspective. I appreciate his contribution, guidance and his belief in me throughout this research tenure.

I am also very grateful to my parents who always remembered me in their prayers. Thanks for showing faith in me and giving me liberty to choose what I desired. Life lessons they taught made me confident enough to accomplish personal goals. Although they hardly understood what I researched on, but without their countless effort and support, I might have not been standing here.

In the end, I would like to express my deepest gratitude to everyone who has been with me during these three years, with any kind of support. My colleagues from both universities, friends, and other family members who have supported me, guided me, helped me and tolerated me all the way long.

Humaira Abdul Salam



# Abstract

The provisioning of on-demand cloud services has revolutionized the IT industry. This emerging paradigm has drastically increased the growth of data centers (DCs) worldwide. Consequently, this rising number of DCs is contributing to a large amount of world total power consumption. This has directed the attention of researchers and service providers to investigate a power-aware solution for the deployment and management of these systems and networks. However, these solutions could be beneficial only if derived from a precisely estimated power consumption at run-time. Accuracy in power estimation is a challenge in virtualized environments due to the lack of certainty of actual resources consumed by virtualized entities and of their impact on applications' performance. The heterogeneous cloud, composed of multi-tenancy architecture, has also raised several management challenges for both service providers and their clients. Task scheduling and resource allocation in such a system are considered as an NP-hard problem. The inappropriate allocation of resources causes the under-utilization of servers, hence reducing throughput and energy efficiency. In this context, the cloud framework needs an effective management solution to maximize the use of available resources and capacity, and also to reduce the impact of their carbon footprint on the environment with reduced power consumption.

This thesis addresses the issues of power measurement and resource utilization in virtualized environments as two primary objectives. At first, a survey on prior work of server power modeling and methods in virtualization architectures is carried out. This helps investigate the key challenges that elude the precision of power estimation when dealing with virtualized entities. A different systematic approach is then presented to improve the prediction accuracy in these networks, considering the resource abstraction at different architectural levels. Resource usage monitoring at the host and guest helps in identifying the difference of performance between the two. Using virtual Performance Monitoring Counters (vPMCs) at a guest level provides detailed information that helps in improving the prediction accuracy and can be further used for resource optimization, consolidation and load balancing. Later, the research also targets the critical issue of optimal resource utilization in cloud computing. This study seeks a generic, robust but simple approach to deal with resource allocation in cloud computing and networking. The inappropriate scheduling in cloud causes under- and over- utilization of resources which in turn increases the power consumption and also degrades the system performance. This work first addresses some of the major challenges related to task scheduling in the heterogeneous systems. After critical analysis of existing approaches, the thesis presents a rather simple scheduling scheme based on the combination of heuristic solutions. Improved resource utilization with reduced processing time can be achieved using the proposed energy-efficient scheduling algorithm.





# Contents

<b>Declaration</b>	<b>iii</b>
<b>Acknowledgement</b>	<b>v</b>
<b>Abstract</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Goal . . . . .	2
1.3 Research Questions . . . . .	3
1.3.1 Power Modeling . . . . .	3
1.3.2 Task Scheduling . . . . .	3
1.4 Thesis Organization . . . . .	3
<b>2 Background and Foundation</b>	<b>5</b>
2.1 Introduction to Virtualization . . . . .	5
2.1.1 Levels of Virtualization . . . . .	5
2.1.2 Virtualization Genres . . . . .	6
2.2 Introduction to Cloud Computing . . . . .	7
2.2.1 Cloud Architecture . . . . .	8
2.2.2 Cloud Deployment Models . . . . .	9
2.2.3 Cloud Services . . . . .	9
2.3 Advantages of Virtualization and Cloud Computing . . . . .	10
<b>3 A Survey: Power Measurement and Power Consumption Models in Virtualized Networking and Computing Environments</b>	<b>13</b>
3.1 Introduction . . . . .	13
3.2 Research Method Overview: Analysis with Structured Coding . . . . .	14
3.3 Analysis through Themes . . . . .	16
3.3.1 State of the art . . . . .	16
3.3.2 Research Domains . . . . .	23
3.3.3 Pitfalls . . . . .	26
3.3.4 Fallacies . . . . .	29
3.4 A Digest of Challenges, Approaches and Developments . . . . .	30
3.4.1 Graphic Device . . . . .	30
3.4.2 Statistics . . . . .	31
3.5 Applications . . . . .	34
3.5.1 Management and Orchestration . . . . .	35
3.5.2 Real-time Control . . . . .	37
3.5.3 Billing . . . . .	38
3.5.4 Macroscopic Analysis . . . . .	40


































3.6	The Engineer’s Kit Bag . . . . .	42
3.6.1	Observations on tools . . . . .	42
3.6.2	A brief methodology: observations on formal methods . . . . .	43
3.7	Conclusion . . . . .	45
<b>4</b>	<b>Improved Power Modeling in Virtualized Environments</b>	<b>47</b>
4.1	Introduction . . . . .	47
4.2	Motivation . . . . .	47
4.2.1	Research Questions . . . . .	47
4.3	In-context . . . . .	48
4.3.1	Performance Monitoring Counters (PMCs) . . . . .	48
4.3.2	PERF - a Linux tool for profiling . . . . .	48
4.3.3	Stress Generator . . . . .	50
4.3.4	Power Distribution Unit (PDU) . . . . .	51
4.4	Proposed Modeling Approach . . . . .	51
4.5	Training Dataset . . . . .	53
4.6	Training Models . . . . .	55
4.6.1	Linear Least Squares Regression (LLSR) . . . . .	55
4.6.2	Linear Support Vector Regression (LSVR) . . . . .	56
4.6.3	Quadratic Support Vector Regression (QSVR) . . . . .	56
4.6.4	Gaussian Process Regression (GPR) . . . . .	57
4.7	Performance Measures . . . . .	57
4.8	Power Modeling . . . . .	58
4.8.1	First Phase: Methodology Selection . . . . .	58
4.8.2	Second Phase: Further Analysis . . . . .	64
4.9	Conclusion . . . . .	71
4.9.1	Findings . . . . .	72
4.9.2	Contributions . . . . .	73
4.10	Future Directions . . . . .	73
<b>5</b>	<b>Overview of Resource Management and Task Scheduling in the Cloud Computing System</b>	<b>75</b>
5.1	Introduction . . . . .	75
5.2	Notation and Terminology . . . . .	75
5.3	Categories of Scheduling Algorithms . . . . .	77
5.4	Related Work . . . . .	78
5.5	Open Issues in Cloud Computing . . . . .	81
5.5.1	Power Consumption . . . . .	82
5.5.2	Network Uncertainty and Heterogeneity . . . . .	82
5.5.3	Security . . . . .	83
5.5.4	Availability and Reliability . . . . .	83
<b>6</b>	<b>Optimal Task Scheduling in Cloud Computing</b>	<b>85</b>
6.1	Introduction . . . . .	85
6.2	Motivation . . . . .	85
6.3	Problem Definition in Scheduling . . . . .	86
6.4	System Overview . . . . .	87
6.5	Simulation Environment . . . . .	89

6.6	Case Study I . . . . .	90
6.6.1	Pseudo Best Fit Scheduling (PBFS) Algorithm . . . . .	90
6.6.2	Results and Discussion . . . . .	92
6.7	Case Study II . . . . .	94
6.7.1	Longest Task First Scheduling (LTFS) Algorithm . . . . .	94
6.7.2	Results and Discussion . . . . .	96
6.8	Case Study III . . . . .	99
6.8.1	Greedy Best Fit Scheduling (GBFS) Algorithm . . . . .	100
6.8.2	Results and Discussion . . . . .	101
6.9	Conclusion . . . . .	105
<b>7</b>	<b>Conclusion</b>	<b>111</b>
7.1	Thesis Contribution . . . . .	111
7.2	Future Work . . . . .	112
	<b>Appendices</b>	<b>115</b>
<b>A</b>	<b>Graph of Problems, Approaches and Developments (PAD)</b>	<b>115</b>
<b>B</b>	<b>Graph Nodes Description</b>	<b>119</b>
<b>C</b>	<b>Task Scheduling</b>	<b>127</b>
C.1	Scheduling Algorithms . . . . .	127
C.1.1	Heuristic Algorithms . . . . .	127
C.1.2	Meta-Heuristic Algorithms . . . . .	129
C.2	Different Machine Environment in Task Scheduling . . . . .	131
	<b>Bibliography</b>	<b>133</b>



# List of Figures

1.1	Market share of cloud infrastructure services (IaaS, PaaS, Hosted Private Cloud), as reported by Synergy Research Group [4]. . . . .	2
2.1	Type 1 and Type 2 hypervisor structure . . . . .	7
2.2	Graphical view of cloud computing architecture . . . . .	8
2.3	Structure of cloud services: IaaS, PaaS, SaaS . . . . .	10
3.1	Frequency of development category . . . . .	32
3.2	Interest in problem categories and their perceived complexity . . . . .	33
3.3	Normalized frequency of occurrence of approaches and their utility metric . . . . .	33
4.1	Map of sources of PERF events [120]. . . . .	49
4.2	A sample output of 'PERF List' command . . . . .	50
4.3	A sample output of 'PERF stat' command . . . . .	51
4.4	Black-box modeling . . . . .	52
4.5	Sum of counter values for all running (08) VMs at different load. Legends: + Instruction Counts, - CPU cycle, * CPU clock, - Stalled In- structions, ▲ Page Faults, - Context Switches, - CPU Migration, * Branches . . . . .	54
4.6	Illustration of the two strategies to model power change from the change in virtual machine MIPS. . . . .	60
4.7	Left: Increasing the vCPU utilization via a stress program leads to an increase in virtual machine MIPS. Right: The same increased vCPU utilization also increases the power consumption at the server running the VM. Legend: Only 1 VM with load (—), 2 VMs in parallel (—) . . .	61
4.8	Model 1 vs. Model 2 and individual components of Model 2. Legend: "x" shows measured data used for training, while the predictions from LLSR and LSVR are shown by "o" and "+", respectively. . . . .	62
4.9	Box-plot of error for Model 1 and Model 2. Legend: Median error (—), 1 <sup>st</sup> – 3 <sup>rd</sup> quartile of errors (—), overall error range (---) . . . . .	64
4.10	Change in VM_MIPS w.r.t total server's CPU utilization . . . . .	67
4.11	Server power consumption vs CPU utilization for different numbers of running VMs. Legend: *8VM, +6VM, *4VM, -3VM, ▲2VM, -1VM . . . . .	67
4.12	Server power consumption vs CPU core. Legend: *4VM_8vCPU, -4VM_4vCPU . . . . .	68
4.13	Response plot of Models. Legend: -LLSR, -LSVR, -QSVR, -GPR and + measured power value . . . . .	69
4.14	Predicted power for test dataset. Legend: -LLSR, -LSVR, -QSVR, -GPR and + measured power value . . . . .	71
5.1	Worldwide power consumption of data centers in GW [167]. . . . .	82

6.1	Case1, total cost of under- and over-utilized cores for Simulated Annealing (SA), Pseudo Best Fit Scheduling (PBFS) and Round Robin (RR) scheduling schemes . . . . .	93
6.2	Case1, the cost for over- and under-utilization of the CPU cores for Simulated Annealing (SA), Pseudo Best Fit Scheduling (PBFS) and Round Robin (RR) scheduling schemes . . . . .	94
6.3	Case1, percentage of CPU utilization rate ( $\phi/s$ ) for Simulated Annealing (SA), Pseudo Best Fit Scheduling (PBFS) and Round Robin (RR) scheduling schemes . . . . .	95
6.4	Case1, total mapping time to map all cloudlets to VMs for Simulated Annealing (SA), Pseudo Best Fit Scheduling (PBFS) and Round Robin (RR) scheduling schemes . . . . .	96
6.5	Case1, total energy consumption for Simulated Annealing (SA), Pseudo Best Fit Scheduling (PBFS) and Round Robin (RR) scheduling schemes . . . . .	97
6.6	Case2, maximum completion time of cloudlets. Legends:  SA,  RR, and  LTFS . . . . .	98
6.7	Case2, total mapping time to map all cloudlets to VMs. Legends:  SA,  RR, and  LTFS . . . . .	99
6.8	Case2, sum of processing time of all 500 cloudlets. Legends:  SA,  RR, and  LTFS . . . . .	100
6.9	Case2, processing time of individual cloudlet (for total of 500 cloudlets) for $\lambda = 16$ . Legend:  SA,  RR, and  LTFS . . . . .	101
6.10	Case2, total energy consumption. Legends:  SA,  RR, and  LTFS . . . . .	102
6.11	Pseudo code for GBFS Algorithm . . . . .	103
6.12	Flow chart of the Greedy Best Fit Scheduling (GBFS) Algorithm . . . . .	107
6.13	Case3, maximum completion of cloudlets. Legends:  SA,  RR, and  GBFS . . . . .	108
6.14	Case3, percentage of CPU utilization rate ( $\phi/s$ ). Legends:  SA,  RR, and  GBFS . . . . .	108
6.15	Case3, sum of processing time of all cloudlets. Legends:  SA,  RR, and  GBFS . . . . .	109
6.16	Case3, sum of lateness of all cloudlets. Legends:  SA,  RR, and  GBFS . . . . .	109
6.17	Case3, total mapping time to map all cloudlets to VMs. Legends:  SA,  RR, and  GBFS . . . . .	110
6.18	Case3, total energy consumed. Legends:  SA,  RR, and  GBFS . . . . .	110
A.1	Problem-Approach-Development (PAD) Graph-part 1 . . . . .	116
A.2	Problem-Approach-Development (PAD) Graph-part 2 . . . . .	117
A.3	Problem-Approach-Development (PAD) Graph-part 3 . . . . .	118

# 1 Introduction

This chapter provides the introduction to this thesis and outlines the motivation and research questions on which this research is based on. The need and growth of data centers are initially discussed in the first section which also highlights the need for further research in this area. In the next section, problems addressed in this thesis are presented, which is followed by the research questions. In the end, the organization of thesis chapters is provided.

## 1.1 Motivation

Virtualization is the key technology behind today's growing data centers and cloud services. The concept of virtualization is not new, although, its use in cloud computing with the introduction of containers has boosted up in the last decade [1]. Cloud computing has opened new ways to access multiple on-demand resources from a remote pool of servers. This technology has brought a new paradigm shift in IT industry which is becoming popular day by day. Cloud services provide a number of facilities to their users with easy access of resources, reduced management and deployment overheads, portability, flexibility, low execution cost, etc. [2]. Enterprises of different scale either small or big, all are now adopting the cloud computing technology to avoid the huge cost of deployment and management of a whole IT infrastructure. These enterprises are running and also developing their applications on these cloud platforms due their reliability and globally available infrastructure. According to Forbes [3], the global market of cloud services is maturing and providing certain support (in terms of speed, security and scaling) for new digital businesses. This emerging demand has increased the number of data centers and cloud significantly worldwide. The worldwide public cloud service market is projected to grow by 36% in 2020 to a total of \$240.3B from \$175.8B in 2018 [1]. Fig. 1.1 shows the revenue of cloud services for different vendors, during the fourth quarter of the year 2017 and 2018.

This global era of new digital business provides flexible ways to access computing resources, but at the same time burdening the service providers with huge power cost. Operational and management cost in these data centers is increasing by billions of dollars each year [5], [6]. Recent studies [7]–[9] estimated that the information and communication technology (ICT) industry might consume 20% of the global electricity, and will contribute up to 5.5% of world's carbon emissions by 2025. In this scenario, data centers on their own might account for more than 3.2% of greenhouse gas (GHG) emissions (corresponding to 1.9 Gtons per year). Therefore, the emerging virtualized environment needs an energy-efficient cloud and virtual networking infrastructure to meet the modern world's requirements. A report on the energy usage of US Datacenters shows that measures can be taken to reduce the power consumption in future data centers [10]. Several energy-aware frameworks and algorithms have already been proposed and implemented which reduce and limit the power usage of servers. However, in this context, it is also important for the predicted or observed power to be precise enough to make

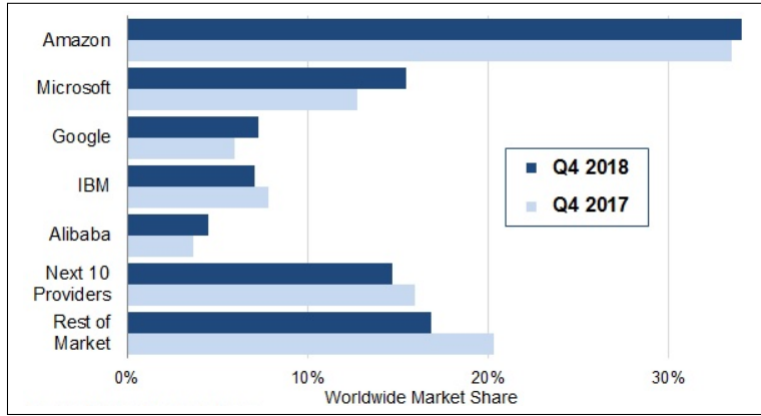


Fig. 1.1: Market share of cloud infrastructure services (IaaS, PaaS, Hosted Private Cloud), as reported by Synergy Research Group [4].

the decision effective. But obtaining high accuracy in estimating power in a virtual environment is challenging. Several power models available can predict the server power with the desired accuracy, but the behavior of server power in a virtual environment does not remain similar [11], [12]. Hence it is difficult to estimate the power consumption profile in a virtualized environment. Since both data centers and the cloud experience variable load and system size, this makes power modeling further challenging.

The heterogeneity also causes difficulty in managing and scheduling resources in the network. Hence, another challenge possessed by huge data centers and the cloud is the management and allocation of resources optimally. The inappropriate allocation of resources and scheduling of tasks in a large-scale distributed system can waste the server resources. According to a study, most of the time, servers in data centers are only 30-50% utilized [13]. Hence, data centers are mostly under-utilized and require efficient management mechanisms to improve their throughput while maintaining their performance. This problem grows exponentially with increasing size of the network hence is classified as an NP-hard problem in several studies [14], [15].

## 1.2 Goal

In this thesis, the research is focused on two major problems of cloud computing. The first objective of this research is to investigate the cause of error and inaccuracy in available power models, as well as, to propose a strategy that can observe the actual resource consumption at hosts and virtual machines. This two-level monitoring is introduced to have a piece of detailed information about system behavior. Analyzing which could help in developing a power model with improved prediction accuracy. This improved power model can be used in any power-aware scheduling or management schemes to make them more effective.

The second part of this thesis is based on resource management in cloud computing. In this part, the complexity and performance of available management models are first analyzed. Later, an effective, less complex scheduling algorithm is proposed that could improve resource utilization in the cloud with relatively short processing time. The proposed scheme will be focusing on maximizing the resource utilization of servers such that the network throughput is increased with minimum energy consumed.



## 1.3 Research Questions

Considering the problems mentioned above, the scope of this thesis focuses on the following identified research questions. These research questions are also categorized into two subparts, each belonging to one of the two research problem addressed in this doctoral work.

### 1.3.1 Power Modeling

- Q1. What are important parameters or features for developing a power model? What are the criteria of selecting the right parameters?
- Q2. Will change in instruction processing speed and resource utilization of virtual machines and the server at similar time instants be the same? If not, then is there a certain relation between the two (host and VM) levels?
- Q3. How the resource monitoring in a virtual environment could be done to obtain actual consumed resources?
- Q4. Does the power of the server remain linear for variable load and different server configuration? How does the number of parallel running virtual machines and their respective load affect the server power and performance?

### 1.3.2 Task Scheduling

- Q1. How can the scheduling of tasks in cloud computing improve the utilization of resources?
- Q2. Does equally balanced load on servers consume less power or does server consolidation provide energy-efficient results?
- Q3. Is it possible to obtain real-time optimal scheduling solutions using heuristic search algorithms?
- Q4. What could be the optimal solution with acceptable latency and network performance for task scheduling in cloud computing?

## 1.4 Thesis Organization

As the research in this thesis is comprised of two major parts, the study of 'power estimation' and its analysis is presented in the first chapters. In later chapters, literature and experiments regarding 'task scheduling' are presented. The organization of the remaining chapters of the thesis is provided below.

*In Chapter 2*, a brief introduction to the concept of virtualization and cloud computing is provided, to make these technologies and their architectures familiar to the reader.

*In Chapter 3*, a detailed survey on servers' power models and measurement methods is presented. An important step to carry out any research is to first identify the problem in that domain; thus, this chapter provides a detailed research survey on available models and measurement methods in a virtual environment.

*In Chapter 4*, the proposed approach for modeling server power is presented. Experiments were carried out to extract the effective parameters to be included in power

## 1 Introduction

modeling. The prediction accuracy of the developed power model is analyzed in detail to validate the performance of the proposed method.

*In Chapter 5*, an overview and background of cloud computing and its management schemes is presented. It further provides the introduction to common terminologies, scheduling literature and open research problems in cloud computing.

*In Chapter 6*, the solution to the scheduling problem is proposed by introducing a new scheduling scheme. Initially, results and analysis of different case studies are presented; later, a combined modeling approach is proposed to improve resource utilization and network energy efficiency.

*In Chapter 7*, the conclusion of the thesis is presented by summarizing the thesis contribution and future work.

# 2 Background and Foundation

## 2.1 Introduction to Virtualization

The virtualization technology is the set of hardware and software tools enabling the abstraction of physical resources into several logical units. These units can be used and operated separately using a different operating system and running different applications. Virtualization is not a new technology, it has rather been in operation for decades. It was introduced in the 1970s by IBM, which allowed the partition of different applications running on similar hardware [16]. However, the introduction of the x86 microprocessor made the processing fast and cheap enough that the need of virtualization was reduced. But digitalization these days has again intensified its need for modern technology. In recent years, modern processor architectures introduced the support of virtualization in their designs. With this, virtualization has become available to everyone everywhere. Continuous evolution and improvements in virtualization technology make it favorable for various domains and at different scales.

In computing, virtualization is a means of creating virtual instances of a device or resources such as a server, network devices, operating system, storage devices, etc. Technically, it enables the sharing of physical resources by providing services to its users worldwide. This spawned a remarkable growth of virtualization in IT infrastructure and is the foundation of the increasingly popular service delivery model known as 'Cloud Computing'. It helps in creating several useful services using dedicated hardware resources of a single hardware unit. Thus, a single physical machine in a virtual environment can act like multiple machines, where each machine is isolated from others. Moreover, it also provides the essentials of consolidation and maximum resource utilization in the cloud and data centers.

To abstract the hardware resources from an operating system, an intermediate layer known as hypervisor is used [16]. This layer lies in between the hardware and Operating System (OS) and is a medium of communication between hardware resources and their virtual counterparts. The hardware where these hypervisors are running is called *host*, whereas the virtual machine where the guest operating system is running is called *guest*. The virtualization layer hides the complexity of underlying hardware architecture and infrastructure providing feasibility and portability to its users.

### 2.1.1 Levels of Virtualization

Different levels of virtualization exist based on the type of services and resources to be virtualized. Starting from the virtualization of instructions set architecture (ISA), it goes to the virtualization of hardware, application level interface (API), process level, operating system (OS), etc. [17]. This thesis, however, considers only the hardware-level virtualization, which enables the creation of virtual machines in a data center or cloud environment. The virtualization of hardware resources can be further categorized depending on the resource used. Three major levels of hardware virtualization are

network virtualization, storage virtualization, and server virtualization [18]. As the name indicates, network virtualization is the creation of a virtual instance for network communication. It shares the available network bandwidth of the host server with the isolated virtual networks. In storage virtualization, virtualization gives access to disk resources on a physical machine to the virtual entity. However, the VM is unaware of the data location and its mapping on the storage device. Another level of virtualization that is server virtualization allows multiple virtual servers (or VMs) to reside on a single physical hardware. This technique is the basis of virtualization in data centers and in the cloud computing environment.

### 2.1.2 Virtualization Genres

There are two different virtualization technologies available to create a virtual image of any instance. One of these technologies is container-based, which is also known as system-level virtualization or containerization. In containerization, users are allowed to create "encapsulated" virtual entities. This encapsulated OS is isolated from the host OS and is called a container. The major advantage of using containers is that they are lightweight and therefore have very efficient system performance. But there is a limitation to this technology, that is, the OS of the host and the container have to be the same.

Another technology is based on the hypervisor where the virtual instance is created using the layer of hypervisor between hardware and virtual level. The hypervisor is a small software layer that enables multiple operating systems, to run alongside each other and share the same physical computing resources. These hypervisors assign the slice of computing resources associated with each VM and isolate them logically. The hypervisor in any virtual environment is responsible for the whole life cycle of a virtual instance. For any virtual machine all steps including its configuration, creation, allocation, termination and even migration are managed by the hypervisor. These hypervisors can be further classified into two types: **native (or bare metal)** and **hosted hypervisors**, which are also known as **Type 1** and **Type 2** hypervisors, respectively [17]. Fig. 2.1 shows the structure of Type 1 and Type 2 hypervisors.

Type 1 hypervisors sit directly on the bare-metal hardware. These native host running hypervisors control the hardware resources and manage the guest operating system by interacting with them directly. With the Type 1 hypervisor, there is no need to load an operating system on the host as this hypervisor itself is an operating system. Type 1 hypervisors are very efficient because of having direct access to physical hardware. Xen, Oracle VM, VMware ESXi are some examples of Type 1 hypervisors. The methodology of how a bare-metal hypervisor allocates available resources, and how it handles driver usage, also depends on whether the hypervisor is a Micro-kernelized or Monolithic Hypervisor [19]. MS Hyper-V Server and Xen are micro-kernelized hypervisors that leverage para-virtualization together with full-virtualization, while VMware ESXi is a monolithic hypervisor which leverages full-virtualization.

Hypervisors that run on top of the OS, similar to other applications are known as Type 2 hypervisors. In this case, the OS itself can abstract its hardware, and can effectively manage the guest OS and VM as an OS process or application. Kernel-based virtual machine (KVM) is one of the most commonly used Type 2 hypervisor, and is often used in conjunction with the QEMU emulator. KVM became part of the Linux kernel machine in 2007 and it supports full-virtualization. Type 2 hypervisors enable

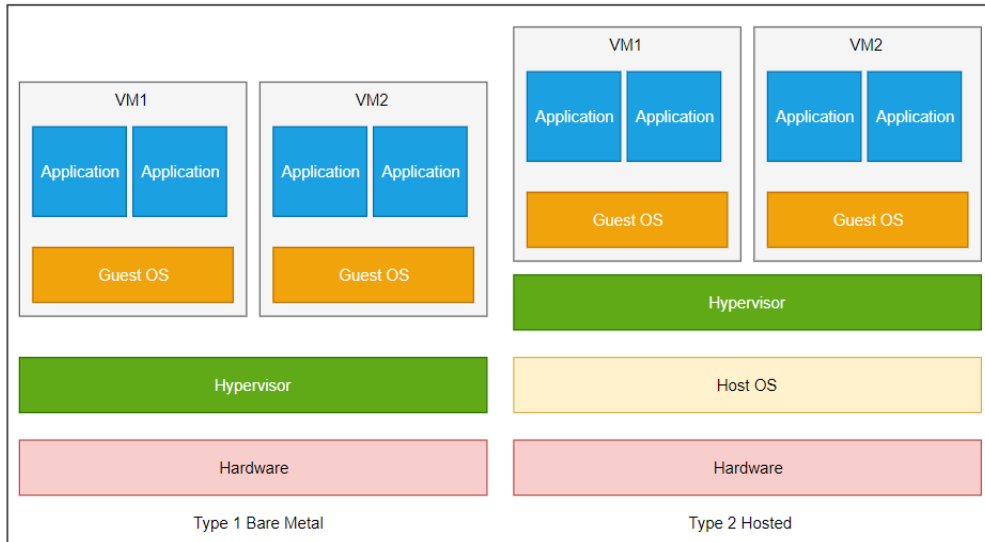


Fig. 2.1: Type 1 and Type 2 hypervisor structure

quick access between parallel running VMs but introduce security risks and latency issues when they need to communicate through the hardware. That is why, this type of hypervisor is usually preferred by individual PC users and professionals but rarely used for servers. Some other examples of Type 2 hypervisors are VirtualBox, VMware Player, and VMware Workstation.

## 2.2 Introduction to Cloud Computing

Building upon the concept of virtualization and virtualized infrastructure, cloud computing has become a widely used source of IT service delivery. It is a dominant heterogeneous and distributed system that offers on-demand resource capacity for different customer requirements.

Cloud computing is the migration of computing capabilities and storage from enterprises and small network to the cloud. The user defines the resource requirement and the cloud provider virtually assembles different requested resources and allocates them to the particular user. The two major factors for clients to rely on cloud services are cost and scalability. Cloud computing is based on a growing infrastructure, providing different service paradigms such as software as a service (SaaS), infrastructure as a service (IaaS), platform as a service (PaaS), data-Storage-as-a-Service (dSaaS), and development-as-a-service (DaaS) [20], with a rapid trend toward "Everything as a Service" (XaaS). The cloud computing environment, now encompassing up to the edge to include Mobile Edge Computing (MEC) [21], is essentially based on a pool of computing, storage and networking resources that need to be dynamically allocated upon user request. Besides a multitude of user application components, a number of enterprises rely on cloud services for their growing IT infrastructure. It is easier for cloud customers to use cloud services rather than deploying and managing their own infrastructure. A recent report on the growth of cloud computing services shows that there was a significant increase of up to 15% of cloud services in Europe in the year 2018 in comparison to 2014 [22].

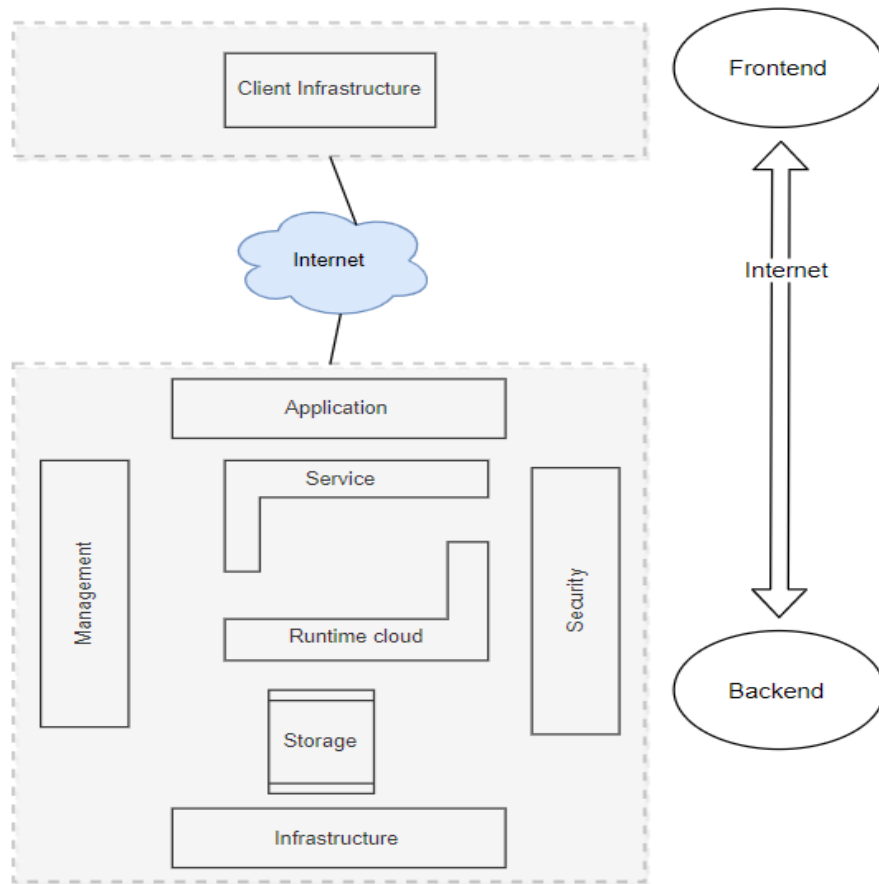


Fig. 2.2: Graphical view of cloud computing architecture

### 2.2.1 Cloud Architecture

The cloud computing architecture is composed of different components and sub-components that are coupled together. There are mainly two major parts of cloud architecture: front-end platform and back-end platform [23], as shown in fig. 2.2. Both these platforms are connected to each other via a network, usually the Internet. The cloud **front-end platform** is the end that is visible to the customer or user. It includes the user interface and the client's computer system, and the network which is being used to access the cloud. Not all clients have the same user interface. The user interface at the front-end could be different as the user choice of accessing web services could vary from Firefox, Internet Explorer, Google Chrome, to others. Clients at this level could be a mobile device or tablet, or a server or group of servers.

On the other hand, the **back-end platform** comprises of components and infrastructure at the service provider end. This is a relatively bigger and much important platform as all functionality of cloud computing and cloud services is delivered from this end. Servers, data storage systems, virtual machines, security mechanisms, and deployment models are the major components at the back-end. A central controller to monitor the network traffic and client requests is also a part of the cloud architecture, which follows a set of protocols rules, and uses a special kind of software called middleware. This is responsible for the control and management of services and ensures everything runs

smoothly.

### 2.2.2 Cloud Deployment Models

Apart from publicly available clouds and cloud services, many organizations have their private cloud infrastructures. Considering the need and use of cloud computing, several models for deployment of the cloud have been developed. Some distinct cloud deployment models are public, private, hybrid, and community cloud [23].

In the public cloud, the cloud infrastructure serves the requests of the general public and shares the resources across multiple tenants. It is a centralized cloud whereby many users concurrently can access different services from across the globe. Some of the services on the public cloud are free but there are also some services that are available on-demand pricing, allowing customers to use them according to the pay-per-usage model. The Amazon Elastic Compute Cloud (EC2) is a primary example of a public cloud. Some organizations, however, build their cloud infrastructure exclusively to be used by their employees, multiple business units, or a distinct group. Services and resources of private clouds are dedicated to a group of users provisioned by the cloud owner. These clouds are usually managed via internal resources; however, sometimes they are also out-sourced to third-parties for management. One of the reasons for using the private cloud is data security and intrusion risks for many enterprises. For many users with sensitive data or large-scale of their own users, private clouds are more efficient, cost-effective and provide higher Quality of Service (QoS) than public clouds. The virtual private cloud service, offered by many public cloud providers, is a special kind of cloud deployment. These private clouds with strong network security and isolation among tenants, provision resources only to the dedicated group of users and lie under the category of hybrid clouds. Hybrid clouds combine the computational power and performance of public clouds, along with flexibility and security gains of private clouds to satisfy their customers.

### 2.2.3 Cloud Services

Virtualization is considered the asset of optimization in IoT. The growing popularity is due to its different levels of services for different users. According to the National Institute of Science and Technology (NIST), the service model of cloud computing is divided into three main types [23]. These three services are Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). The structure of these services is shown in fig. 2.3.

***Infrastructure-as-a-Service (IaaS)*** is popular among enterprises for providing the solution of resource management and cost management for their large network. This model leases the infrastructure as a service, which includes guaranteed processing power, storage, and network resources. The cloud consumers have control over the computing environment such as running OS and application on a server, as well as virtual network and storage, but they do not have any control over the underlying virtualized infrastructure. They are not allowed to choose physical servers on which they could provision their VMs. OpenStack is one open-source cloud management software that provides Infrastructure level services.

***Platform-as-a-Service (PaaS)*** provides the facility to its users to develop and run web services on the cloud. These services are focused on the particular application as they provide a specific OS to their users which unlike IaaS users, cannot choose their

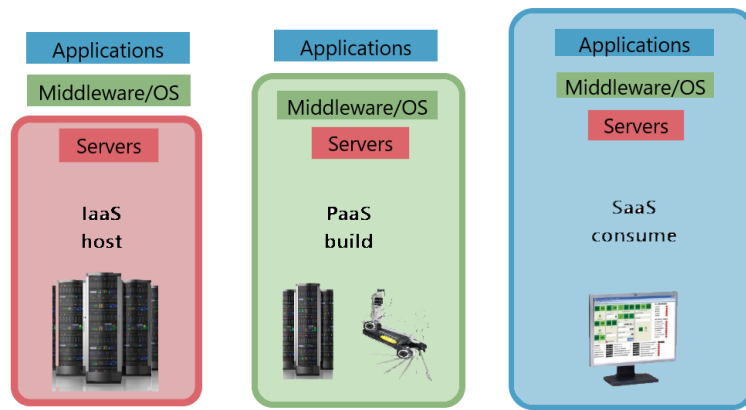


Fig. 2.3: Structure of cloud services: IaaS, PaaS, SaaS

own OS. PaaS is technically a version of IaaS with a custom software stack for a given application with access to the necessary services it requires. It allows cloud consumers to deploy applications using programming environments, libraries, services and other tools provided by service providers. Examples of PaaS include but are not limited to Google App Engine, Heroku, and IBM BlueMix.

The simplest of these services is **Software-as-a-Service (SaaS)** that provides the application to its users. The application is deployed from the centralized system and later provided as a software (or application) to run on a local computer. It allows users to lease the application and pay according to the model "pay-as-you-go". In SaaS, cloud consumers have no control over the infrastructure nor over the application environment. Google Gmail and Salesforce CRM are two typical examples of SaaS.

### 2.3 Advantages of Virtualization and Cloud Computing

The emerging growth of virtualization in different domains is due to its flexibility and feasibility in the management and deployment of infrastructure [2], and these properties are the basic requirements for any modern IT infrastructure. Cloud providers give their clients a competitive advantage by providing the most innovative technology available. In a virtualized environment, decoupling users from physical hardware infrastructure while giving access to the cloud makes it flexible for the client to access and use the services with enhanced portability, anywhere anytime. One of the benefits of isolation either in physical or virtual environments is security. Consider a virtual machine running on a host, along with many others, is being attacked. This affected VM will not impose security risks to other parallel running VMs on a similar host. Similarly, any software failure on a single VM will not effect the performance of other VMs.

One important factor for service providers and users in the cloud is the management and operating cost of the system. Moving to the cloud using virtualization technology has significantly reduced the management and maintenance costs of IT infrastructure [24]. Eliminating the use of hardware devices, networking devices and cables in the virtual environment is an efficient way of reducing capital and deployment cost. The management cost, floor space and operational cost are also reduced at a personal level. Most of the enterprises, small organizations and new businesses save the purchasing, deployment and



managing cost of the systems and equipment using the cloud infrastructure. Startups can save their cost on investment; hence, they do not have to wait for years to get the payback from it.

Different workload patterns and traffic intensity are also a major concern to maintain system performance. The scalability in the virtualized environment makes it feasible for applications with dynamic load to efficiently scale-up or scale-down the resource request [2]. Software-based resource orchestration tools in a virtual environment further provide rapid resource provisioning with an optimal solution. Monitoring of resources and performance using software-based tools makes it relatively easy to manage, migrate and reschedule VMs for load balancing. Load balancing and consolidation are two common ways to improve resource utilization in these environments. Consolidation further reduces the computing cost in the cloud by possibly activating only a small number of servers.

Considering the environmental issues of the modern world, sustainability is also a major concern in all domains. With reduced hardware components and minimizing the power consumption (by running several applications or VMs on a single server), the negative impact of IT infrastructure on the environment can be reduced [24]. However, researchers are still investigating to further improve the energy-efficiency of algorithms and frameworks to reduce the carbon footprint of data centers and virtualized technology.



# 3 A Survey: Power Measurement and Power Consumption Models in Virtualized Networking and Computing Environments

## 3.1 Introduction

The emergence of cloud computing has led to massive virtualization in data centers. Network Function Virtualization (NFV) [25] and mobile edge computing [26] are extending the paradigm to both access and backhaul networks. Thanks to such increasingly available computing facilities, scientific, consumer and business domains are using software, applications and infrastructure as services offered by their providers [27]. This recent trend of growing virtualization and virtual networking has increased the power expenditure at data centers and its adverse effect on the environment, owing to the increased carbon footprint. Already in the year 2014 [6], the power consumption of cloud computing system and networks was about 100 billion kWh per year, with a global expenditure of about 40 billion US dollars. According to a survey by the Natural Resources Defense Council (NRDC) [5], data centers worldwide consumed in 2016 about 1.3% of the total electricity. With this trend it is predicted that the total power consumption of US data centers will most probably increase to 140 billion kWh by 2020. More recent studies estimate that the ICT industry may reach 20% of the worldwide electrical requirements, and emit up to 5.5% of the world's carbon emissions by 2025 [7]–[9]. Within this scenario, data centers on their own might account for more than 3.2% of greenhouse gas (GHG) emissions (corresponding to 1.9 Gtons per year). Therefore, the emerging virtualized environment needs an energy efficient cloud and virtual networking infrastructure to meet the modern world's requirements.

Strategic planning and developing methods for green computing in data centers have been and still are under investigation. Recent data [10], [28] show that these advancements in energy efficient frameworks and computing techniques might reduce the overall power consumption of US data centers by 33 billion kWh in 2020. As regards the integrated mobile and fixed networking environment of fifth generation mobile networks (5G), it has been noted that virtualization, though capable of increased energy efficiency, owing to consolidation of resources, risks to increase the power expenditure and carbon emission, unless proper optimization and dynamic adaptation strategies are put in operation [29], [30].

This high-level objective of remedial action therefore consists of optimization of power usage in (a) data centers and (b) softwarized and virtualized networks. These broad application scopes rest on a foundation of accurate power measurements of individual virtual components. Accuracy in energy measurements facilitates devising energy-efficient algorithms and effective VM consolidation methods. Precise modeling of power

consumption of a virtual machine or a physical device is essential for optimization. It is also essential for billing in multi-tenant environments, so that the Infrastructure Provider (IPr) can charge customers the fair amount for the resources (including energy) they consume.

However, it is difficult to measure power precisely for virtual components since current power meter measurements cannot be related directly to them. Furthermore, power consumption of similar virtual components differs based upon the running application and hosting machine. Most power measurement models available measure the consumption of the whole system. There is no accurate power meter available to measure the consumption of a single core of a server or for a single VM or container.

We observe more challenges in the use of multi-core processors. Parallel processing with an increased number of cores has increased speed and performance; however, power consumption has also increased, owing to growing traffic volumes and power-hungry tasks, unmatched by the hardware energy efficiency [31]. Further: with multi-core processors, efficient resource utilization and fair task allocation is a challenge [31]. This shared environment makes the power estimation of a virtual component even more challenging, as different virtual components share the same host resources.

The survey in this chapter indicates that recent research has started to provide directions to improve the accuracy of power-measurement tools. We suggest a unique tool, a problem-approach-development (PAD) triad, which, to the best of our knowledge, we are the first to use, to identify these directions. In the process of discovering triads, we identify a number of interests, or research domains, which can be delineated within the maturing research space on power consumption in virtualized environments. The volume of works in these research domains of the space is prohibitively large for a single survey. Here, we limit ourselves to one research domain: power models and meters for virtual computing and networking environments. We do, however, identify the other research domains in sec. 3.3.2.

This chapter is organized as follows. In sec. 3.2, we describe a unique method for carrying out a qualitative analysis. To facilitate readability of this work, we precede the detailed results with our analysis, presented in sec. 3.3. There, we give a qualitative assessment of the primary research domain through our reflections on themes which emerged as we organized the data. We have classified these themes as “state of the art”, “fallacies”, “pitfalls” and “research domains”, to suggest guidance and warnings which we were able to glean from others’ experiences. Sec. 3.4 carries the base product of this survey. We provide a digest, through a balanced set of a synthesis, a graphic device and statistics. The use of power models in real scenarios are presented in sec. 3.5. We wrap up our reporting in sec. 3.6, where we present our observations on favored tools and methods. Conclusion to the survey is provided in sec. 3.7.

## **3.2 Research Method Overview: Analysis with Structured Coding**

We base our method on (a) the coherence amongst widely-cited texts [32]–[35] in recommending thematic analysis as an introduction to the methods of qualitative analysis, and (b) upon its application in [36]. This form of qualitative analysis was selected as the primary method of gathering insights. We used thematic analysis on a body of works gathered from the ACM, IEEE and other sources (the “corpus”) to

produce our results. Use of a recognized method of analysis provides the needed structure and technique and supports the in-depth reading that, jointly, facilitate both objectivity in our results and critique of our results. The corpus is our raw qualitative data. It consists of a set of papers, which we refer to as *research units (RUs)*, i.e. publications (excluding surveys) in conference proceedings and journals. The prescriptions of sound qualitative analysis for systematic review require a choice of *coding method* of this data. The process of coding collates the diversity of the bodies of text into smaller *codes or labels*. Codes are terse, dense representations of a verbose articulation of a concept, and resemble jargon without the elitist connotation which this word carries.

Codes must have the following characteristics.

1. **They must be semantically rigorous** i.e. the meaning they represent must be clear and their use (application) must be unconfutable.
2. **They must be universally applicable across research units** i.e. they must provide a uniform means of dissecting publications. Use of more than one coding system (i.e. two or more non-universal coding systems) may create a split in the coded data with in-comparable parts across the split.

We satisfy these two requirements through the elemental coding method of structural coding. Structural coding poses a series of questions relevant to the inquiry in hand and is well suited to any problem which can be described using a standardized set of questions. We deconstruct the problem of literature review into a standardized question-and-answer protocol that can be directly converted into a structural coding approach. The questions are the following:

1. What is the topic in which the researcher is interested?
2. What is the problem which the researcher(s) saw as an opportunity for study?
3. What approach(es) did the researchers take in an attempt to solve the problem?
4. What development(s) and/or contributions derive from the researcher(s) work?

The corresponding structural codes are:

1. *I*nterest
2. *P*roblems and derivatives (challenges)
3. *A*pproaches
4. *D*evelopments / contributions

We refer to this protocol as the *IPAD review protocol*, and use it to survey research into power modeling and measurement in virtualized environments. Each RU (paper) is mined for “the topic in which the researcher is interested” (I-nodes), “the problem which the researcher(s) saw as an opportunity for study” (P-nodes), the “approach(es) . . . [taken by] the researchers . . . in an attempt to solve the problem” (A-nodes) and the “developments(s) and/or contributions deriv[ing] from the researcher(s) work”. The product of mining an RU is therefore one or more bound collections, or tetrads, that facilitate

a good apprehension of the RU. More importantly, as we explain shortly, it facilitates the act of locating this RU within the greater landscape of research (here: into power measurement and modelling in virtualized networking and computing environments).

In this particular study, diversity of interests was very limited. We refer to this observation again while introducing sec. 3.3.2, “Research domains”. Therefore, we chose to separate the interest-nodes from the rest of the nodes of the tetrad, to avoid repeating essentially the same two interests with every RU analyzed. From here on, we will focus on the P-A-D triads.

The act of coding is carried out as an intermediate step on the way towards categorization. For example, given any number of research units, a set of approach codes is collected. Minimally, one approach per RU can be discerned. Within the full set of approach codes, some codes will be identical. Others can be reduced to a code which carries enough of the original, unreduced meaning, to be non-trivial. This process of proximation (of the codes therefore leads to the aforementioned categorization of the codes into a condensed set that is conducive to (a) assimilation by a viewer, as well as (b) further rationalization.

We embark upon such a rationalization. We observe that each research unit may be represented as one or more triads and that the frequency of a triad within the overall set identified, is itself highly representative of the state of the art. The frequency of individual nodes (i.e. individual problem, or approach, or development) is itself meaningful. We also attempt to interpret frequency of occurrence of pairs of these nodes, or dyads. We expect the observed nodes, dyads, triads and their relative frequencies to be fertile grounds for grounded reflection about the state of research. This culminates our thematic analysis.

## 3.3 Analysis through Themes

### 3.3.1 State of the art

#### Trends

The concern about how to attribute system power to the virtual entities (VEs) in the process of developing a model of power consumption. The most common approach is that of attributing dynamic consumption only to guest VEs and static consumption to the host, or to a privileged VE (root in Hyper-V, dom0 in Xen). This coarse attribution must be complemented by one that is capable of dividing dynamic consumption amongst the individual VEs. The approach differs depending on whether VMs or containers are under study. With VMs, a common approach is to allocate processor events occurring within a VM’s scheduled time and “space” (the executing cores) to the account of that VM. In this manner, all three aspects of the model are captured: the events, the period within which the events are registered, and the overall system power consumption.

The second most significant research path departs from the problem of modeling VM power consumption in terms of its own resource use. The impact of utilization of a specific resource (usually, the processor) on power consumption is investigated (resource isolation) and used to develop a linear model.

A somewhat incidental path departs from concern with the relationship between workload type and power. In the course of isolating specific resources while approaching this problem, researchers observe aspects of the behavior of power consumption on

specific processor hardware. For example, it was observed that the Xeon Core 2’s exclusive-caching implementation activates L1 caches on cores that would otherwise be idle [37]. Suppose that core “A” is loaded with a memory-intensive thread. Then core “B”, which would otherwise be idle, is activated not for processing but to operate a sub-unit that searches L1 cache. This activation of otherwise idle cores leads to a rate of growth of power consumption with workload that is sub-linear. This follows because the rate of growth observed when the core “B” is intentionally loaded, is less than the rate of growth when core “A” is loaded. Several other highly-specific observations like this can be found [38], [39]. Their use is limited to specific hardware and the general lesson that accurate, broadly-scoped modeling of power consumption cannot be a highly abstracted study. We feel compelled to elaborate on this while discussing fallacies (sec. 3.3.4), as it seems that there is need for greater awareness of these limitations.

We further observe that the challenge of power consumption as a function of VM resource use is approached through three groupings of approach categories, which we enumerate below. Before presenting the enumeration, it must be added that this challenge has the complementary one of system power attribution (as described in this sub-section’s first paragraph). A fourth approach category, i.e. the set of those used to deal with system power attribution, should be added to our enumeration of groupings. Therefore, a basic methodology of the study of modeling power consumption in virtualized environments includes four complementary efforts:

1. Use of microarchitectural instrumentation and/or architectural instrumentation
2. Use of synthetic benchmarks (resource isolation) and/or representative workloads
3. Pre-selection of model type
4. Coarse (VE / non-VE) and granular (intra-VE) system power attribution

Our final observation concerns the scope for research in this field. Linear models are represented in about 35% of all developments observed. Non-linear models (polynomial regression or regression to some other closed form) are present in about 15%, while alternative models (Gaussian Mixture Model, Support Vector Machine, etc.) are present in about 10%. Meanwhile, researchers presenting alternative models have claimed that these are more accurate than linear regressions. Polynomial and other types of regression to closed-form are highly suspect, as they often coincide with modeling that is narrow in scope. In sec. 3.6, we evaluate formal methods at some length and conclude that it is time for future research to use the more sophisticated regression techniques.

### **Core challenge in VM power modeling and measurement**

The core challenge common to all problems is that modeling power consumption by a virtual machine has several dimensions of variability. Comparison with the power consumption of physical machines throws this core challenge into sharper relief. With physical machines:

- power consumption can be measured directly;
- the foundational dependence on virtualizing agent is ontologically foreign and

- the activity of other physical machines (that do not send or receive workload) is irrelevant.

It may be useful to think of the research spaces as possessing five dimensions of variability:

1. workload
2. virtualization agent (genre and technology)
3. host (resources and architecture)
4. power attribution approach (e.g. dynamic only or static + dynamic; processor only or processor and I/O)
5. co-hosted VMs

Each model's scope covers only a subspace of the ideal 5-dimensional space but the extent is not usually stated. We touch upon this issue briefly in sec. 3.3.4.

### **Three levels of abstraction**

We note that existing power models may be classified into one of three levels of abstraction.

1. Microarchitecture and architecture
2. Simple characterization of workload
3. Complex characterization of workload

Models at the microarchitecture and architecture levels of abstraction are low-level models. Power consumption is expressed in terms of variables that are defined at sub-CPU and computer system levels. The granularity of this level holds the greatest potential for accuracy but the rate of change of observed variables poses significant communicational and computational overhead. As a result, such models are unlikely to be used in centralized, real-time deployment but may be used in a layered power-control architecture as contemplated in ETSI Std. 203 237 [37]. Enokido, Takizawa and various others with whom they have co-published use simple characterization of workload e.g. [39]–[41]. These models describe power consumption in terms of fundamental descriptors of workload, e.g. number of processes and transmit/receive data rate. The least granular model uses a complex characterization of workload [30], [42]. The objective here is to quickly proceed to a good estimate of the power or energy required to produce the workload. This kind of model has no use in real-time control but it is useful for macroscopic comparisons, i.e. comparisons between two disparate systems for provision of a service. Possibly, the disparity is paradigmatic; for e.g.: classical vs virtualized implementations. Thus in [30], the implied unit is the amount of power required to deliver 1 million packets per second of throughput through an evolved packet-core's (EPC) serving gateway (SGW). In [42], while units of energy efficiency are not specified, the objective is to minimize the amount of power consumed for baseband processing functions in a radio-access network.



### Architectural or Microarchitectural Instrumentation

Many power models use hardware resource consumption to estimate the power consumed by virtual components. A study was conducted in [43] to evaluate the available power models for processors, VMs and servers. According to this survey, most of the power models for virtual machines use physical machine counters to estimate the corresponding resource utilization by the virtual components. We offer further insight on this matter. We concur in the observation that much current research is concerned with modeling power consumption of virtual machines. The approach may succinctly be described as estimates obtained from models trained out of collected data. We further indicate that approaches may be broadly divided into two groups, according to whether *architectural or microarchitectural* instrumentation data is collected. We therefore divide the works into these two groups, with further sub-division to create a representative set of samples. Within each sample's analysis, we present references to similar works.

#### 1. Architectural instrumentation

- (a) Joulemeter is a software power meter that was developed to measure the power of running processes, applications and different components on a physical host [44]. It has low overhead and uses a built-in power sensor available in modern processors to measure the power drawn. It monitors the run time resource usage to compute the resources used by any running process or a virtual machine. For the virtual machine, the virtual processor's activity is tracked at the hypervisor where the resources consumed by each VM, such as CPU utilization, memory and network are observed. Other components are assumed static and included in the idle power of the server. The percentage error for different workloads running on a host machine was in the range of 2 – 5%. This power monitoring tool can measure the power consumption by a VM in a heterogeneous VM network as well. Joulemeter was also used for the management and provisioning of a VM in a virtual environment in [44]. Experimental results show that VM management based on power estimation by Joulemeter can reduce power consumption by up to 8-10%. Joulemeter is no longer being developed by Microsoft as a standalone tool. Microsoft is integrating energy-awareness directly into Visual Studio [45].
- (b) Another approach is presented in [38]. Here, the power model of virtual machines is designed by monitoring the power consumption of the server in two phases. In the initial phase, no VM is running. The idle power of server is measured, which includes the power consumed by CPU, hard disk, network and other hardware components. Different polynomial degrees are allocated automatically to the model variables to estimate the whole system power on stressing CPU. Unknown parameters of the system's power function are found using multiple regression techniques. These components or variables are assumed to provide the combined resource consumption of all VMs running. In the second (runtime) phase, each virtual machine is started after some amount of time, and then the effect of the VM on the counter values and measured power, is tracked. Results show that a linear regression for power consumption values with CPU intensive workload gives a higher error of up to 10%, equivalent to an root mean square error of 20W. However, using polynomial regression in the initial phase reduces the error to 1.9% with

an average measurement deviation of 0.7W. The power estimation is also observed by switching on the turbo mode of the processor. It was found that with higher polynomial degree in the regression model, the estimation error in turbo mode can be reduced too. The drawback of this approach is that only the behavior of a single VM running alone on a server is considered. Several VMs running concurrently on a server may consume the resources in a different manner. Such loading is not considered in this study.

- (c) Another power model was designed to estimate the power consumed by a physical host by varying the number of VMs running on it [46]. The power profile of the host was created corresponding to the type and number of loads running on each virtual machine. Results show that power consumption increases non-linearly with the number of VMs. Linear regression was applied to the observed data, which shows the logarithmic relation between VM resource utilization and host power. However, this model is only useful as long as the concurrent VMs consume equal processor resources. Hence this model is limited in practice, since resources utilized by concurrent VMs may vary. Also, it is difficult to estimate the consumption of individual VMs when more than one VM is running on a host as the aggregated virtual load is considered in the experiment.
- (d) Since the number of CPU cores also affects the power and performance of the server and VMs, a core-aware power model (CAM) is proposed in [47]. This model profiles the power behavior of a VM and the server, considering the CPU utilization and number of cores present. For the VM power model, the CPU utilization of each VM is observed using the software tool Sysstat running inside each VM, and the server power is calculated using the derived core-aware power model. The resulting model can be used to study the impact of varying the core count on a VM's power consumption. The model has low overhead, as it has just two features to train for different scenarios, although in networks with dynamic resource allocation this model might not be feasible.

## 2. Microarchitectural instrumentation

- (a) In [48], a distributed accounting framework of system power consumption is proposed, in order to manage guest machines. This framework is designed for systems with hypervisor-based virtual machines, where energy management is implemented at host-level and at guest-level. Host-level energy management is responsible for controlling CPU and disk usage of the virtual machines, whereas the guest-level monitors the same resources inside the VM. At the host-level, CPU power is modeled using processor counter values. Each counter is given a weight according to its contribution to power consumption. For disk usage contribution to server power, the model uses disk usage and transfer rate for idle and active state for a calculated time. The framework also considers the recursive energy at the hypervisor layer which is usually the amount of energy required for the transfer of information from virtual to physical device and vice versa. Similarly, at the guest level, the vCPU and vDisk are monitored, with the difference that host-level used real hardware counters whereas guest-level uses the virtual performance counters. Using this framework, the study proposed the energy-aware management of physical

resources for a virtualized environment.

- (b) A power model for a virtual machine is presented in [49], which sums the power consumption by 22 major sub-units of the Pentium 4 processor, based on NetBurst microarchitecture. The authors select counters to represent the different sub-units of the CPU. These include: instructions per second, L1 and L2 cache, Branch, Bus control, etc. Values obtained for each sub-unit are weighted and aggregate power is calculated for the server.
- (c) Another study in the survey [37] suggested that different cache levels consume different amounts of power. These metrics were incorporated in the research design to develop a power model by observing the different levels of memory hit by the running VM processes. Instructions retired and last-level cache miss counter events were monitored to obtain CPU and memory utilization by a process. Memory utilization shows significant change in the slope of its linear model when used with parallel cache access compared with sequential memory access. Hence, two different power models are derived for both cases. Power bounds (power consumption range of a server) for each case are defined using the particular case's linear model. This model can estimate the power required by any process through the number of memory hits at a particular level of cache.
- (d) Another power model for VM was proposed using hardware counters in [50]. Instructions per cycle (IPC) and memory counters are used as the input feature for training the prediction model in this research. For the training phase, the power model for the physical machine and its hosted VMs, is modeled on a service machine (a "server") using a Gaussian Mixture Model (GMM). Each model consists of several Gaussian distributions, corresponding to possible microarchitecture level interactions among components and reflected in counters values such as instruction per cycle (IPC), memory, cache etc. For the prediction phase, these counter values are mapped using Gaussian Mixture Vector Quantization (GMVQ) classifier to find the optimal power value for the given input metrics. Results shows that this model can achieve prediction accuracy with less than 10% error and outperforms linear regression models. Since this model uses a separate power model for each client on a server, it can be computationally exhaustive for multi-tenant servers such as in a cloud data center. Data collection and modeling is carried out on a service machine which is also a 'server' to reduce the impact of resource usage by the model in power prediction. Host machines in the cluster send architecture metrics of their constituent VMs to the server.
- (e) Another study proposed a software framework to collect the consumption of resources by VMs [51]. The power model was then developed using resource usage counters (oprofile for hardware events and iostat for disk usage) for the VMs and dependencies of these counters on each other. A non-linear regression model (Support Vector Regression (SVR)) was used to develop the power model for this system. The study shows that different resource usage counters are highly correlated and incorporating their relation in the model can increase the accuracy of power consumption modeling.

### Direct or indirect measurement of power consumption in virtualized environments?

We observe that most research in modeling power consumption seeks to obviate the need for direct measurement through indirect measurement, namely: of resource use which has a discoverable relationship with power consumption by the entity hosting the resources. modeling, here, has the objective of indirect measurement of a variable that is not directly accessible (power consumption by VEs), through others which have convenient and reliable instrumentation. The accessible variables are referred to as (power) proxies. Intel's Running Average Power Limit (RAPL) interface provides a unique approach to measurement as it directly addresses power consumption. The accessible variables are referred to as (power) proxies. Intel's RAPL interface provides a unique approach to measurement as it directly addresses power consumption. However, notwithstanding appearances of direct measurement, RAPL is actually based on a software model that uses performance monitoring counters (PMCs) as predictor variables to measure power consumption [52]. It is available in processors starting from the Sandy Bridge microarchitecture. RAPL measures the power consumption of different physical domains, where each domain consists of either cores, sockets, caches, or GPU. We briefly denote its accuracy through references to research that has investigated them.

1. A study was conducted to find the advantages and drawbacks of using RAPL [53]. Different Intel's architectures such as Sandy Bridge, Haswell and Skylake were used in the experiments to analyze the RAPL's accuracy and its overhead. Data collected were modeled using linear model and Generalized Additive Model (GAM). Accuracy of predicted results was compared with the measured power consumption from precise external hardware power meter where RAPL based model shows 1.8-4.3% of error for the various architectures. Prediction accuracy of RAPL based power model was also compared with power model based on OS counters, where OS based models show high error of 5-16%. Also, the performance overhead (in terms of wall clock time) of using RAPL was studied at different sampling frequencies and for different application runs. Results show that even with high sampling frequency of 1100Hz, RAPL incurs overhead of not more than 2%. Some limitations of using RAPL include: poor driver support to read energy counters, overflow of registers due to its 32bit size and measurement of energy consumed by individual core.
2. Another study to analyze the precision of RAPL is presented in [54], where only the dynamic change in power consumption is observed. WattsUp Pro, an external power measurement unit is used as a reference for power measurement values. Intel Haswell and Skylake servers were used in the experiments to run different applications and to find the reliability of counter based power model and external power meter. However, in this research work, only two power domain packages (power consumption of whole socket) and DRAM domain of RAPL were observed. Applications such as dense matrix multiplication and 2D Fast Fourier Transform were used for server power profiling. Results shows that power measurement error varies with changing application and its workload size. For different applications the average measurement error using RAPL was in the range of 13-73% considering WattsUp power meter as the ground truth. It was concluded that with the modern multi-core parallel processing and resource contention for shared resources,

there is a complex non-linearity between performance, workload size and energy consumption.

### Lack of use of metrics of energy efficiency

We note several experiments [39], [40], [43], [55]–[58] that target power consumption but less than 5% of our corpus approach the problem in terms of some energy efficiency metric [59]–[61] (see fig. 3.3 in sec. 3.4). It is necessary to move beyond measurements of how much power was consumed, to measurements of how much power was consumed to carry out a specific task. This change in approach facilitates comparison between research works. More importantly, it directly addresses the question about cost of operation of infrastructure.

This approach requires identification of a unit of comparison, that transcends the boundaries of disparate systems that deliver this unit. This unit of comparison is referred to in the LCA framework (ISO 14040) [62] as the functional unit. A definition specific to telecommunications equipment is given in [63]: the functional unit is defined as “a performance representation of the system under analysis”. Since this definition is too broad to serve as a standard, units specific to a variety of classes of equipment are defined [63]. Two approaches we have seen are hash/J [28] and J/Web Interaction [64]. The functional unit in these cases are performance of one crypto-hash and one web interaction respectively. Another is to define a functional unit specific to a digital service delivered over a telecommunications network, e.g. ten minutes’ time of browsing [65].

### 3.3.2 Research Domains

In the introduction (sec. 3.1), we claim to identify a number of interests, or research domains within the “maturing research space on power consumption in virtualized environments”. In this sub-section, we separate the interests from the rest of the nodes of the tetrad (as we have pointed out while describing our method, in sec. 3.2), to give them some prominence and avoid repeating essentially the same two interest categories with every RU analyzed. We suggest that the interests are a theme in their own right and briefly describe them in sub-sub-sections below where the interests / research domains are named as their titles. Samples from the literature are used to better illustrate the research domain referred to.

#### Models of power consumption

Models of power consumption are investigated at several levels of granularity, with a variety of approaches.

1. *Power models for servers* [66], [67] are tackled in early work that attempted to predict power consumption in computing machinery. One approach [66] is “power profiling” of a physical host. Power profiling is a process that produces a qualitative apprehension of the effect of a change in an independent variable upon the power consumption of the entity under test. In so far as it remains qualitative, it can only be used to guide design rather than produce a numerical model in the toolbox of power-aware design.
2. *Power models for processors* are widely investigated. Although extensive research has been carried out on the accurate measurement of power consumption

of a physical system, the research question about precise measurements of a single core's power consumption is still open. Without having precise power estimation or measurement, it is challenging to optimize the resource allocation, load distribution and task management at the hypervisor level, where load balancing is one method to achieve power efficiency in most processors and their networks.

- (a) Moreover, with the advent of multi-core systems, these measurement tasks are now much harder [31]. Available power models consider the power consumption of the whole system and balance the load with consideration of aggregated traffic. Several studies, however, have been carried out to make the task allocation among cores fair by developing power models for multi-core systems [68]–[70]. Fair traffic allocation policies have been designed to share load among the cores of a processor in order to have distributed power consumption.
  - (b) In [71], the authors approach the problem using CPU hardware counters. They found that the counter value for the number of instructions fetched can affect the estimation of the system's power consumption. A power model using instructions fetched and instructions processed per CPU cycle was developed and trained with different SPEC2000 benchmarks. Experimental results obtained from the test dataset show that the model can estimate the server power with an error of around 2.6%.
  - (c) Another study [72] derives a simple power model for a multi-core microprocessor with static and dynamic power consumption. The number of active cores in a multi-core server can affect the dynamic power of the system, but the static power consumption will remain constant by varying the number of cores in idle state. Since memory components are also fabricated on chips, researchers also included the power consumption by caches in the power expression.
  - (d) In the research work presented in [55], an optimal server configuration for power and performance trade-off in the cloud environment is obtained. A multi-core server is analyzed using an M/M/m queuing model where the average task response time and its associated average power are measured. The waiting time for the task is calculated using the queuing model and the power consumption model is derived using the digital circuit power dissipation expression, considering constant speed at all cores. This research shows that power and performance optimization for a server can be obtained through optimal selection of the server size (i.e. number of server cores) and core speed. For servers with similar core speed, it is concluded that fewer cores with high speed perform better than more cores at slower speed. A limitation of this work is the consideration of an ideal case, where the speeds of all cores are similar at every time instant, which is not the case in a real-life scenario. Moreover, the resource utilization is measured for the whole system and not on an individual core basis. Also, each core receives the same load; hence, this model might not be valid for cores experiencing different workload and power consumption.
3. **Identification of good proxies for power consumption** is another important aspect to consider for power measurement. The objective is identification of those parameters which have at least a clear correlation with power consumption. Keong

et al. [56] take the approach of seeking software application metrics which fulfill this proxy role.

### **The effect of parallel processing and architecture on system performance and power**

The effect of parallel processing and architecture on system performance and power consumption is studied in [73]. Starting from Amdahl's law, the authors observe how the system's energy can be affected by parallelism and traded off with delay cost. This research work first derives an expression for a processor architecture's generalized energy-delay-product cost function. It then proceeds to determine expressions particular to specific microarchitectures for the energy- and the delay components of the cost function. Furthermore, two forms of the cost function are employed, to reflect the difference in importance afforded to performance and energy savings, respectively. The Energy-Delay (ED) cost function, where both energy and delay are equally weighted, and the Energy-Delay square (ED2) cost function, with double contribution of delay, are used to evaluate the processor speed and its corresponding performance. In ED2, more emphasis is placed on performance (speed) than power. Since these generalized expressions embed the trade off between power and delay, they can be used to optimize the processor performance by considering these design parameters (as was done, e.g., in [74]–[76] in the context of NFV).

### **Real-time control of power consumption**

Real-time control of power consumption (using various approaches, particularly schedulers) is widely investigated. One particular approach considered load balancing and task-cache affinity [68]. To balance the load among cores, load assigned to different threads may need to be switched across caches. It is desirable to balance two objectives: (a) improve affinity between thread data and cache in a multi-core (multi-cache) system, hence improving energy efficiency while (b) keeping load balanced among cores. Towards this end, scheduling of request queues using dynamic weights is proposed. Tasks of similar type are directed towards the same thread to avoid back and forth movement (across cores) of arriving tasks. However, this may add network latency and increase power depletion of the system as tasks with higher incoming frequency need to wait longer at their thread queue. This problem is tackled by assigning weights to all tasks at the scheduler, based on offline web traces. The selection of weight for each type of task is based on how much time and CPU resources the task will take to process. Since the mean service time is calculated from off-line log files, which may not closely match the runtime dynamic system, the model appears to be suitable if the network has frequent and similar types of traffic arrivals. Also, this model enforces hard affinity, i.e., each core will process only a dedicated type of task, which is another limitation of this research work.

### **Open and flexible programming interfaces**

Open and flexible programming interfaces are of particular interest, as they abstract away the diversity in parameterization of hardware power models, to facilitate interaction with a control plane entity. Furthermore, abstraction can be used to convey the effect on power consumption of control parameters, between the network management plane and

the control plane of network devices. In this regard, the Green Abstraction Layer (GAL) architecture was introduced in [77], and later adopted as an ETSI standard [78]; Power management capabilities of the future energy telecommunication fixed network nodes. It defines an abstract interface to represent power and performance parameters at different levels of complexity of a network device (such as entire device, chassis, line cards, single hardware component). The GAL represents the energy capabilities along the chain of architectural hardware components of networking devices. The GAL concept can also be extended to embrace virtual entities, and it might be an effective method to represent different granularity levels of power consumptions also in virtualized environments.

### 3.3.3 Pitfalls

#### **Power consumption does not in general increase linearly with processor utilization**

Notwithstanding advances made in identifying operating contexts that manifest a sub-linear power-utilization relationship [37], [50], recent publications [76], [79]–[85] persist in using the linear model without acknowledging its limitations. The model is simple to use and has some foundations in research [67]. It has three premises, described here with regard to the operation of Microsoft Windows:

1. When Windows has no threads to run on a logical core, it schedules the idle thread [86]
2. The idle thread keeps the processor in a low-power state [87]. The specific state depends on the processor’s green capabilities.
3. In the complement (non-idle time), the processor issues instructions at a constant rate.

This simple model has limitations ([45], p.6/12, [50], p.808). It fails to take into account diverse processor operating contexts, some of which are coming to bear on current use cases. Specifically, the third premise is true only to the extent to which instructions are being fetched and data are being loaded from/stored to instruction and data cache respectively. Consider the context of 90% and greater hit ratios. Even the short time periods over which rate of instruction issue is computed, can be expected to lead to a narrowly distributed probability density function of such a rate. By contrast: the lower the hit ratio at the cache level before main memory, the lower the fraction of non-idle time at which power consumption saturates. This saturation is strikingly illustrated in ([50], fig.1). Variation of power consumption due to execution of tests from the SPEC CPU2000 benchmark suite is shown. The power consumption diverges at 25% CPU utilization and the consumption of the processor-bound test (mesa) is greater than that of the memory-bound test (mcf) by a factor of about 2.6.

For Intel processors with Intel® Hyper-Threading technology enabled, the operating context under which a linear relationship is subject to the lowest error includes at least the following two conditions.

1. The processor cores are increasing their instruction issue rate in proportion to the fraction of time they spend busy [37]. This implies that instruction and data cache hit ratios are high. This is simply the third premise.



2. Only one logical core is active per physical core at any given time [37], [50]. Expressed alternatively, actual utilization must lie below half maximum utilization. The underlying cause is that activation of the second logical core employs fewer organizational units of the processor than activation of the first logical core.

The first condition is particularly problematic, as cache miss ratios are likely to be much higher in the context of virtualized environments. In such environments, the number of runnable threads is the sum of runnable threads controlled by independent operating systems. Evidently, this is higher than the expected number of runnable threads on a single server instance. Other evidence of this “utilization trap” is not hard to find. In [79], the compute resource is stressed using `cpulimit` and `stress-ng`. The “`cpulimit`” utility runs a specified process image, then pauses and resumes it until a certain percentage utilization is reached [82]. The repetitive execution of a single process is highly likely to create conditions for very high instruction- and data-cache hit ratios.

### **Application characteristics significantly impact power consumption**

In [58], very high time overheads and significant power consumption overhead were observed in the virtualized implementation of a web server with respect to its physical counterpart. The empirical context consists of software network switches, which are part of the agent of virtualization. While this is a representative context, it is noteworthy that this is a network-intensive application that uses the agent of virtualization’s software layer-2 switch. This is an emulation of a device. As such, this component (the emulation) of the agent of virtualization does not meet the second of Popek’s and Goldberg’s three characteristics of virtualized environments: “show[s] at worst only minor decreases in speed” [88]. As regards power, the implication of the overhead in instructions is overhead in consumption. Indeed, the authors recognize the scope of their work: they plan to carry out research in “advanced [VM] packet switching techniques”.

The impact of application on server power consumption is central to the approach of Enokido and Takizawa, who have published or co-published a series of works on models of power consumption under different operating conditions, e.g. compute-intensive [40], communication-intensive [39] and storage-intensive applications [39]. They explicitly accentuate the relationship between power consumption model and application process [40]. We address the fallacy of the generalized power consumption model in sec. 3.3.4.

### **Benchmarks may skew power consumption according to their organizational dependencies**

In sec. 3.3.3, it was seen that both “`cpulimit`” and “`stress-ng`” do not produce generally representative measurement of power consumption. This observation is not limited to measurement of power consumption. Use of kernels, toy programs and synthetic benchmarks to measure performance has been identified as unrepresentative ([89], p.40) of general performance. Benchmarks are standardized workload generators that are used for comparison of computer systems for a highly specific class of application. Unless this application class is a good representative of the productive application of the computer system, the power consumption measured under test is not a reliable predictor of that obtained during productive use. It is necessary to plan test workload generators in advance and state the limits of validity of results. In [64], TPC-W is used which is a transactional web benchmark that can simulate the business oriented online web-servers.

The MySQL++Java version of TPC-W benchmark, suitable for cloud applications, is used to generate the online traffic, where three different traffic profiles based on browsing, purchasing and ordering of books are generated. The throughput measure for these servers are observed through the metric Web Interaction Per Second (WIPS).

### **Processor organization significantly impacts power consumption**

We illustrate this point with a wide-ranging example. The Intel Xeon X5670 and AMD Opteron 2435, both on x86-64 microarchitecture are compared in [90]. Different idle loops (using no operation, pause, repetition, etc.) were tested to see their effect on power consumption of both systems. It was observed that the Intel Xeon has a loop stream detector, which disables the processor's features like fetch and decode. On the other hand, the AMD processor has no hint to process these loops efficiently; hence, it consumed more power than the Intel processor. Furthermore, different ALU operations (such as load, addition, multiplication, etc.) consume different amount of power depending upon the instructions they require and memory location from where they are accessed. For Intel's processor bandwidth consumption of all ALU processes from a particular memory location is almost same but there is a difference in their power consumption. Such that 'load' operation consumes lowest power compare to other ALU operations performed, and this held true for all memory locations. The reason for this is since 'load' instruction just needs to load the content in to the processor registers whereas 'add' and 'mul' operations are more computation demanding. This behavior was however opposite on AMD processor, where the 'load' operation when accessed from L1 cache consume almost double the bandwidth than other operations hence also consume more power. This difference in resource utilization is due to the different microarchitecture. In AMD processors, the 'load' instructions is handled by many floating-point pipelines, and other instructions just uses single pipeline for their operations. Also AMD processors, specifically, have an exclusive cache level design, which requires write-back function when evicting data among different cache levels. On the other hand, Intel's inclusive cache design does not require this function; therefore consume less power. However, when accessed from L2 cache, L3 cache, and RAM on AMD, all performed ALU operations consume almost similar amount of bandwidth, and consume similar amount of power.

### **Power consumption or energy consumption?**

Power consumption and energy consumption are evidently closely related yet concern with one is not identical to concern with the other. Concern with energy consumption is inherently a concern with the continued supply of energy whereas concern with power consumption is a concern with the implications of using it at the current rate. Concern with power consumption is less about the availability of energy and more about the cost of its use and the rate at which it produces other undesirable side-effects, notably GHG release. Therefore, power consumption is the focus of research in situations of regular (if not guaranteed) supply; energy consumption is the focus where the energy source is finite over a timespan of hours, or days. This distinction is perhaps a little pedantic yet there is value in clearly conceiving of, and communicating, a research interest.

### 3.3.4 Fallacies

#### A universal power model

We have suggested that the core challenge in modeling power consumption by virtual entities is its number of dimensions of variability. This has been demonstrated throughout this survey, where a number of generalizations have been addressed. Summarizing, literature shows that:

- host power consumption does not generally have a linear relationship with processor utilization;
- CPU-intensive workloads that repeatedly execute the same code skew power consumption results;
- network-intensive workloads are power- and time-consuming because they employ emulations of network switches, but the root cause (emulation in the hypervisor software switch) disappears with SR-IOV ([91]. p.5);
- host saturation must be taken into account in predicting VM power consumption;
- processor utilization (an architectural attribute) is insufficient to predict host power consumption and microarchitectural attributes, such as last-level-cache misses, are necessary to predict host power consumption even for the same level of processor utilization.

This list, while not exhaustive, amply illustrates that the several dimensions of variability are significant in determination of VM power consumption. A model claiming to determine power consumption as a function of fewer variables than the dimensions we have pointed out must be accompanied by a scoping region that limits its use. While a precise scope may be an unrealistic demand, looser conditions of use of the model are essential. We now illustrate using two examples from the corpus. Khan compares ([28], p.51) energy efficiency (hash/J) obtained by scheduling process threads on additional cores, with that obtained by scheduling them on that hardware threads on active cores (through Intel Hyper-Threading). He shows that the former is greater than the latter. Enokido and Takizawa show ([40], p.279) that for a given data transmission rate through the uplink of a software virtual switch, greater energy efficiency (W/bps) is obtained by operating an additional hardware thread on an active core (through Intel Hyper-Threading), than operating an otherwise idle core. An important difference lies in the task's processing "intensity", i.e. the rate of supply of instructions. While Khan's operations are tightly bound to the processor (cryptographic hashing), Enokido's and Takizawa's operations are distributed over the processor and network input/output. Without delving into detail, it is realistic to hypothesize that the average instructions per second demanded are far lower in the networking application, since transmission of a large file (as is the case here) does not take place in one processing burst. Operating time is divided between the processor and the media channel. In such a scenario, the added capacity of the same-core hardware thread suffices.

#### Research on power models without knowing power-relevant context

We start with an example. Enokido's and Takizawa's work [39] derives a power consumption model for a server while VMs run computation-bound processes. The servers

used run on Intel Core i5-3230M processors. These processors are used in the mobile device market [92]. They are capable of low-power idle states [93]. CentOS 6.5 uses a tickles kernel [94]. Combined, these facts, relevant to the context of power consumption, provide a plausible explanation for the observed increment in power, (denoted, in [46], by  $\min C_t$ ), when a core in a package is activated.

Another example regards [46]. No reference is made to whether Hyper Threading is enabled. This is essential to understanding how the ESXi vCPUs are created. Neither is any information given about how the vCPUs are related to physical (logical) cores. Nor are we told how virtual network interfaces and switching are set up. ESXi version 5 offers both paravirtualization (“vmxnet”) and emulation (“e1000”) to implement virtual network interfaces. The impact on energy consumption of selecting a virtual network interface implemented by emulation, can be expected to be high [58].

The most notable general shortcoming regards the exploitation of processor low-power modes of operation by hypervisors and guest operating systems. Such exploitation must be investigated and conclusions stated as part of the declaration of experimental setup. This facilitates the reproducibility of results and (as indicated within the fallacy about a universal model) partially defines a scope for the model obtained. Both the above examples can be applied to illustrating the fallacy of the universal power model but they emphasize a particular aspect that merits special attention. A researcher into power models is expected to qualify his/her results within the physical reality in which power is being consumed. Research into power models involves hard components and a diligent characterization thereof is essential to the acceptance of work as scientific research.

### 3.4 A Digest of Challenges, Approaches and Developments

This section contains our digest of research produced by the IPAD protocol. The digest consists of:

1. the graphic device
2. statistics

#### 3.4.1 Graphic Device

Fig. A.1, Fig. A.2, and Fig. A.3 are the product of this research coding structure referred as IPAD (in sec. 3.2). These graphics in Appendix A are a node graph (or a mind map) that shows the state-of-the-art of research in our chosen scope. In particular, these graphs are encoding of surveyed research units, that strives to relieve a profile not only of current knowledge (the developments) but also of what has been found a fruitful pursuit (the approaches) thereof. This is obtained through the relationships that are illustrated in the graphic between the problems addressed, the approaches to solutions, and the knowledge obtained in pursuit of solutions. Each node in the graph is enumerated as category or sub-category. Each problem, approach and development in the graph is called a node and is represented by letter 'P', 'A', and 'D' respectively. A full tabulation of the nodes included within the categories in presented in Appendix B.

### 3.4.2 Statistics

The frequency of occurrence of aspects of data collected is examined here. We combine the structural codes into the following groups:

1. individual problems
2. individual developments
3. dyads of problems and approaches
4. triads of problems, approaches and developments

In the following sections, we interpret the statistics and suggest meaningful metrics and report our results alongside.

#### Metric of research interest: Individual problems

For each problem (challenge) identified, we determine the number of times in which it has been tackled in research units (RUs). We suggest this as a metric, denoted by  $R_k$ , of the **attention**, or **research interest**, which this challenge is receiving. We also observe that a solution to a problem can be approached, within a single RU, in multiple, complementary ways. Indeed, this diversity is reflected in our data but the approach diversity factor is not significant to the interest which this challenge attracts. We avoid the approach diversity factor and measure attention by incrementing the metric only once per RU in which it is tackled.

$$R_k = \frac{\sum_{j=1}^{N_{RU}} P_k^{(j)}}{\sum_{i=1}^{N_p} \sum_{j=1}^{N_{RU}} P_i^{(j)}} \quad (3.1)$$

where  $P_k^{(j)}$  is a binary variable that represents the presence (or lack thereof) of a problem  $k$  within a single RU  $RU_j$ , over the corpus of  $N_{RU}$  unique RUs, with a total of  $N_p$  unique, identified challenges/problems.

#### Frequency of occurrence of categories of development

Development statistics are distributed thinly unless developments are categorized. However, when grouped into **meaningful clusters**, conclusions can be drawn about the frequency with which developments take place in sub-spaces of this research space (see Fig. 3.1). Thereby, a prospective researcher is guided through grounded insight into works that cover this space. We avoid further interpretations of development statistics in the interest of objectivity.

#### Metric of challenge complexity: Weighted challenges

We consider the diversity of approaches through which a challenge is tackled, as a **metric of challenge complexity**. The set of all unique problem-approach pairs in the triads is collected first. Then, for each problem, we add up the total number of pairs within which that problem is found. We suggest a normalized complexity metric,  $C_k$ , as follows:

$$C_k = \frac{\sum_{j=1}^{N_{PA}} P_k^{(j)}}{\sum_{i=1}^{N_P} \sum_{j=1}^{N_{PA}} P_i^{(j)}} \quad (3.2)$$

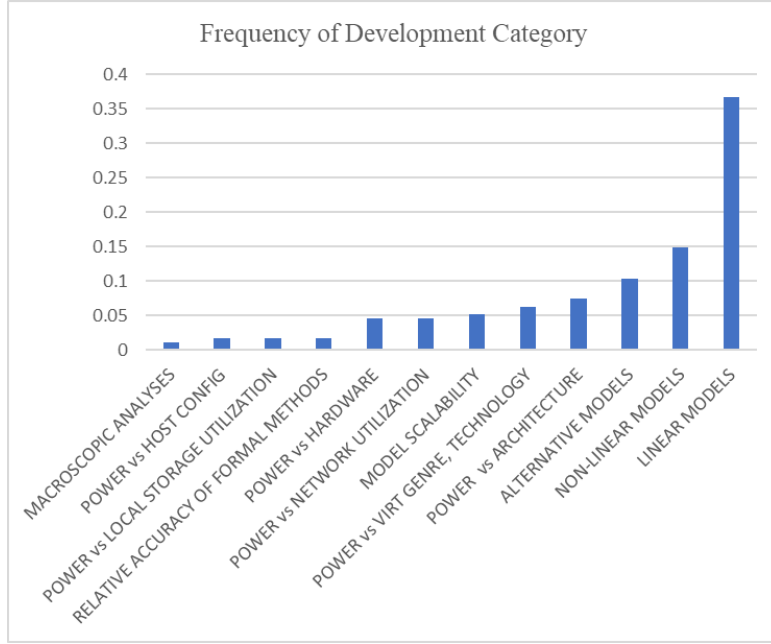


Fig. 3.1: Frequency of development category

where  $P_k^{(j)}$  is a binary variable that represents the presence (or lack thereof) of a problem  $k$ , within a single P – A pair  $PA_j$ , over the set of all  $N_{PA}$  unique P-A pairs within the corpus; with a total of  $N_p$  unique, identified challenges/problems.

Fig. 3.2 shows the research interest and complexity observed for the different categories of challenges.

### Metric of utility of an approach: Weighted approaches

We analyze approaches in terms of their **utility** i.e. how useful they are in the overall motion between problems and developments, and denote this metric as  $U_k$ . We increment the metric of utility each time a particular approach is a component of a triad within an RU. Therefore, a single RU may increment the metric several times. The utility metric of a specific approach  $k$  is the normalized metric:

$$U_k = \frac{\sum_{j=1}^{N_{RU}} \sum_{l=1}^{N_{triads_{RU_j}}} A_k^{(l)}}{\sum_{i=1}^{N_A} \sum_{j=1}^{N_{RU}} \sum_{l=1}^{N_{triads_{RU_j}}} A_i^{(l)}} \quad (3.3)$$

where  $A_k^{(l)}$  is a binary variable that represents the presence (or lack thereof) of approach  $k$ , within any of the  $N_{triads_{RU_j}}$  triads within a single research unit  $RU_j$ , over the corpus of  $N_{RU}$  unique  $RU_s$ ; with a total of  $N_A$  unique, identified approaches.

We emphasize that a single research unit may be described by several such triads that include approach  $k$ . Fig. 3.3 shows the normalized frequency of occurrence and utility metric observed with regard to approaches detected in the corpus.

### 3.4 A Digest of Challenges, Approaches and Developments

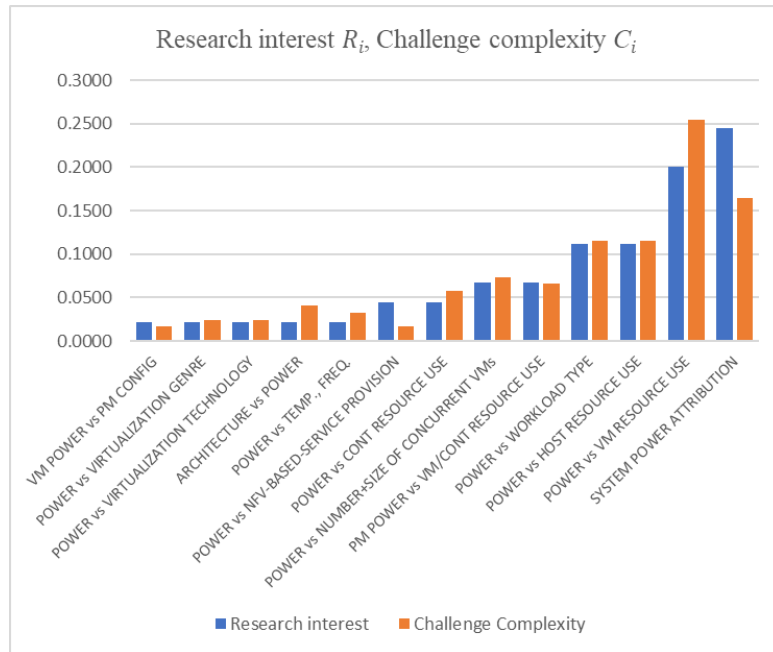


Fig. 3.2: Interest in problem categories and their perceived complexity

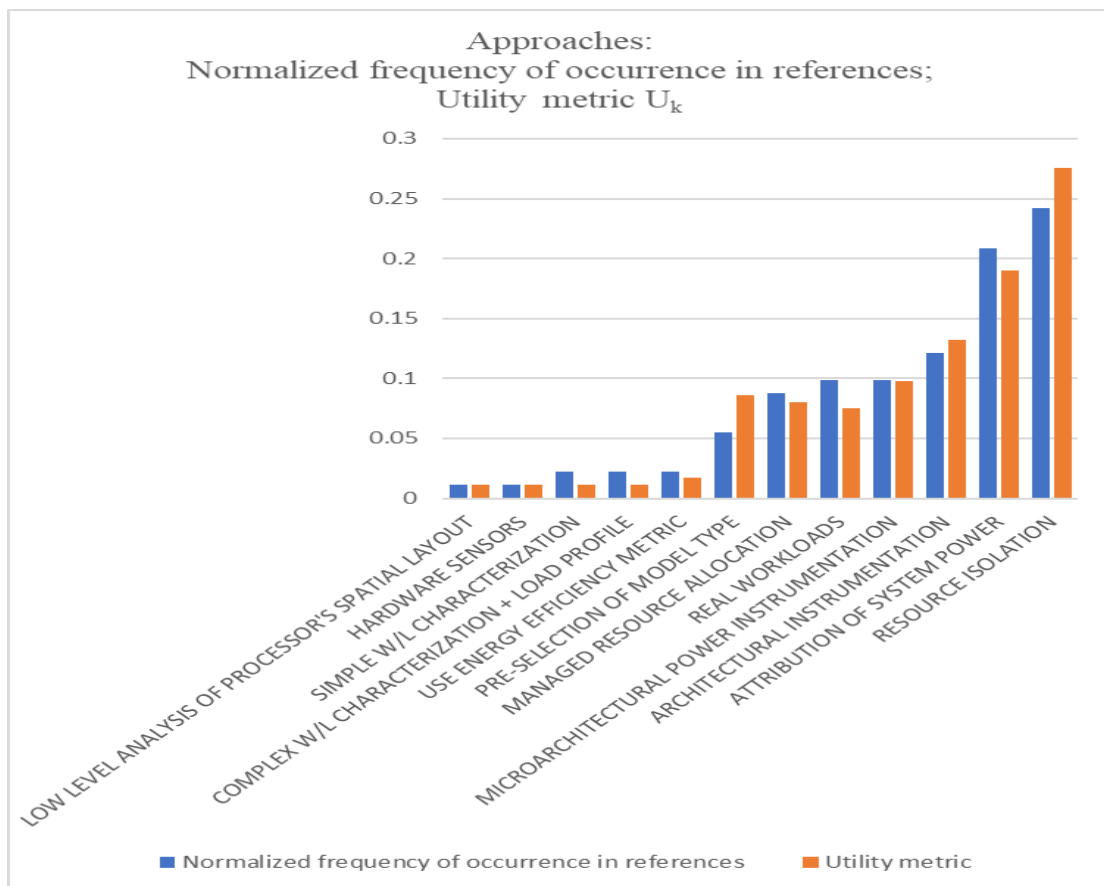


Fig. 3.3: Normalized frequency of occurrence of approaches and their utility metric

### 3.5 Applications

Interest in measuring and modeling power consumption by virtual machines and containers derives from the relationship which power consumption has with the *cost of operations* of information and communication technology infrastructure. This statement is relevant to *infrastructure provision* across the scale, whether in Cloud data centers or virtualized infrastructure running in points of delivery within the access segment of a public telecommunications network (PTN). Measurement and modeling are applied to *optimization* of these operations. We distinguish between optimization and minimization through *service constraints*: optimization is the minimization of costs subject to constraints on operations explicitly or implicitly obtained from service-level agreements (SLAs). A well-known example of the impact of SLAs on power consumption regards the use of redundant equipment to meet service availability requirements. Service availability (an important parameter of SLAs) depends critically on the absence of single points of failure (SPOFs). The redundant physical equipment deployed to avoid SPOFs increases power consumption. Power-aware operation of redundant infrastructure hinges on models of physical infrastructure's power consumption as a function of VM workload (measured in terms of architectural or microarchitectural entities). From this observation, it may be seen that application of modeling and measurement reviewed in section 4, lies in the *power-aware operation of virtualized environments*.

Here, we survey research into power awareness of operations characteristic to virtualized environments, where these characteristic operations have significant power consumption that must be measured and modeled. Three concerns emerge:

1. Creation, deletion, startup, shutdown, hibernation, resumption and migration operations characterize the dynamicity of virtualized environments. The significance of their *overhead* in power consumption must be quantified to determine its impact on energy efficiency. In particular, migration's cost is not trivial, although acceptable [95], [96].
2. Furthermore, in order to operate a commercial virtualized environment, *heterogeneity* in both VMs and hosts must be characterized in models. Primarily, heterogeneity refers to variability in allocation of resource quantities: a virtualized environment supports diverse VM sizes. Secondly, there are also implementational differences in physical machines and these differences manifest themselves in the power models. This dimension of variability must be included in modeling to support frameworks of operation, administration and management (OAM).
3. *Pay-per-use* is one of the key tenets of Cloud Computing. Implementation of this tenet depends upon the facility to account for resource usage. This facility may be passed on to the consumer or reserved by the provider for internal cost control. We observe an evolution of research into comprehensive frameworks capable of automating the transformation of resource usage into cost.

The first two concerns are part of the broader scope of a Network Functions Virtualization Management and Orchestration framework (NFV-MANO) [97] whereas the latter concern is part of a Business Support System (BSS). We divide reviewed research along these two system classes.



### 3.5.1 Management and Orchestration

#### VM Migration

We first consider a sample of works that deal with power-aware VM migrations [80], [98]–[100]. Their approach is divided into two parts. The first part regards basic research, where the researchers develop a power model to guide the second part. This latter part may be described as an approach to applied research, where the power model(s) is(are) used to quantify the impact of migrations on power consumption.

1. Problem in [98]: How does VM migration affect power consumption?
2. Approach:
  - a) Workload Model: Historical data about static and periodic cloud workload patterns was collected and used in an ARIMA model to predict workload. The predicted workload was transformed into a prediction of power consumption through the linear/polynomial model of power consumption as a function of virtual processor utilization.
  - b) Power Model: A relationship was obtained, through linear or polynomial regression, between the workload (utilization of virtual processors hosted by a physical machine), and its power consumption.
  - c) Applied research: The approach taken is to reduce frequency of migration. The SLA is respected by bounding server power below a threshold. Server power consumption is used since the relationship between workload and host power consumption is determined during basic research, and measurement of host power consumption is less invasive and less demanding than measurement of several individual VM workloads.
3. Key Developments:
  - a) The migration algorithm does not migrate the VM as soon as the high CPU utilization peak is detected. The algorithm waits for some time and if the high utilization persists, then the VM is migrated. This facilitates improved energy efficiency through optimal migration of heterogeneous virtual machines under different workloads.
  - b) VMs can be migrated without violating performance terms of the Service-Level Agreement (SLA) by limiting the server's power consumption below a given threshold. This framework considers the heterogeneity of virtual machines and predicts the variation in key parameters such as power, performance and migration cost, for different usage and size of the VM.
4. Limitations: The workload prediction model is limited to estimating the precise power cost of periodic workloads; therefore, it might not be useful for a dynamic scenario.

#### Orchestration in data centers

Deploying single service in a single host is fairly simple to manage, however, these days in large data centers and cloud network no single server is serving a single application. Virtualization has open new ways of service provisioning and application processing on a

single server where several users can run number of different applications, each isolated from other. The optimal resource management and task scheduling in such scenarios may took infinitely long time and consequently may consume more energy. Hence, an energy efficient management scheme, also called as orchestration tool, is needed that can find the optimal solution of scheduling problem within polynomial time limit. Although, some researchers proposed management schemes in [60], [101], [102], but there are also some orchestration tools [103]–[105] available in the market, which helps to manage the large network of cluster applications, multiple data centers, public and private clouds etc. These tools are energy efficient choice for VM consolidation and load balancing due to their robustness. Some proposed strategies and performance comparison of recent management schemes are as follow,

1. Problem addressed in [60]: Develop an energy efficient management scheme for large network with minimum impact on system performance.
2. Approach: Developed an energy-efficient cloud orchestrator (e-eco) which opt for an appropriate management scheme considering the cloud application behavior at a given instant. Two energy saving management schemes used in proposed orchestrator are VM consolidation and DVFS.
3. Key Developments:
  - a) Proposed strategy will allow VM consolidation for hosts running at low CPU usage rate when images of VM are centralized. It is an energy efficient technique as it only requires memory pages to transfer through the network. However, when VM images are only available where they were initialized, it is preferred to reduce the processors frequency using DVFS to save energy.
  - b) In comparison to previous studies, e-eco reaches the same energy saving rates with far less impact on the application performance. Performance trade-off between power saving and SLA violation was about 25% and 6% respectively.
1. Problem addressed in [102]: Energy efficient orchestration tool for VM consolidation in cloud
2. Approach: The optimize solution for VM placement and its network paths (required band width for consolidation) is found by combining Greedy Bin Packing (GBP) and Multi-Commodity Flow (MCF) algorithms. For making consolidation decision, an additional information from user's end is retrieved, that declare the used or idle state of the resources. Different approaches for consolidation categorized as max power saving (aggressive), balanced load and max performance (less aggressive), are proposed based on the power and performance tradeoff.
3. Key Developments:
  - a) For critical and latency sensitive applications less-aggressive consolidation approach (i.e. using maximum available servers) show better performance, although energy efficiency is compromised to some extent in this case. Aggressive consolidation (i.e. allocate maximum VMS to a single server) saves the maximum energy while maintaining the performance when VMs are not fully (100%) utilized.

- b) VM consolidation when considering the context information from users help in reducing the power consumption if some idle VMs are present in the system. The algorithm group all idle VMs together and place them on the server with lowest power state to reduce power consumption.
1. Problem in [106]: Performance evaluation of orchestration tools available in market for large network management.
  2. Approach: Comparison of four most used orchestration tools Docker Swarm, Kubernetes, Mesos and Cattle, has been done based on their framework complexity, scheduling and service layer performance, network scaling and failover time. Applications running on the servers includes Jenkins, WordPress and GitLab.
  3. Tools: Rancher software platform is used to organize the cluster and its UI tool is used to observe performance parameters of servers.
  4. Key Development:
    - a) The deployment of Kubernetes is more resource hungry than other tools because of its complex architecture. Whereas Cattle took the least time which might be because it is the Rancher's native tool. Hence, deploying Kubernetes cost for more power resource than other available tools.
    - b) Time required to scale-up the network by creating replica images (up to 100) was also observed. When the images are available locally, Kubernetes took the least time to deploy all replica images, whereas it took the most when images are being downloaded from 'Docker registry'. Hence, Kubernetes when communicating outside the cluster have a huge overhead. However, the increase in time was linear for increasing number of replicas over all platforms.
    - c) Recovery on failure of container and host server is also observed. For container failure, Kubernetes again is the fastest to recover than Docker Swarm (other two tools don't have this feature), since it has a local Kubernetes agent to monitor cluster health. Whereas host failure recovery was quick on Docker Swarm than Kubernetes. This might be because Docker architecture has the heartbeat functionality providing short time to report the failure, whereas for Kubernetes manager is notified after a series of events which introduce latency.

### 3.5.2 Real-time Control

#### Schedulers

In automated virtualized environments, scheduling is a set of activities in the control plane, guided by a management policy, that instantiates, monitors, migrates and terminates the virtual machines and/or containers which encapsulate the workload. One useful implementation of a power-aware scheduler would operate either to minimize a (cost) function of power under service constraints or a multi-objective function of power and some key performance indicator.

1. In [107], Weighted Round Robin (WRR) scheduling of dynamically arriving tasks among VMs is proposed. Variants of the Round Robin (RR) approach have been

used in several scheduling processes. The algorithm presented in [107] combines classical Round Robin and Weighted Round Robin and proposed and Improved Weighted Round Robin (IWRR). The static scheduler in IWRR consider the resource capacity, task length, priority and load on VMs to allocate task to an appropriate VM. To further optimize the system performance, a dynamic scheduler monitors the system status. It balances the load among VMs on completion of each task. In this way unbalanced resource usage that may occur because of poor estimation of task completion time or processing delay, could be fixed immediately.

2. To deal with dynamic load and resource management in virtualized environments, a discrete state transition model is designed in [101]. This model provides a predictive controller for virtual systems based on a non-linear power model. The states are defined based on the server state (on or off), the number of VMs and their frequency. The processor can switch among these states during runtime to balance the workload among different virtual machines. This model, however, has the limitation of being unsuitable for a large state space, as this will make scheduling and design complex.

### Server Power Capping

We apply the IPAD review protocol here. Since this triad regards an application, rather than a model, we do not include the nodes mined here in the statistics.

1. Problem, in [99], Host power capping in dynamic environment
2. Approach:
  - a) Basic research: the effect of hyper-threading and processor fan speed on VM power consumption is analyzed. The study found that the varying power consumption of a fan changes the power consumption of a server. Also, to attribute fan power consumption fairly among all VMs, each VM's share is found using the dynamic power behavior for each virtual machine.
  - b) Applied research: This power model was used to cap server power consumption. A host near its power threshold is detected with assigned maximum and minimum power of the server. The VM to migrate is chosen by monitoring its dynamic power and its effect on server power after migration. The virtual machine selected for migration is that with the least dynamic power necessary to reduce its host power consumption below the upper threshold. The VM consuming least dynamic power is chosen as that which takes least time to migrate
3. Key developments: Results show that this power measurement scheme for virtual machines is accurate enough to support effective server power capping through the use of VM migration.

### 3.5.3 Billing

Fair billing of power consumption is enabled by granular tracking of the financial cost of power consumption by resources (compute, storage and network) used by a customer. The emphasis on usage is purposeful. Even with the earliest accounting method of

billing by active time, it was recognized that resource allocation does not impinge on an infrastructure provider’s operating cost until it is active, i.e. in use. At this level of granularity, accounting is indiscriminate of power consumption. With respect to some referential time period, whether the resource is active and idle or active and fully utilized, only the active status affects the charge imposed on the customer. This is gratingly unjust and calls for remedial action. Indeed, significant effort has been brought to bear in at least two channels of effort: the academic, wherein modeling and measurement is the primary means to enabling this application, and the industrial, where, concurrently, efforts to harness academia’s (and industry’s) developments in this space are under way in the form of a green abstraction layer for virtualized environments. A brief review of these two channels follows.

### Development within academia

1. We observe firstly several works directed specifically at coupling power consumption, virtual resource usage and accounting for costs. Here, the intention is not solely to relate virtual resource usage and the ensuing power consumption but rather a more ambitious infrastructural combination of measurement and cost into a single system.
  - a) In [108],
    - i. Problem: The authors tackle per-VM energy consumption.
    - ii. Approach: They extend earlier work [109] on modeling power consumption by multi-core processors through microarchitectural instrumentation of processor subunit activity. The monitoring tool, `perfmon2` [110], enables gathering of resource usage counters per process. The agent of virtualization, KVM [111], encapsulates all thread activity deriving from a single VM, into a single process. Thereby, the Linux operating system running KVM – the “host OS” [108], provides a single process of reference for `perfmon2`’s tracking.
    - iii. Formal method: Linear regression fits the training `perfmon2` data into a model of power consumption.
    - iv. Development: The authors shows their model remains accurate with simultaneous VMs and when tested under conditions where DVFS is applied to processor operations. They observe that the principal limitation regards tracking “brownouts” under conditions of moderately dense VM packing per host.
  - b) In [112]
    - i. Problem: Billing cost for cloud service is not based on CPU/resource usage, instead it is fixed based on VM size and time duration it is used.
    - ii. Approach: Authors identified that the cost models such as ‘pay-as-you-go’ and even ‘pay-as-you-use’ does not charge fair amount to its users, as these models do not consider the workload on processor or consumed resources. These models charge same amount to their users, those having similar VM configuration even if they utilize different CPU percentages. Consider the case where two users using similar VM on cloud but running

different applications, such as webserver and CPU intensive workload. The VM running web-server will use around 5% of the CPU with idle period whereas other VM might use 90% of the CPU resources. The current models such as ‘pay-as-you-go’, will charge the same price to both users despite both having distinct workload, profiles and consequent energy consumption. Hence authors in [112], proposed a cost model for VM instances based on ‘Proportional-Shared Virtual Energy (PSVE)’, which is composed of CPU energy consumption and traditional commodity prices such as management and operational cost of hypervisor.

- iii. Method: Developing an analytical cost model for the VM instances, to have fair distribution of energy consumption between hosted tenants.
- iv. Development:
  - A. The energy consumed by a single vCPU is dependent on its workload.
  - B. Considering the CPU usage knowledge available at hypervisor, a proportional energy-cost sharing model for a VM can be developed.
  - C. Proposed model can also be used to evaluate the efficiency of services by service providers and to develop optimized applications to reduce energy consumption.

### Industrialization Development/Green Computing

The industrial effort is visible in the attempt to extend the European standard for green abstraction of non-virtualized environments [37] to virtualized environments. This second version of the standard [78] for a green abstraction layer aims to facilitate the development of software for automation of control and accounting functions. It is tightly integrated into ETSI’s group of specifications on management and orchestration of virtual functions, e.g. [97], adhering to the architecture described therein. Along with improving architecture for green computing, use of renewable energy source (RES) to operate cloud data centers is also proposed. Renewable energy generators using solar panels were installed at data centers located in different geographical locations [113]. These data centers were using a flexible load management tool EcoMultiCloud [114], that balance the load considering objectives such as energy cost variation and renewable energy production in the region at a particular time. Performance evaluations shows that use of RES in distributed data center network can save energy cost significantly without comprising on network performance, if smart management strategy is implemented.

#### 3.5.4 Macroscopic Analysis

We have referred to a complex characterization of workload (sec. 3.3.1) as a means to “quickly proceed to a good estimate of the power or energy required to produce the workload”. Such an approach abstracts real-time operation of a computing or telecommunications (sub-) system, by reducing the characterization of the power-performance relationship of its major components to a conservative ratio of power consumed to produce a unit of workload. Therefore, the aim of such research is not to facilitate real-time energy-aware control (through the development of models of real-time, dynamic power consumption) but to justify the use of an (alternative) architecture, organization or implementation. Two examples are described briefly below.

In [30], the concern regards whether a telecommunications system built around virtual network functions can readily be claimed more energy efficient than one built around physical network functions. A case study is undertaken with a focus on the evolved packet core's (EPC) serving gateway (SGW). Part of the modus operandi of this application of power modeling is explicitly declared in [30]:

1. a mathematical model, and
2. the energy consumed to meet
3. the performance levels demanded of the SGW.

The performance levels demanded are derived from the workload. Therefore, a profile of workload is required, to determine the performance of the system components over a temporal (and possibly geographical) range. The conjunction of a temporally unfolding workload and power consumed to meet this workload lead to the energy consumption. Hence, a figure of merit can be ascribed: the energy consumed to process the workload is an indicator of energy efficiency.

The missing piece in this puzzle is how to “fit” the workload into the investigated (sub-) system (this is the first phase) and therefrom, to its power consumption (this is the second phase). Herein lies the key abstraction inherent to the macroscopic analysis. In the first phase, the maximum throughput of a server bridges the gap between hardware and workload. The second phase transforms a performance level into a power consumption. In this case, the approach is relatively simple. The maximum throughput is bound, reasonably albeit sweepingly, to the maximum power consumption of the implementing device (here, a commodity-off-the-shelf server, or a chassis-bound SGW). We identify this two-phase transformation from workload to power consumption as a model in the highest of the three levels of abstraction into which we have classified power models earlier.

In [42], the telecommunications sub-system is a centralized- (cloud-) radio access network (C-RAN). The workload consists of a volume of baseband signal processing at the virtual baseband unit (vBBU). The baseband signal processing function is divided into four transmission steps (coding, modulation, mapping and multiple-input-multiple-output processing, followed by a Fast Fourier Transform); there are also the inverse four reception steps. Each step loads a single vBBU with a certain amount of a unit of operation: one million operations per time slot (1 MOPTS). This is the first phase particular case of the key abstraction: the computing hardware that performs any one of the four steps, is reduced to a capacity in terms of MOPTS. The second phase (transforms performance into a demand for power) maps the operation of the vBBU to a power consumption that is in part independent of baseband-signal-processing-load (“static”) and in part directly proportional to this load (“dynamic”).

Given these abstractions, and provided that realistic figures can be sourced to represent them, then the power demands of a load profile can be estimated. Here, the ultimate objective is not the consumption estimate itself but a comparison with alternative algorithms for dispatching the steps of baseband signal processing to the vBBUs in a pool.

## 3.6 The Engineer's Kit Bag

### 3.6.1 Observations on tools

#### Workload Generators/Benchmarks

Reliability of the studies, analyses and development of models is greatly dependent on the environment and workload considered. Setting up the real environment is relatively easy than finding appropriate workload. A well-suited benchmark suite that has the various stress ratio of different aspects, and have similar computing requirement as of real-time applications in data centers should be consider.

The Standard Performance Evaluation Corporation (SPEC) provides the number of benchmarks for different applications and performance evaluation. SPEC CPU benchmarks are industry standards and well accepted in research studies [115]. For cloud applications, available benchmark suite SPEC Cloud IaaS, stresses the resource provisioning, storage, network resources and computation. For emerging multi-core processors and parallel processing applications, PARSEC benchmark is developed which incorporates the workloads from multiple domains [61]. Another benchmark suite derived from NASA real fluid computational applications is NPB-MP, which is suitable for traditional HPC workloads [61].

Cloud providers and cloud sites were also evaluated using these benchmarks, in a study [116], BitCurrent bench mark was used to evaluate nine different cloud providers and CloudHarmonics was used to evaluate 144 different cloud sites. Another study [117] evaluate the cloud performance models and different cloud benchmark suites on Amazon EC2. Five widely used benchmarks used in [117] includes BenchCloud, this was developed in academia under USC and contains a workload from real-time social media applications for big data processing. Another academia based benchmark is CloudSuite that was developed under EPFL, Lusanne. The workload pattern for Cloudsuite is based on Media streaming, web services and Data/Graphic analytics. Two industry based workload pattern generated by Yahoo and Trans. Proc. Council are YCSB and TPC-W respectively. Where TPC-W is a webserver based benchmark and YCSB is a synthetic workload for cloud serving evaluation. HI Bench is a workload developed specifically to run Hadoop programs in clouds.

Selection of benchmark suite for evaluating modern processor must also consider the processor configurations such as multi-core or single core, SMT and Turbo Boost enabled, etc. Considering these requirements, workload pattern of different benchmarks also have different stressing capability such as serial stressing and paralleling stressing. The analysis of multi-core system using several instances of serial benchmark suite (such as SPEC CPU) is not preferred, as these instances will be treated as multiple applications [61]. It further increases the energy consumption due to unexpected contention for shared resources. Therefore, parallel benchmark suites such as NPB-MPI, PARSEC etc. should be used to analyze the performance of parallel processing.

#### Resource Monitoring Tools

Several methods of monitoring performance are available which either observe parameters inside the VM or through its host [6]. Methods that monitor the VM performance from outside a VM are known as black box methods. These use different software and hardware power meters to observe performance and resource usage. On the other hand,



white box methods run a proxy program that collects resource utilization data. However, these proxy programs can also impact measurements, which is not considered in many studies.

According to the survey in [6], it was found that the black box methods using software meters or hardware counters are more reliable. Moreover, they impose little overhead on the system, and their own effect is minimized. Different software tools of this kind, such as OProfile [118], PERF, sysstat [119] are available to collect data about different resources like memory access, storage and other performance metrics. Several power models have also been developed using these tools, with some of them accounting for the effect of multi-core processors in their model. Considering the available literature, Linux perf [120] is a widely-used performance-monitoring tool; this latter observation is confirmed in [121]. It is integrated in the kernel and hence provides deep insight of system behavior. Care must be exercised in selecting the number of counters to observe and the sampling frequency. Inappropriate selection can adversely affect the accuracy of observed values.

Software based meters are also used to monitor hardware resources and network traffic, to estimate power consumption of virtual machines [122]. Since these software meters run on the object of study, their own resource consumption must also be considered [43]. Moreover, algorithms and languages chosen for designing a software meter also have significant impact on power consumption [123]. This study finds that recursive algorithms are more energy efficient than the iterative ones. Furthermore, implementations of tools in C and C++ are the most energy efficient. Implementations in other languages consumed significantly more power. Perl consumes the most power, followed by OCaml, Python and Prolong. The Perl implementation consumes 25516 J, while the most energy efficient implementation (C++) consumes just 53 J (the C implementation consumes 54.5 J).

### 3.6.2 A brief methodology: observations on formal methods

We comment here on another compartment of the researcher's toolkit: a robust methodology of formal methods. We refer to methodology here in the classical sense of it being a study of alternative methods used for modeling, with the intention of picking the method most suited to the task at hand. Indeed, we have observed in the statistics that some research is seeking alternatives to linear regression as a means of fitting relationships between power and its determinants. Both linear and non-linear methods are being used to model power consumption of modern processors. The most common methods used to develop power models are linear models includes Least Square Estimation (LSE), Lasso, Mantis etc. [6]. We have confirmed this in our work here, where linear models are far more numerous than the rest. However, non-linear models such as Polynomial Regression, Multi-Gaussian Regression and Exponential Regression are also being used. The following are some studies that used different modeling methods to improve precision of server power models.

1. The dynamic behavior of traffic and heterogeneity of network in cloud computing and data centers cannot rely on power model based on particular workload behavior. This requires the use of reinforcement learning for continuous improvement of the model with the changing environment and network behavior. One approach uses an Elman Neural Network (ENN) is proposed to estimate the power consumed by

servers in cloud network [124]. The Elan Neural Network consists of input, output and state (instead of hidden layer) layers, where the state layer is a combination of several hidden layers with local feedback within these hidden layers. This local feedback helps model to learn the temporal pattern of the dataset for the specified time interval, which is neglected in other regression methods. Server resources such as CPU utilization, memory and disk usage, and I/O request rate are fed as inputs to the model which has the power of the system as the output. Two hyper-parameters, number of neurons in the state layer and set back time of Back Propagation Through Time (BPTT), are optimized for the training. When this model was used to estimate power consumption for two different servers, it showed better accuracy than existing Artificial Neural Networks (ANN) and regression models.

2. It is also evident from literature that servers with different configurations and multi-core functionality, may possess linear or non-linear behavior in their power consumption depending on the offered load. Hence different studies suggest both linear and non-linear regression methods to develop a power model for the server. One such study used Simple Linear Regression and Generalized Additive Models (GAM) to estimate the power consumption of the server under different load conditions [53]. RAPL counters were used to collect the training data, later the accuracy of power estimation for different servers was compared using an external power meter as the ground truth. The model was developed for memory intensive applications, using the memory statistics and power consumption as the predicted and response variable, respectively. Since the power profile of the server had some non-linearity, hence GAM predicted power of the server more precisely in non-linear region than the linear model.
3. It is challenging to predict real-time (runtime), optimal, task scheduling with Dynamic Voltage Frequency scaling (DVFS) and Dynamic Power Management (DPM) enabled. Hence machine learning based models are preferred as they can be trained and adapted. The Ordinary Least Square (OLS), Support Vector Machine (SVM), Artificial Neural Network (ANN), Naive Bayes, Multinomial Logistic Regression (MLR) are some of the many supervised learning methods that have been used in different environment to optimize runtime scheduling [125]. Considering the dynamic behavior of servers, different reinforcement learning (RL) methods are also being used. Reinforcement learning algorithms such as Q-learning, On-line distributed, Modular Q-learning, Back Propagation Neural Network (BPNN) and Multi-Level RL (MLRL) are mostly used to minimize the energy consumption of the server while continuously improving the model [125]. These reinforcement models have been found effective in power management of data centers by through consolidation and optimal VM allocation.

Architecture, system deployment, and workload patterns in computing environment are so diverse that one modeling approach and method cannot be identified as the universal solution for all circumstances. However, model selection diagnostic measures need to be used to assess the reliability of the selected model. The basic rule of thumb is to select the simplest model and only move to a more complex one if the desired accuracy is not met. The dataset itself contains information such as multicollinearity, number of outliers, size of dataset, non-linearity, etc. that points towards the right choice. The

nature of data points, that is whether the input variables are independent of each other or not, is one important factor. Analyzing the relationship between input and output variables is also useful to find an appropriate method. If the relationship between the two is continuous and linear, simple linear models could be effective. The size of dataset also plays a role. For some complex models, computation time can increase exponentially. Hence for big data analysis clustering, classification algorithms, frequent pattern mining are some techniques to divide the dataset into smaller set and make it feasible for data analysis [126]. Moreover, big data require traditional machine learning algorithms to be modified so they can work in parallel computing environments. Machine learning algorithms generally used for big data analysis are evolutionary algorithms, regression trees, and neural networks. Among these, the evolutionary algorithms such as genetic algorithm, swarm intelligence, and ant clustering are considered more robust methods as their sub-populations can run on parallel computing resource. Hence, these modeling methods reduce computation time when run on a GPU, instead of a CPU, on a parallel computing platform.

The ratio of size of dataset and number of input features is also important to assess before moving forward. Studies suggested that a single increment in number of input features will lead to an exponential growth in the dataset. With multiple input features, the dataset must be big enough to avoid the ‘curse of dimensionality’. Other options could be to reduce number of features based on statistical information such as Principal Component Analysis (PCA), probability value (p-value), correlation, stepwise regression, and vector quantization and mixture model [127], [128]. When the relationship among input and output variable is far more complex to identify using collected data points, non-parametric methods could be useful. Instead of learning pattern of given data points, these models generate different distributions to extract more useful information. An example of one such method is Gaussian Process Regression, that creates the Gaussian distribution for the set of selected data points.

With this we observed that no one method is perfect for modeling in all cases. Model accuracy is dependent on the right choice of modeling method, which can be made via thorough assessment of raw data and post processing methods available.

### 3.7 Conclusion

A recent study [129] found that, during the period 2010 – 2015, the power consumption by Information and Communication Technology (ICT) in various parts of the world was less than estimated. This is, at least in part, due to the use of energy efficient IoT communication modules and devices and in part to the effect of data centers and virtualization. This is encouraging and motivates further research into development of Green in IT.

This chapter highlights the current state-of-art of the power measuring models and methods at the hardware and virtual level. Energy-efficient data centers and virtualization depend on modeling that is fit for purpose. Such models support the power-performance trade-off in single as well as distributed systems. In turn, modeling depends on power measurement. Power models and meters available to measure the resource consumption of the system are based on different parameters (e.g. architectural and microarchitectural) and approaches (e.g. resource isolation, resource management, workload adaptation). Many power models are available but their accuracy under advanced microarchitectures

and system organization is questionable. Multi-processor CPUs make it challenging to estimate the power consumption of the system precisely, further increasing complexity at the finer granularity levels such as that of cores.

Moreover, use of emerging virtualization has several benefits such as running different processes or applications on a single server, isolation among VMs and reduced power consumption but still power metering for virtual machines is not yet available. For power estimation in a virtual network, extant power models for hardware components cannot be used, since resource mapping and its consumption at the virtual level differ. Software and hardware power meters are being used to measure the power consumption of virtual components, but observation and estimation of a single virtual component is still an open issue. Power models for virtual networks are also available but there is room for improvement in accuracy. Further precision in these networks can be obtained by introducing power- and energy-aware network management. Moreover, excessive training algorithms are being used to develop a more general power model for virtual machines. These model can be useful to estimate the power for even untrained input data set. Power-aware network management can also help in reducing the growing issue of  $CO_2$  emission. Thus, measurement of accurate power consumption for virtual machines can serve as the basis for fair cost to its users and will also facilitate green computing.

# 4 Improved Power Modeling in Virtualized Environments

## 4.1 Introduction

The growing use of virtual resources either in private or public data centers or cloud services has drastically increased the power usage and, hence, the cost of their operation and management. To reduce the power consumption at data centers, research has been carried out at infrastructure, software and hardware level. Researchers are developing more reliable power models along with power-aware allocation and scheduling schemes for optimal power usage under different scenarios. And also for trading among power consumption and performance in terms of computational effectiveness and processing delay, among other indicators. It is also important in these infrastructures to charge customers, availing cloud services, with a fair amount for the resources they consume. And this could be made possible by more accurate power measurements.

To effectively improve the efficiency of any system, it is necessary to be aware of the refined relation between system resource usage and its power consumption. This will help in making more informed decisions to improve energy efficiency and such awareness can be achieved through a comprehensive and fine granularity level of system monitoring. As discussed in the previous chapter, monitoring can be done using either software power meter or hardware power meter. Further to incorporate energy awareness in decision making, power models are required to estimate the server power at run time.

The chapter below further discusses research questions tackled in this research work. Later on, the chapter proceeds by describing approaches and tools that are used in an attempt to derive the power model. The proposed methods and models are also presented, and their performance and precision are discussed later on.

## 4.2 Motivation

### 4.2.1 Research Questions

This chapter describes and develops the power model for the server in a virtual environment considering the detailed monitoring of resources at both guest and host levels for precision. This improved prediction accuracy could also be useful to make an effective energy-aware decision and to develop an improved cost model for cloud service availing clients. To capture details regarding resource consumption and their corresponding significant metrics, this research proposes some questions which could be beneficial in determining and justifying the requirement of the proposed power model. The questions are as following,

1. How does the behavior of server power consumption changes in a virtual environment? If it differs in various situations, what are the factors causing this variation and relationship among them?

2. What is the significance of using related features and performance counters based on different kinds of stress/load or applications?
3. How the system-wide effect can be captured using some effective and significant features in a virtualized environment, such that it provides sufficient information for power modeling?
4. How the multi-core and core sharing in modern processors effect the power consumption?
5. What is the impact of selected method of modeling on the prediction accuracy of the power model?

This work aims to derive, via system performance monitoring, a power model for run-time power estimation of a server in a virtual environment. Also, to identify the change of performance at the guest and host level which could help in load balancing and resource optimization through two-levels monitoring.

### 4.3 In-context

#### 4.3.1 Performance Monitoring Counters (PMCs)

As the development continues into virtual environments, the need of tools and methods to analyze their performance characteristics grows. At hardware level, many processor architectures have now integrated hardware counters to observe the system behavior [130]. Some of these counters can also be exposed at the guest level with the support of hypervisors, and are called virtual Performance Monitoring Counters (vPMCs) [131]. One such hypervisor providing the facility of vPMCs is KVM (Kernel-based Virtual Machine). The PMCs are located in the Performance Monitoring Unit (PMU) of Intel and AMD processors. On physical host, these counters are easily accessible, but are inherently not exposed at the guest level due to security concern. However, monitoring of these counters at the guest level is possible by enabling the vPMCs at the host. This feature allows software running inside virtual machines to access this performance information, just as it would when running on a physical machine. Different processor architectures provide visibility of different vPMCs to the guest. PMCs are actually the Model Specific Registers (MSR), which keep on counting different microarchitectural events. An interrupt is raised if the counter overflows by reaching the defined number of maximum counts. Some of the most common events available in most of the processors include CPU cycle count, instruction count, CPU clock, translation lookaside buffer (TLB) accesses and misses, branch retired count and misses, and cache misses. Studies show that PMCs can be used for effective runtime optimization. By leveraging these counters, the Operating System (OS) is capable of making resource-aware scheduling decisions.

#### 4.3.2 PERF - a Linux tool for profiling

There are many tools available to collect the information from these counters. The type of tool can be selected depending upon the sampling frequency and granularity level of information required. In this study, PERF [120], a profiling tool for Linux based system is chosen. PERF is a non-invasive performance analyzing tool in Linux. This userspace

controlling utility is capable of statistical profiling of the entire system. It is known by different names as PERF, Linux perf event, and Performance Counter for Linux (PCL). It provides access to the Performance Monitoring Unit (PMU) in kernel space, and thus provides a low-level insight to the system behavior considering events happening and resource usage. Fig. 4.1 below shows the different sources of data collection in PERF,

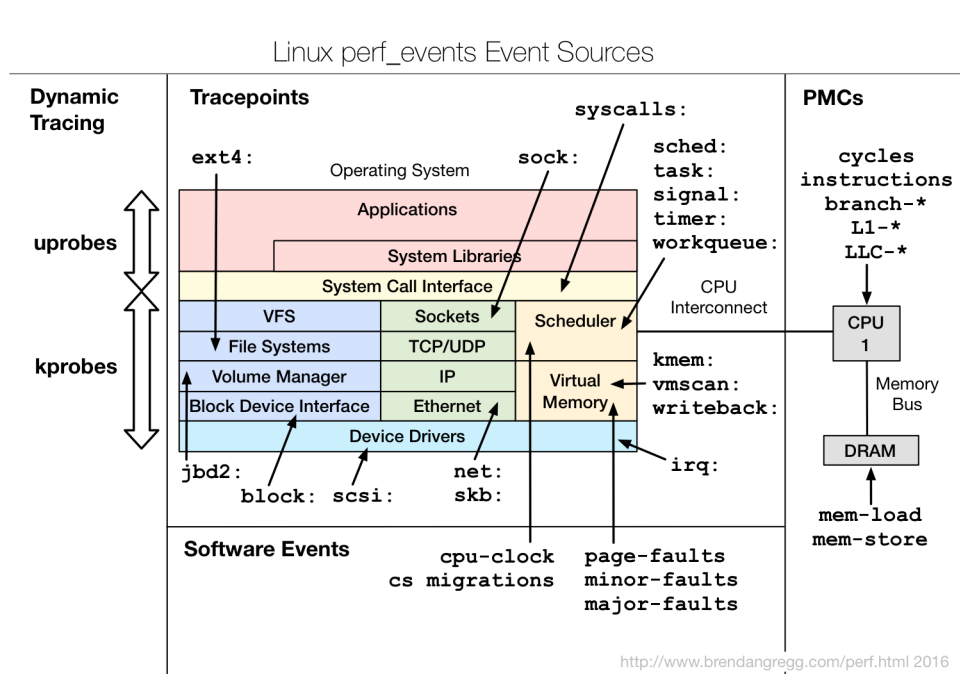


Fig. 4.1: Map of sources of PERF events [120].

The Linux PERF tool monitors the hardware and software counters in the CPU and generates the report of the collected data. A list of available events can be obtained by running the 'list' command. A sample output of running this command is presented in fig. 4.2.

From fig. 4.2 we find out that the particular system is able to collect counter values for branch-instructions, bus cycles, branch-misses, cpu-cycles, instructions, and many others. The counters collected in this research are low level counters such as CPU cycles, instruction, page faults, etc., instrumented using the perf 'stat' command. PERF, when run with the 'stat' command, collects the statistics for the operations executed in the command. An example output of a perf 'stat' command is shown in fig. 4.3.

The command in fig. 4.3 runs perf stat to collect system-wide counter statistics for 5 seconds. Hundreds of different PMCs are available; however, any processor can record only few of them at a time. This is due to the limited processor capacity and also to avoid the frequent interrupts generated while collecting counter values. The amount of time taken by the tool for data sampling is its 'overhead'. The Linux PERF tool is selected in this thesis due to its very little overhead and less impact on performance. Studies show that PERF incurs 5%(or less) overhead for the sampling frequency of 100,000Hz, which is way smaller than the other profilers and is acceptable in many cases. However, sampling frequency is an important factor for causing overhead; hence, it must be chosen wisely. PERF provides far more information than other profilers with minimum overhead, and it supports both hardware events (branch miss, instruction counts) and software events

```

humaira@humaira-ThinkPad-W530:~$ perf list
List of pre-defined events (to be used in -e):

branch-instructions OR cpu/branch-instructions/ [Kernel PMU event]
branch-misses OR cpu/branch-misses/ [Kernel PMU event]
bus-cycles OR cpu/bus-cycles/ [Kernel PMU event]
cache-misses OR cpu/cache-misses/ [Kernel PMU event]
cache-references OR cpu/cache-references/ [Kernel PMU event]
cpu-cycles OR cpu/cpu-cycles/ [Kernel PMU event]
cstate_core/c3-residency/ [Kernel PMU event]
cstate_core/c6-residency/ [Kernel PMU event]
cstate_core/c7-residency/ [Kernel PMU event]
cstate_pkg/c2-residency/ [Kernel PMU event]
cstate_pkg/c3-residency/ [Kernel PMU event]
cstate_pkg/c6-residency/ [Kernel PMU event]
cstate_pkg/c7-residency/ [Kernel PMU event]
instructions OR cpu/instructions/ [Kernel PMU event]
mem-loads OR cpu/mem-loads/ [Kernel PMU event]
mem-stores OR cpu/mem-stores/ [Kernel PMU event]
msr/aperf/ [Kernel PMU event]
msr/mperf/ [Kernel PMU event]
msr/smi/ [Kernel PMU event]
msr/tsc/ [Kernel PMU event]
power/energy-cores/ [Kernel PMU event]
power/energy-gpu/ [Kernel PMU event]
power/energy-pkg/ [Kernel PMU event]
ref-cycles OR cpu/ref-cycles/ [Kernel PMU event]
stalled-cycles-frontend OR cpu/stalled-cycles-frontend/ [Kernel PMU event]
topdown-fetch-bubbles OR cpu/topdown-fetch-bubbles/ [Kernel PMU event]

```

Fig. 4.2: A sample output of 'PERF List' command

(context-switches, page misses).

### 4.3.3 Stress Generator

The stress or workload generator is a tool that can run different programs, mimicking the real workload [132]. These tools are useful in research when the user knows the characteristics of its application or workload, and can use these tools to generate related load. These tools provides a variety of stress programs for different kinds of applications. The stress generator used in this thesis to produce load on VMs is stress-ng. This is a command-line tool and is designed to stress the system in different ways. It has a wide range of stress programs to stress computer systems with computation of floating point, integer, bit manipulation and control flow. The user can select the type of stress, such as CPU, memory, and can also change the intensity level to stress these resources. For CPU-intensive stressing, the number of CPUs to stress and their utilization percentage can be chosen using the simple stress command. The selected stress methods with integer and floating point operations runs on the VM in sequence. For these experiments only CPU-intensive workload is considered, where the workload is increased by increasing the virtual CPU (vCPU) utilization on the VM. The intensity of stress is increased by increasing the vCPU utilization percentage from 0 to 100% with step of 10 units. A sample command to create stress on a single CPU core with 50% CPU utilization for 30 seconds is,

```
$ stress-ng -c 1 -l 50 -t 30
```



```

humaira@humaira-ThinkPad-W530:~$ sudo perf stat -a sleep 5

Performance counter stats for 'system wide':

   40008,806856      cpu-clock (msec)          #    7,998 CPUs utilized
         8.401      context-switches        #    0,210 K/sec
          197       cpu-migrations         #    0,005 K/sec
          188       page-faults            #    0,005 K/sec
   777.726.449      cycles                   #    0,019 GHz
  1.308.117.681    stalled-cycles-frontend  # 168,20% frontend cycles idle
   242.363.685     instructions             #    0,31 insn per cycle
                                     #    5,40 stalled cycles per insn
   50.273.829      branches                 #    1,257 M/sec
   3.336.571      branch-misses            #    6,64% of all branches

5,002043849 seconds time elapsed

```

Fig. 4.3: A sample output of 'PERF stat' command

#### 4.3.4 Power Distribution Unit (PDU)

A power meter is a hardware device that reports the information about the power consumption, in watts. In contrast, a Power Distribution Unit (PDU), is a device used to distribute electric power to the system, more specifically to the racks of computers and networking equipment, and to the servers in data centers. These PDUs monitor the power consumption of the system and manage power resources accordingly. They are connected to the power socket of the system, and are used for power capping and sensing system's power consumption for various other controlling options. These PDUs can also be used as a power measurement device, to measure the power value for respective load on the server, hence are useful for power profiling of the server. As discussed in chapter 3, the power measurement of servers can be done either internally using integrated sensors and power models, or externally through separate power meters such as power distribution units (PDUs) for data centers. In most cases, the power measured using external power meters is considered more accurate than that stemming from mathematically derived and developed power models. This is because an external hardware meter does not require any hardware changes on the system or running software on the device, which reduces the overhead and the influence of the measuring device on the measurement values. Usually, high-end PDUs offer  $\pm 1\%$  of error and 0.1 watt precision (e.g. Raritan Dominion PX-5367) [133]. Hence, the validity of the derived power model in research must be compared with the external hardware power meter. Sampling rate is also important while collecting data from PDUs, as with small sampling rate, surges in power consumption can be missed, and on contrary high sampling rates would incur extra overhead on the system performance. This research work also uses the PDU for system-wide power profiling and its modeling.

## 4.4 Proposed Modeling Approach

Accurate measurement of energy consumption during an application execution is a key to energy minimization techniques at the software level. There are three popular approaches for this: (a) System-level physical measurements using external power meters, (b) Measurements using on-chip power sensors and (c) Energy predictive models. Data collected through (a) and (b) can be used to construct a real-time power estimation model

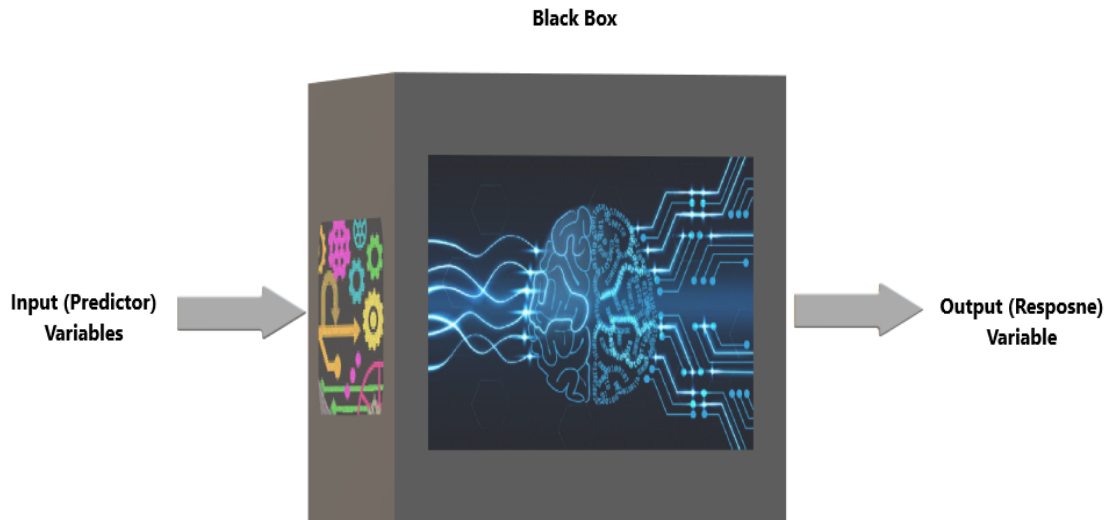


Fig. 4.4: Black-box modeling

such as (c). This power modeling can be accomplished generally in two ways: detailed analytical modeling and high-level black-box modeling. For analytical modeling, detailed knowledge of hardware components is required, but there are generally limitations in generality and portability. On the other hand, black-box modeling is feasible for scalable and dynamic networks, and are flexible to modify for different servers and environments. Hence, to avoid the complexity of analytical modeling, this research focuses on the black-box modeling method. The black-box modeling is a learning-based approach where the collected data is trained through the selected method. These models are not aware of the internal working of the system or about the relation among different system variables, instead, they are representative of the functional relationship between system inputs and outputs. In particular, black-box models are derived from the observed experimental data, which has the capability to define the system behavior with sufficient accuracy. Although using black-box modeling might sacrifice some accuracy, but it favors simplicity by avoiding reliance on detailed knowledge of the hardware's implementation. In a virtualized environment, the identification of the relationship among different mutually interacting variables increases the complexity of modeling. Black-box modeling with little architecture and configuration knowledge could be useful in such a scenario.

Sensor-based counters integrated into system architecture are used to monitor the system performance. Studies [6], [109], [134] show that using built-in performance counters are a reasonable and cost-effective choice for power and performance modeling; therefore this work uses PMCs for collecting resource usage information. Also, it is evident from literature that performance monitoring at the host as a whole is not sufficient in virtual environment; therefore, the observation of virtual performance counters (vPMCs) at the guest-level is also introduced. We did two-level (host and guest levels) resource monitoring to analyze how the system behavior changes in two environments. Performance monitoring at two levels and the selection of effective performance counters could help in developing an improved power model.

Many hardware and software counters are available to monitor several events and resource usage; however, using too many performance counters simultaneously does not

necessarily provide accurate counts. The selection of effective performance counters is therefore important. Since different kinds of workloads have different resource usage, this work is concentrated on CPU-intensive workload, as it is a dominant contributor to server power consumption. After analyzing the effect of all observed counters, non-significant counter values can be neglected before modeling. Only a few significant counters are valuable for modeling; otherwise, the higher number of counters with the limited amount of dataset could cause the 'curse of dimensionality', since with the growing number of inputs the size of the required dataset to be trained grows exponentially. Hence, the ratio of a number of input features and dataset size must be appropriate to avoid complexity and retain model performance. Datasets collected in this research consist of around 1600 data points for several counters. After analyzing observed data, the number of input features is later reduced to avoid unnecessary complexity in the model. Moreover, the monitoring of performance counters and measurement of power consumption using the PDU is done simultaneously, for different load intensities and host configurations. These measured values of power and performance were then used to extract the power profile of the server under different configurations. This system profiling is a powerful tool that is meant to derive the power model in this work.

Moreover, use of machine learning algorithms in recent modeling methods is also increasing. These models are feasible for large and heterogeneous systems; hence, they are useful for power and performance modeling in virtual environments, as well [135]. This could also reduce the model complexity for large virtualized systems [136], [137]. Hence, the power consumption of the server as a function of workload and other parameters might be better represented using regression models, depending upon the active number of cores, CPU utilization, and model selection. Several linear and non-linear methods are available that can be selected for an appropriate modeling. This model selection considers the type of processor architecture, system behavior and load on the system. This is an important step, as the selection of complex non-linear models for a simple system might incur extra overheads. On the other hand, a simple linear model for a complex resource-sharing system could result in inefficient modeling. As the power behavior of the system varies from linearity to non-linearity based on its load and environment, both kinds of models are used in this study. In many cases, non-linear models can provide more accurate results, but they are also more complex than linear ones. Hence, there will always be a trade-off between computational complexity and accuracy for these models. Further, to evaluate the accuracy of models using the proposed approach, estimated power values of the models are compared with the external connected PDU, which has the precision of about 0.1 W, with a low internal power consumption of about 5 W to 7 W.

The section below discusses how the training dataset is generated for modeling and which methods are used in the proposed approach for modeling.

## 4.5 Training Dataset

The training dataset and the training methods are the essence of modeling. For a good regression, identification and using of effective features is very important. According to [138], micro-operations fetched per cycle or instruction counts could be useful for modeling by using regression techniques, for CPU-intensive workload. Also, [139] presents some effective counters for CPU-intensive load including instruction count, CPU cycles,

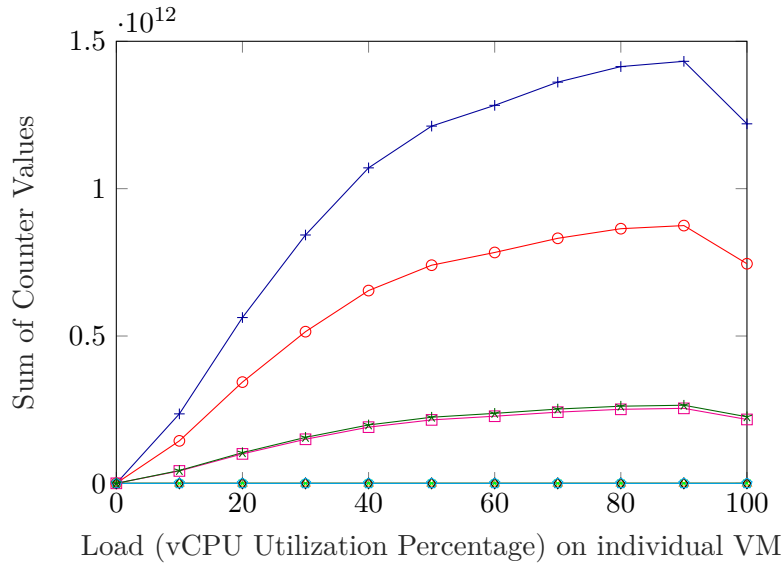


Fig. 4.5: Sum of counter values for all running (08) VMs at different load. Legends: —+— Instruction Counts, —o— CPU cycle, —\*— CPU clock, —x— Stalled Instructions, —x— Page Faults, —o— Context Switches, —o— CPU Migration, —\*— Branches

branch count and misses, cache misses, and stalled instructions. Since our research is targeting attributed power consumption by virtual components, associated virtual memory counters as a page fault, CPU migration major fault and minor faults were also considered [120]. Several counters were observed in this research for different CPU workload intensity, and later the effect on all counter values with respect to the changing load was analyzed. Although many counter values vary accordingly, only counters which have significant impact are considered. In most researches, a number of variables are collected but only few are used for modeling. This is because more number of features in modeling can cause complexity and reduce accuracy (if data the set is small). Also, only significant features should be extracted to have a simple model. This feature reduction can be done in several ways; one way of doing this is using p-value that indicates the significance of relationship among variables [127]. Features having higher p-value have no significance and can be removed from the training dataset. Another method of feature selection could be adjusted R-squared and predicted R-squared. The value of R-squared of the model shows how well the model is explaining the data [127]. One can also use the stepwise regression and select the best subset of features for modeling. In the stepwise method, different features are selected and used to train the model. The model giving minimum error for the set of features is selected. This method is used in this research, where the power modeling is done using different sets of input features. A set of features is selected by analyzing their variance at different load intensities. Fig. 4.5 shows the change in different counter values for different load across all running VMs.

As we can see from fig. 4.5, the most prominent variation is observed in instruction counts, and the CPU cycles constitute the second most dominant feature. The number of stalled cycles and branch counts are next to show increase with increasing load. So, we neglect the other non-effective counters in modeling for this research to avoid complexity. Only the most effective counters i.e. stalled instructions, total instructions and CPU cycles at the host and guest level are used in the approach this thesis propose.

The number of Instructions Per Second (IPS) is an approximate indicator of the likely performance of a load and is obtained considering the total instructions processed (including stalled instructions and all retired instructions). Where stalled instructions are the instruction cycles that are wasted due to operations such as cache miss, and retired instructions are the actual cycles that process the request.

Instructions per second (IPS) can be calculated by using eq. (4.1) or eq. (4.2) below,

$$IPS = \frac{\text{total instructions retired} + \text{stalled instructions}}{\text{total time}} \quad (4.1)$$

$$IPS = \frac{\text{instructions} \times \text{cores} \times \text{cpuclock}}{\text{cpu cycles}} \quad (4.2)$$

The IPS of virtual machines and the host are fed as the input feature (predictor variable) for training. The power measured for the same instant using the external PDU will be treated as the response variable. To the best of the authors' knowledge, performance monitoring of individual VMs (using vPMCs) has not been done earlier for power modeling. This novel approach has the effect of reducing the prediction error and providing solutions to develop power models for even larger systems by using machine learning methods.

## 4.6 Training Models

The selection of regression technique is another challenge in machine learning. Number of independent variables, type of dependent variables and shape of the regression line are some parameters that can be used to select an appropriate regression method. A linear model of the power consumption of processors can be used for relatively small CPU utilization; however, as the CPU utilization increases, power consumption tends to be better approximated by a non-linear model. The effect is more significant especially in modern processors with multi-cores and the tendency to share cores among different processes. Considering this behavior, both linear and non-linear modeling techniques are used in this research.

### 4.6.1 Linear Least Squares Regression (LLSR)

When the nature of dependent variable is continuous and the regression line for input and output variables is linear, then the most simplest and widely used Linear Least Square Regression (LLSR) can be a good choice. LLSR model is the simplest and most commonly used predictive analysis model that represents the linear relationship between a single dependent variable  $y$  and a single (or many) independent variable(s)  $\mathbf{x} = [x_1, x_2, \dots, x_n]$  [140]. Instead of using absolute difference values, LLSR uses the square values of the differences between measured and predicted points. The sum of square distances between the data points and predicted values should be minimum to get the best approximate value from the model. A simple LLSR model with single dependent and single independent variable can be written as eq. (4.3),

$$y = \beta_0 + \beta_1 x + \epsilon \quad (4.3)$$

where  $\beta_0$  is a constant,  $\beta_1$  relates the change in predictor variable  $x$  to the mean of the response variable  $y$ , and  $\epsilon$  is the random noise error. The output function of this model is the best fitted line for the given data points.

#### 4.6.2 Linear Support Vector Regression (LSVR)

The Support Vector Machine (SVM) is a supervised learning algorithm which is being used for classification and regression in datasets. SVM is useful for datasets where prediction of response is difficult due to random system response [141]. In simple words, it is commonly used for data where the output variables are not easily separable in one dimension. Since we use different combinations of load on a number of VMs in our experiments, this may generate similar number of MIPS (Million Instructions Per Second) for different experiments. So, the effect of VM instructions on the physical device and consequently on the power consumption of the server is analyzed using the Support Vector Regression (SVR). This would help in distinguishing among different environments for similar output. SVR works with the same mechanism as SVM and projects the given one dimension data onto a higher dimensional plane. This makes the identification and classification of data points easier.

To study in depth the effect of one variable on the other, SVM maps data points of the training set into multi-dimensional feature space using some fixed mapping. The hidden relation among these data at higher dimension space is then determined and a model based on kernel function is constructed in this feature space. Different kinds of kernels, linear or nonlinear, can be used depending upon the nature of data points. The so-called primal formula for linear SVM to derive the vector  $\beta$  of the coefficients of the linear regression model through (constrained) minimization is,

$$f(\beta) = \frac{1}{2}\beta'\beta + C \sum_{n=1}^N (\epsilon_n + \epsilon_n^*) \quad (4.4)$$

where prime denotes transpose. The quality of the estimated values is measured with a loss function, for which slack variables are introduced. The slack variables  $\epsilon_n$  and  $\epsilon_n^*$  defines the allowed range of error. The data points outside this range are penalized. This penalty in the model is determined by the constant  $C$ , a positive numeric value that controls the penalty imposed on observations that lie outside the  $\epsilon$  margin.

#### 4.6.3 Quadratic Support Vector Regression (QSVR)

As we know from the literature that modern processors with core sharing capability may have non-linearity in dynamic power consumption, we also use some non-linear models to train our dataset. One non-linear model used in this research is Quadratic SVR [142], where the kernel function for the SVR is quadratic instead of linear. When the relation between dependent and independent variables in feature space is non-linear, non-linear kernel functions provide more precise estimation. In QSVR a non-linear kernel constructs the hyperplane using a quadratic function to map the data points. This can be expressed as eq. (4.5),

$$f(\beta) = \frac{1}{2}(\beta'\beta + 1)^2 + C \sum_{n=1}^N (\epsilon_n + \epsilon_n^*) \quad (4.5)$$

The quality of the model in SVM is measured with the distance of the estimated value to the hyperplane, using slack variables. These slack variables  $\epsilon_n$  and  $\epsilon_n^*$  define the allowed range of error across the hyperplane. The data points outside this range are penalized by a constant,  $C$ , a positive numeric value.

#### 4.6.4 Gaussian Process Regression (GPR)

In some complex architecture, it is difficult for many methods to model correctly or to identify incorrect modeled points. In such scenario, a model is required which could self-test its validity and be able to correct its response. One such model is Gaussian Process Regression (GPR), which is a non-parametric approach to regression. GPR can work well on small data sets as well and has the ability to predict precisely. GPR does not learn the given data points of training data, but it rather infers a probability distribution over all possible values. GPR also has the ability to distinguish among output variables generated in different environments. The Gaussian process based on prior information keeps improving the posterior information, and model the output of the given dataset only when it is completely certain about it. Because of this, GPR is also computationally expensive and its complexity grows with the size of the data set.

The Gaussian Process (GP) is a collection of multivariate Gaussian distributions, where each distribution is defined by a finite subset of data points [143]. GP can be formally defined by the mean and covariance (or kernel) function of all Gaussian distributions of data points. The kernel function of the GP is however parameterized using parameters such as scale length to normalize all data points and standard deviation in the dataset, as shown in eq. (4.6). The covariance function shows the relation of neighboring input values to their corresponding output value. This covariance function, is selected as to make the Gaussian distribution for each dataset smoother. A 'Squared Exponential' function is used as the kernel for GPR in this research, expressed mathematically in eq. (4.7).

$$f(x) = GP(m(x), K(x, x')) \quad (4.6)$$

$$K(x, x'|\theta) = \sigma_f^2 \exp\left[-\frac{1}{2} \frac{(x_i - x_j)^T (x_i - x_j)}{\sigma_l^2}\right] \quad (4.7)$$

where prime denote transpose,  $x = x_1, x_2, \dots, x_n$  is the set of  $n$  data points,  $m$  and  $K$  are the mean and covariance function of the GP,  $\theta$  is the hyperparameter, which consist of scale length  $\sigma_l$  and standard deviation in data points  $\sigma_f$ .

For LLSR computations for 100 and 1000 data points will remain same, whereas in GPR number of computation for such case increases significantly. This is because for each set of data point the Gaussian distribution is calculated for modeling and prediction, which becomes more complex with large amount of data.

## 4.7 Performance Measures

Different linear and non-linear models are evaluated based on their prediction accuracy and model complexity. Two error metrics, root mean square error (RMSE) and mean

absolute error (MAE) are used, along with their percentage, and can be calculated as in eq. (4.8) and eq. (4.9):

$$RMSE = \sqrt{\frac{\sum_{i=1}^N (y_{i_{pred}} - y_i)^2}{N}} \quad (4.8)$$

$$MAE = \left| \frac{\sum_{i=1}^N (y_{i_{pred}} - y_i)}{N} \right| \quad (4.9)$$

where  $y_i$  is the measured power,  $y_{i_{pred}}$  is the estimated power and  $N$  is the total number of samples.

While training, the reliability of the model is achieved using cross-validation, which helps in generalization of the model for new data points. In cross-validation the training data set is divided into two subsets (train and test), one for training and other for testing. For k-fold cross-validation, the dataset is divided and trained k times. The model developed is based on the summarized evaluation of the results of each  $K - th$  set. However, there is a bias-variance trade-off associated with the choice of k in k-fold cross-validation. Typically, given these considerations, one performs k-fold cross-validation using  $k = 5$  or  $k = 10$ , as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance [144]. Thus in this research work, 5-fold cross-validation is used.

Furthermore, model complexity is evaluated considering the training time and prediction speed for different models. Training time is the time that model takes to train the whole data set and is measured in seconds. The model's prediction speed is defined as the number of predictions a model can make for a given input in one second, hence represented in units of observations/second (obs/s).

## 4.8 Power Modeling

To develop a power model, the study was carried out in two phases. In the first phase, the strategy to train the model is selected by implementing different modeling approaches. Later, the chosen modeling approach is further analyzed in depth to answer the research question of this work.

### 4.8.1 First Phase: Methodology Selection

The instruction count per second for a VM can change by increasing the workload on a similar virtual machine, and also if other virtual machines are running in parallel. Hence, the approach of measuring instruction counts at VM level helps in obtaining the detailed insight of VM performance and corresponding resource usage. Likewise, monitoring resource counters for individual VM at the physical level helps in identifying the relation between resource requested (processed) by the VM and its corresponding physical resource consumption. This rises the question whether power modeling by considering the resource usage at guest level could produce better results, or the model requires the status of consumed resources at host level, as well. Hence, to create a better model from the collected data, initially some hypotheses were tested to identify which approach and what features are effective.



## Hypotheses

Below mentioned are some hypotheses that were tested in this phase of the experiment.

- **H1:** Training the power model with the resources monitored at VM level can provide acceptable results;
- **H2:** Modeling in two steps, mapping virtual MIPS to physical MIPS, then mapping physical MIPS to the power consumed could provide a more precise estimation.

To test the above mentioned hypotheses, an experimental test-bed was setup at the CNIT National Laboratory of Smart and Secure Networks (S2N) in Genoa, and experiments were carried out at a small scale.

During the data collection phase of the experiments, vPMCs for each VM were monitored individually at VM level. Instruction changes for a particular VM were also monitored at the physical machine at the same time. The change in instructions' rate at both levels is calculated by using Eqs. (4.10) and (4.11), where  $i$  is the number of VMs running and  $j$  represents the vCPU utilization (or workload on the VM). Furthermore, the power consumption of the server is obtained using the PDU, and the change in power is obtained using Eq. (4.12).

$$\Delta\text{MIPS}_{\text{VM}_i} = \text{MIPS}_{\text{VM}_i,j} - \text{MIPS}_{\text{VM}_i,\text{idle}} \quad (4.10)$$

$$\Delta\text{MIPS}_{\text{server}} = \text{MIPS}_{\text{server},j} - \text{MIPS}_{\text{server},\text{idle}} \quad (4.11)$$

$$\Delta P_{\text{server}} = P_{\text{server},j} - P_{\text{server},\text{idle}} \quad (4.12)$$

where,  $i \in \{1, 2\}$  and  $j \in \{10, 20, \dots, 100\}$ .

To determine the change in power and instructions, idle values for both variables were also obtained. Idle server power is found to be 17 W for the machine whose characteristics are reported in tab. 4.1, and MIPS at the server when idle were 0.4 MIPS. Also, idle MIPS observed at each VM were 0.5 MIPS on average.

Two different modeling methods were used for the datasets, which will be referred to as Model1 and Model2 from now on.

**Model1:** In Model1, a dataset containing the instruction count collected at the VM level is used. The sum of the increase in instruction rates from idle level of all VMs running the specific experiment,  $\sum_i \Delta\text{MIPS}_{\text{VM}_i}$ , will be fed as the predictor. The response variable for Model1 is the change in power at the server corresponding to the varying load. Hence, this model comprises of one step modeling, and maps VM MIPS to their corresponding power consumption.

**Model2:** For training of Model2, the model is divided into two sub-models, which will be referred to as Model 2a and Model 2b. In model 2a, the relation between  $\sum_i \Delta\text{MIPS}_{\text{VM}_i}$  and the  $\Delta\text{MIPS}_{\text{server}}$  is determined, whereas, in model 2b, the server power change,  $\Delta P_{\text{server}}$ , is modeled corresponding to the  $\Delta\text{MIPS}_{\text{server}}$ .

Fig. 4.6 illustrates the considered modeling approaches. Both models are trained using linear regression models, LLSR and LSVR. For blind test validation of trained models, measured data is divided into training and testing datasets, with 80% and 20% of data, respectively.

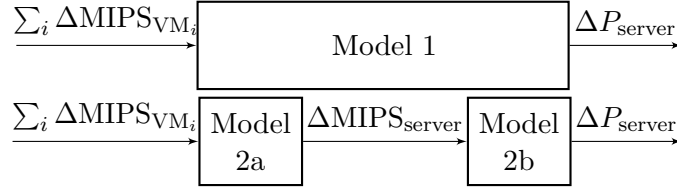


Fig. 4.6: Illustration of the two strategies to model power change from the change in virtual machine MIPS.

Tab. 4.1: System configuration for the Server and VM

	Server		VM
Processor	Intel(R) Core(TM) i7-6770HQ	CPU	Intel(R) Core(TM) i7-6770HQ CPU
No. of CPUs		8	1
OS		Linux 4.9.0-3-amd64	Linux 4.4.0-134-generic
Cache		6144KB	16384KB
RAM		32GB	2GB
Hypervisor		Nova (qemu-KVM)	-

### Genova Test Bed

The experimental test-bed has been set up using the OpenStack software. The “Intel(R) Core(TM) i7-6770HQ” CPU is used as the server, whereas “compute (NOVA)” [145] serves as the hypervisor. The server has two VMs running, both with similar configuration. Further configuration details about the server and VMs are listed in Tab. 4.1.

The program “stress-ng” is used to generate the workload on VMs. The selected stress methods with integer and floating point operations runs on the VMs in sequence. For these experiments only CPU-intensive workload was considered, where the workload is increased by increasing the virtual CPU (vCPU) utilization on the VM. In each experiment workload on a VM is increased by 10% of virtual CPU utilization and goes up to 100%. The power behavior of the server is also observed in different scenarios, such as no load, single VM, two VMs. The virtual machine instructions processed by a specific VM are measured using the Linux PERF tool inside the VM. MIPS of each VM are also monitored at the server level using the same tool. At the same time dynamic power of server is monitored remotely using the external Power Distribution Unit “Raritan JSON-RPC(PX2)”.

### Results and Discussion

Instructions measured at the VM by changing its workload are presented in fig. 4.7 (left). It shows the increase in number of instructions at the VM with increasing load on its vCPU. The number of instructions further grows by increasing the number of VMs running on the server. As can be observed in fig. 4.7 (left), the processor needs to process a maximum of around 3000 million instructions per second (MIPS) to reach the 100% load on the vCPU for a single VM. However, a similar server when running 2 VMs at 100% of their vCPU utilization requires more than 5000 million instructions to process per second. This increase in processor speed is obvious, as with two VMs the processor

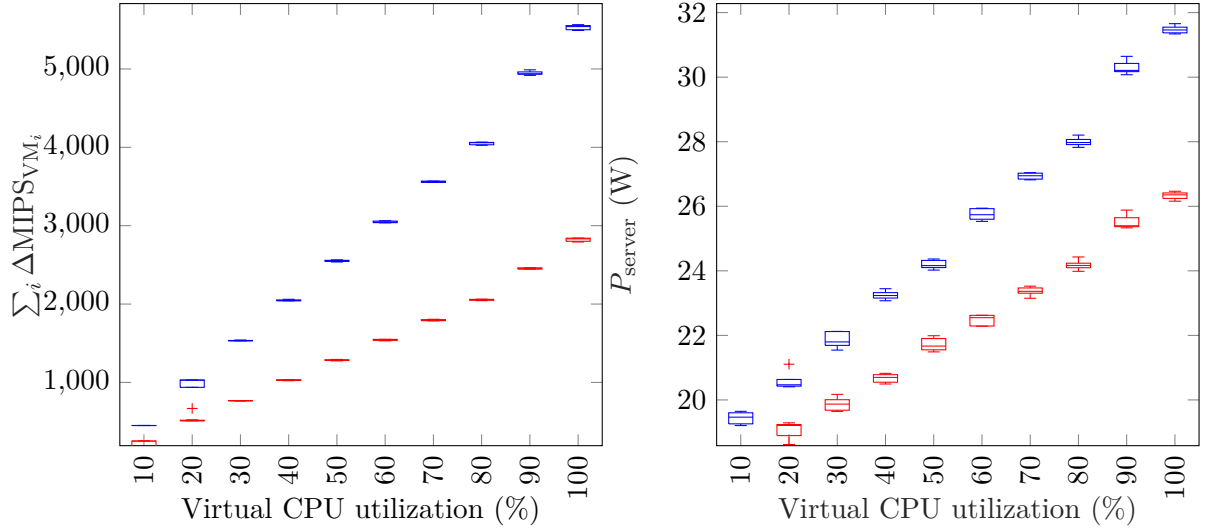


Fig. 4.7: Left: Increasing the vCPU utilization via a stress program leads to an increase in virtual machine MIPS. Right: The same increased vCPU utilization also increases the power consumption at the server running the VM. Legend: Only 1 VM with load (—), 2 VMs in parallel (—)

needs to process the load of two CPU cores. Moreover, when a processor executes more instructions per cycle, it also consumes more power. The change in power consumption at the server with changing load at the VMs is shown in the right part of fig. 4.7. The power for a single VM with increasing load changes from 0.49 W to 8.5 W, whereas this change for 2 VMs running in parallel is from 1.6 W to 13.3 W. Although MIPS processed at the server for two VMs are almost doubled with respect to those of a single VM, the change in power consumption is increasing with different rate. With these observations, the VM power models are trained with two different regression methods. At first, LLSR is used to train the dataset for model1 and model2. The parameters obtained for these models are listed in the top part of tab. 4.2. A similar dataset is then used to train for LSVR for both models; model parameters obtained for this training are listed in the bottom part of tab. 4.2.

Tab. 4.2: Models' Parameters

LLSR				
Parameter	Model1	Model2 (equiv.)	Model 2a	Model 2b
$\beta_0$	0.5369	1.4395	32.769	0.1687
$\beta_1$	0.0025	0.0021	0.0533	0.0388
LSVR				
$\beta$	0.1074	-	2.8876	0.5398
$\epsilon$	0.4030	-	10.807	0.403
$C$	4.0305	-	108.07	4.0305

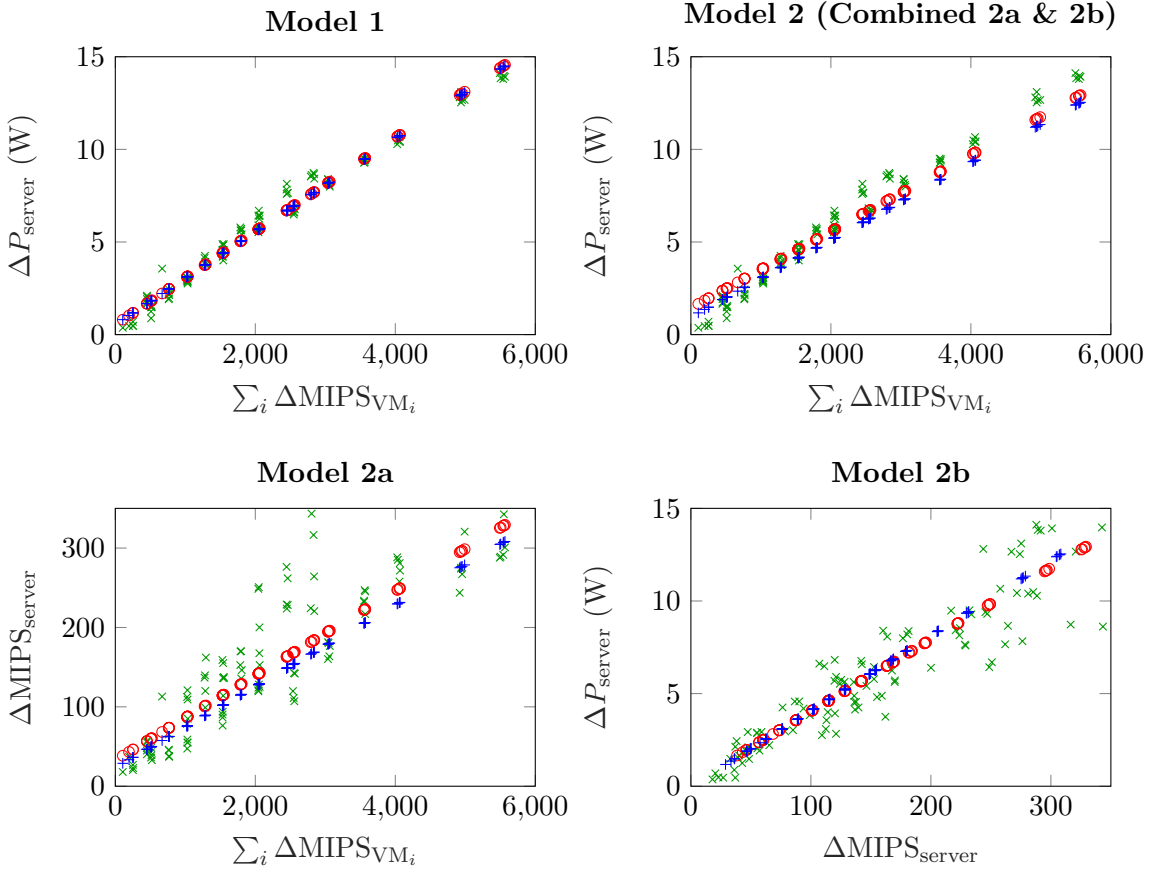


Fig. 4.8: Model 1 vs. Model 2 and individual components of Model 2. Legend: “x” shows measured data used for training, while the predictions from LLSR and LSVR are shown by “o” and “+”, respectively.

The mapping of data points for Model1 obtained after training is shown in the top left part of fig. 4.8. As the measured data points show almost a linear behavior, the predicted values for both models are very close to the measured ones. The Root Mean Square Error (RMSE) for both models is also small, as can be seen in tab. 4.3. Both LLSR and LSVR models perform almost similarly, with RMSE of about 0.5 W. We can also observe that there is not much variation in prediction for test data in both cases, as shown in fig. 4.9. Hence, for some small scale system, linear models can predict with limited error, which is acceptable in many cases. However, we can not conclude this for much bigger systems.

Measured and predicted values for Model2a and Model2b are shown in the lower part of fig. 4.8. Here we can see that the measured data for Model2a is scattered for some data points. The scattering of these data points show that there is no defined relation between the MIPS processed at the virtual level and processed MIPS at the physical level. Clearly, the relation between the two is not linear; as a matter of fact, there is a lot of uncertainty between the two variables. Hence, this model might impose high error on prediction. The LSVR regression model tries to compensate this scattering in training by increasing its tolerable error margin  $\epsilon$  (high value of  $\epsilon$  in Model2a). But

Tab. 4.3: Models' Prediction Error

	Reg. Model	Cross- Validation RMSE	Testset RMSE
Model 1	LLSR	0.532 W	0.504 W
	LSVR	0.545 W	0.507 W
Model 2a	LLSR	44.4 MIPS	43.5 MIPS
	LSVR	45.5 MIPS	55.5 MIPS
Model 2b	LLSR	1.510 W	1.2 W
	LSVR	1.580 W	1.39 W
Model 2 (combined)	LLSR	-	0.640 W
	LSVR	-	1.070 W

still some data points are extremely far away, which causes the cross-validation RMSE of the model to be around 44 MIPS. A similar effect was also observed for test data where the predicted RMSE value is 43 MIPS, shown in table 4.3. The second part of Model2, i.e. Model2b, however performs much better, as shown in the bottom right corner of fig. 4.8. Variation in measured data points is not as much as in Model2a. This can be supported with the observed data in fig. 4.7, where the change in MIPS was higher than the change in power by increasing the number of VMs. Hence, the error percentage for this model could be less than Model2a, but will be higher than with Model1. As observed, the cross-validation RMSE of model2b is 1.5 W for both LLSR and LSVR (refer to tab. 4.3). The prediction error for Model2b is also small, around 1.2 W. However, since Model2a and Model2b alone have no significance and since they are designed to be used together, their combined effect on prediction must also be analyzed. In the top right corner of fig. 4.8, data points for combined Model2 are shown. It can be observed that the predicted values are overlapping the measured data points in most cases. The combined model shows that the negative effect of Model2a is balanced a little with Model2b; however, the prediction at extreme data points (such as for minimum MIPS and for maximum MIPS) still exhibits more deviation than with Model1. The RMSE of test data for the combined Model2, is 0.6 W for LLSR and 1.07 W for LSVR, which is significantly improved from individual Model2a and Model2b performance. However, the box-plot of prediction error for combined Model2 in fig. 4.9 show more variation as compared to Model1, but the average prediction error still lies within the uncertainty of  $\pm 1.5$  W for both regression types for Model2.

From the results of these models it can be concluded that Model2, through the combined parts, improves the prediction error from individual Models (2a and 2b). However, Model1 performs better than the combined Model2 when trained with LLSR. With LSVR, the combined effect in Model2 is almost similar to Model1. The percentage prediction error for Model1 is found to be 6-7%, whereas the error for Model2 lies in the range 8-14%. These modeling approaches are less complex in comparison to other existing models which uses several counter values and their dependencies at the server level. The model provides the similar prediction accuracy (even improved prediction error in Model1), while using a less complex single counter value at the VM end.

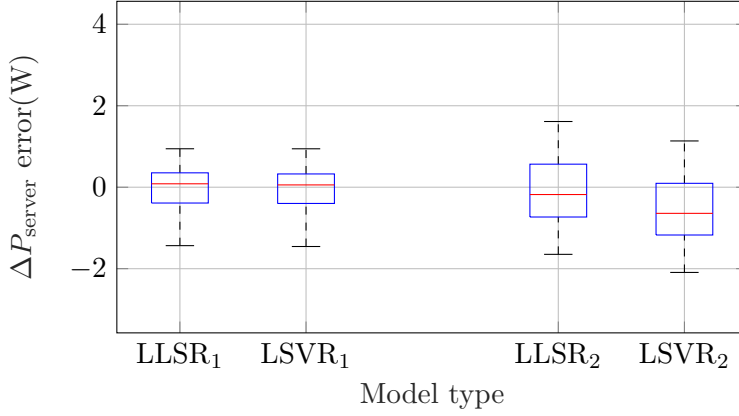


Fig. 4.9: Box-plot of error for Model 1 and Model 2. Legend: Median error (—), 1<sup>st</sup> – 3<sup>rd</sup> quartile of errors (—), overall error range (---)

#### 4.8.2 Second Phase: Further Analysis

The results in the previous section suggest that the introduction of two-step modeling has no significant positive effect on power estimation. Indeed, there are more details of input features (two-step model) but, since the relation between the host and virtual MIPS is not clearly defined, Model2 is performing no better than Model1. Hence, the selection of a less complex modeling approach could be a smart choice for further analysis. Considering this, further experiments for detailed analysis were carried out using the single-step approach of Model1. These experiments were carried out on the setup at 'Hamburg University of Technology (TUHH)', with increased number of VMs and more combinations of load intensity on all VMs. Also, the rate of change in MIPS at the host is fed along with the VM MIPS as the input feature to the model.

#### Hypotheses

For this phase, hypotheses considered includes,

- **H1:** The resource consumption at the host level differs from that of the guest level for a given load.
- **H2:** At higher load, the power consumption behavior of the server becomes non-linear.
- **H3:** The prediction accuracy can be improved using a smaller number of features

The methodology in this phase is similar to the one adopted in sec. 4.8.1, where the modeling consists of three steps: server power profiling, model training and then power estimation. For the power profiling of a server, a set of measurements are taken by running different workloads at the server. Various workload intensities are used to cover the whole range of CPU utilization. Along with that, performance counters at the host and guest machines and power consumption of the server are also observed. A bigger dataset is collected in this phase than in the previous experiments, as in these experiments all possible load combinations (from 0-100% CPU utilization, with intervals of 10 steps) were used on all eight virtual machines. Virtual PMCs (vPMCs) at individual

VMs are monitored along with PMCs at the host to observe resource utilization with much finer granularity. Then, the power consumption relation corresponding to the instruction count is modelled. The change in instructions' count and power is calculated using the similar equations (eqs. (4.10) to (4.12)) used in the previous experiment phase (in sec. 4.8.1).

For blind test validation of trained models, measured data is divided into training and testing dataset, with 80% and 20% of data, respectively. For training, features including  $\Delta \sum \text{MIPS}_{\text{VM}_i}$  of individual VMs and  $\Delta \text{MIPS}_{\text{Server}}$  are fed as predictors, whereas  $\Delta P_{\text{server}}$  at the server corresponding to the varying load is the model's response variable. Moreover, the relation between consumed resources at host and guest level at a similar instant is determined. Model reliability and validity is later evaluated based on different performance metrics.

### TUHH Experimental Setup

For these experiments a new system was setup at TUHH, Hamburg. It includes a single server with KVM [11] as a hypervisor, and different numbers (1-8) of virtual machines. Server and VM configurations are listed in tab. 4.4 below.

Tab. 4.4: Server's and VMs' Configuration

	Server		VM	
	Intel(R)	Xeon(R)	Intel(R)	Xeon(R)
Processor	CPU E3-1270 V2		CPU E3-1270 V2	
No. of CPUs	4		1-2	
OS	Ubuntu 18.04.1 LTS		Ubuntu 18.04.2 LTS	
RAM	31GB		2GB	
Storage	916GB		53GB	
$Power_{\text{server},\text{idle}}$ (W)	33.6		-	
$\text{MIPS}_{\text{idle}}$	80		0.3	

The workload generator “stress-ng” is used to stress the vCPUs. The selected stress methods with integer and floating point operations run on each VMs. For these experiments only CPU-intensive workload is considered, where the workload is increased by increasing the virtual CPU (vCPU) utilization from 0% to 100%, with all possible load combinations (in intervals of 10 units) among all VMs. With increased number of load combinations the size of the dataset also grows, which can improve the model reliability. In each experiment, different combinations of CPU workload run on active VMs. Resource monitoring using PMCs is carried out at both levels, host and guest, using the advanced observability Linux tool ”PERF”. At the same time, the dynamic power of the server is monitored remotely using an external power distribution unit “GUDE Expert Power Control 8226-1”. Further experiment's parameters are presented in tab. 4.5.

Tab. 4.5: Experiment's Parameters

Parameter	Value
No. of VMs	1-8
CPU load	0-100%
No. of repetitions	5
Linear regression models	LLSR, LSVR
Non-linear regression models	QSVR, GPR
No. of input features	1-10
No. of output features	1
Cross validation	5-fold
Run time	60 s

### Training Data Analysis

In this section, we first discuss the measurements obtained for power profiling of the server. Graphs below show how the server behaves with different numbers of VMs running in parallel and how is the performance changing in terms of MIPS (million instructions per second) for different scenarios.

From fig. 4.10 we can observe that by increasing the number of virtual machines the total number of instructions processed in one second also increases, even if the total CPU utilization remain similar. Considering the case of 50% CPU utilization in fig. 4.10, when this 50% of CPU usage is distributed among four virtual machines more instructions are processed. In comparison, if the same CPU load is distributed on two virtual machines, the number of processed instructions is reduced. The reason is that, with more VMs, the hypervisor needs to process more requests from individual VMs to the physical host. This effect becomes more significant with increasing number of VMs. This increase in MIPS also depends on how the CPU load is distributed among different VMs. In the same figure, considering the similar case of 50% CPU utilization for four VMs, there are obviously a number of ways to distribute this 50% usage among them. Some of these combinations were selected for the experiments and results show that the required MIPS vary slightly for different combinations. Also, in modern machines, the processor adjusts its processing speed depending upon the load; the effect of CPU usage and its performance does not remain linear as a consequence of the adjustment strategy. As it can also be observed from the same figure, the change in MIPS processed is increasing non-linearly at higher CPU utilization for four VMs.

Fig. 4.11 shows how different numbers of VMs and load affect the server power. It can be observed that for lower number of VMs, the server power consumption increases almost linearly. However, the increase becomes non-linear for higher numbers, such as with 6 and 8 VMs. Power consumption also tends towards non-linearity with over-utilization of processor capacity. Many datacenters perform CPU overcommitment using hypervisors, running multiple virtual machines on a single computer where the total number of virtual CPUs exceeds the total number of physical CPUs available. This aspect is studied by using the same number of VMs but with different numbers of vCPUs in an experiment. Considering fig. 4.12, four VMs each having a single core constitute 4 vCPUs in total, which is also the maximum CPU capacity of the server (refer to tab. 4.4). The change in power consumption in this case is also practically linear. However, when the total of



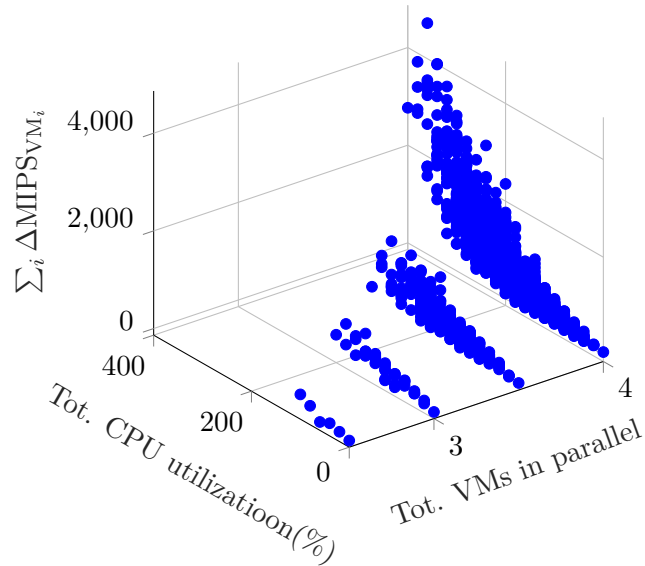


Fig. 4.10: Change in VM\_MIPS w.r.t total server's CPU utilization

8 vCPUs are assigned to these 4 VMs, the power consumption trend against utilization becomes non-linear. Hence, when the number of vCPUs exceeds the available physical CPU capacity, VMs start contending for CPU resources. In such scenarios, modern processors share the cores among vCPUs, and this core sharing is causing the non-linear behavior in power consumption.

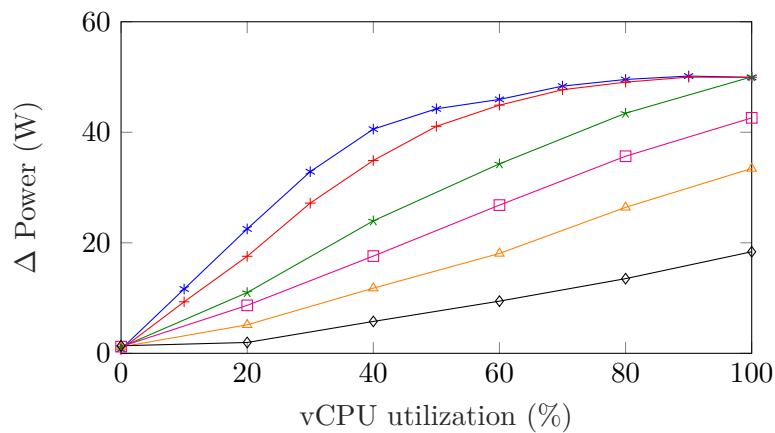


Fig. 4.11: Server power consumption vs CPU utilization for different numbers of running VMs. Legend: —\*— 8VM, —+— 6VM, —\*— 4VM, —□— 3VM, —△— 2VM, —◇— 1VM

### Models Analysis

As we observed from fig. 4.11 and fig. 4.12, the power consumption curve of the server against vCPU utilization changes from almost linear to non-linear with increasing number of parallel running VMs and vCPUs. Also, the required MIPS tend toward a non-linear increase with increasing load and number of VMs (refer to fig. 4.10). Therefore, both

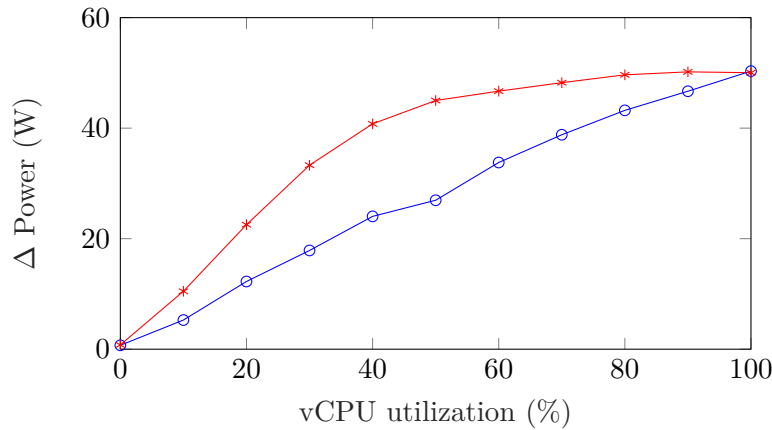


Fig. 4.12: Server power consumption vs CPU core. Legend: —\*4VM\_8vCPU, —○4VM\_4vCPU

linear and non-linear models are used in this research for training purpose.

Fig. 4.13 shows the response plot of different models after training for all data points included in the dataset. From these responses we found that almost all of the models estimate the values precisely in the middle range of the data points, i.e. when the total MIPS processed are not at extreme limits. However, for small MIPS values, linear models are prone to some error. Also, with higher MIPS there were some scattered estimated values from all models. Linear models have more erroneous values at higher MIPS, owing to non-linearities occurring in the power consumption, which are more accurately captured by non-linear models. This deviation of estimated value for different models is further analyzed for test the dataset in fig. 4.14 which will be discussed later in this section.

Other performance metrics considered for the evaluation of these models are model's training time and prediction time. The training times for non-linear models are found to be longer than those of linear models. This is because non-linear models are more complex and require more computations and processing. Indeed, linear models such as LLSR and LSVR have simpler functions; hence, they can model the response value using simple linear equations. On the other hand, for non-linear models kernel functions are more complex. The kernel function for GPR is based on finding a multi-variate Gaussian for a small set of data points, which becomes computationally more expensive with increasing number of data points. Also in QSVR, the kernel function is a quadratic equation, which is simpler than GPR, but is more complex compared to linear functions. Hence, these complex functions take longer for non-linear models to get trained and also take more time for prediction. Comparisons of training and prediction times for these models are presented in tab. 4.6. From this table it can be observed that linear models (such as LLSR and LSVR) took 1-8 seconds for training, and the similar dataset took around 23 seconds of training with QSVR. However, the maximum training time of about 84 seconds is taken by the GPR model, since it has the most complex kernel function. Hence, the fastest training can be done with the simplest linear functions (such as LLSR) which take only 1 sec to train the whole dataset. Also, for the validation (or testing) phase it is evident (from tab. 4.6) that GPR is the slowest to make predictions for the given data. The prediction rate of the GPR is  $23 \times 10^3$  per second whereas the LLSR

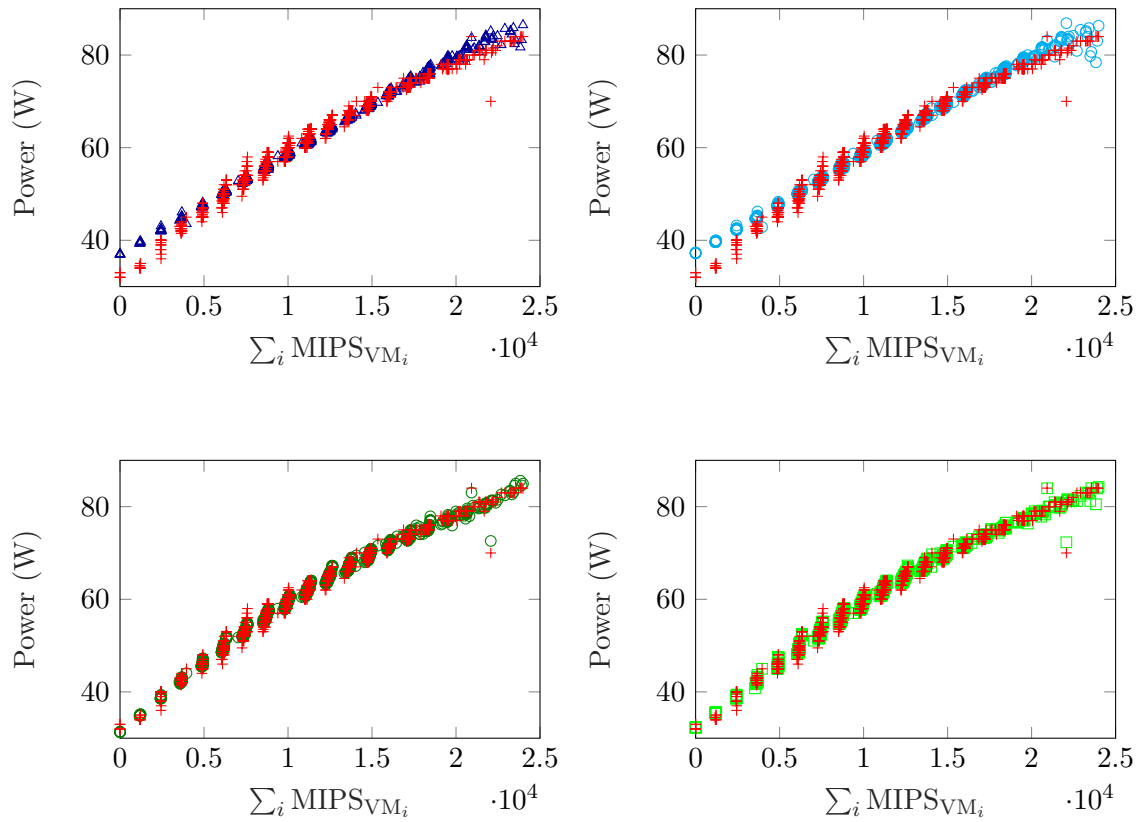


Fig. 4.13: Response plot of Models. Legend:  $\circ$ -LLSR,  $\triangle$ -LSVR,  $\circ$ -QSVR,  $\square$ -GPR and  $\times$  measured power value

Tab. 4.6: Model's Performance Indicators

	<b>LLSR</b>	<b>LSVR</b>	<b>QSVR</b>	<b>GPR</b>
Training time (s)	0.964	7.918	22.7	83.8
Prediction time (obs/s)	17e4	18e4	24e4	23e3

is fastest with  $17 \times 10^4$  predictions per second. Hence, the use of simple functions for modeling can make the estimation process 10 time faster compared to complex functions.

These models were further tested with both seen and unseen data to validate their reliability. Linear models, when used for seen datasets, estimate the power values for the middle range of load with minimum error. However, the error values increase as the load on the server increases. As we mentioned, modern processors change their power state and speed to adapt to load variations, which introduces non-linearities in the power consumption curve; hence, linear models do not fit well in those regions. In comparison, non-linear models when tested with the same dataset estimate power more precisely. The error percentage of both linear models is almost similar; however, the modeling complexity of LSVR is far greater than LLSR. Furthermore, both non-linear models also behave similarly for the given dataset. Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) of all four models are presented in tab. 4.7, where we can see that GPR performs best among all models with RMSE% of about 0.98%, whereas other models have RMSE% in the range of 1.2-2.9%. This precision in GPR comes at the cost of time and complexity, as can be seen in tab. 4.6.

The test dataset was also used to analyze the model performance and find the model's precision for unseen data. Fig. 4.14 shows the measured and estimated values of different models for the test dataset. We can see that linear models for the test data have predicted values further away from measured ones, especially at extreme data points. Results shows that prediction accuracy is also decreased in non-linear models with the test dataset. Hence, for unseen data all models exhibit some increased error percentage. From tab. 4.7, we can see that RMSE% for linear models increases up to 3.4%, for GPR to 1.34%, and there is a very little increase in QSVR.

Models studied in the literature [135], [146] that used the machine learning approach and performance counters for power modeling have error percentages in the range of 1.5-10%. In comparison, the proposed methodology predicts power consumption more accurately with minimum error of 0.9%, especially with non-linear models.

Small prediction errors in the proposed research approach are due to the selection of effective performance features in training. Performance counters used in the data have high correlation with the changing workload. Also the performance monitoring at both host and guest level helps in estimating the real amount of resources requested and processed. Models with complex kernel functions such as non-linear ones take more time to train and predict; however, they can give more precise estimated results. From results it can be concluded that the model selection for power estimation comes with a trade-off between accuracy and complexity. The modeling method for a specific workload or application can be chosen based on the acceptable error percentage and prediction latency. In most virtualized systems, servers use core sharing for processing multiple requests; non-linear models appear to be more suitable for these cases.

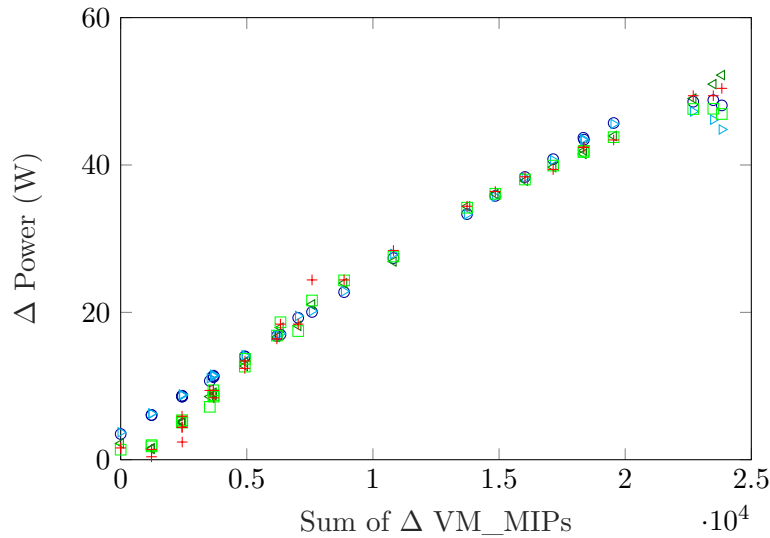


Fig. 4.14: Predicted power for test dataset. Legend:  $\circ$ -LLSR,  $\blacktriangleright$ -LSVR,  $\blacktriangleleft$ -QSVR,  $\square$ -GPR and  $+$  - measured power value

## 4.9 Conclusion

Measurement of power attributed to specific virtual components has become necessary and critical at the same time. With growing virtualization in servers and in the network, energy efficient load scheduling and optimized resource management is required. The estimated power consumption caused by the incoming workload to a VM can help servers making smart decisions upon VMs' initialization or their migration. This also requires precise knowledge of the actual resources consumed by individual virtual components. In this research work, we tried to highlight the primary factors that can improve the power estimation of any server. In a virtual environment, it is essential to observe the behavior of both host and guest machines to develop any power and performance model. Hence, we accounted for both hardware performance counters of servers and virtual performance counters of guest machines, to gain insight of generated requests and their processing at both levels. Monitoring too many counters simultaneously does not necessarily provide accurate counts; therefore, effective counters based on application type must be selected. This research further analyzed the performance difference between the host and guest level, and studied the effect of using counter values of both levels. Our aim while developing the power model emphasize on the prediction precision along with less overhead and complexity in the model.

As for big systems, such as data centers, server power modeling could be critical, with hundreds of VMs with several different configurations, machine learning algorithms can be useful in these scenarios. The selection of less complex models can reduce the training time even for huge datasets. The proposed approach uses machine learning algorithms to develop a power model and observe the performance counters inside VMs. The dataset consists of effective features (counters) for training helps improving the overall error percentage of the proposed approach. However, the results cannot identify one model to be generalized for any kind of server, although using the proposed methodology and training features can help developing accurate power models. Model selection should be

Tab. 4.7: Models error

	LLSR	LSVR	QSVR	GPR	LLSR, ANN [135]	SPTL [146]	
<b>Seen dataset</b>							
Cross-Validation Error (Watt)	1.68	1.58	0.78	0.84	-	-	-
RMSE (W)	1.51	1.65	0.70	0.64	-	-	-
RMSE %	2.9	2.98	1.2	0.98	-	-	-
MAE (W)	1.13	1.12	0.55	0.48	-	-	-
MAE %	1.97	1.96	0.92	0.83	-	-	-
<b>Unseen dataset</b>							
Cross-Validation Error (Watt)	1.76	2.0	1.2	0.9	-	-	-
RMSE (W)	1.61	1.62	0.71	0.78	-	-	-
RMSE %	3.39	3.46	1.29	1.34	-	-	-
MAE (W)	1.22	1.2	0.54	0.5	-	-	-
MAE %	2.21	2.18	0.94	0.93	5.42	1.83	4

based on the consideration of trade-off between prediction accuracy and model complexity, acceptable in the particular system for a specific application. Some findings and major contribution of this study are presented below.

#### 4.9.1 Findings

After analyzing the experimental results, we can conclude the following:

- The number of operations processed at hosts is generally different from that processed at VM level. So, in order to do effective resource scheduling and load balancing among VMs, performance and resource monitoring at VM level is also necessary.
- The power consumption of the server becomes non-linear at higher load. However, it is unfair to say that only an increasing number of VMs causes the non-linear behaviour of server power consumption. Instead, considering fig. 4.11 we can say that non-linearity is also introduced in modern processors when the offered load exceeds the available core capacity of the server. That is, when a server starts sharing its core among different entities, the power profile of the server changes to non-linear; this non-linearity increases with increasing over-utilization of core resources.
- Considering the above two points we can see that in modern processors the server behavior changes towards non-linearity. Therefore, one has to be careful while selecting modeling methods as the processor properties and load on the processor can dynamically change the server profile. Considering this the selection of different models for different load intensities can be made.

- The significance of using effective counters was also analyzed. Collecting a number of counters is necessary but extracting only effective ones from them is also important. The results of using several counters show that their number may become redundant in terms of features for training. Hence, it is useful to reduce the training features for the sake of reducing model complexity.

### 4.9.2 Contributions

We summarize below some major contributions of this section.

- The research has analyzed in some detail the behavior at two system levels, i.e. the host level and the guest level. On identifying the performance difference between the two, one can estimate the actual resources required at the host level to process the requested resources at the guest level. This estimation could help service providers in establishing management policies at datacenters and the cloud to use the system resources more efficiently and to optimally balance the load among different entities.
- The power model proposed in this research work exhibits reduced error in comparison to existing models. The precision of the proposed power model is between 97-99%, even for the unseen data points. This accuracy in power estimation could help improving the effectiveness of making power-aware decisions such as scheduling, power-capping, consolidation, etc. The proposed model also did not compromise on the complexity of the model and achieved high accuracy in power estimation with simple modeling techniques. Hence, high performance at the cost of low overhead can be obtained while using a minimum number of features and simple regression models.

## 4.10 Future Directions

In future, this work can be extended to derive the power model for an individual VM, by more selective exploitation of performance monitoring counters. These VM power models would be beneficial for both, service providers and clients. Service providers can have effective resource management and they can also develop cost models for their services based on actual resource usage. Such improved cost models will help clients to pay a fair amount for the resources they used. Moreover, models considering different types of workloads such as network- or memory-intensive ones can be developed, by finding their corresponding performance counters since, besides CPUs, memory and network are two dominant components in server power consumption.

With the development in energy management services, it is also expected that in future the introduction of energy saving technologies will make the CPU power consumption less relevant. In such scenarios, the power models will require to incorporate other dynamic factors contributing to power consumption, for accuracy and model effectiveness. These might include the effect of components such as disk, network cards, RAM, etc. In addition, if a large part of these power consuming components is not considered, power models could become relatively unpredictable. This will certainly increase the difficulty to develop and design an accurate power model and therefore it is an important and challenging research direction.





# 5 Overview of Resource Management and Task Scheduling in the Cloud Computing System

## 5.1 Introduction

Distributed systems offer a tremendous processing capacity. However, to realize this tremendous computing capacity, and to take full advantage of it, good resource allocation schemes are needed. Cloud scheduling is a research area that targets the efficient utilization of cloud resources. In the cloud, *Scheduling* is the process of detecting resources and allocating them to users, whereas *resource management* is a way to do it appropriately by optimizing resource utilization and improving system performance. An unbalanced server load could result in over- or under-utilization of resources, performance degradation, higher operation and execution costs, and may also lead to violation of Service Level Agreements (SLAs) [147]. Hence, task scheduling is an important functionality to avoid unbalanced load and inappropriate allocation of resources, in a system with hundreds or thousands of servers. The size of cloud networks is growing to deal with the increasing demand of incoming traffic, which makes optimal resource allocation further complex. Cloud scheduling is therefore categorized as an NP-hard problem in the literature [14], [15]. Variable workload patterns, uncertainty and heterogeneity of resources are major hurdles in the development of an efficient and optimal scheduling algorithm. Cloud providers and consumers both can get benefit from effective scheduling, such as by having minimum makespan and computation time, reduced latency and robustness, maximum resource utilization, scalable and heterogeneous services, and reduced cost [147].

In this chapter, introduction to scheduling in cloud computing is provided. The first section of the chapter describes some related notations and terminologies which are being used in this environment. Chapter further proceeds with the categorization of algorithms, which are usually based on the characteristics and nature of the variables in the system. Some scheduling algorithms are then briefly discussed, followed by the section presenting the existing literature and challenges posed by scheduling of tasks and allocation of resources in cloud computing.

## 5.2 Notation and Terminology

Just as the services, demands, and framework changes in different computing systems and communication networks, performance metrics and terminologies relating objectives, constraints and the environment also change. This section discusses some notations, performance metrics and terminologies used in scheduling in cloud computing. The main characteristics of any scheduling scheme in the cloud comprises of objectives, current environment, and constraints. Some most targeted objectives in these systems include (but are not limited to) reduced energy consumption, waiting time and makespan

minimization, overhead and complexity reduction, lower CAPEX and OPEX, maximum resource utilization and fault tolerance. However, cloud providers must consider the Service Level Agreement (SLA) of their clients while providing high Quality of Service (QoS).

The optimization of the objectives, however, is restricted with given constraints which must be considered while finding an optimal solution. The constraints are the strict system features that can not be compromised. Constraints considered in the system could be due time, deadline, processing speed and processing capability of a machine, preemption, priority, power threshold, etc. Processing capability refers to the fact that not all machines can process any task. Some machines are dedicated to process a specific kind of task and vice versa.

There are also several primary and derived performance metrics available that are evaluated to analyze the system performance. Similar to the problems' objectives and constraints, performance metrics also vary for different users and applications. Task completion time, throughput, makespan, cost, energy consumption, and resource utilization can be listed as some of the widely used metrics in cloud computing. Some of the metrics have high significance as they might also be used to derive the performance of other metrics. Once such example is task completion time that may affect the throughput and power consumption of the network as well.

Discussed below are some notations and terminologies which are highly related to scheduling in cloud computing.

**Processing Speed:** It is the processing speed of each VM in the system and is measured in million instructions per second (MIPS). The processing speed of any machine can be represented either as the sum of the speed of all cores or processing speed of a single core. In the latter case, if all cores operate at the same speed, processing speed can be multiplied by the total number of cores to get the maximum speed of the machine.

**Cost:** It is one major objective for service providers. The cost of the system is defined as the combination of VM migration cost, failure cost, operational cost, cost of execution and communication. There are various factors that could affect it. Cost reduction can be made at software level, hardware or architectural level, legal processing, etc.

**Arrival and Completion Time:** The time at which the task enters the system is task's arrival time. Similarly, when a task finishes its processing and leaves the system, that instant is called completion time of the task.

**Execution Time:** This is the actual time at which the task starts processing. The tasks are not scheduled immediately upon their arrival in the system; there is some scheduling delay or some time the scheduler waits for a batch of tasks to complete. This usually add delay for the task to execute.

**Waiting Time:** The time during which the task wait in the system queue, before start processing is called its waiting time. It is the period between the arrival and execution time of the task. In the case of preemption, an additional waiting time is added if the task is paused and resumed again later.

**Processing Time:** The processing time of the task is the time any task takes to finish its job. It is the time between the execution and completion time of the task.

**Deadline:** The deadline of the task is the strict restriction for the completion of a specific task. The tasks associated with the deadline constraint should finish their processing before reaching their deadline. Tasks that cross their deadline are considered as failed or lost, hence degrading the system throughput.

**Due Time:** It is defined as the time by which a task should try to finish its processing. If any task finishes after its due time it will be considered late and will be evaluated for its lateness.

**Lateness:** The lateness of the task is defined as an additional time any task takes to finish after its due time.

**Makespan:** The Makespan of the task is the total span of time that any task spent in the system. The time from the arrival of the task into the system until it finishes its processing and leaves. However, some applications also consider the makespan of the process (or for a batch), where the makespan is the time when the first task arrives in the system and the last task leaves the system.

**Energy (or Power) Consumption:** This is the one important performance metric, from both economical and environmental point of view, and is also used as an objective function in many applications. Power and energy are two dependent variables, which should be reduced to minimize operational cost and carbon footprint. The energy consumed by the system is the amount of power resources used by the system during an interval of time. The major sources of power consumption in cloud computing are computation resources, cooling and management system, and networking devices.

**Migration:** It is an act of moving a VM from one server to another server either to achieve load balancing or for other performance gains. Migration is a complex process where the selection of VM to be migrated and of the server on which to migrate is a strategic decision. Moreover, migration cost is an overhead itself, which should be considered precisely before making any decision. Migration of a VM can be done in a number of ways such as offline and online migration, pre-copy and post-copy migration, etc. Several models are available to calculate the migration cost and to make a right consolidation decision.

### 5.3 Categories of Scheduling Algorithms

Scheduling algorithms can be categorized based on the characteristics of tasks, objectives of scheduling algorithms, approaches and methods adopted for scheduling, etc. Some of these categories [148] are elaborated below.

- **Dependent and Independent:** These two algorithms are differentiated from each other based on the dependency of their precedence tasks. With the introduction of containers and virtual network functions (VNFs), different applications divide the task into sub-tasks for fast processing. These sub-tasks can run in parallel but sometimes have a dependency on other tasks. In this context, the dependent tasks can not be executed until all their precedence tasks have finished, i.e. they depend on their parent tasks. On the contrary, independent tasks do not have any dependency on other tasks and can be executed unconditionally.
- **Online and Offline:** The offline schedulers have all required details of tasks such as their arrival rate, processing time, required resources, etc., before the start of scheduling. With the complete set of input information, the algorithm finds a way to efficiently process the inputs and obtain an optimal solution. However, online schedulers only receive information about the task on its arrival. The workload in these cases is unpredictable and schedulers on the run-time decides the mapping of a task to a VM. Since these schedulers have no information about the task

arrival rate in advance, they just try to optimize the solution for a given instant. Considering the dynamics in a cloud network, online schedulers are mostly used by service providers.

- **Meta-heuristics and Heuristics:** These methods are used to find the solution of the scheduling problem in polynomial time. Heuristic algorithms systematically explore the search space and have a definite searching rule to find the solution. Meta-heuristic is a higher level of heuristic search that searches for the optimal solution in an iterative manner. The iterative process guides heuristics to explore the search space efficiently. It considers the feedback from the last iteration, objective function, and prior performance of the system to find the optimal solution. The combination of exploration and exploitation in meta-heuristics, although providing the best solution even for the complex problem (where heuristics might fail to provide one), requires a significantly long time.
- **Resource Oriented and Application Oriented:** In any system, clients and providers are two entities that can be benefited from the algorithms and methods used. The QoS and SLAs are concerned with the satisfaction of clients; however, reducing the OPEX and other costs concerns the service providers. Models that consider clients' satisfaction as their objective are called as application-oriented models, whereas objectives such as load balancing and optimal resource usage fall under the category of resource-oriented models.
- **Single Objective and Multi-Objective:** Since different applications and workload have different requirements and performance criteria, there could also be multiple objectives for a single application. Hence single- and multi-objective applications require different methodologies to maximize their performance. In multi-objective algorithms, there is a trade-off between two or more gains, and an optimal point is selected through different strategies.

As discussed above, the scheduling algorithms can be divided into different categories, where a single algorithm could possess the characteristics of more than one category. Some widely used scheduling algorithms are described in Appendix C.1.

## 5.4 Related Work

In recent years, virtualization technology has been widely applied in data centers, however, the CPU utilization of traditional servers is only 40% on average [13]. Besides, the Gartner report show that, underutilized servers have high energy consumption, and virtualization can reduce energy use as high as 80%, and save the deployment (area) space up to 85% [1], [149]. To resolve the contradiction between the number of servers and resource utilization, many service providers adopt virtualization to integrate server resources and save costs. Though cloud computing is no longer a new research area, its growing demand continues to challenge for better performance, reliable and more energy-efficient solutions for service providers and their users. Initially, scheduling techniques such as first-come first-served, round-robin, and various other schemes that do not consider the system parameters, were being used in cloud computing [150]. However, with the expanding use of cloud services, researchers realized that the cloud environment requires more dynamic, efficient and reliable algorithms. Task scheduling in cloud

computing is an NP-Hard problem that exhibits a large state space that complicates the search of optimal solutions in polynomial run-time [14], [15]. These problems are often addressed by using heuristic methods and approximation algorithms.

According to a study in [150], two major categories of scheduling are based on deterministic and evolutionary approaches. Deterministic approaches schedule the incoming tasks using algorithm based on priority, integer programming, max-min algorithms, etc., whereas evolutionary algorithms include, among others, genetic methods, particle swarm, ant colony, and cuckoo search. Both kinds of approaches are being used to improve the scheduling by maximizing resource utilization and by reducing delay in task execution. Also, different *workload patterns* in cloud computing have different resource requirements and also different SLA requirements which lead to several different application-based scheduling schemes. According to a survey on cloud computing [151], scheduling of tasks based on types of resources may include networking, computing, power, and storage. Researchers proposed different models to optimally use these resources and maximize the performance of servers in the cloud. According to a study [13], most of the time servers in data centers utilize only 30-50% of their total processing capacity. And this might be due to the inappropriate allocation of resources. Hence, future data centers need to increase their utilization rates for effective processing.

A study presented in [152] considered sensitive tasks scheduling based on their *deadlines*. They used the Open Nebula platform that uses the backfilling algorithm to schedule processes based on their deadline. The backfilling algorithm is an optimized FCFS algorithm that first sorted all processes in queue and then allocate them sequentially. However, owing to the conflicts with similar kind of task arrival, the increased failure rate in backfilling is addressed in the proposed scheme [152]. The proposed approach improved the failure rate by avoiding sorting of tasks, which was done earlier based on the task arrival time. Instead, the proposed scheme tries to schedule all the tasks at run time. Each arriving task is scheduled considering its deadline and excess time it has. The excess time of a task, also called 'gap time' in the paper, is the difference between the expected execution time and deadline of a given task. Tasks with smaller gap time are allocated first to the VMs so that they can finish their job before approaching their deadline. Results show that this scheme improves VM utilization and also the successful allocation of tasks for the various workloads. Another task scheduling algorithm considering the available *network bandwidth* is proposed in [153]. All incoming tasks with the same execution time are grouped together and are checked for the availability of bandwidth resources on each VM. The simulation shows improved bandwidth utilization, but lacks in providing the utilization of other resources that were not considered for optimization. Also, tasks with similar task length were considered in this study, which might not be applicable in real-world cloud computing. One resource management technique using 'Dynamic Voltage and Frequency Scaling' (DVFS) is proposed in [154]. In this technique, the voltage of the processor is adjusted to maximize the utilization based on the current load. One of the recent studies [155], proposed a heuristic-based scheduling scheme which comprises an analytic hierarchy process to optimize the resource usage. The scheduling process starts with the ranking of each incoming tasks, based on their size and run-time. Later, tasks are assessed for their resource requirements such as CPU, memory, and bandwidth, and assigned to available virtual machines. The load on each VM is also constantly monitored, which helps to balance the tasks running on them. The experimental evaluation of this scheme shows that the processing time of tasks by this approach is reduced by 20-25%, CPU utilization

is increased by 4%, whereas memory and bandwidth utilization rate are doubled.

Yet another study analyzed the effect of *performance interference* on the energy efficiency of the system [156]. Performance interference occurs in virtual environments when several VMs running on a similar server compete for the resources. A workload heterogeneity based scheduling is proposed that allocates the workloads based on their characteristics and their resource consumption on different servers. Results show that this scheme can reduce performance interference by 27.5%, and it improves the energy efficiency of a data center by 15%. One way to avoid the resource wastage in cloud computing is to have a balanced network load and avoid under- or over-utilization of CPU on any server. Balancing the load on servers has also been used in several studies to reduce power consumption. *Load balancing* is achieved usually considering the CPU usage of the server, while ignoring the other resources such as memory, bandwidth, and disk. An algorithm to attain the optimal energy solution of VM scheduling based on multi-objective is presented in [157]. The Multi-Resource Energy Efficiency Particle Swarm Optimization (MREE-PSO) method is proposed based on optimizing CPU and disk usage. Furthermore, local and global fitness of the swarm particle is assessed at each iteration to increase the probability of having the best solution by avoiding local minima. The MREE-PSO maximizes the resource utilization of the hosts by reducing the number of active servers which consequently reduces power consumption. Improved energy efficiency was observed for the proposed scheme compared to Modified Best Fit Decreasing (MBFD) and Modified Best Fit Heuristic (MBFH).

*Time* is also a constraint in scheduling algorithms. Services in the cloud are typically on a pay-per-use basis. So, tenants do not want to stay longer in the network, and hence reducing processing time is also a concern for both consumers and service providers. This requires scheduling algorithms not to be computationally expensive, as this could lead to increased latency [157]. Its effect could be more significant for dynamic traffic arrival and high traffic density. A Heuristic-based approach is used in [158], to solve the task scheduling problem in the cloud computing environment. Authors proposed an enhanced 'Fruit Fly Optimization Algorithm (FOA)' as 'Pareto FOA'. Two objective functions minimized in this approach are rent cost and makespan of the task. At first, random number of tasks are generated, where each task further has random number of sub-tasks or services. Later, each sub-task searches for enough available resources to execute. The cost of this heuristic search is minimized by allocating the task to a machine with a minimum cost to capacity ratio. Then, the population for the next heuristic search is selected by non-dominant sorting of tasks. However, reallocating tasks each time for a new heuristic solution will incur additional cost in this scheme. Results show that Pareto FOA reaches the optimal solution faster with reduced network cost. However, the algorithm is more complex than many other existing approaches.

The *cost* of running servers in the cloud network is also an important performance metric that can be computed in several ways depending on the service provider and the services offered. One of the research works determines the cost of a virtual machine in the cloud based on the processing delay and energy consumed [75]. Some parameters affecting the server's cost [159] includes processing time, the power consumed, processing capacity and execution cost. The proposed scheme tries to reduce the cost by reducing the processing time and maximizing CPU utilization.

Power consumption in the cloud can be broadly classified into three categories, pertaining to software, servers, and cooling [160]. Optimizing energy at any level can make a significant reduction to the power cost and it can also contribute towards the green

computing environment. **Power capping** of a server is also useful in many scenarios such as to avoid any breakdown due to surges, high traffic load, etc. Hence, most of the data centers and cloud networks have their server power consumption limited up to 75-85% of their total power [161]. Power threshold examples can also be seen at a tech-giant stores like Amazon's EC, where amazon's largest instance 'm1.xlarge' when allocated to the server, had been restricted to use only 75% of the CPU [162]. The server's power threshold is set based on applications' trade-offs in terms of latency and power [163]. For critical applications, a meta-heuristic solution is sometime not feasible as they take long time to find a solution. The greedy heuristic solutions are preferred in these scenarios, where a scheduling decision is made based on the current system status and satisfaction of predefined rules. A greedy task-scheduling approach is presented in [164] to reduce the **power consumption** of the system by minimizing the active number of servers. With the rule of most-efficient server first, the algorithm sort the available servers by their corresponding computing capacity. Considering the deadline of the task as a constraint, a model is designed to schedule the incoming tasks to the most efficient server (first server in the sorted list) first, until its saturation point is reached. Later, other servers from the sorted list are considered for scheduling of the remaining tasks. This work only consider the offline task arrival, where the resource requirement of tasks were known in advance.

In the cloud, scheduling process is required for two functions, scheduling of a VM and of a task. However, models and methods used for scheduling in both cases are often similar. The Best Fit algorithm for VMs scheduling is used in [165], where hosts are first arranged in ascending order of their capacity and the requested VM is then allocated to the host which has the maximum unused resources available to accommodate new VMs. Results show that the time complexity for a single VM allocation is reduced to  $O(\log n)$ , where  $n$  is the number of virtual machines. However, this allocation time is further reduced to a constant  $O(1)$  for the Worst Fit scheme, where hosts are arranged in descending order before scheduling. Both Best Fit and Worst Fit strategies show reduced allocation time as well as the reduce number of active hosts when compared with 'Balance' and 'Greedy' algorithms, and hence can be considered as an optimal choice for similar scenarios.

## 5.5 Open Issues in Cloud Computing

A methodological survey [147] on challenges and issues of task scheduling in cloud computing analyzed its literature from various aspects. They studied by categorizing the research as the classification of resources, resource distribution policies, QoS parameters, resource scheduling tools, etc. Based on different techniques used in each scheme, the authors in [147] classify the group of schemes as cost-effective, time-based, SLA- and QoS-dependent, optimization, energy, and dynamic network behavior-based. Furthermore, the survey highlights the significant components to consider while designing a scheduling algorithm. These components include availability, reliability, security, cost, execution time, energy, resource utilization, SLA violation rate, throughput, bandwidth, and user satisfaction. The study [147] suggested two new aspects to be considered for task scheduling; namely: QoS-aware scheduling, and self-management of cloud services. According to the authors, cloud services are provisioned according to the available resources and not ensuring the performance. Hence, there is a need to evolve scheduling

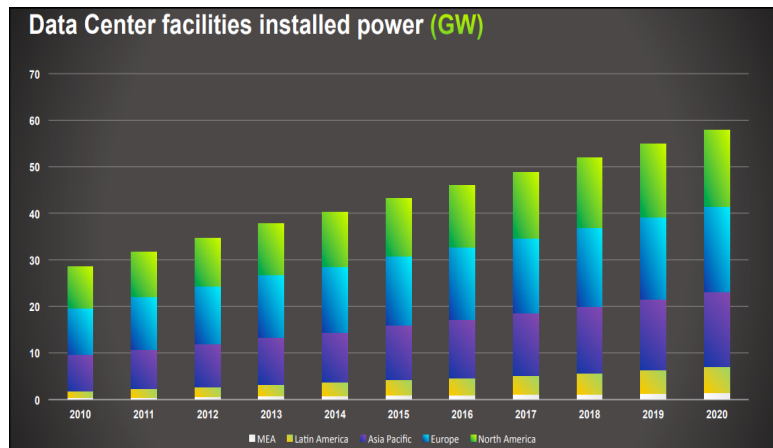


Fig. 5.1: Worldwide power consumption of data centers in GW [167].

schemes based on QoS requirements. Secondly, the self-management of system resources will help in keeping the system stable in unpredictable conditions and it will adapt new environment quickly. The survey concludes that self-management service, migration, and server consolidation, QoS and energy efficiency are still greater challenges in cloud computing. Discussed below are some focused open issues in cloud computing of the recent era.

### 5.5.1 Power Consumption

We already studied in the literature that the growing server farms are significantly increasing the rate of power consumption by data centers each year. This huge power consumption rising both economical and ethical issues those concerns the energy cost and the carbon footprint [166]. This domain has been remained under-focused at all times. A report on data center's power consumption by Lawrence Berkeley National Laboratory in 2014 [167] estimated that by 2020 data centers would most likely be using 200 billion kWh of energy. Power consumption estimation stated in this report is present in fig. 5.1.

Increasing power consumption led to the increased server's temperature, which in the long run could degrade the server performance, and also increase the cost of the cooling system. The power consumption at data centers is constituted of different factors including processors, cooling systems, network equipments, and other secondary resources [168]. But two components contributing to major power consumption are computing resources and cooling services. Improving energy efficiency on a large-scale distributed system is challenging. Therefore, the problem of power consumption even after being studied for years still requires more effort. Moreover, a maximum of about 18% of the existing research work is using energy-aware scheduling for cloud computing [147], indicating that the cloud still needs energy-efficient algorithms to reduce its execution and operational cost.

### 5.5.2 Network Uncertainty and Heterogeneity

Each virtual machine in the cloud may have different configuration, and similarly, each incoming task may have different requirements. Hence, the heterogeneous nature of the



system environment is another challenge. The scheduling methods and their objective functions depends upon the workload type and its Quality of Services (QoS) requirements. However, in any case, the maximization of the CPU utilization and reduction of the execution and operational costs are among the major concerns for most researchers. Since the cloud facilitates the users across the globe, hence with users having different region and applications have different resource requirements which constitute a dynamic and heterogeneous cloud environment. Network bandwidth demand for different multimedia applications, variable workload patterns, user's location, and device type, all have become factors for creating uncertainty in the cloud [20]. Another changing variable in the virtual network is VM migration and consolidation. Since huge data centers have hundreds and thousands of servers distributed over several racks, this also varies the cost of VM migration. Also, there are numerous factors involved for the network heterogeneity, finding a holistic solution while optimizing all dynamic variables is difficult to attain. Also, according to a survey [147], not many of the studies have considered the heterogeneity of workload in their research, which is, however, one major characteristic of the cloud.

### 5.5.3 Security

The developments in cloud computing are also speeding up the rate of outsourcing of services in several organizations. This ultimately is increasing the security risks such as network and other infrastructural vulnerabilities, user access, authentication, and privacy. The cloud users engaged in web services and applications need to trust service providers for their data sharing and privacy. Data security and privacy are traditional issues in computing and networking, which has been addressed by a large number of researchers and yet is a matter of investigation and lively discussion. Along with that, the advent of virtualization, multi-tenancy and shared resource pools has imposed new threats to the privacy of cloud computing [169]. These non-traditional issues are usually over-looked in security service provided. Data stored on the cloud has a major concern about the privacy of shared data. An attack on a single user can affect multiple tenants. Outsourcing of privacy services is also a threat where the third party is involved for data management. Several controlling schemes and architectures have been proposed dedicated to a single cloud framework and addressing particular attack problems, but a more generalized architecture dealing with different security issues in the cloud is still missing. One survey on security challenges of the cloud [170] concludes that cloud services regarding legal and administrative concerns have been studied well enough and solutions to them are available. However, domains like secure virtualization are still an open issue. Legal issues, compliance, and loss of data are three major issues in cloud security, where legal issues and compliance appear in 73% of the studies and 12% of the studies cover the issue of secure virtualization. Network and data security have also been given little attention and hence require more research efforts.

### 5.5.4 Availability and Reliability

High availability and reliability are among the great expectations of customers in cloud computing. Reliability in the cloud can be defined as the probability that the system is operational at all time without any failure, whereas the availability of the system is the probability that at a given instant the system is accessible, and functioning correctly

[171]. The idea of having no physical offices or companies came to existence due to the availability of resources anywhere, anytime. In recent years, cloud computing has gained great attention from global businesses and government agencies, thus it is becoming essential to provide high availability for cloud services to maintain customer's confidence. For these remote customers, their data, quality of data, reliability of data, and computing resources is all their worth. They can only be satisfied if high availability of resources, the possibility of data back up in case of failure, fast response time and high performance can be ensured [172]. Failures that could affect the reliability in the cloud are hardware failure, software failure, cloud management system failure, security failure, environment failure, and human faults. Future research in these directions can be carried out to improve the system reliability [171].

# 6 Optimal Task Scheduling in Cloud Computing

## 6.1 Introduction

Scheduling is the process of detection, selection, and allocation of appropriate resources to the users. Task scheduling in any network is important; otherwise, a network with a bunch of servers can have an unbalanced load and this will result in over- or under-utilization of resources, performance degradation, higher operational and execution cost, and may also violate Service Level Agreement(SLA) [147]. Efficient scheduling schemes, however, can reduce the cost, processing time, power consumption and can also improve other service requirements such as security, reliability, and scalability.

The problem of resource allocation in large-scale shared cloud infrastructures is known to be NP-hard and has been studied in many contexts in the past [14], [15]. This chapter aims to find a reasonably good heuristic solution in a relatively short time. The available heuristic solutions considering different objectives are targeted to efficiency in energy and performance. One goal of this research is to look for a less complex scheme, that can make the scheduling decision quicker and less resource consuming.

In this chapter, we first look at the issues and motivations for scheduling in cloud computing environments. Considering those, the problem definition for this research work will be stated in a later section. The chapter further proceeds with the description of the system which is being analyzed and present different case studies done to achieve final goals. The proposed scheduling method is then proposed in the last case, which is evaluated for different performance metrics.

## 6.2 Motivation

As discussed in chapter 5, resource scheduling in the cloud possesses several challenges. Considering the SLAs and network architecture, constraints and objectives of the system vary; however, the goal in all systems is to find optimal task to VM mapping and task scheduling, usually for heterogeneous networks. A few of the issues motivated to carry out this research includes the under- and over-utilization of resources, model complexities, and energy-efficient scheduling, while maintaining the performance.

According to a study [13], most of the time servers in data centers are being utilized only 30-50% of their total capacity. This might be due to the inappropriate allocation of resources. Hence, for optimal resource utilization, future data centers need to increase their utilization rates; thus, shifting towards the cloud will benefit with lower costs and increase services. Moreover, the complexity of scheduling schemes introduces a delay in task execution. This optimization problem is NP-Hard, as it is similar to the bin packing problem which becomes more complex with different sizes of bins and boxes. Most researchers propose optimal search solutions to find the best possible mapping for incoming tasks to available resources. However, using these optimal search heuristics

could be costly, especially for huge data set such as in the cloud. Also, the optimal solution is only possible in limited cases of NP-Hard problems in polynomial time. So, to avoid uncertainty and huge computation delay, some research proposed heuristic rules to find the optimal solution in a relatively short time. The solution obtained using these rules might not necessarily be the optimal one, but it provides good enough results in most cases. The worst-case scenarios for these heuristics have also been analyzed, which provide bounds for their performance. The time complexity for scheduling  $n$  tasks using 'Simulated Annealing', which is an optimal search method, is  $O((n^2 + n) \log n)$  [173], whereas the solution using the heuristic rule such as 'Longest Processing Time (LPT) first' costs  $O(\log n)$  [174].

Moreover, the effect on power consumption using different approaches is also studied in the literature. The factors affecting the power consumption of the server include the system internal power usage, application resource requests and schedulers processing [136]. Hence, we can reduce the overall power consumption by selecting energy-efficient scheduling schemes. But, since the major contribution of power in data centers and cloud networks is through computing sources and their management equipment, the problem resource allocation is also addressed and tackled in this research. A significant amount of idle power is consumed by the server, i.e. when the servers are powered on but not executing any task. This is usually static but varies from server to server, depending upon its configuration and architecture. Mostly the idle power ranges from 30 W to 140 W which is usually 60% of the server maximum power [175]. This power wastage can be reduced by powering up only a few servers. According to the [136], the total power consumption of the data center is dependent on static and dynamic power consumed by an individual server, where a significant amount of static power is comprised of server idle power.

$$P_{total} = \sum_{j=1}^M (P_{jstatic} + P_{jdynamic}) + P_{jsecondary} \quad (6.1)$$

In eq. (6.1) ([136]),  $P_{total}$  is the total power consumed by the cloud or data center,  $P_{static}$  is power consumed at power-on state of the server,  $P_{dynamic}$  is the change in power caused with resource usage, and  $P_{secondary}$  is an additional power used for cooling, management of the system and network devices, etc., and  $M$  is the total number of servers in the system.

Considering the above-mentioned issues and possible solution, this thesis research tries to solve the problem through combination of heuristic rules that are proved to provide the sub-optimal solution even in the worst case.

### 6.3 Problem Definition in Scheduling

A scheduling problem can be defined using three characteristics  $\alpha | \beta | \gamma$ , which represents the system environment and its objectives. The environment of processing machines is represented by  $\alpha$ , where different configuration of the system could be a single machine (1), identical machines in parallel ( $P_m$ ), parallel machines with different speeds ( $Q_m$ ) and unrelated machines ( $R_m$ ) (further detail in Appendix C.2). The field  $\beta$  represents the system constraint(s) and,  $\gamma$  field contains the objective function(s). Depending upon the priorities and SLAs, the system implies different constraints such as task preemption, task's due time and deadline, processing delay, waiting time, power consumption, etc.

The objective function for the system also varies for different users and their applications. This concerns the performance and QoS which is maximized to achieve maximum gain(s) for a particular system. The most common objectives in scheduling problems are task makespan, total completion time, maximum completion time, energy consumption, cost, load balance, etc.

In the cases below, we will formally present our problem using the formulation described above. We describe a non-preemptive dynamic task scheduling problem for heterogeneous computing systems, such as the cloud. The problem considers a system of parallel machines with the objective of minimizing the overall execution time and maximizing the resource utilization of servers. A set of 't' independent tasks is to be allocated on 'm' non-preemptive machines, where  $t > m > 2$ . The task scheduler is used to map tasks on selected virtual machines for execution. Preemption of the task, considering any sort of priority is not considered. Any task, once started processing on any of the machine, will finish its processing without any pause in between. The system is studied under different machine environments, homogeneous ( $P_m$ ) and heterogeneous ( $R_m$ ), with and without any constraints, and considering different objective functions. The constraints considered include task due time and objectives in different cases are resources optimization, task completion time and lateness. Each case-study analyzes the system performance under defined objectives and strategy used to find the optimal solution. The final scheduling algorithm is then proposed considering multiple objectives derived from the results of previously studied cases. The proposed algorithm in each case is compared with the two most commonly used scheduling algorithms, Round Robin(RR) and Simulated Annealing(SA) Heuristic approaches (refer appendix C for details). These models are selected to compare the performance of proposed methods with the simplest one (RR) and one complex (SA) scheme and to observe where the proposed algorithm lies in terms of complexity and performance in comparison.

## 6.4 System Overview

The study considers the system with several parallel servers (or host machines) where each server hosts a different number of running virtual machines. Both cases of homogeneous (similar VMs) and heterogeneous (different VMs) systems are considered in this research. Also, the dynamic and heterogeneous nature of tasks is considered to incorporate the real-world scenario for the case studies. Tasks are supposed to arrive in the system according to a Poisson distribution with arrival rate  $\lambda$  and are collected in a single system queue  $T$ . The scheduling is initiated on completion of a batch, where the size of the batch is defined in simulation parameters. In case of a shortage of computation resources, all remaining tasks in the queue are scheduled later on the availability of resources i.e. when any task finishes its job and leaves the system. Considering a point in time of the execution of the scheduling process, the time indexes in system variables below are avoided. The system consists of a set of servers  $S$ , comprising  $n$  servers:

$$S = \{s_1, s_2, \dots, s_n\} \quad (6.2)$$

where each server  $s_i \in S$  has  $m_{s_i}$  VMs running on it. These VMs on a single server can be represented as a set  $V_{s_i}$ :

$$V_{s_i} = \{v_{s_i,1}, v_{s_i,2}, \dots, v_{s_i,m_{s_i}}\} \quad (6.3)$$

The VMs are allocated on the servers based on their core capacity, where each core of VM can be allocated to a single core of a server. The resource capacity of each VM may be different and is defined by the number of cores possessed by a VM and processing capacity in Million Instruction per Second (MIPS), where all cores are supposed to have the same processing speed. The  $j^{\text{th}}$  VM of server  $s_i$  can be represented as  $v_{s_i,j}(c_{s_i,j}, \rho_{s_i,j})$ , where  $c_{s_i,j}$  is a positive integer and represents the number of cores, and  $\rho_{s_i,j}$  is the processing speed of each single core in the VM.

Furthermore, there is a single task queue  $T$  consisting of the number of tasks present in the system at the observation instant:

$$T = \{t_1, t_2, \dots, t_k\}, \quad 1 \leq k \leq k_{\max} \quad (6.4)$$

where  $k_{\max}$  is the maximum allowed number of tasks in queue at any given instant of time.

Each task  $t$ , has a certain number of sub-tasks that defines the number of cores that a task can use on a given VM to run sub-tasks in parallel. The scheduler tries to allocate each sub-task on a separate core to reduce the task's processing time. The  $k^{\text{th}}$  task can be represented as  $t_k(c_k, \vec{l}_k)$ , where  $c_k$  is the number of sub-tasks (or cores) for task  $t_k$ , and the vector  $\vec{l}_k = [l_{k,1}, l_{k,2}, \dots, l_{k,c_k}]$  is the number of instructions or task length for all sub-tasks in a task  $t_k$ . The scheduling process is initiated after the batch of task (with  $k=10$ ) has arrived in the system queue 'T' or when any tasks leaves the system. When there is no suitable VM available to process the given task the task remain in queue and will be processed in the next cycle. The processing time ( $p_k$ ) of a task allocated to  $VM_j$  of server  $s_i$  can be determined using the eq. (6.5),

$$p_k = \frac{\max(\vec{l}_k)}{\rho_{s_i,j}} \quad (6.5)$$

Constraint in the system as task 'due time ( $d_k$ )' is also considered, where the task, if finishing after its due time, is evaluated for its lateness ( $l_k$ ). The due time of the task is the task's processing time with some delay. The equations below define the lateness and due time of the task,

$$l_k = C_k - (a_k + d_k) \quad (6.6)$$

$$d_k = p_k + rD \quad (6.7)$$

where  $C_k$  is the completion time of any task or the time when the task leaves the system and  $a_k$  is the arrival time of the task in the system. The value ( $rD$ ) is a randomly assigned delay within the range of 1-5 seconds. The performance of the system based on resource utilization is also studied while considering the core capacity of the server as a resource. The utilization cost (or core cost) is determined by the amount of cores being under- or over-utilized. The core utilization cost of the system with a total of  $n$  servers,  $m_s$  VMs, and  $t$  tasks can be expressed as

$$Core\ Cost_{sys} = \sum_{s_i=1}^n \sum_{j=1}^{m_{s_i}} (c_{s_i,j}) - \sum_{k=1}^t (c_k) \quad (6.8)$$

Considering eq. (6.8), the system will be over-utilized if the value of  $Core\ Cost_{sys} < 0$ , and if the  $Core\ Cost_{sys} > 0$  then the system is considered as underutilized. The system

is said to be maximum utilized for  $Core\ Cost_{sys} = 0$ . Furthermore, the complexity of each algorithm is measured in terms of the time taken by the algorithm to map all the incoming tasks to the available VMs. Moreover, power consumed and energy consumption of the system are also analyzed. The linear power model is used to measure the power consumed by each server, where the power is a linear function of the server CPU utilization and can be expressed as,

$$Power_{system} = \sum_{s_i=1}^n \left( P_{static_{s_i}} + (b * \phi_{s_i}) \right) \quad (6.9)$$

where  $b$  is a constant and  $\phi$  is the server CPU utilization.

## 6.5 Simulation Environment

A recent survey on cloud computing shows that about 81% of researchers are using the CloudSim simulation tool to carry out simulation-based analysis for cloud [147]. Thus, the Cloudsim-Plus 4.0 simulation tool is used in this thesis to implement the proposed scheduling schemes. Incoming tasks in this tool are referred to as 'Cloudlets', so task and cloudlet will be interchangeably used in the later sections of this research. The simulation network considered consists of 50 servers, where 150 virtual machines are distributed among them. Further, servers and VMs configurations are shown in table tab. 6.1. The network is heterogeneous; that is, the virtual machines have a different number of CPU cores and hence different processing capacity (MIPS). Also, all incoming tasks have different processing length (required number of processing Instructions) and can process on different number of cores based on the number of sub-tasks. Variable task length is assigned according to continuous distribution and is used to analyze the performance of the network for different task sizes. Also, a Poisson's distribution of task arrivals with different mean values is used in the simulation to analyze the scheduling behavior for various traffic densities. To observe different system scenarios, static and dynamic tasks are processed. Tasks already present in the system before the simulation starts are referred to as static tasks, and the system knows about these task requirements in advance. Tasks arriving in the system according to the Poisson distribution are considered as dynamic tasks. The simulation will end when all the generated tasks are processed and leave the system. The results of the proposed algorithms are compared with the Simple Round Robin (RR) and Simulated Annealing (SA) Heuristic Scheduling schemes in each of the following cases.

Further simulation parameters are presented in table tab. 6.2, and parameters for simulated annealing heuristic scheduling approach are presented in table tab. 6.3.

Tab. 6.1: System configuration for the Server and VM

	Server	VM
No. of machines	50	150
No. of CPU cores	32	1 - 16
Core speed ( $\rho$ )	1000 MIPS	1000 MIPS
RAM	2048 GB	512 GB
Hypervisor	Xen	-

Tab. 6.2: Simulation parameter

Parameters	Value
Simulation tool	CloudsimPlus 4.0
No. of repetitions	3
Max no. of task	50 - 500
Static task	50-500
Dynamic task	50-450
Task arrival rate per second ( $\lambda$ )	1 - 16
No. of sub-tasks ( $c_k$ )	1 - 8
Sub-task length/instructions ( $l_k$ )	1000 - $10 \times 10^3$
Task due time in sec ( $d_j$ )	1-5
Server power model	Linear model
Task batch size	10
Server static power	37%
Server Power Threshold ( $\delta$ )	85%

Tab. 6.3: SA Heuristic scheduling parameters

Parameter	Value
Initial temperature	1
Cold temperature	0.0001
Cooling rate	0.003
Neighbour searches	total no. of task

## 6.6 Case Study I

To have a stable system, we know that the total utilization of the whole system must not exceed its total capacity. We also analyzed from the literature that factors causing waste of CPU capacity include the inappropriate allocation of resources. Hence, in this case, the proposed algorithm tries to maximize the CPU utilization by avoiding the resource wastage in the contention for their use. The approach proposed is a modified best-fit scheduling algorithm for task allocation. For this case, we have a system of 'm' parallel machines where each machine has a different number of cores; hence the machine environment for the given system corresponds to  $R_m$ . The objective function for this scheduling scheme is to maximize the servers' utilization ( $\phi$ ) while avoiding under- or over-utilization. Hence the problem for this case can be stated as,

$$R_m | \phi \tag{6.10}$$

### 6.6.1 Pseudo Best Fit Scheduling (PBFS) Algorithm

The Pseudo Best Fit Scheduling (PBFS) algorithm tries to find an appropriate VM that has minimum unused resources available for task allocation. The over- and under-utilization of the resources can be measured in several terms; in this research, the number of cores of a VM is considered as a parameter. If the number of tasks allocated to any VM requires a number of cores greater than the available cores of the VM, then the VM



is said to be over-utilized. Similarly, if a VM at any time instance has some cores unused, it will be considered as under-utilized. The proposed optimal scheduling will allocate the tasks such that the number of cores of a VM is equal to the number of cores required by the task(s) allocated on it. Hence, the proposed method optimally uses the resources to the maximum available capacity which could also serve as a consolidation strategy.

The scheduler for this scheme starts scheduling by detecting the available and required resources in the system. It first takes a task  $t_k(c_k, \vec{l}_k)$  from the queue  $T$  in FCFS (First Come First Served) order and allocates it. The algorithm does not look for the optimal best fit solution; instead, it allocates the task to the first available machine satisfying the given conditions. Hence, it is called 'Pseudo Best Fit (PBF)'. The residual core capacities of the VMs on the available set of servers  $S$  are detected and compared with the required resources of a task as,

$$V_{\text{sel}} = \{v_{s_i,j} \mid v_{s_i,j} \in V_{s_i} \wedge c_{s_i,j} \geq c_k, \forall s_i \in S\} \quad (6.11)$$

where  $V_{\text{sel}}$  contains the list of VMs that have enough available free resources. A task-to-VM mapping function  $f$  can be represented as,

$$f : T \times V_{\text{sel}} \rightarrow \{0, 1\} \quad (6.12)$$

where,

$$f(t_k, v_{s_i,j}) = \begin{cases} 1, & \text{if } t_k \text{ is mapped to } v_{s_i,j} \\ 0, & \text{otherwise} \end{cases} \quad (6.13)$$

The objective function to this problem is expressed in eq. (6.14), that is to find an optimal mapping of tasks to VMs such that the difference between required cores by tasks allocated to VM and available cores of a VM is kept to a minimum.

$$f_{\text{opt}} = \min_f \left( \sum_{v_{s_i,j} \in V_{\text{sel}}} \left( c(v_{s_i,j}) - \sum_{t_k \in T_{v_{s_i,j}}} c(t_k) \right) \right), \quad (6.14)$$

$$T_{v_{s_i,j}} = \{t_k \mid t_k \in T \wedge f(t_k, v_{s_i,j}) = 1\}$$

where  $f_{\text{opt}}$  is the optimal task-to-VM mapping,  $c(\cdot)$  is the function to calculate the cores required by a task ( $c_k = c(t_k)$ ) or cores available for a VM ( $c_{s_i,j} = c(v_{s_i,j})$ ), and  $T_{v_{s_i,j}}$  is the set of tasks that are mapped to a VM  $v_{s_i,j}$ . When there are no suitable resources available for any task in the queue, the task will remain in the queue. From the list  $VM_{\text{sel}}$ , the VM with minimum available resources is selected as the optimal VM for the given task. Hence, the proposed PBFS allocates the task to the maximum utilized VM. But it does not allocate more than the available capacity to any VM, thus attempting to achieve the maximum utilization of the CPU resources per unit time. The scheduler repeats the process of the task scheduling until all the generated tasks are processed. It further schedules the waiting tasks in the queue when any finished task leaves the system. Hence, the scheduler does not check the availability of resources at every instant, which reduces the computational overhead; instead, it looks for the resources when there are possible free resources available or when any new batch (of tasks) arrives.

## 6.6.2 Results and Discussion

The simulation environment and system is similar to parameters defined in tab. 6.1, tab. 6.2, and tab. 6.3. However, simulation for this case is carried out with static cloudlets (tasks) only i.e. all the cloudlets will be present in the system queue at the start of the simulation. A total of 500 cloudlets were generated, where each cloudlet has a different cloudlet length following a continuous distribution in the range of 1000 to 10,000 instructions. Simulation will stop once all the cloudlets finish their service.

The simulation results obtained are analyzed for different performance metrics and the proposed scheme is compared with RR and SA scheduling schemes. Since the objective of this case is to maximize the resource utilization, we observe the utilization of cores for each scheme and later analyze the effect on resource utilization per second. Fig. 6.1 shows the total number of cores that were either over- or under-utilized in each case and is referred to as core cost (refer eq. (6.8)). The SA and RR algorithms allocate all the available cloudlets to the number of machines available in the system and do not consider the over- or under-utilization of resources. Nonetheless, SA tries to optimize the resource usage while searching for the solution by minimizing the under- or over-utilization. As we observe from fig. 6.1, the maximum core cost is around 1300 on average for the SA algorithm, whereas RR has cost around 1100 on average which is a little lower than SA. For the PBFS algorithm, the cost is very much lower and is around 100. As mentioned earlier, this cost includes both the over- and under-utilization of cores; therefore, fig. 6.2a and fig. 6.2b show these costs separately. In fig. 6.2a we see that the SA algorithm has the maximum over-utilization of resources and RR has a little lower than that. Also, the major contribution in total core cost in both cases is due to the over-utilization of resources, where the under-utilization cost is significantly smaller than over-utilization, as shown in fig. 6.2b. Since the PBFS works on the principle of avoiding over-utilization, it does not allocate resource more than the available capacity of the VM. Hence, the over-utilization cost for this scheme is zero. However, an adverse effect of this approach could be a long waiting time for cloudlets in some cases.

The system behavior for resource wastage due to under-utilization shows that the proposed scheme has some underutilized core cost (fig. 6.2b). This might be due to excessive resources which may not be required for the given number of tasks. Another reasonable factor could be that the remaining resources at a given moment might not be enough for the cloudlet to be scheduled, due to which these resources remain unused and the cloudlet is scheduled in the next cycle. However, still, the performance of the proposed scheme provides a better mapping of cloudlets to VMs in comparison to the other two approaches. The under-utilization cost for SA and RR is 200 and 80 on average, respectively, where the PBFS cost on average is similar to RR but it goes as low as 30 in some cases.

As observed from the above results, utilization cost in terms of cores is minimum for the PBFS; so, its effect on the resource utilization rate is also analyzed. Fig. 6.3 shows the CPU utilization per second for all three schemes. We can observe from these results that the maximum utilization rate obtained for the PBFS is around 18%, whereas the SA has the worst utilization rate of 2%, and RR is performing almost similar to SA with 3% utilization. As SA and RR allocate all available cloudlets to the available VMs, without considering the over-utilization of resources, this causes the sharing of core resources among multiple cloudlets. The contention of resources creates delays and depletes energy in control and management rather than job processing. Another

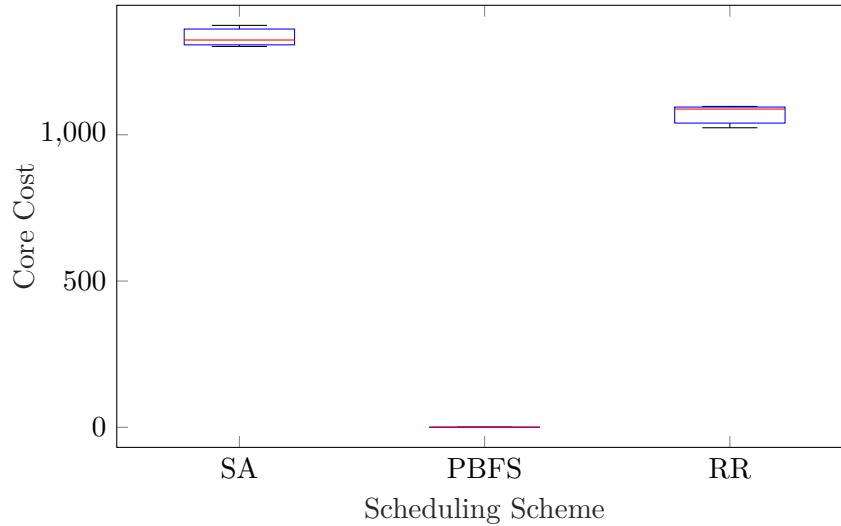


Fig. 6.1: Case1, total cost of under- and over-utilized cores for Simulated Annealing (SA), Pseudo Best Fit Scheduling (PBFS) and Round Robin (RR) scheduling schemes

important factor affecting the utilization rate adversely for SA is its complexity; the algorithm took a longer time to find an optimal solution, which in turn increases latency and worsens the utilization rate. On the contrary, the PBFS is decision-based scheduling and is therefore faster. Also, it only allocates the cloudlet until the availability of resources, hence avoiding unnecessary resource contention at the core-level.

The time complexity of the proposed model is also evaluated based on the results shown in fig. 6.4. Total mapping time, that is the time required to allocate all the cloudlets to the available VMs is observed. The PBFS, adopting a simple decision-based policy, took less time compared to SA that took much longer time while searching for the optimal solution in the whole solution space. The total time taken by the SA algorithm for this case is about 40s, whereas the PBFS took only a negligible amount of time i.e. around a few seconds. The major factor of increased mapping time in SA is the optimal search for a solution in a solution space. This solution space gets bigger and more complex with larger number of tasks in the system. The RR scheme here also took almost a similar time as that of the proposed scheme, as it also allocates a task to a VM in a cyclic round-robin fashion, without optimizing the resource. However, the proposed algorithm will be preferred over RR because of its improved performance in CPU utilization and energy minimization. Fig. 6.5 shows the energy consumption by all three schemes during the process. The energy consumed by RR is higher than the energy consumed by the proposed scheme. It is because RR took longer to process all cloudlets, due to core sharing, which is avoided in the PBFS scheme by avoiding over-utilization. However, the maximum energy is consumed by the SA algorithm to process a similar number of cloudlets.

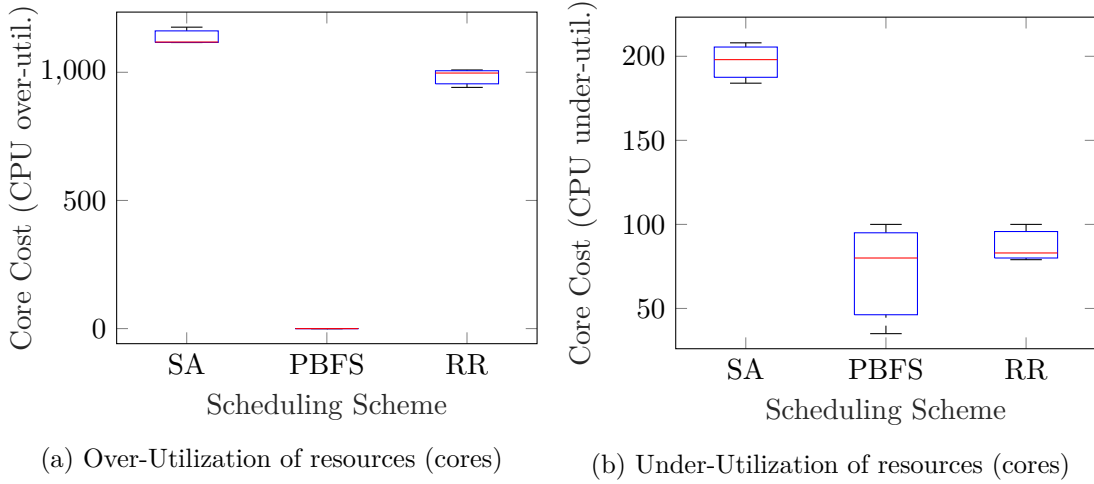


Fig. 6.2: Case1, the cost for over- and under-utilization of the CPU cores for Simulated Annealing (SA), Pseudo Best Fit Scheduling (PBFS) and Round Robin (RR) scheduling schemes

## 6.7 Case Study II

In the second case, the effect of heuristic solutions on improving the completion time of tasks is studied. The completion time of any task is referred to as the time at which the task finishes its job and leaves the system, which, of course, depends on their schedule. The completion time can be optimized in different ways depending upon the application requirement. The applications' interest in minimizing the makespan, i.e. to process all tasks in minimum time without considering the completion time of an individual task, focuses on the maximum completion time ( $\min(C_{max})$ ) as their objective function. The maximum completion time is the completion time of the last task processed on any of the machines, and the makespan is the period at which the first job starts its processing till the last job finishes and leaves the system. The objective of minimizing the  $C_{max}$  is an important one in practice as it has an effect of load balancing over various machines. On the other hand, time-critical applications where the user wants to minimize the completion time of individual task will have the objective to minimize the sum of completion time of all tasks ( $\min \sum C_j$ ). Both of these objectives have optimal heuristic rule solutions available (chapter 05 in [174]) as Long Processing Time (LPT) first and Short Processing Time (SPT) first, respectively. In this case, we would like to reduce the makespan in order to finish the processing of all tasks in minimum time. Also, there is no priority assigned to any task, therefore, optimizing the completion time of individual task is of no interest in this case. Considering this, the problem addressed, in this case, can be defined as

$$R_m | C_{max} \quad (6.15)$$

### 6.7.1 Longest Task First Scheduling (LTFS) Algorithm

As discussed earlier, the problem of finding an optimal solution to map all incoming tasks to the available machines is an NP-Hard problem, even if only two parallel machines are available in the system [174]. This motivates the design of heuristic methods where with

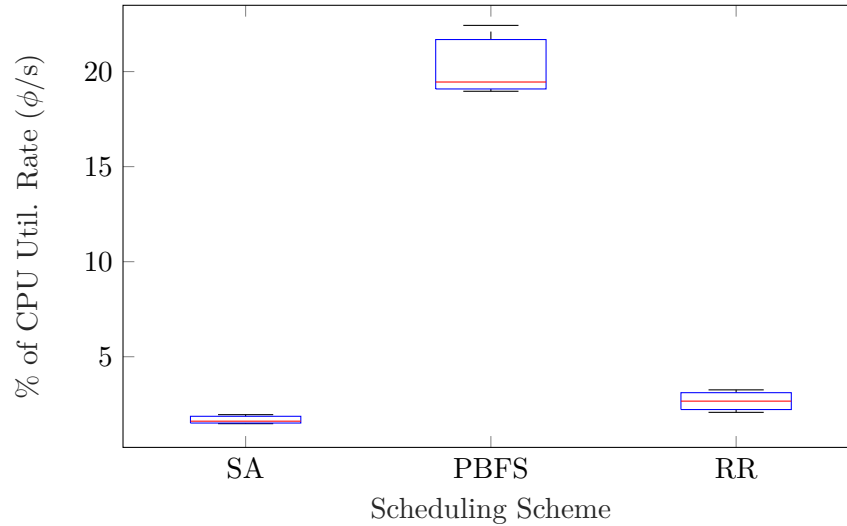


Fig. 6.3: Case1, percentage of CPU utilization rate ( $\phi/s$ ) for Simulated Annealing (SA), Pseudo Best Fit Scheduling (PBFS) and Round Robin (RR) scheduling schemes

moderate computational effort can produce a solution that is reasonably close to optimal. The extreme research in an optimization problem in the last decades has provided some heuristics solutions to solve these problems in polynomial time. In this case, we revisit the famous heuristic solution LPT, proposed in 1969 (by Graham) [176], and which is still considered as the optimal one for identical parallel machines with the objective of minimizing the makespan. At first, the LPT rule sorts the list of tasks in order and then schedule them one after another. Initial probabilistic studies to analyze the validity of the LPT rule considered the tasks with independent processing time, generated from some given probability distribution. Detailed analysis of these problems strengthens the intuition that LPT is a reasonable heuristic for scheduling problem  $P_m | C_{max}$ . In this case, the Longest Task first Scheduling (LTFS) algorithm is proposed, which is based on the LPT rule. The algorithm first sorts all the tasks present in the queue at a given time instant, in descending order of their processing time. After that, the scheduler takes a single task from the queue and schedules it on the available machine. In such a scenario, the shortest task will start processing in the end, hence it will be the last one to finish the process. As according to the LPT heuristics the shortest tasks are scheduled in the end, they can be used for load balancing on the machines. When there is no appropriate machine available to process the task, the task will remain in the queue and the scheduler will try to schedule it when any tasks leave the system.

The LPT heuristic solution is proved to provide a near-optimal solution. The maximum diversion of heuristic solution from the optimal one, even in the worst-case scenario is proved to be (Theorem 5.1.1 in [174]),

$$\frac{C_{max}(LPT)}{C_{max}(OPT)} \leq \left( \frac{4}{3} - \frac{1}{3m} \right) \quad (6.16)$$

where  $C_{max}(LPT)$  is the makespan obtained through the LPT rule and  $C_{max}(OPT)$  is the optimal makespan of the problem, which is possibly unknown. The above eq. (6.16), is the bound provided and proved by Graham for the LPT heuristics. However, empirical experiments have shown that the LPT performs much better in practice than

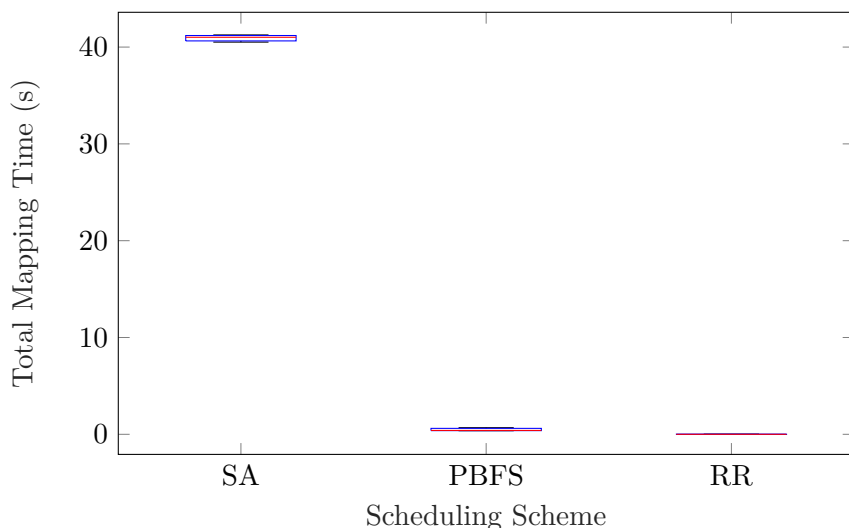


Fig. 6.4: Case1, total mapping time to map all cloudlets to VMs for Simulated Annealing (SA), Pseudo Best Fit Scheduling (PBFS) and Round Robin (RR) scheduling schemes

its worst-case scenario ratio. Under different studies, heuristic rules are proven to be asymptotically optimal. This worst-case bound is, however, inherently pessimistic and does not necessarily provide performance information of the method. Since the scheduling decision is based on a simple rule this solution is not computationally expensive, with time complexity of order  $O(n \log n + n \log m)$ , where  $O(n \log(n))$  is the time complexity to sort tasks,  $m$  is the number of machines and  $n$  is the number of tasks.

### 6.7.2 Results and Discussion

In this case, the dynamic arrival of cloudlets is considered where the cloudlets are arriving in the system according to a Poisson distribution. The total number of generated cloudlets is 500, which are generated at different arrival rates. The simulation will stop once all the cloudlets finish their services. The proposed algorithm, in this case, is LTFS that follows the LPT rule to schedule the incoming cloudlets, and the proposed scheme is compared with the SA and RR scheduling algorithms. Since the objective, in this case, is to reduce the maximum completion time, we first analyze its effect in fig. 6.6. For different arrival rates, we observed that the proposed LTFS is performing better than the other two, where the maximum time taken by the LTFS is around 200s, whereas it is around 400s for the other schemes. Since the solution obtained using SA does not perform significantly better than other schemes, it is assumed that the solution provided by SA is not the optimal one. SA might need more iterations to reach the near-optimal solution, which definitely will cost more resources and incur a delay. However, the effect of LPT in these results is significant, where LTFS minimizes the maximum completion at different arrival rates with a significant difference. The time taken by LTFS, in this case, is almost three times less than RR and SA, which is reduced further at higher rates. For the arrival rate of sixteen cloudlets per second ( $\lambda = 16$ ),  $C_{max}$  is around 50s for LTFS, 200s for SA, and 400s for RR. The reason for increased completion time at a lower arrival rate is the long waiting time at the scheduler for batch completion before it

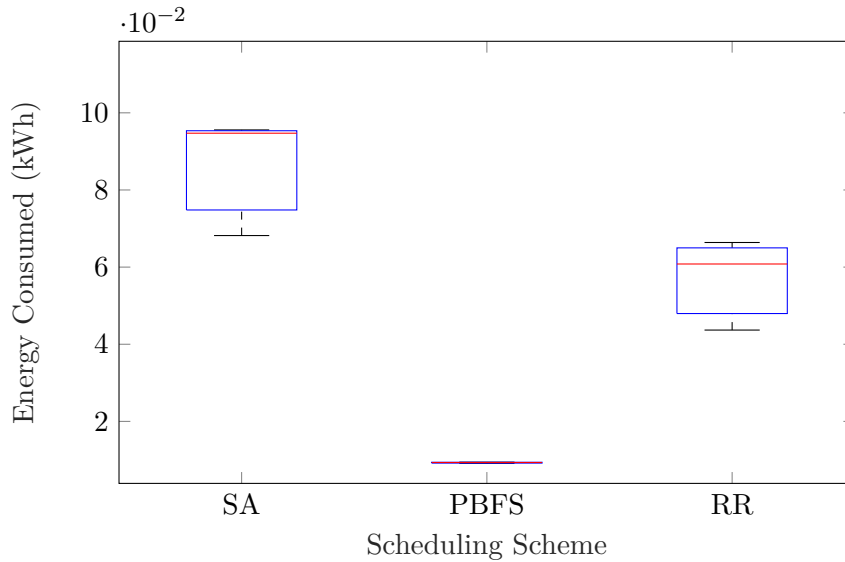


Fig. 6.5: Case1, total energy consumption for Simulated Annealing (SA), Pseudo Best Fit Scheduling (PBFS) and Round Robin (RR) scheduling schemes

could start scheduling them. However, at a high arrival rate, the cloudlets in a batch are collected faster and the scheduler can schedule them more frequently, reducing the delay.

Further, the complexity of schemes in terms of their mapping time is shown in fig. 6.7. The complex algorithm of SA took more time to schedule the given cloudlets in comparison to RR and LTFS. But here we could see that the mapping time of cloudlets in case2 is much lower than case1 and it is due to mapping a smaller number of cloudlets at a given time. In case1, 500 cloudlets were to be mapped at a given time instant, which increased the solution space for SA whereas, in case2 cloudlets are arriving and being mapped in small batches; hence, their mapping requires less time. However, in both cases, RR and LTFS perform much better than SA as they use simple mapping policies based on heuristic rules. Another important point to note here is that the mapping time, in this case, is not varying with an increasing number of cloudlets per second ( $\lambda$ ); this might be because before starting the scheduling process for a new batch, the cloudlets from the previous batch have been scheduled and started their processes already, or there might be a smaller number of cloudlets left in the queue.

The effect on the processing time of the cloudlets is also analyzed for all schemes. The sum of processing time of all cloudlets is presented in fig. 6.8 and the individual processing time of cloudlets are shown in fig. 6.9. From fig. 6.8 we observe that the total processing time of cloudlets is increasing almost linearly for SA and RR, with increasing arrival rate, as with increased number of cloudlets in the system, congestion in the system increases causing a delay in task processing. However, the strange behavior of constant processing time is observed for LTFS. This again could be due to the fact that, before scheduling a new batch, all previous cloudlets in the system might have been allocated. This is further analyzed from the results of the cloudlets' individual processing time.

In the simulation, a total of 500 cloudlets were generated dynamically, hence there is a probability that at high arrival rate, cloudlets arriving later in the system will have to wait longer before execution. The cloudlets processing already in the system might add

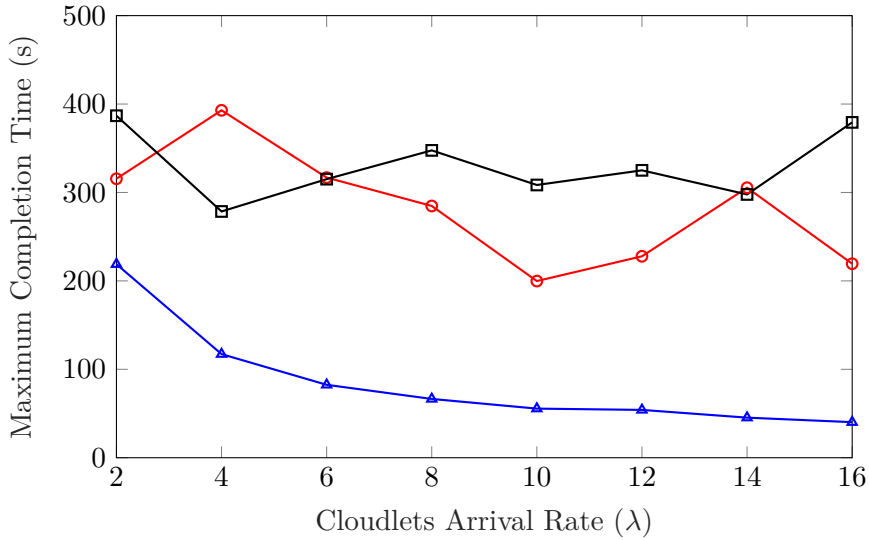


Fig. 6.6: Case2, maximum completion time of cloudlets. Legends: —○— SA, —□— RR, and —▲— LTFS

a delay in execution on tasks arriving later. Especially in the case of LTFS, where the over-utilization of resources is not allowed, cloudlets might need to wait longer. However, in SA and RR, the arriving task may be scheduled soon after the batch completion, but due to over-utilization of core resources, cloudlets contend for the resources which in turn affect their processing time. But, as we already observed from fig. 6.8, cloudlets in the LTFS case are processed fast enough and do not encounter any congestion. So, the effect of resource contention and task density is further analyzed using the individual processing time of cloudlets. Fig. 6.9 shows the processing time of an individual cloudlet for the arrival rate of 16 cloudlets per second. We can see that for SA and RR schemes the processing time of an individual cloudlet is growing exponentially. For a total of 500 cloudlets, cloudlets till 300 have almost similar processing time in all three cases. Afterward, however, the processing time of cloudlets starts growing for SA and RR algorithm, whereas there is a negligible effect in the LTFS scheme.

Another metric of interest is the operating and management cost of the system, which is based on power consumption. The total power consumed by the system over time is observed and energy consumed is calculated. If we analyze the graph in fig. 6.10, we can observe that the energy consumption of the proposed scheme is the lowest among all. Also, with increasing arrival rate the energy consumed in all cases is decreasing significantly in the LTFS scheme, where the energy consumption drops from  $8 \times 10^{-2}$  kWh to  $2 \times 10^{-2}$  kWh for different arrival rates. Whereas, the total energy consumption in SA and RR cases is almost similar and lies in the range of 0.10 kWh to 0.13 kWh. The reduction in consumption at a higher rate is due to the frequent scheduling of cloudlets, where the batch is completed quicker with fast arriving cloudlets. The proposed scheme in this case, is again the simple decision-based policy, which tries to optimize the completion time of the cloudlets. The minimal completion time of the LTFS scheme and less complex algorithm reduces the total time required to map and process all cloudlets, which as a result reduces energy consumption.



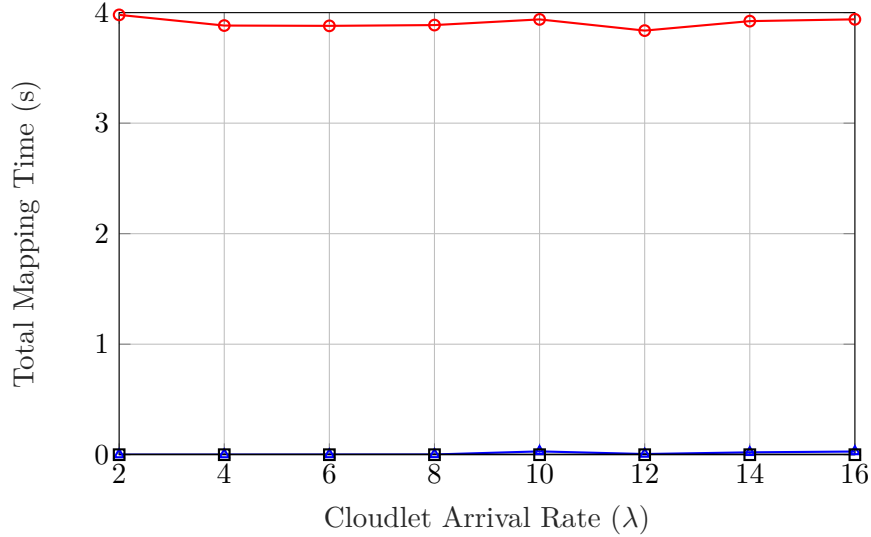


Fig. 6.7: Case2, total mapping time to map all cloudlets to VMs. Legends:  $\text{---}\circ\text{---}$  SA,  $\text{---}\square\text{---}$  RR, and  $\text{---}\triangle\text{---}$  LTFS

## 6.8 Case Study III

Results obtained from previous cases are used to derive a new improved scheduling scheme proposed in this case. The Greedy Best Fit Scheduling (GBFS) algorithm proposed here considers the objective of reducing maximum completion time and maximizing resource utilization to obtain the gains of both. Combining the rules of sec. 6.6 and sec. 6.7, this algorithm acts greedily in finding the near-optimal solution. The GBFS algorithm does not search for the best solution; instead, it makes the scheduling decision based on defined rules and conditions. This scheme aims to maximize the rate of CPU utilization and reduces the overall completion time using a fairly simple approach. In this case, the power consumption of the servers is also monitored to avoid exceeding the server's power threshold. Most of the servers in data centers and cloud networks have a power threshold limit to avoid the server breakdown due to system uncertainties and unpredictable workload surges. Hence power monitoring and management is an important aspect in these environments. Moreover, an additional system constraint of the task's due time has also taken into account to make the system comparable to the real environment. The due time related problems typically have maximum lateness as the objective. The lateness of the task ( $l_k$ ) can be investigated by comparing the completion time of the task ( $C_k$ ) with its due time ( $d_k$ ) eq. (6.6). It implies that to minimize the lateness of any task, the task must finish its processing before reaching its due time. Here it should be noted that the due time is not the deadline for the task. So, if the task does not start processing before its due time, it will not be removed from the queue or considered lost; instead, the delay will be evaluated considering its lateness. According to [174], finding an optimal solution with lateness as an objective requires to schedule all tasks considering minimizing their completion time. Hence, the heuristic solution for both objectives, lateness and maximum completion time, is to apply the LPT rule. This case further analyzes the performance of each scheme in more detail

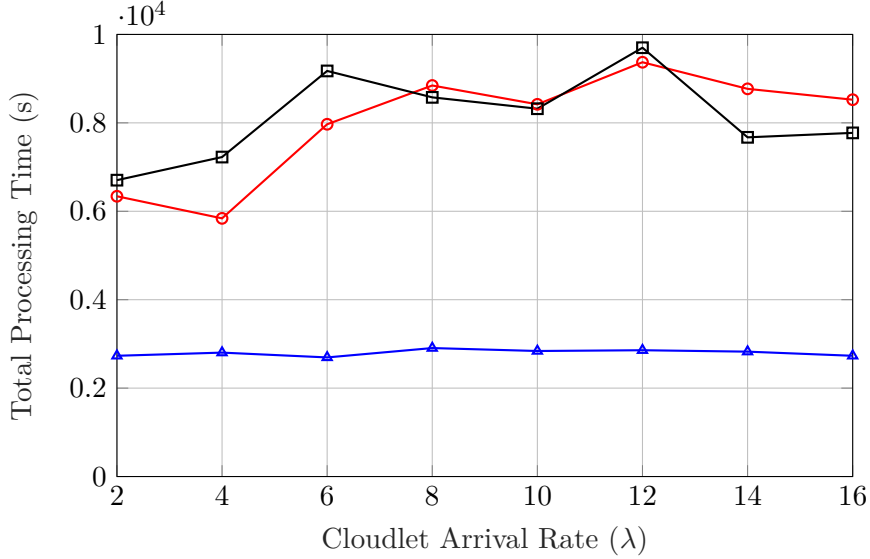


Fig. 6.8: Case2, sum of processing time of all 500 cloudlets. Legends:  $\circ$  SA,  $\square$  RR, and  $\triangle$  LTFS

considering several performance metrics. Here we can define the problem for this case as,

$$R_m | d_k | \phi, C_{max}, l_k \quad (6.17)$$

### 6.8.1 Greedy Best Fit Scheduling (GBFS) Algorithm

The algorithm is based on several steps to include the effect of all heuristics applied. Before starting scheduling, the algorithm first applies the LTFS (sec. 6.6) rule and sorts the available cloudlets in the system. The cloudlets in the system queue are sorted in descending order of their processing time. Also, the limited power resources of the servers are used by capping the server at a given power threshold ( $\delta$ ) during task scheduling. The power consumption of the servers as a function of CPU utilization is estimated using a linear power model, which is the function of CPU utilization (eq. (6.9)). Servers those already have reached their power threshold are not considered further in the scheduling process until their power consumption is reduced after finishing some running tasks. Hence, the algorithm uses the concept of power capping of the server to avoid excessive power consumption. The proposed scheduler first selects the servers having power consumption under the threshold limit,

$$S_\delta = \{s_i | s_i \in S \wedge P_{s_i} < \delta\} \quad (6.18)$$

where  $S_\delta$  is the list of selected servers,  $P_{s_i}$  is the power of the server  $s_i$  and  $\delta$  is the power threshold. Afterward, the scheduler starts scheduling by detecting the available and required resources. It first takes a task  $t_k(c_k, \vec{l}_k)$  from the sorted queue and allocates it based on Pseudo Best Fit Scheduling (PBFS) (sec. 6.7). The algorithm does not look for the optimal best fit solution; instead, it allocates the task to the first available machines satisfying the given conditions. After extracting the available servers for scheduling and task from the sorted queue, it extracts the list of VMs capable of processing the selected

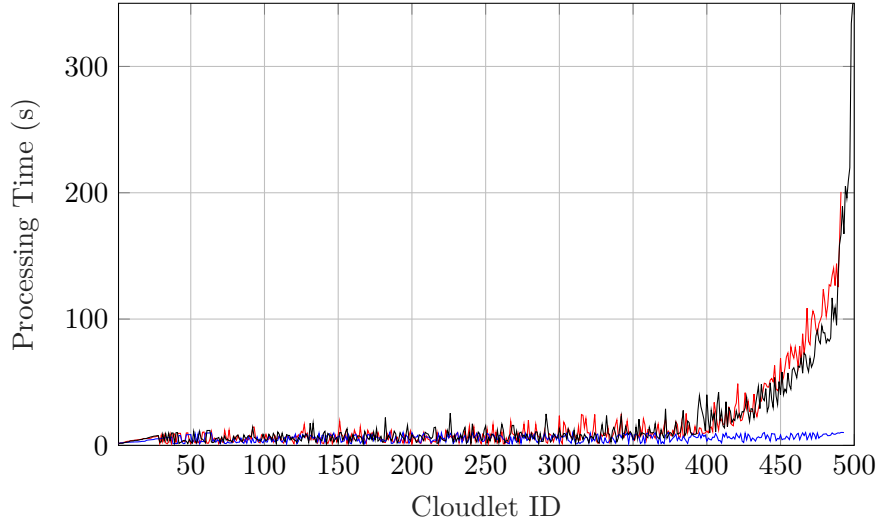


Fig. 6.9: Case2, processing time of individual cloudlet (for total of 500 cloudlets) for  $\lambda = 16$ . Legend:— SA, — RR, and — LTFS

cloudlet. The residual core capacities of the VMs on selected servers in  $S_\delta$  are detected and compared with the required resources of a cloudlet,

$$V_{sel} = \{v_{s_i,j} \mid v_{s_i,j} \in V_{s_i} \wedge c_{s_i,j} \geq c_k, \forall s_i \in S_\delta\} \quad (6.19)$$

where  $V_{sel}$  contains the list of VMs that have enough available free resources. A similar mapping objective of minimizing the waste resources as in case2 is used, and the optimal cloudlets to VM mapping solution is obtained using equation eq. (6.14). This function assigns an optimal VM,  $VM_{opt}$ , to each cloudlet from the list  $VM_{sel}$ . The scheduler repeats the process of cloudlet scheduling until all the generated cloudlets are processed. It further schedules the waiting cloudlets in the queue when any finished cloudlet leaves the system. Hence, the scheduler does not check the availability of resources at every instant, which reduces the computational overhead; instead, it looks for the resources when there is a possibility of free resource available or when any new batch (of cloudlets) arrives.

Further, the mechanism of the proposed GBFS scheme is explained using the pseudo-code in fig. 6.11 and flow chart in fig. 6.12.

### 6.8.2 Results and Discussion

The system is evaluated for dynamic cloudlets arrival with the system parameters and configurations as mentioned in tab. 6.1, tab. 6.2, and tab. 6.3. At first, the overall completion time of all cloudlets is discussed using the results presented in fig. 6.13. These results are for dynamic cloudlets arrival with different mean arrival rates. The worst performance of SA can be observed from the graph, where it takes approximately 3-4 times longer than the proposed algorithm to process all cloudlets. Round Robin in this case performs a little bit better than SA, compared to case1 of static cloudlets where it performs almost similar to SA. This shows that SA performance is further degraded in a dynamic situation with continuous task arrivals. The effect of LPT is significant here, as it reduced the completion time for GBFS significantly at all arrival rates. Much lower

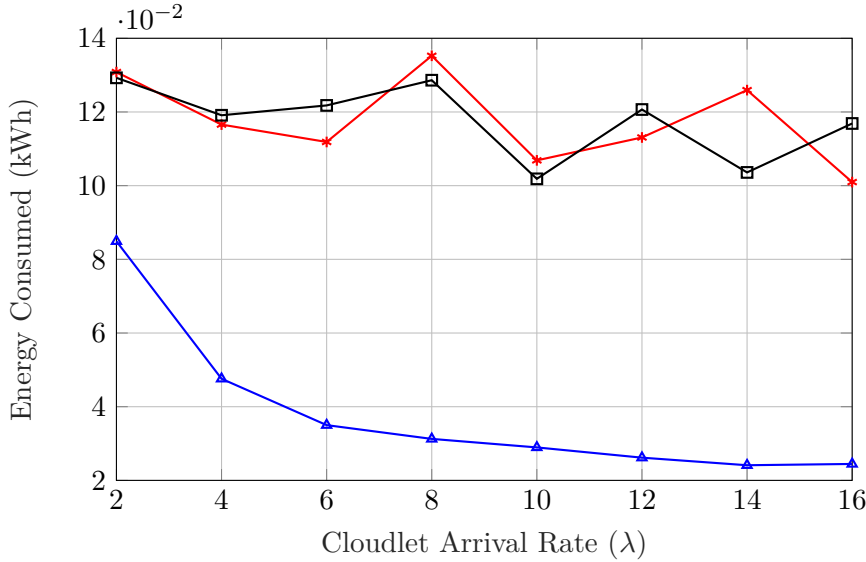


Fig. 6.10: Case2, total energy consumption. Legends:  $\circ$  SA,  $\square$  RR, and  $\triangle$  LTFS

completion time at a higher rate is observed due to frequent scheduling with a large number of cloudlets coming into the system in less time. This might, however, create a bottleneck if the arrival rate of tasks is too high.

Next, we analyze the usage of CPU resources by comparing the rate of CPU utilization in each scheme. Fig. 6.14 shows the CPU utilization per second for all three schemes, from which we can observe that the GBFS uses more CPU resources per unit time than other scheduling methods. However, this is less in comparison to case1, which might be due to the dynamic arrival of cloudlets. In case1, there were more cloudlets to be processed at a specific moment, whereas in this case only a small number of cloudlets (one batch and cloudlets in the queue if present) are to be processed at any given time. The CPU utilization rate was around 20% in case1, which is reaching a maximum of 15% in this case for the proposed algorithm. However, most of the resources are wasted in SA and RR schemes, where the CPU utilization is only 2-3% per second, SA and RR being unaware of the resource utilization causing delay and increased processing time for the cloudlets. In this way, both algorithms process a similar number of cloudlets in a much longer time, reducing the CPU utilization rate. This behavior of SA and RR remains similar for different arrival rates, whereas the GBFS is improving the server resource utilization for larger numbers of cloudlets in the system.

To see if this low utilization rate in SA and RR algorithms is the effect of longer processing time of cloudlets, we analyze the sum of processing times of all cloudlets in the system. Fig. 6.15 shows the sum of processing time of all cloudlets of case3, where the processing time for SA and RR is increased a little compared to case2 but the trend is almost similar (a linear increase). On the other hand, the processing time of GBFS is again a constant, which is because any cloudlet will take the same amount of resources even if arriving at different arrival rates, as there is no effect of resource contention and core sharing in the proposed scheme. Also, it is expected that there would be a little delay in the execution of arriving cloudlets, as the cloudlets arrived earlier in the GBFS system might have finished before scheduling a new batch. While the effect of core sharing with a larger number of cloudlets is visible in SA and RR, the sums of

Fig. 6.11: Pseudo code for GBFS Algorithm

```

1: procedure GBFS( $T, S, \delta$ )
2:    $T \leftarrow$  Tasks in queue
3:    $S \leftarrow$  Set of servers in a network
4:    $\delta \leftarrow$  Power threshold of a server
5:    $task(t_k)$  finish listener  $\leftarrow$  false
6:   while  $T$  is not empty do
7:      $t_k \leftarrow$  Task from queue
8:      $\vec{S}_\delta \leftarrow$   $S_i$  for  $P_{s_i} < \delta$ 
9:      $\vec{VM}_{sel} \leftarrow c(v_{s_i,j}) \geq c(t_k)$ 
10:     $VM_{opt} \leftarrow \min(c(v_{s_i,j}))$  in  $\vec{VM}_{sel}$ 
11:    if  $VM_{opt} \neq$  empty then
12:      assign  $t_k \rightarrow VM_{opt}$ 
13:    else
14:      if  $\vec{VM}_{sel} \neq$  empty then
15:        goto 9
16:      else
17:        goto 6
18:      end if
19:    end if
20:  end while
21:  goto Task finish listener
22:  Task finish listener :
23:  while  $t_k$  finish listener = true do
24:     $t_k$  finish listener = false
25:    goto 6
26:  end while
27:  return
28: end procedure

```

$\triangleright$  Trigger when any task finishes  
 $\triangleright$  No task left in queue  
  
 $\triangleright$  All Tasks finished

processing times in these two schemes are increasing with increasing arrival rate, as with more cloudlets resource sharing will observe more contention. Hence, we see that over-utilization of resources can increase the processing time, but on the contrary, the waiting time for the cloudlets could grow exponentially at a very high arrival rate.

As mentioned earlier, we analyze the performance in terms of the lateness of cloudlets, as well. The sum of the lateness of all cloudlets at different arrival rates is shown in fig. 6.16. Since lateness of the cloudlet is the delay in completion of a task concerning its defined due time, we consider the cloudlets finishing with lateness less than zero were scheduled efficiently. From fig. 6.16 we can see that almost all the cloudlets in the GBFS scheme finished before their due time, and lateness in SA and RR is increasing with increasing number cloudlets in the system. This behavior corresponds to the processing time of the cloudlets; since the cloudlets at SA and RR took longer to process, they are finishing late and consequently are considered delayed. On the contrary, cloudlets in GBFS are processed fast and they finish their processing before reaching their due time. This also shows that for time-critical applications, where tasks are associated with deadline, the number of cloudlets expired in the system will be greater in SA and RR due to increased latency imposed by these schemes.

After analyzing the cloudlets related performance metrics, the system computational cost for these schemes is evaluated. Results are shown in fig. 6.17 reporting the total mapping time of each scheduling scheme required to map all cloudlets entered into the system, to the available VMs. Since we know that the SA is an optimal search based algorithm and RR and GBFS are condition-based scheduling policies, the computational complexity of SA is very much higher ([173]) than the other two ([174]). Furthermore, the complexity of SA increases more with dynamic cloudlets arrival as compared to static ones. Hence, SA in this case took maximum time to search for an optimal solution and the other two schemes took almost similar time to map all the cloudlets, which is nearly a few seconds.

One of the important aspects for service providers is the system cost that we also tried to reduce using less complex energy-efficient scheduling. Fig. 6.18 shows the energy consumed in the mapping and processing of all generated cloudlets for case3. Since with the dynamic arrival of cloudlets, scheduling is to be initiated several times as compared to the static environment, where all the cloudlets are generated at start and the scheduler has all the details of cloudlets, this one time scheduling makes it less resource consuming in static scenarios. However, frequent scheduling using complex schemes costs for more overheads and more system resources. This we can observe from fig. 6.18, where the SA consumed high energy compared to the proposed GBFS scheme. And we also observe the high energy consumption with RR, which is a relatively very simple scheduling method. High consumption in the RR case is due to inappropriate scheduling, although it maps the task to the VM using a simple algorithm, but it does not consider the available and requested resources to optimize the scheduling solution. In contrast, the GBFS algorithm uses heuristic rules and also considers the available and requested resources to improve the CPU utilization and reduce the system cost overall. Since energy consumed is dependent on the time taken to process all cloudlets, therefore the consumed energy in GBFS is further reduced at a high arrival rate as it processes all generated cloudlets in less time.

## 6.9 Conclusion

Cloud computing environments require energy-efficient algorithms and maximum resource utilization to improve the network's reliability and QoS at a low cost. Since task scheduling is an NP-Hard problem, most of the existing algorithms use evolving heuristic approaches to obtain the optimal solution. These meta-heuristic solutions for scheduling are complex, and also their complexity increases exponentially for high density networks; therefore they are not suitable for critical applications or making real-time scheduling decisions. This research work is particularly focused on optimizing the resource provisioning and processing time in the cloud. Through different case studies, a relatively simple and improved scheduling algorithm, *Greedy Best Fit Scheduling*, is proposed. The proposed method in sec. 6.8 is derived from the observations and results of case studies carried out in sec. 6.6 and sec. 6.7. The GBFS algorithm is less complex, as it uses a decision-based heuristic approach to find the sub-optimal solution. It allocates the tasks, to the VM having minimum available resources and hence it attempts to maximize the utilization the available capacity of an individual server. The overall processing time is also reduced by first arranging the tasks in a specific order. The GBFS further considers the power threshold of the server while scheduling to secures the server's unused resources for unpredictable high traffic density.

Listed below are some major findings of this research,

- Resource utilization in cloud computing can be improved using the best fit strategy. The Pseudo Best Fit algorithm reduces the resource wastage by trying to allocate all small unused chunks of resources. The time complexity of this approach is also relatively less than that of meta-heuristic methods (such as SA). And reason for this is that PBFS does not search for the optimal solution, instead it allocates a VM to the task when the given conditions are satisfied.
- Using the simple LTFS rule, the overall processing time of all tasks can be reduced. The LTFS rule is based on arranging tasks in a specific order, which is descending based on their processing time in this research. This is one optimal way to improve the response time of the system when there are a number of tasks with different characteristics in the system queue. Results show that in system with parallel machines of different speed and capacity, this rule reduces the average processing time and hence the cost and energy consumption of the system.
- The experiments were also carried out at different task arrival rates (in sec. 6.7 and sec. 6.8) which could help in choosing the optimal arrival rate for a given system. At very low mean arrival rate processing time and energy consumption are higher, whereas this could be improved with increasing arrival rate such as for  $\lambda = 8$  to  $\lambda = 16$ . However, arrival rates much higher than this could cause long waiting times for the tasks, especially in the proposed schemes where the tasks have to wait for free resources to be allocated.
- Combination of different heuristic rules can provide acceptable results in complex system. It reduces the time and computational complexity of the scheduling scheme and improves the system performance. The solutions obtained using a heuristic rule might not be the optimal ones, but they provide quick and good enough solutions for task scheduling, where long scheduling delay cannot be tolerated.

Results of all presented case studies shows that the heuristic rule based scheduling performs well among existing Round Robin and Simulated Annealing approaches. It reduces the scheduling complexity, improves the system cost by utilizing the maximum server capacity and also improves the processing time of individual tasks by avoiding the over-utilization of the server resources. Running a minimum number of servers (using the best fit strategy) in the system further reduces the power consumption and hence the energy consumed by the overall system. Hence, it can be concluded that the GBFS algorithm is an efficient choice for both service provider and consumer from economical and performance points of view. In the future, it could be of interest to further improve the objective function by introducing constraints such as task priority and delay. Different applications have different constraints and objectives, hence these metrics specific to any application can be considered for a particular scenario. Also, the performance of schemes were not analyzed for time critical applications. But results shows that the more number of tasks are delayed in SA and RR compared to GBFS, which is an indication that the proposed method can perform well for time critical applications as well. The performance of the proposed scheme can further be validated by doing experiments on real system.

This research work provides an initial step toward using multiple rules to achieve network gains; different heuristics rules can be used to find the optimal combination for any system. Moreover, this research work has just targeted the assignment of incoming tasks; however, the algorithm could also be used for VM migration and server consolidation. Reallocation of running tasks among different VMs and migration of VMs for consolidation using the proposed scheme may provide efficient solutions.



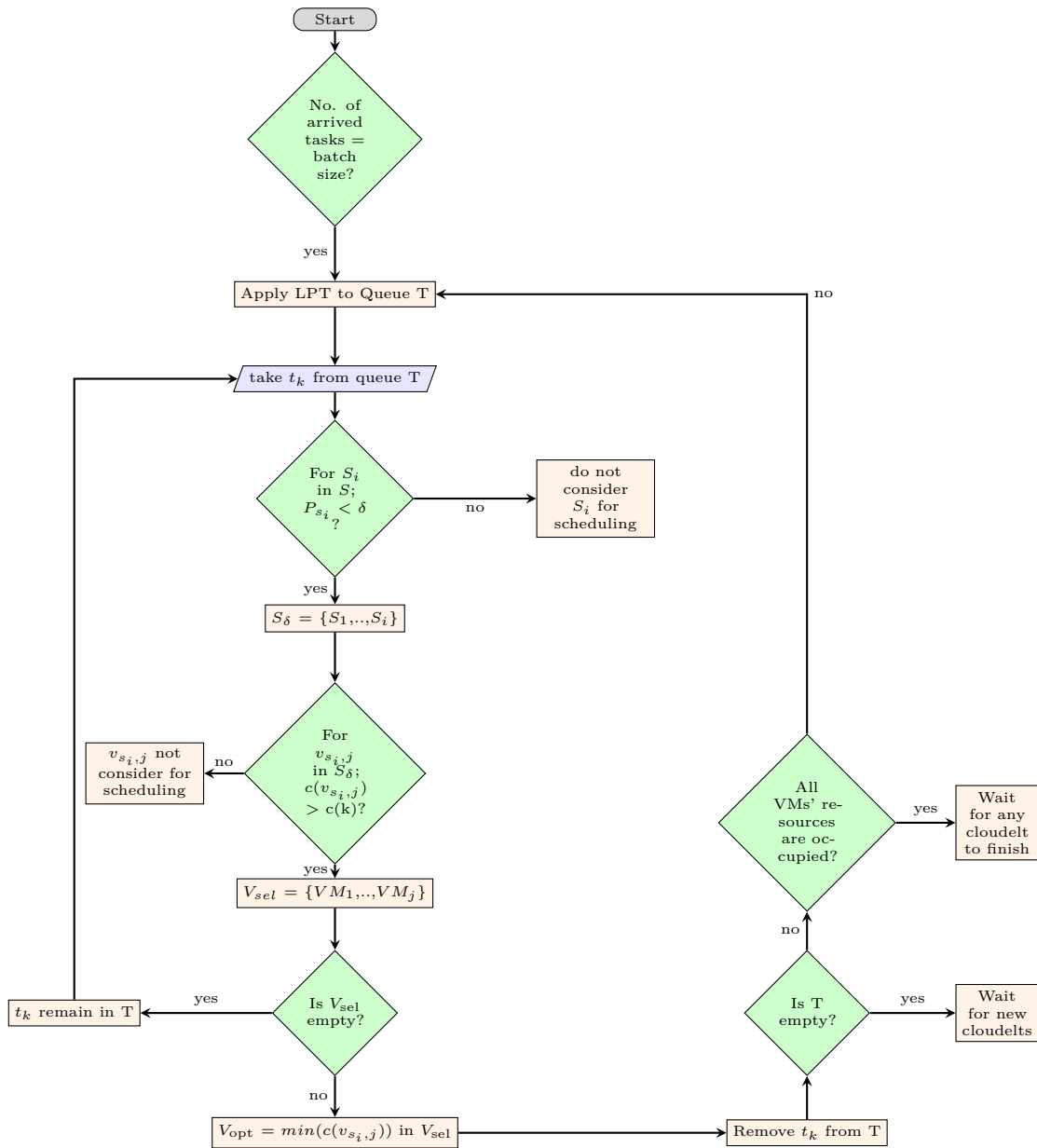


Fig. 6.12: Flow chart of the Greedy Best Fit Scheduling (GBFS) Algorithm

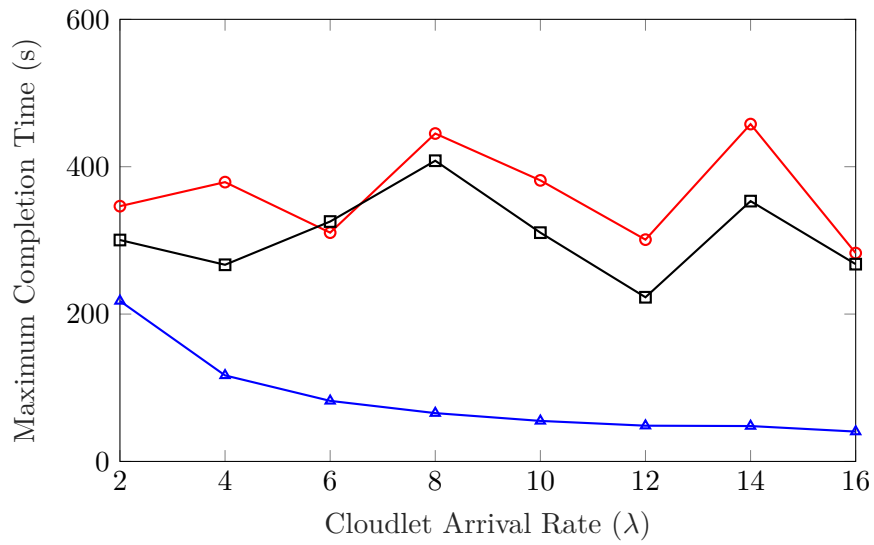


Fig. 6.13: Case3, maximum completion of cloudlets. Legends:  $\circ$  SA,  $\square$  RR, and  $\triangle$  GBFS

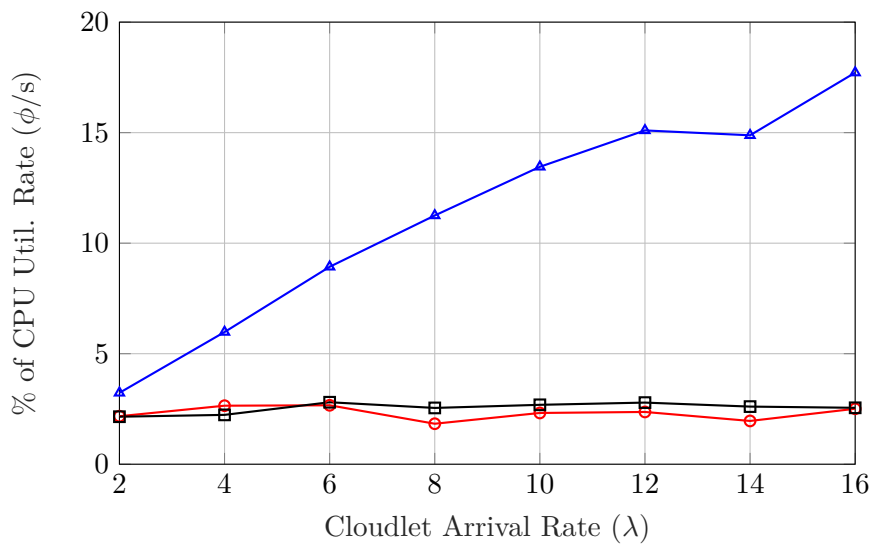


Fig. 6.14: Case3, percentage of CPU utilization rate ( $\phi/s$ ). Legends:  $\circ$  SA,  $\square$  RR, and  $\triangle$  GBFS

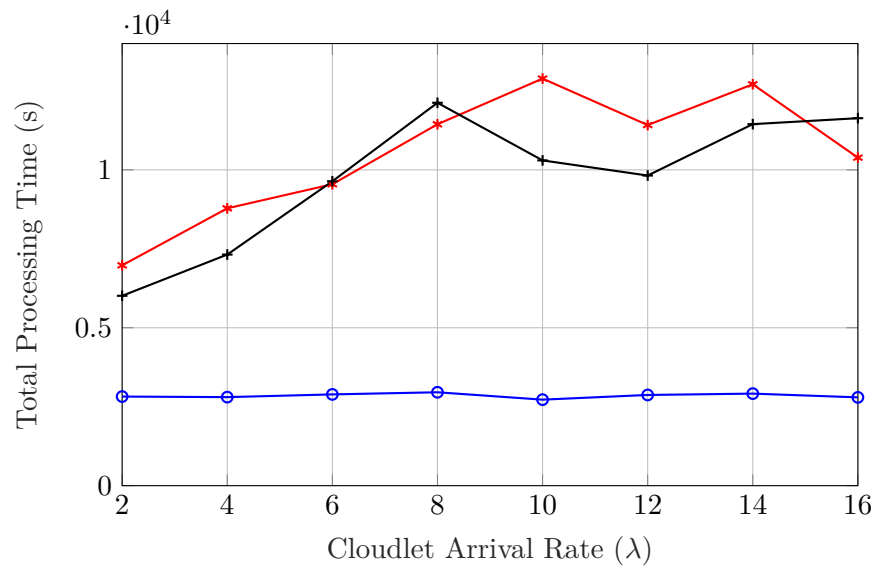


Fig. 6.15: Case3, sum of processing time of all cloudlets. Legends: —\* SA, —■ RR, and —○ GBFS

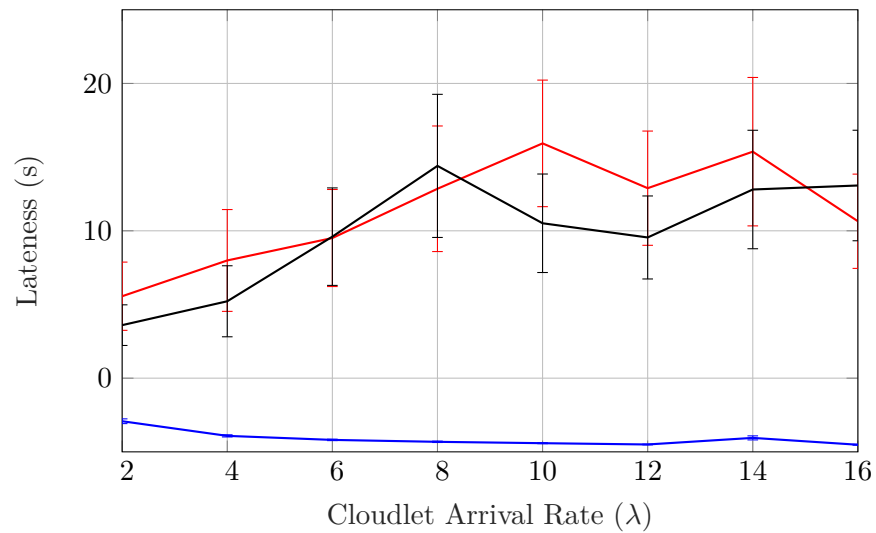


Fig. 6.16: Case3, sum of lateness of all cloudlets. Legends: —\* SA, —■ RR, and —○ GBFS

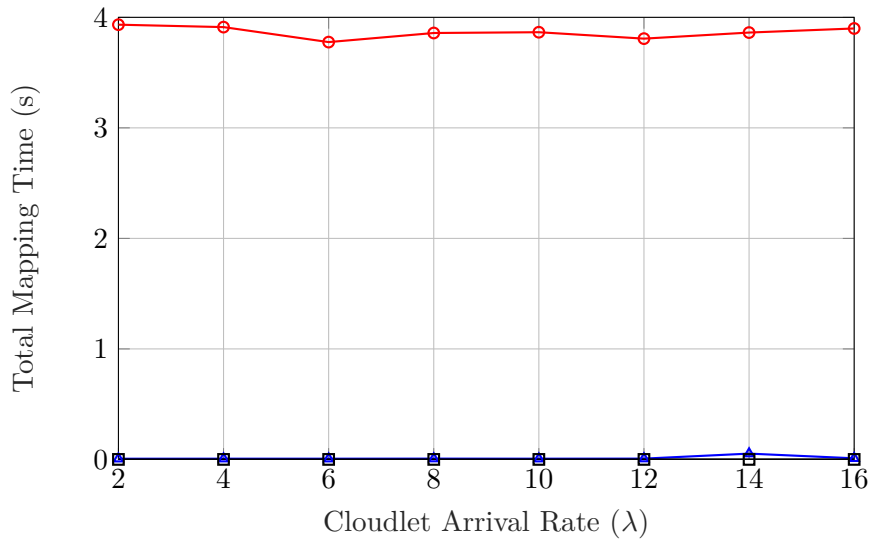


Fig. 6.17: Case3, total mapping time to map all cloudlets to VMs. Legends:  $\circ$  SA,  $\square$  RR, and  $\blacktriangle$  GBFS

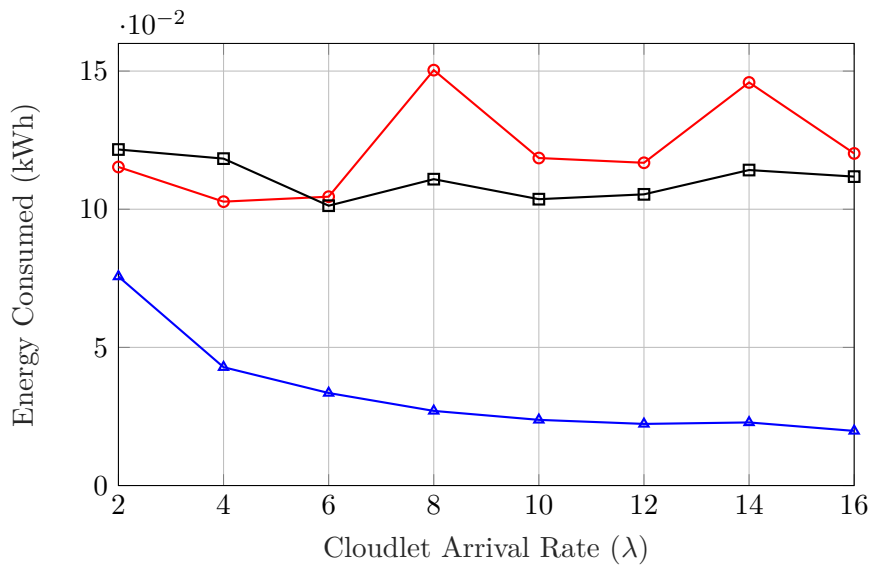


Fig. 6.18: Case3, total energy consumed. Legends:  $\circ$  SA,  $\square$  RR, and  $\blacktriangle$  GBFS

# 7 Conclusion

This thesis is mainly focused on the challenge of improving resource utilization in data centers and the cloud with reduced energy consumption. Two basic techniques to achieve this goal are energy-aware or power-aware algorithms and energy-efficient mechanism and methods. The power-aware algorithms require precise power measurement and estimation to make the effective power-aware decision at run-time. This domain is discussed in the first research phase of this thesis where detailed research is first carried out on challenges and later an improved power model is developed. The accurate power estimation using the proposed model can make the power-aware decision in cloud management more effective. The research also deals with the second technique that rely on energy-efficient algorithms. Attention is given to the energy-efficient scheduling scheme in this research. An efficient, less complex and fairly simple scheduling scheme is presented in the second research phase of the thesis. However, a trade-off between waiting time and performance has to be made at a higher arrival rates.

## 7.1 Thesis Contribution

With respect to the defined objectives, this thesis brings the following key contributions,

1. At first, a research survey on prior work of power modeling and measurement methods was done. It is a detailed study on how the server power modeling could be affected by various factors and what is the effect of virtualization on the server power profile. The survey categorized the research problems, approaches taken and key developments, of the literature into major themes to improve its readability. It also shed light on the use of power modeling in macroscopic analyses of the system. The survey further summarizes the useful tools and methods based on the statistical result of the studies considered during the survey. In short, the survey contains concrete information, starting with a deep analysis of the available methods and techniques to the summary of widely used and useful methods.
2. The next research contribution is the development of the power model for a server in a virtual environment by introducing resource monitoring at the virtual level. The proposed model uses the concept of resource monitoring at two different levels of architecture (host and virtual level), which helps in identifying the missing components in the development of the power model. The research work first identifies and collects different parameters that are later used to learn the system profile and extract the significant variables. A power model for the server is developed using different regression techniques. The accuracy of the proposed model shows improved results which can estimate the power precisely from 96 % to 99.1 %. This is 1 % to 4 % improved precision compared to some existing models. Thus, this research work provides a less complex and improved power estimation model for a virtualized environment. With these results, it is expected that the proposed method will help in making effective power-aware decisions.

3. In the second phase of this thesis, research focuses on the scheduling problem of a cloud computing environment. Considering the under-utilization and waste of server resources in data centers, an efficient scheduling algorithm is proposed based on the Greedy and BestFit strategies. Two objectives considered in this research for task scheduling are processing time and resource utilization. A step-by-step study was carried out, starting with the simple system and single objective, to analyze the gains in different cases. An improved scheduling scheme 'GBFS' is then proposed in the last case study that combine the gains of previous cases. The proposed algorithm finds the task to VM mapping solution in such a way that maximizes resource utilization and also reduces the total completion time. Hence, it reduces the power consumed by the system, as well. The proposed method is relatively a less complex approach compared to meta-heuristic methods in terms of time and computational complexity. Power capping of the servers is also considered in the proposed scheme to avoid sudden server failure.

## 7.2 Future Work

This research in scheduling and power modeling in the virtual environment does not end here with these contributions, on the contrary; it opens new ways to develop and implement algorithms based on these methods. Inaccurate estimation in power-aware algorithms can sometimes cause ineffective results. One such scenario is the use of a power estimation model for power capping of the server where inaccurate prediction could either overload the server or might waste some server resources. Thus, an improved power model could make more effective decisions in such cases. Moreover, the proposed power model has introduced resource monitoring at the guest level, which could be used in developing cost models (to charge cloud customers) by service providers. Currently, cloud customers are charged based on different cost models but none of them considers the actual resources consumed by the user, which is sometimes unfair to the user.

Moreover, the scheduling scheme using heuristic rules could prove useful in critical applications, where meta-heuristics fail to provide a quick solution. Since the proposed scheme reduces the processing time of the tasks, it could also help in improving the throughput and reliability of time-critical applications. The reliability of the proposed scheduling scheme can also be compared to the experimental setup to study its effectiveness in a real environment. This would help in analyzing further limitations of the scheme and opens area for further development.

Further possible extension of this work could be to use the proposed methods in different environments and for different applications that can provide a new ground to analyze the validity of these models. Some additional limitations might arise for the proposed models when using them in a different system environment and for different user applications. Further enhancements to improve the gains of the proposed models for a particular application or service can also be made.

# Appendices





# A Graph of Problems, Approaches and Developments (PAD)

The node graph (or mind map) below illustrates the detailed information about each Problem-Approach-Development (PAD) path (triad). These maps give an overview of estimated problem complexity: a problem tackled by numerous approaches is considered more complex (see eq. (3.2)). The three figures (Fig. A.1, Fig. A.2, and Fig. A.3) show the various approaches adopted in the literature, departing from a specific problem category, to obtain different developments. Given any one problem node, denoted by  $P_j$ , one or more of a number of approach(es)  $A_k$ , contribute (at least in part) to achieve the development  $D_l$ , where ' $j$ ', ' $k$ ' and ' $l$ ' represent the respective node numbers.

A description of each nodes' label is provided in Appendix B.

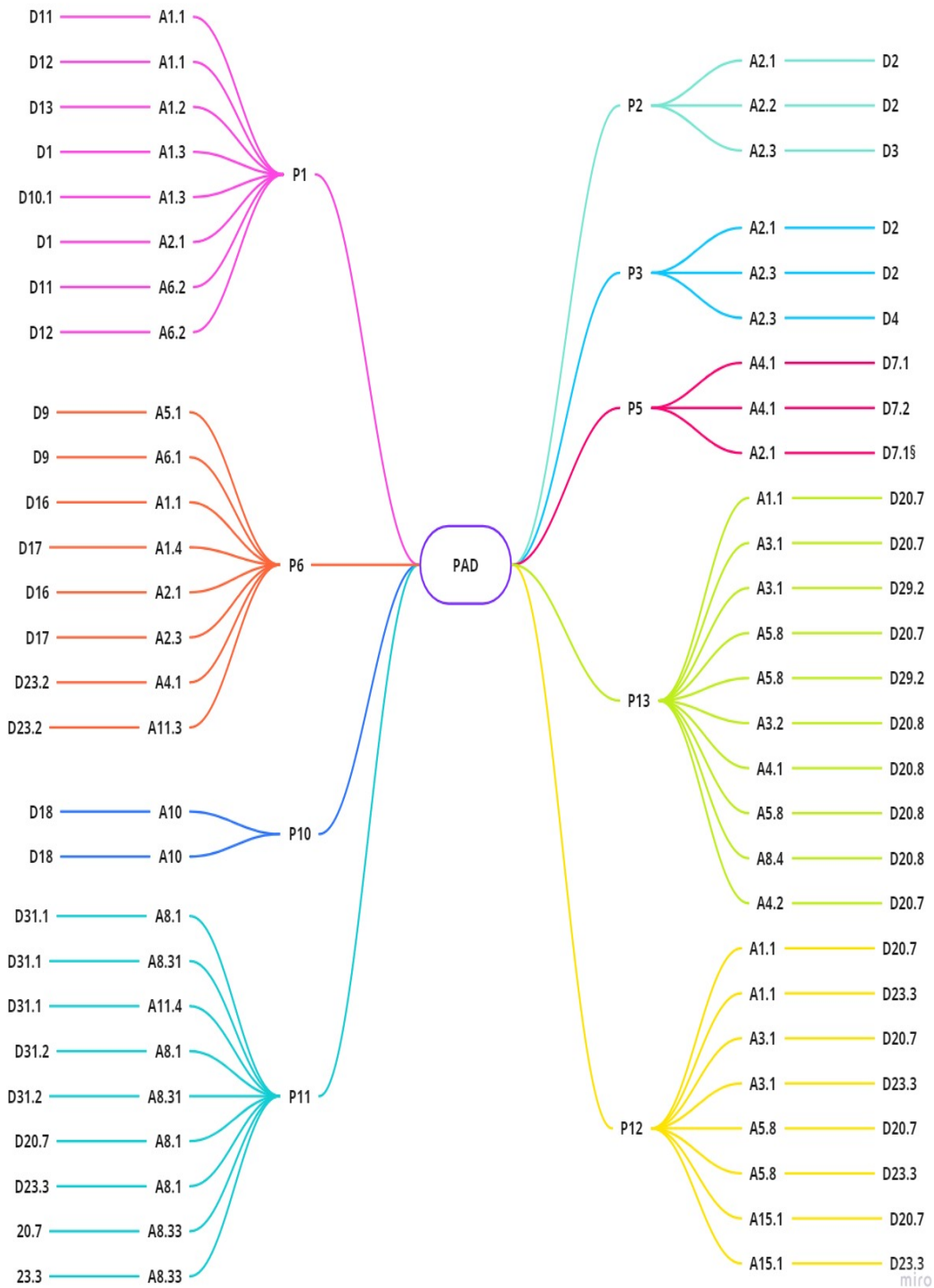
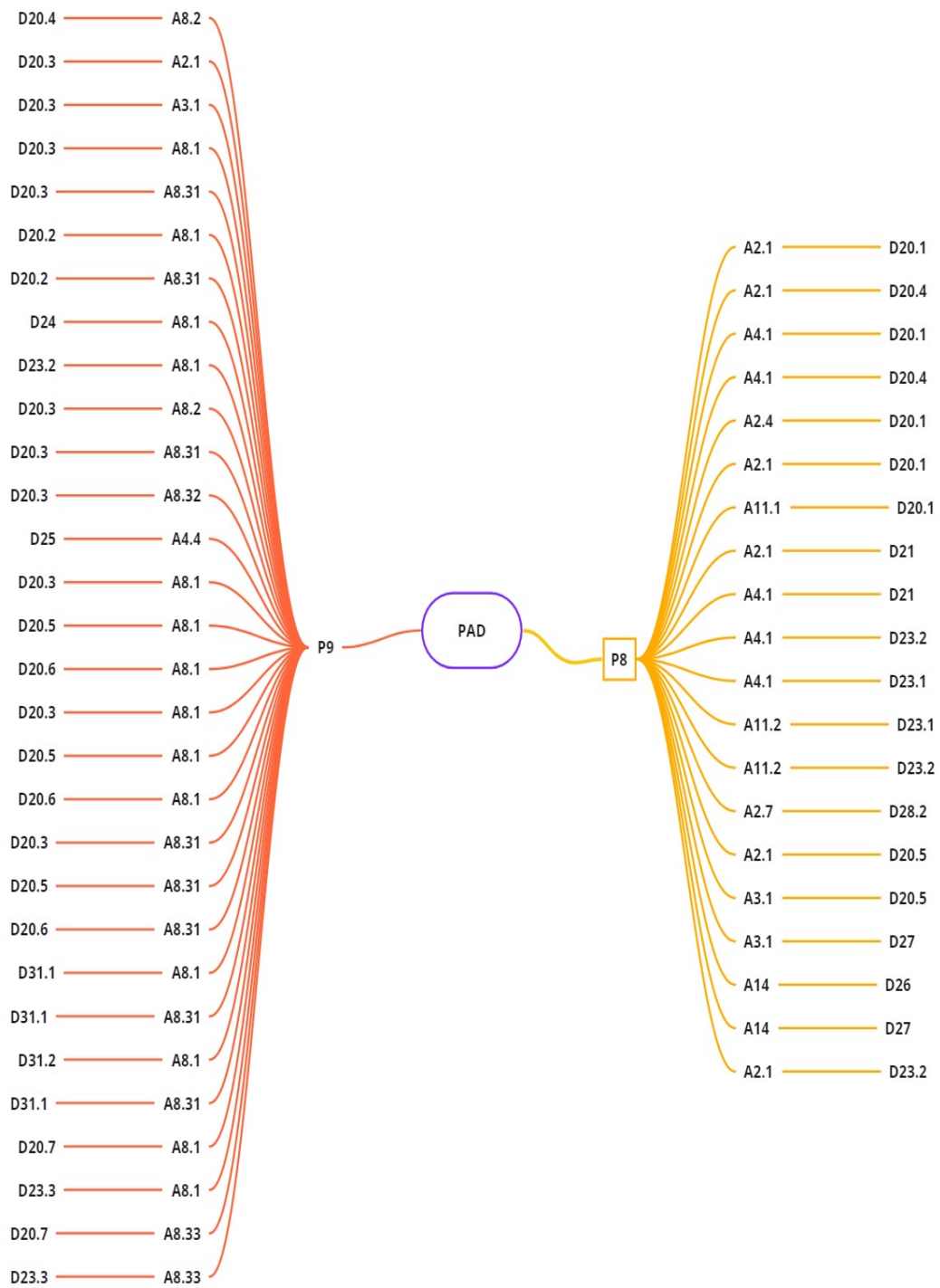


Fig. A.1: Problem-Approach-Development (PAD) Graph-part 1



miro

Fig. A.2: Problem-Approach-Development (PAD) Graph-part 2

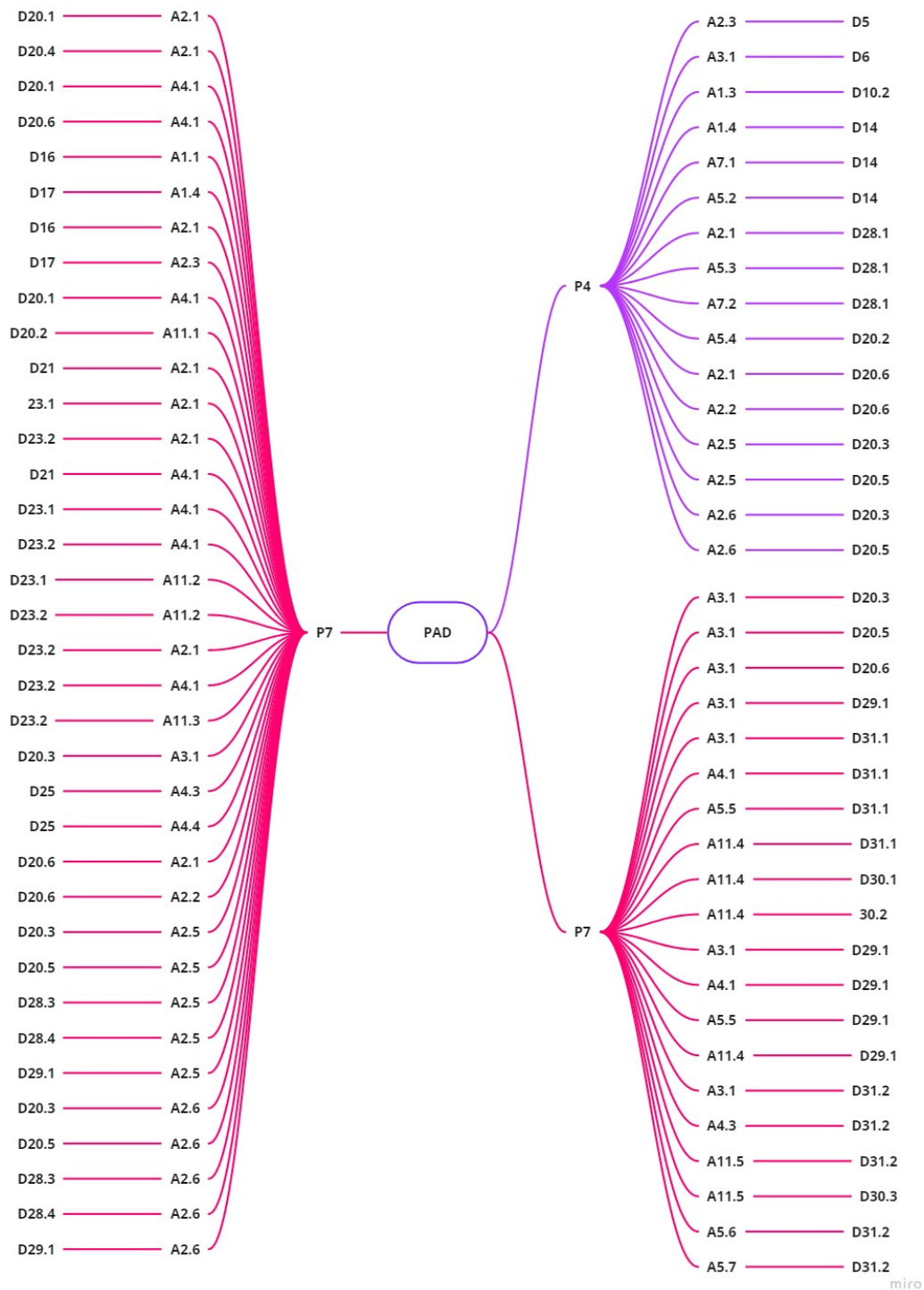


Fig. A.3: Problem-Approach-Development (PAD) Graph-part 3

## B Graph Nodes Description

Tables given below complement the node graphs in Appendix A. Description to the Problem, Approach and Development nodes are provided in tab. B.1, tab. B.2 and tab. B.3, respectively.

Category	Node no.
Architecture vs power for VMs and PMs	P1
Power vs virtualization genre	P2
Power vs virtualization technology	P3
Power vs workload type	P4
VM's power vs physical host's configuration	P5
Power vs number and size of concurrent VMs	P6
Power as a function of VM resource use	P7
Power as a function of host resource use	P8
System power attribution	P9
Power as a function of provision of a service by virtual systems	P10
Host power as a function of VM or container resource use	P11
Power as a function of temperature and frequency	P12

Power as a function of container resource use	P13
---	-----

Tab. B.1: Problem nodes, enumerated and showing category names

Category	Node no.	Sub-category
Managed resource allocation	A1.1	Frequency-scaling
	A1.2	NUMA
	A1.3	Simultaneous multi-threading
	A1.4	NIC data rate capacity or utilization
Resource isolation	A2.1	Processor-intensive workload
	A2.2	Memory-intensive workload
	A2.3	Network-intensive workload
	A2.4	Disk-intensive workload
	A2.5	CPU-core-specific workload
	A2.6	Memory-hierarchy-level specific workload
	A2.7	Multi-core: obtain a given CPU utilization in different core load distribution
Micro-architectural instrumentation	A3.1	Microarchitecture event instrumentation
	A3.2	Microarchitecture power instrumentation
Architectural instrumentation	A4.1	Processor utilization
	A4.2	Container utilization
	A4.3	Hard disk drive utilization
	A4.4	Instrumentation inside custom i/o device emulator
Workloads representative of real use	A5.1	TPC-W
	A5.2	File transfer
	A5.3	Floating-point operations
	A5.4	SPEC CPU2006
	A5.5	SPEC CPU2000
	A5.6	NPB
	A5.7	GCC
	A5.8	PARSEC

Identification of metric of energy efficiency	A6.1	J/Web interaction
	A6.2	hash/J
	A7.1	Network transmit bandwidth
	A7.2	Number of workload processes
Simple workload characterization	A8.1	Dynamic power only to guest VEs and static to host/privileged VM
	A8.2	Both static and dynamic power to guest VEs
	A8.31	Events occurring in a VM's time of operation
	A8.32	Events arising out of the activities of emulating drivers obo VM
	A8.33	Events occurring in a thread's time of operation
	A8.4	Decomposition of system power into container power through CPU utilization
	A10	Identification of a unit of workload and procurement of a load profile
Various attribution techniques	A11.1	Linear
	A11.2	Polynomial
	A11.3	Sub-linear
	A11.4	Gaussian mixture
	A11.5	Support vector regression
Selection of model type	A14	Low-level analysis of physical nature of processor, including implicit use of Dennard's law.
	A15.1	CPU package temperature
Hardware sensors		

Tab. B.2: Approach nodes, enumerated and showing category and sub-category names

Category	Node no.	Sub-category
----------	----------	--------------

## B Graph Nodes Description

Observations on dependency of power consumption on architecture	D32.1	Host energy efficiency (EEy), measured in hash/J, peaks at some frequency for any number of threads >1
	D32.2	VM EEy, measured in hash/J, peaks at some frequency for any number of threads
	D32.3	For the same number of running threads, use of Hyper-Threading less energy efficient than scheduling the threads onto added cores
	D32.4	For the same average transmit data rate, use of Hyper-Threading more energy efficient than using processing on added cores
	D32.5	NUMA has no effect on EEy of CPU-intensive activity unless it causes frequent thread-switching
	D32.6	Processor power varies non-linearly with utilization if DFS active but linearly if DFS inactive
	D32.7	If the number of threads demanding 100% utilization is greater than the number of processor logical cores, the EEy of processor-bound tasks decreases
Observations on dependency of power consumption on specific hardware	D28.1	Ivy Bridge laptop processor shows step increment in power consumption between idle and one active core
	D28.2	AMD Phenom II multi-core processor power varies if the same overall utilization is obtained by different core load distributions
	D28.3	Xeon Core 2 cache coherency architecture activates cores that would otherwise be idle, therefore introducing sub-linear growth of power consumption when previously idle cores are loaded
	D28.4	Nehalem core's deep-sleep (C-) states add step transition in power consumption from purely idle to active
Groups of developments that regard power consumption in IT system implementation	34.1	Power vs network-utilization slope increases sub-linearly with the number of VMs



	D34.2	Power required for packet delivery in virtualized environments using software network switches >>power required in non-virtualized environments
	D34.3	Power increases linearly with Linux vbr's average transmit data rate
	D7.1	For a given VM, power consumption depends on its host
	D7.2	VM power consumption on any given host is positively correlated with VM size
	D33.1	Power vs CPU utilization slope increases sub-linearly with the number of VMs, on VMware ESXi version 5
	D33.2	Power consumption does not depend on virtualization genre or technology for processor- and memory-intensive workloads
	D33.3	Power consumed by containers is less than power consumed by VMs for network-intensive workload
	D33.4	Both hardware-assisted virtualization and paravirtualization are less efficient than non-virtualized operation in use of processor caches.
	D33.5	Power consumed by KVM less than power consumed by Xen on network-intensive workload
Scalability of models	D29.1	Power model scalable to multiple concurrent VM operation
	D29.2	Power model scalable to multiple concurrent container operation
Groups of developments that regard models of power consumption in virtualized environments:  20: Linear 21: Non-linear 31: Beyond linear		

18: Macroscopic	25	Disk i/o energy modelled as function of transfer time and power consumed in an active state
	D20.1	Linear model of host power in terms of architectural instrumentation, obtained using a resource-homogeneous workload
	D20.2	Linear model of VM power in terms of architectural instrumentation, obtained using a specific workload
	D20.3	Linear model of VM power in terms of microarchitectural instrumentation, obtained using a specific workload
	D20.4	Linear model of VM power in terms of architectural instrumentation, obtained using a homogeneous workload
	D20.5	Linear model of host power in terms of microarchitectural instrumentation, obtained using a specific workload
	D20.6	Linear model of host and VM power in terms of microarchitectural instrumentation, obtained using a CPU + memory workload
	D20.7	Linear model of thread power in terms of microarchitectural instrumentation, obtained using a mixed CPU and memory workload
	D20.8	Linear model of container power in terms of microarchitectural instrumentation and processor utilization, obtained using a mixed CPU and memory workload
	D23.1	Non-linear model of host power in terms of architectural instrumentation, obtained using a homogeneous workload
	D23.2	Non-linear model of VM power in terms of architectural instrumentation, obtained using a specific workload
	D23.3	Non-linear model of host power in terms of microarchitectural instrumentation, temperature and frequency, obtained using a mixed CPU and memory workload
	D31.1	Gaussian mixture model, in terms of processor utilization, instructions per cycle, memory accesses per cycle, cache transactions per cycle and obtained using a mixed CPU and memory workload

	D31.2	Support Vector Regression model, in terms of microarchitectural instrumentation and obtained using a mixed CPU, memory and disk workload.
	D18	Service-specific macroscopic power model
Relative accuracy of formal approaches	D30.1	Accuracy of Gaussian Mixture model exceeds that of linear regression model, whether uni- or multivariate
	D30.2	Accuracy of linear, CPU-utilization-only regression is highly dependent on host configuration
	D30.3	Accuracy of Support Vector Regression model exceeds that of linear regression model
	D26	Power analysis of P4 processor in terms of area of, and rate of access to, sub-units of the processor
	D27	Processor power model tested under desktop applications and found to have modest accuracy.
	D9	Highest EEy (J/Web interaction) occurs when the VM hosting the web application is being hit with more traffic than its capacity.
	D24	Predicted host power in terms of total resource utilization is slightly less than the total predicted VM power in terms of individual VMs' resource utilization

Tab. B.3: Development nodes, enumerated and showing category names



# C Task Scheduling

## C.1 Scheduling Algorithms

The scheduling algorithms considered in this thesis are majorly differentiated as heuristic and meta-heuristic methods. Hence, major scheduling algorithms that lie in these categories of scheduling are presented below.

### C.1.1 Heuristic Algorithms

A heuristic technique is an approach to problem-solving that employs a practical method but does not guarantee an optimal solution. Considering the fact that finding an optimal solution in many problems is impractical in a limited amount of time, these heuristic methods help in providing a good quality approximation to the exact solutions. These approaches are problem-dependent and find the solution by effectively applying problem features. One drawback of using these methods is they might sometime get trapped in local optima, and hence be unable to find a globally optimal solution. Discussed below are some widely used heuristic methods, which have been adopted in several domains and are still considered as a good choice for less complex problems.

#### **First Come First Served**

The First Come First Served (FCFS) strategy is the simplest scheduling method where each incoming task is scheduled on the basis of its arrival order. All arriving tasks are collected in the system queue through the tail, and the scheduler selects the task from the head of the queue [177]. This scheme is usually effective in systems where the arriving tasks have no priority and have no objective function to optimize. This is why it is rarely used in scheduling problems that require an optimal solution, but it may be rather effective for systems with task having the same priority and requiring a fair treatment.

#### **Best Fit**

The Best Fit (BF) scheduling algorithm allocates the task to the minimum of available resources. All the available resources of the system which has the capacity to process the given task are first extracted. Required resources by the task are then allocated to the machine which has the least unused resources left. In such a way, the BF algorithm fills out the small chunks of unused resources and improves the overall utilization of the system [177]. This strategy is useful for consolidation and load balancing as well, where a task (or a group of tasks) is used to avoid resource wastage.

## **Linear Programming**

Linear Programming (LP) is an approach to obtain the maximum or minimum of a linear function with some given constraints, over a set of variables [178]. The constraints in the linear programming should be linear, they just cannot follow other curvature or shape. Integer Programming (IP) is a kind of Linear Programming where all the variables are restricted to be integers. It is a very powerful tool to find the solution for problems that contain both continuous and discrete variables, such as scheduling in a heterogeneous environment. Mixed Integer Programming (MIP), however, has the flexibility of having some variable as integers and some as non-integers.

## **Shortest Processing Time / Longest Processing Time**

These two heuristic rules were proposed in the early 1970s considering the objective of the makespan of the system and task. The Shortest Processing Time (SPT) first algorithm is suitable for the system where the completion time of individual tasks matters and needs to be reduced. On the contrary, for the objective of minimizing the overall completion time of the system the Longest Processing Time (LPT) first rule is applied. The scheduling of tasks in these models is initiated by sorting all the given tasks either in ascending (for SPT) or descending (for LPT) order and then scheduling them one after another [174]. These methods are proved to provide a good solution even in a worst-case scenario within a bounded condition. However, studies show that the solution obtained in many cases is much better than the worst-case bound.

## **Min-Min / Max-Min**

Min-min and max-min algorithms work on the similar principle of SPT and LPT heuristic rules. In the min-min algorithm, tasks with minimum completion time are selected to schedule first. Max-min works in the opposite way to min-min and schedules the tasks with maximum processing time first [177].

## **Round Robin**

The Round Robin policy allocates a fixed time slot to each task for processing. The fixed time slot is called Time Quantum and is usually between 10-100 milliseconds [177]. Once the assigned QT for any task is finished, the task is preempted and its current state is stored. The rounds continue until all the tasks in the system finish processing. One drawback of this scheme is that it does not consider the available resources, task priority, required resources, and hence it does not provide an efficient schedule in many cases. Several variants of round-robin have been proposed, which consider system parameters and dynamics to improve the response time of tasks and other parameters. One improved RR model is Weighted Round Robin (WRR) that considers the resource availability of VMs before scheduling and assigns more load to the VM having high resource capability [107].

## **Gradient Descent**

Gradient Descent (GD) is a classical mathematical-based approach to find the local minima of a function. It is an iterative optimization method that takes a step proportional

to the negative gradient of the function at the current point [179]. The GD algorithm is designed on the theory that if a function with multiple variables is distinct and differentiable in a neighborhood of a certain point, then the search can reach the minimum faster if it follows the negative gradient from that point. This algorithm can be caught in local minima sometime, but jumping randomly in a few iterations can reduce this risk. An algorithm that works opposite to GD is the Steepest Ascent Method (SAM), in which the iterative step is taken in proportion to the positive gradient. SAM is used for the problems needs to maximize the objective function.

### Priority Based

With the long delay in execution and processing, some tasks take much longer to finish. So, in critical applications, tasks are associated with a deadline and they are restricted to finish their job within a given deadline, otherwise, they will be considered as failed or lost. Moreover, in the cloud, different customers may have different priorities based on the services they purchased. Such tasks are typically scheduled based on their priority to satisfy the customer requirement. Scheduling in Priority Based (PB) algorithms starts by assigning a fixed priority score to each task and arranging them in order [177]. Tasks with higher priority are then scheduled first to reduce their completion time and meet the client's requirement.

### C.1.2 Meta-Heuristic Algorithms

Meta-heuristics is a higher-level procedure to find an optimal solution, using an iterative approach. These methods are often based on the learning of the concept of natural living and evolutionary science. They are the combination of local search and randomization to reach the global optimal solution. They can provide the optimal solution to the problem but might take much longer time in some cases. Two major characteristics that usually affect the computational time of these models are the complexity of their approach and the number of system variables.

### Simulated Annealing

Simulated Annealing (SA) is a probabilistic search algorithm. The concept of SA is derived from the formation of the crystalline structure into an ordered state. It uses the annealing process that iterates the heating and gradually cools down the structure. A similar physical phenomenon is adopted in SA, where the temperature is set high in the initial stage of the execution due to which the system has a high probability to accept solutions. The jump occurs in the system to avoid local minima when there is a high probability of poor solution in a region. The temperature of the system decreases as the time elapses and a point is reached where iterations can be terminated. At this point, the simulation attains a solution where no further improvement can be made. The number of successive iterations in finding an optimal solution could be very high; therefore, some researchers compromise on the solution and limit the number of iterations. The rate of cooling and terminating cooling temperature can also affect the accuracy and time complexity of the process.

## Genetic Algorithms

A Genetic Algorithm (GA) is a meta-heuristic algorithm, inspired by Darwin's theory of natural evolution, genetics, and population growth. It reflects the process of natural life where the selected genes produce the offspring for the next generation [180]. The GA works with an initial population of variables known as chromosomes. Once the initial generation is created, the algorithm evolves using three operators: selection, crossover, and mutation. In the selection operator, the most fit chromosomes with good fitness scores are selected as parents at the reproduction stage. The selected individuals are now allowed to pass their genes to successive generation. These selected individuals are randomly mated in crossover operations and create a completely new individual (offspring). The concept of mutation is then brought to bring diversity in the next generations. An arbitrary change of the genes is implemented at this stage, but keeping its probability low to avoid potential disruption in a good solution. The process is terminated when the termination criteria are met or the number of desired generations are created.

## Pareto Optimal Front

Pareto Optimal Front (POF) provides a solution to problems with multiple objectives. There is no single solution that can optimize all objectives of the problem. In such a scenario, several solutions are possible with conflicts among objectives and one has to compromise over one or other objectives. To solve these multi-objective problems, several algorithms derived functions to trade-off among different objectives. One such method is the Pareto Optimal Front that analyzes the trade-offs of the stated problem using the given function. It then provides the optimal front which contains the optimal solution to the problem [181], [182]. The solution can not be improved further in any other dimension, although, based on acceptable trade-off, any point on that front can be chosen as an optimal solution.

## Swarm Intelligence

The optimization methods inspired by the social behavior of animal species living in large colonies fall under this category. Two swarm intelligence methods, Particle Swarm Optimization and Ant Colony Optimization are discussed in this chapter.

*Particle Swarm Optimization:* Particle Swarm Optimization (PSO) is a robust evolutionary strategy based on the behavior of a group of birds. The swarm is formed by a bunch of particles, where each particle is allowed to move around and explore the search space [182]. Each particle in the system distributes its information to its immediate neighbors, and they communicate collectively to converge to a single point. Each particle has its own position and velocity. The best-known solution for PSO is the solution where the most number of particles have converged, and best-position of any particle is the position of that particle from the best-known solution. The direction of the particle is determined by three components, own previous velocity of the particle, the distance of best-known solution from the individual particle itself, and the so far best-solution distance from swarms. The particles in swarm avoid the local minima if their own best solution is better than the swarm's solution. The optimal solution is achieved once all the swarm particles converge to a single point.



*Ant Colony Optimization:* Ant Colony Optimization (ACO) is also a population-based meta-heuristic model that can be used to find the optimal solution for complex problems. It is based on the behavior of ants searching for food [182]. Ants while searching for food first wander randomly, on finding food they walk back to the colony while leaving behind markers. These markers, which is actually a chemical known as pheromones are the main contributor in ACO. The other ants come across these pheromones with some probability to reach the food, while populating the space with their own markers and paths. As the number of ants and pheromones increases, the probability of having the shortest path in the search space increases. The ACO exploits a similar mechanism to find an optimal solution, by using agents (as software ants) to search for a good solution.

## C.2 Different Machine Environment in Task Scheduling

A scheduling problem in any domain can be defined using three parameters  $\alpha|\beta|\gamma$ . The first parameter in this definition ' $\alpha$ ' defines the environment of processing machine(s). There could be numerous machine environments depending upon number of processing machines and their characteristics. The second parameter ' $\beta$ ' is the system constraints. It provides the detail of processing different application based on user requirement. This field can be left empty if no particular characteristics or constraints are applied on the system. Some example of processing characteristics are preemption of tasks, tasks deadline, due time of task and precedence. The objective of the system is defined using the term ' $\gamma$ '. The  $\gamma$  field can have single or multiple objectives that are to be optimize in the given system. Some widely used scheduling objectives are average completion time, maximum completion time, lateness, tardiness, resource utilization, and makespan. Described below are some different machine environments for task processing.

- **Single machine (1):** In this case, there is only one processing machine available in the system. Hence, this is the most simplest environment for task scheduling where all incoming task can only be allocated to one single machine. This case could also refer to special case for other complicated multiple machine environments.
- **Identical machines in parallel ( $P_m$ ):** A system with ' $m$ ' number of parallel processing machines where all machines have same processing speed and similar configuration falls under this category. Any task in this case can be assigned to any of the machine available in the system.
- **Machines in parallel with different speed ( $Q_m$ ):** A system of parallel running processing machines where all machines have different processing speed comes under this classification. The search space for optimization (task scheduling) in this case is much broader as processing of tasks can be optimize considering different machines' speeds. But this heterogeneity also increases the complexity of the system, thus the scheduler in these cases will require longer time to find an optimal solution. A special case of this category with all machines having similar speed will be similar to the previous case of  $P_m$ .
- **Unrelated machines in parallel ( $R_m$ ):** This is the special case of parallel machine environment where each machine has some processing constraints. Not all available machines in the system can process any given task, and these machines are limited to process only particular kind of task(s). The optimal scheduling of

## *C Task Scheduling*

tasks in this case is also complex due to more number of variables in machine environment.

# Bibliography

- [1] Stamford. (April 2, 2019). Gartner Forecasts Worldwide Public Cloud Revenue to Grow 17.5 Percent in 2019, [Online]. Available: <https://www.gartner.com/en/newsroom/press-releases/2019-04-02-gartner-forecasts-worldwide-public-cloud-revenue-to-g>.
- [2] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, “A survey of mobile cloud computing: Architecture, applications, and approaches”, *Wireless Communications and Mobile Computing*, vol. 13, no. 18, pp. 1587–1611, 2013, ISSN: 1530-8677. DOI: 10.1002/wcm.1203.
- [3] L. Columbus. (Sep. 23, 2018). Roundup Of Cloud Computing Forecasts And Market Estimates, 2018, [Online]. Available: <https://www.forbes.com/sites/louiscolombus/2018/09/23/roundup-of-cloud-computing-forecasts-and-market-estimates-2018/#17475361507b>.
- [4] R. NV. (February 2019). Fourth Quarter Growth in Cloud Services Tops off a Banner Year for Cloud Providers | Synergy Research Group, [Online]. Available: <https://www.srgresearch.com/articles/fourth-quarter-growth-cloud-services-tops-banner-year-cloud-providers>.
- [5] S. R. Jena, V. Vijayaraja, and A. K. Sahu, “Performance Evaluation of Energy Efficient Power Models for Digital Cloud”, *Indian Journal of Science and Technology*, vol. 9, no. 48, December 29, 2016, ISSN: 0974-5645, 0974-6846. DOI: 10.17485/ijst/2016/v9i48/105639.
- [6] C. Gu, H. Huang, and X. Jia, “Power Metering for Virtual Machine in Cloud Computing-Challenges and Opportunities”, *IEEE Access*, vol. 2, pp. 1106–1116, 2014, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2014.2358992.
- [7] “‘Tsunami of data’ could consume one fifth of global electricity by 2025”, *The GuardianClimate Home News, part of the Guardian Environment Network*, ISSN: 0261-3077.
- [8] J. M. Lima. (December 12, 2017). Data centres of the world will consume 1/5 of Earth’s power by 2025, [Online]. Available: <https://data-economy.com/data-centres-world-will-consume-1-5-earths-power-2025/>.
- [9] R. Danilak. (December 15, 2017). Why Energy Is A Big And Rapidly Growing Problem For Data Centers, [Online]. Available: <https://www.forbes.com/sites/forbestechcouncil/2017/12/15/why-energy-is-a-big-and-rapidly-growing-problem-for-data-centers/>.
- [10] A. Shehabi, S. Smith, S. Sartor, R. Brown, M. Herrlin, J. Koomey, E. Masanet, N. Horner, I. Azevedo, and W. Lintner, “United States Data Center Energy Usage Report”, Ernest Orlando Lawrence Berkeley National Laboratory, 2016.

- [11] S. Mazaheri, Y. Chen, E. Hojati, and A. Sill, “Cloud benchmarking in bare-metal, virtualized, and containerized execution environments”, in *Proc. of 4th International Conference on Cloud Computing and Intelligence Systems (CCIS)*, Beijing, China, August 2016, pp. 371–376. DOI: 10.1109/CCIS.2016.7790286.
- [12] Z. Li, M. Kihl, Q. Lu, and J. A. Andersson, “Performance Overhead Comparison between Hypervisor and Container based Virtualization”, in *Proc. of IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*, Taipei, Taiwan, March 2017, pp. 955–962. DOI: 10.1109/AINA.2017.79.
- [13] M. Pawlish, A. S. Varde, and S. A. Robila, “Analyzing utilization rates in data centers for optimizing energy management”, in *Proc. of International Green Computing Conference (IGCC)*, San Jose, CA, USA: IEEE, Jun. 2012, pp. 1–6, ISBN: 978-1-4673-2154-9 978-1-4673-2155-6 978-1-4673-2153-2. DOI: 10.1109/IGCC.2012.6322248.
- [14] T. Mandal and S. Acharyya, “Optimal task scheduling in cloud computing environment: Meta heuristic approaches”, in *Proc. of 2nd International Conference on Electrical Information and Communication Technologies (EICT)*, Khulna, Bangladesh, December 2015, pp. 24–28. DOI: 10.1109/EICT.2015.7391916.
- [15] D. Gabi, A. S. Ismail, A. Zainal, and Z. Zakaria, “Solving Task Scheduling Problem in Cloud Computing Environment Using Orthogonal Taguchi-Cat Algorithm”, *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 7, no. 3, p. 1489, Jun. 1, 2017, ISSN: 2088-8708, 2088-8708. DOI: 10.11591/ijece.v7i3.pp1489-1497.
- [16] D. Tigano, “An innovative approach to performance metrics calculus in cloud computing environments: A guest-to-host oriented perspective.”, Doctoral thesis, University of Genoa, Italy, March 2018.
- [17] A. J. Younge, “Architectual Principles and Experimentation of Distributed High Performance Virtual Clusters”, Doctor of Philosophy, Indiana University, U.S.A, 2016.
- [18] S. T N and N. Ramasamy, “Literature Survey and Review: Virtualization Technologies”, *International Journal of Printing, Packaging & Allied Sciences*, vol. 4, pp. 1135–1149, December 1, 2016.
- [19] H. Fayyad, LucPerneel, and M. Timmerman, “Benchmarking the Performance of Microsoft Hyper-V server, VMware ESXi and Xen Hypervisors”, *Journal of Emerging Trends in Computing and Information Sciences*, vol. Vol. 4, no. 12, pp. 922–933, December 1, 2013, ISSN: 2079-8407.
- [20] S. Sharma, V. Chang, U. S. Tim, J. L. H. Wong, and S. K. Gadia, “Cloud-based emerging services systems”, *International Journal of Information Management*, pp. 1–12, 2016.
- [21] “Multi-access Edge Computing (MEC); Framework and Reference Architecture”, European Telecommunications Standards Institute, Group Specification ETSI GS MEC 003 V2.1.1, January 2019, Reference: RGS/MEC-0003v211Arch, pp. 1–21.
- [22] M. Kaminska and M. Smihily, “Cloud computing - statistics on the use by enterprises”, December 2018, p. 9.

- [23] I. Odun-Ayo, M. Ananya, F. Agono, and R. Goddy-Worlu, “Cloud Computing Architecture: A Critical Analysis”, in *2018 18th International Conference on Computational Science and Applications (ICCSA)*, Melbourne, Australia, Australia: IEEE, Jul. 2018, pp. 1–7, ISBN: 978-1-5386-7214-3. DOI: 10.1109/ICCSA.2018.8439638.
- [24] S. B. Alam, “Cloud Computing – Architecture, Platform and Security Issues: A Survey”, *World Scientific News*, vol. 86, no. 3, pp. 253–264, 2017.
- [25] M. Chios, D. Clarke, P. Willis, A. Reid, J. Feger, M. Bugenhagen, W. Khan, M. Fargano, D. C. Cui, D. H. Deng, J. Benitez, U. Michel, H. Damker, K. Ogak, T. Matsuzaki, M. Fukui, K. Shimano, D. Delisle, Q. Loudier, C. Koliass, I. Guardini, E. Demaria, R. Minerva, A. Manzalini, D. López, F. J. R. Salguero, F. Ruhl, and P. Sen, “Network Functions Virtualisation. An Introduction, Benefits, Enablers, Challenges & Call for Action”, in *SDN and OpenFlow World Congress*, Darmstadt, Germany, October 22–24, 2012.
- [26] “Mobile Edge Computing (MEC); Technical Requirements”, 2016, ETSI GS MEC 002 V1.1.1.
- [27] A. Uchechukwu, K. Li, and Y. Shen, “Energy consumption in cloud computing data centers”, *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, vol. 3, no. 3, p. 19, 2014. DOI: 10.11591/closer.v3i3.6346.
- [28] N. Khan, “Investigating Energy Efficiency of Physical and Virtual Machines in Cloud Computing”, University of Oslo, Norway, 2017.
- [29] E. Hernandez-Valencia, S. Izzo, and B. Polonsky, “How will NFV/SDN transform service provider opex?”, *IEEE Network*, vol. 29, no. 3, pp. 60–67, May 2015, ISSN: 0890-8044. DOI: 10.1109/MNET.2015.7113227.
- [30] R. Bolla, R. Bruschi, F. Davoli, C. Lombardo, J. F. Pajo, and O. R. Sanchez, “The dark side of network functions virtualization: A perspective on the technological sustainability”, in *Proc. of IEEE International Conference on Communications (ICC)*, Paris, France, May 2017, pp. 1–7. DOI: 10.1109/ICC.2017.7997129.
- [31] K. M. Attia, M. A. El-Hosseini, and H. A. Ali, “Dynamic power management techniques in multi-core architectures: A survey study”, *Ain Shams Engineering Journal*, vol. 8, no. 3, pp. 445–456, Sep. 2017, ISSN: 20904479. DOI: 10.1016/j.asej.2015.08.010.
- [32] V. Braun and V. Clarke, “Using thematic analysis in psychology”, *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, January 2006, ISSN: 1478-0887, 1478-0895. DOI: 10.1191/1478088706qp063oa.
- [33] J. W. Creswell, *Qualitative Inquiry and Research Design: Choosing Among Five Approaches*. SAGE, 2007, 417 pp., ISBN: 978-1-4129-1607-3. Google Books: DetLkgQeTJgC.
- [34] T. Palys, “An Introduction to Codes and Coding”, in *The Coding Manual for Qualitative Researchers*, vol. 3, 2009.
- [35] R. E. Boyatzis, *Transforming Qualitative Information: Thematic Analysis and Code Development*, ser. Transforming Qualitative Information: Thematic Analysis and Code Development. Thousand Oaks, CA, US: Sage Publications, Inc, 1998, ISBN: 978-0-7619-0960-6 978-0-7619-0961-3.

- [36] B. Knowles, L. Blair, M. Hazas, and S. Walker, “Exploring Sustainability Research in Computing: Where We Are and Where We Go Next”, in *Proc. of ACM International Joint Conference on Pervasive and Ubiquitous Computing*, (Zurich, Switzerland), ser. UbiComp '13, New York, NY, USA: ACM, 2013, pp. 305–314, ISBN: 978-1-4503-1770-2. DOI: 10.1145/2493432.2493474.
- [37] B. Krishnan, H. Amur, A. Gavrilovska, and K. Schwan, “VM power metering: Feasibility and challenges”, *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 3, p. 56, January 3, 2011, ISSN: 01635999. DOI: 10.1145/1925019.1925031.
- [38] I. Waßmann, D. Versick, and D. Tavangarian, “Energy consumption estimation of virtual machines”, in *Proc. of the 28th Annual ACM Symposium on Applied Computing - SAC '13*, Coimbra, Portugal: ACM Press, 2013, p. 1151, ISBN: 978-1-4503-1656-9. DOI: 10.1145/2480362.2480579.
- [39] T. Enokido and M. Takizawa, “Power Consumption and Computation Models of Virtual Machines to Perform Computation Type Application Processes”, in *Proc. of Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, Santa Catarina, Brazil: IEEE, Jul. 2015, pp. 126–133, ISBN: 978-1-4799-8870-9. DOI: 10.1109/CISIS.2015.18.
- [40] —, “Power Consumption Model of a Server to Perform Communication Type Application Processes on Virtual Machines”, in *Proc. of 10th International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA)*, Krakow, Poland: IEEE, November 2015, pp. 275–282, ISBN: 978-1-4673-8315-8. DOI: 10.1109/BWCCA.2015.67.
- [41] T. Inoue, A. Aikebaier, T. Enokido, and M. Takizawa, “Algorithms for Selecting Energy-Efficient Storage Servers in Storage and Computation Oriented Applications”, in *Proc. of IEEE 26th International Conference on Advanced Information Networking and Applications*, Fukuoka, Japan, March 2012, pp. 920–927. DOI: 10.1109/AINA.2012.136.
- [42] W. Al-Zubaedi and H. S. Al-Raweshidy, “A parameterized and optimized BBU pool virtualization power model for C-RAN architecture”, in *Proc. of IEEE EUROCON 2017 -17th International Conference on Smart Technologies*, Ohrid, Macedonia, Jul. 2017, pp. 38–43. DOI: 10.1109/EUROCON.2017.8011074.
- [43] C. Mobius, W. Dargie, and A. Schill, “Power Consumption Estimation Models for Processors, Virtual Machines, and Servers”, *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 6, pp. 1600–1614, Jun. 2014, ISSN: 1045-9219. DOI: 10.1109/TPDS.2013.183.
- [44] A. Kansal, F. Zhao, J. Liu, N. Kothari, and A. A. Bhattacharya, “Virtual machine power metering and provisioning”, in *Proc. of the 1st ACM Symposium on Cloud Computing - SoCC '10*, Indianapolis, Indiana, USA: ACM Press, 2010, p. 39, ISBN: 978-1-4503-0036-0. DOI: 10.1145/1807128.1807136.
- [45] (February 23, 2010). Joulemeter: Computational Energy Measurement and Optimization, [Online]. Available: <https://www.microsoft.com/en-us/research/project/joulemeter-computational-energy-measurement-and-optimization/>.

- [46] S. Chinprasertsuk and S. Gertphol, “Power model for virtual machine in cloud computing”, in *Proc. of 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, Chon Buri, Thailand, May 2014, pp. 140–145. DOI: 10.1109/JCSSE.2014.6841857.
- [47] W. Wu, W. Lin, and Z. Peng, “An intelligent power consumption model for virtual machines under CPU-intensive workload in cloud environment”, *Soft Computing*, vol. 21, no. 19, pp. 5755–5764, October 2017, ISSN: 1432-7643, 1433-7479. DOI: 10.1007/s00500-016-2154-6.
- [48] J. Stoess, C. Lang, and F. Bellosa, “Energy Management for Hypervisor-Based Virtual Machines”, in *Proc. of USENIX Annual Technical Conference*, Santa Clara, CA, Jun. 2007, pp. 1–14.
- [49] C. Isci and M. Martonosi, “Runtime power monitoring in high-end processors: Methodology and empirical data”, in *Proc. of 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, San Diego, CA, U.S.A, December 2003, pp. 93–104. DOI: 10.1109/MICRO.2003.1253186.
- [50] G. Dhiman, K. Mihic, and T. Rosing, “A system for online power prediction in virtualized environments using Gaussian mixture models”, in *Proc. of the 47th Design Automation Conference*, Anaheim, CA: ACM Press, Jun. 2010, p. 807, ISBN: 978-1-4503-0002-5. DOI: 10.1145/1837274.1837478.
- [51] T. Veni and S. M. S. Bhanu, “Prediction Model for Virtual Machine Power Consumption in Cloud Environments”, *Procedia Computer Science*, vol. 87, pp. 122–127, 2016, ISSN: 18770509. DOI: 10.1016/j.procs.2016.05.137.
- [52] M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, “A Comparative Study of Methods for Measurement of Energy of Computing”, *Energies*, vol. 12, no. 11, p. 2204, Jun. 10, 2019, ISSN: 1996-1073. DOI: 10.3390/en12112204.
- [53] K. N. Khan, M. Hirki, T. Niemi, J. K. Nurminen, and Z. Ou, “RAPL in Action: Experiences in Using RAPL for Power Measurements”, *ACM Trans. Model. Perform. Eval. Comput. Syst.*, vol. 3, no. 2, 9:1–9:26, March 2018, ISSN: 2376-3639. DOI: 10.1145/3177754.
- [54] M. Hähnel, B. Döbel, M. Völp, and H. Härtig, “Measuring energy consumption for short code paths using RAPL”, *ACM SIGMETRICS Performance Evaluation Review*, vol. 40, no. 3, p. 13, January 4, 2012, ISSN: 01635999. DOI: 10.1145/2425248.2425252.
- [55] K. Li, “Optimal configuration of a multicore server processor for managing the power and performance tradeoff”, *The Journal of Supercomputing*, vol. 61, no. 1, pp. 189–214, Jul. 1, 2012, ISSN: 0920-8542, 1573-0484. DOI: 10.1007/s11227-011-0686-1.
- [56] C. K. Keong, K. T. Wei, A. A. A. Ghani, and K. Y. Sharif, “Toward using software metrics as indicator to measure power consumption of mobile application: A case study”, in *Proc. of 9th Malaysian Software Engineering Conference (MySEC)*, Kuala Lumpur, Malaysia, December 2015, pp. 172–177. DOI: 10.1109/MySEC.2015.7475216.

- [57] R. Morabito, “Power Consumption of Virtualization Technologies: An Empirical Investigation”, in *Proc. of 8th IEEE ACM International Conference on Utility and Cloud Computing*, Limassol, Cyprus, November 4, 2015. DOI: 10.1109/UCC.2015.93.
- [58] R. Shea, H. Wang, and J. Liu, “Power consumption of virtual machines with network transactions: Measurement and improvements”, in *Proc. of IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, Toronto, ON, Canada: IEEE, April 2014, pp. 1051–1059. DOI: 10.1109/INFOCOM.2014.6848035.
- [59] C. Jiang, Y. Wang, D. Ou, Y. Li, J. Zhang, J. Wan, B. Luo, and W. Shi, “Energy efficiency comparison of hypervisors”, *Sustainable Computing: Informatics and Systems*, vol. 22, pp. 311–321, Jun. 1, 2019, ISSN: 2210-5379. DOI: 10.1016/j.suscom.2017.09.005.
- [60] F. Rossi, M. Xavier, C. De Rose, R. Calheiros, and R. Buyya, “E-eco: Performance-Aware Energy-Efficient Cloud Data Center Orchestration”, *Journal of Network and Computer Applications*, vol. 78, pp. 83–96, November 16, 2016. DOI: 10.1016/j.jnca.2016.10.024.
- [61] S. Yu, H. Yang, R. Wang, Z. Luan, and D. Qian, “Evaluating architecture impact on system energy efficiency”, *PLOS ONE*, vol. 12, no. 11, e0188428, November 21, 2017, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0188428.
- [62] “Environmental management — Life cycle assessment — Principles and framework”, ISO Standards ISO 14040:2006, 2006–2007.
- [63] (2011). L.1310 : Energy efficiency metrics and measurement methods for telecommunication equipment, [Online]. Available: <https://www.itu.int/rec/T-REC-L.1310/en>.
- [64] M. Callau-Zori, L. Samoila, A.-C. Orgerie, and G. Pierre, “An experiment-driven energy consumption model for virtual machine management systems”, *Sustainable Computing: Informatics and Systems*, vol. 18, pp. 163–174, Jun. 2018, ISSN: 22105379. DOI: 10.1016/j.suscom.2017.11.001.
- [65] D. Schien, P. Shabajee, M. Yearworth, and C. Preist, “Modeling and Assessing Variability in Energy Consumption During the Use Stage of Online Multimedia Services”, *Journal of Industrial Ecology*, vol. 17, December 1, 2013. DOI: 10.1111/jiec.12065.
- [66] Z. Zhang and S. Fu, “Macropower: A coarse-grain power profiling framework for energy-efficient cloud computing”, in *Proc. of 30th IEEE International Performance Computing and Communications Conference*, Orlando, FL, USA, November 2011, pp. 1–8. DOI: 10.1109/PCCC.2011.6108061.
- [67] A. Vasani, A. Sivasubramaniam, V. Shimpi, T. Sivabalan, and R. Subbiah, “Worth their watts? - an empirical study of datacenter servers”, in *HPCA - 16 2010 The Sixteenth International Symposium on High-Performance Computer Architecture*, Bangalore, India, January 2010, pp. 1–10. DOI: 10.1109/HPCA.2010.5463056.
- [68] G. You and Y. Zhao, “A Dynamic Weight-Based Dynamic Requests Scheduling Model in Multi-core Web Server”, in *International Conference on Service Sciences*, Wuxi, China, May 2014, pp. 112–117. DOI: 10.1109/ICSS.2014.20.



- [69] A. Pathania, V. Venkataramani, M. Shafique, T. Mitra, and J. Henkel, “Distributed fair scheduling for many-cores”, in *Proc. of Design, Automation Test in Europe Conference Exhibition*, Dresden, Germany, March 2016, pp. 379–384.
- [70] C. Kim and J. Huh, “Exploring the Design Space of Fair Scheduling Supports for Asymmetric Multicore Systems”, *IEEE Transactions on Computers*, vol. 67, no. 8, pp. 1136–1152, August 2018. DOI: 10.1109/TC.2018.2796077.
- [71] W. L. Bircher, M. Valluri, J. Law, and L. K. John, “Runtime identification of microprocessor energy saving opportunities”, in *Proc. of the 2005 International Symposium on Low Power Electronics and Design.*, San Diego, CA, USA, August 2005, pp. 275–280. DOI: 10.1145/1077603.1077668.
- [72] M. Kim, Y. Ju, J. Chae, and M. Park, “A Simple Model for Estimating Power Consumption of a Multicore Server System”, *International Journal of Multimedia and Ubiquitous Engineering*, vol. 9, no. 2, pp. 153–160, February 28, 2014, ISSN: 19750080, 19750080. DOI: 10.14257/ijmue.2014.9.2.15.
- [73] A. S. Cassidy and A. G. Andreou, “Beyond Amdahl’s Law: An Objective Function That Links Multiprocessor Performance Gains to Delay and Energy”, *IEEE Transactions on Computers*, vol. 61, no. 8, pp. 1110–1126, August 2012, ISSN: 0018-9340. DOI: 10.1109/TC.2011.169.
- [74] R. Bruschi, A. Carrega, and F. Davoli, “A Game for Energy-Aware Allocation of Virtualized Network Functions”, *Journal of Electrical and Computer Engineering*, vol. 2016, pp. 1–10, 2016, ISSN: 2090-0147, 2090-0155. DOI: 10.1155/2016/4067186.
- [75] M. Aicardi, R. Bruschi, F. Davoli, P. Lago, and J. F. Pajo, “Decentralized Scalable Dynamic Load Balancing Among Virtual Network Slice Instantiations”, in *Proc. of IEEE Globecom Workshops (GC Wkshps)*, Abu Dhabi, United Arab Emirates: IEEE, December 2018, pp. 1–7, ISBN: 978-1-5386-4920-6. DOI: 10.1109/GLOCOMW.2018.8644472.
- [76] T. Zhao, J. Wu, S. Zhou, and Z. Niu, “Energy-delay tradeoffs of virtual base stations with a computational-resource-aware energy consumption model”, in *Proc. of IEEE International Conference on Communication Systems*, Macau, China, November 2014, pp. 26–30. DOI: 10.1109/ICCS.2014.7024759.
- [77] R. Bolla, R. Bruschi, F. Davoli, L. Di Gregorio, P. Donadio, L. Fialho, M. Collier, A. Lombardo, D. Reforgiato Recupero, and T. Szemethy, “The Green Abstraction Layer: A Standard Power-Management Interface for Next-Generation Network Devices”, *IEEE Internet Computing*, vol. 17, no. 2, pp. 82–86, March 2013, ISSN: 1089-7801. DOI: 10.1109/MIC.2013.39.
- [78] “Environmental Engineering (EE); Green Abstraction Layer (GAL); Power management capabilities of the future energy telecommunication fixed network nodes”, ETSI Standard ETSI ES 203 237 V1.1.1, March 2014, Reference: DES/EE-0030, pp. 1–34.
- [79] M. Aldossary, K. Djemame, I. Alzamil, A. Kostopoulos, A. Dimakis, and E. Agiatzidou, “Energy-Aware Cost Prediction and Pricing of Virtual Machines in Cloud Computing Environments”, *Future Generation Computer Systems*, vol. 93, pp. 442–459, November 9, 2018, ISSN: 0167-739X. DOI: 10.1016/j.future.2018.10.027.

- [80] N. K. Sharma, P. Sharma, and R. M. R. Guddeti, “Energy efficient quality of service aware virtual machine migration in cloud computing”, in *Proc. of 4th International Conference on Recent Advances in Information Technology (RAIT)*, Dhanbad, India, March 2018, pp. 1–6. DOI: 10.1109/RAIT.2018.8389047.
- [81] M. Zakarya and L. Gillam, “An Energy Aware Cost Recovery Approach for Virtual Machine Migration”, in *Economics of Grids, Clouds, Systems, and Services*, J. Á. Bañares, K. Tserpes, and J. Altmann, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2017, pp. 175–190, ISBN: 978-3-319-61920-0. DOI: 10.1007/978-3-319-61920-0\_13.
- [82] (May 23, 2012). CPU Usage Limit for Linux, [Online]. Available: <http://cpulimit.sourceforge.net/>.
- [83] W. Dargie, “A Stochastic Model for Estimating the Power Consumption of a Processor”, *IEEE Transactions on Computers*, vol. 64, no. 5, pp. 1311–1322, May 2015, ISSN: 0018-9340, 1557-9956, 2326-3814. DOI: 10.1109/TC.2014.2315629.
- [84] P. Garraghan, I. S. Moreno, P. Townend, and J. Xu, “An Analysis of Failure-Related Energy Waste in a Large-Scale Cloud Environment”, *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 2, pp. 166–180, Jun. 2014, ISSN: 2168-6750, 2376-4562. DOI: 10.1109/TETC.2014.2304500.
- [85] A. F. Monteiro and O. Loques, “Quantum Virtual Machine: A Scalable Model to Optimize Energy Savings and Resource Management”, in *2015 27th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, Florianopolis, Brazil: IEEE, October 2015, pp. 194–201, ISBN: 978-1-4673-8011-9. DOI: 10.1109/SBAC-PAD.2015.24.
- [86] B. Catlin, J. E. Hanrahan, M. E. Russinovich, D. A. Solomon, and A. Ionescu, *System Architecture, Processes, Threads, Memory Management, and More*, Seventh edition, ser. Windows Internals ; Part 1. Redmond: Microsoft, 2017, 784 pp., OCLC: on1012946187, ISBN: 978-0-7356-8418-8.
- [87] (May 5, 2017). CPU Analysis, [Online]. Available: <https://docs.microsoft.com/en-us/windows-hardware/test/wpt/cpu-analysis>.
- [88] G. J. Popek and R. P. Goldberg, “Formal Requirements for Virtualizable Third Generation Architectures”, *Communication ACM*, vol. 17, pp. 412–421, 1974.
- [89] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 6th ed. Morgan Kaufmann, November 23, 2017, 939 pp., ISBN: 978-0-12-811906-8. Google Books: cM8mDwAAQBAJ.
- [90] D. Molka, D. Hackenberg, R. Schone, and M. S. Muller, “Characterizing the energy consumption of data transfers and arithmetic operations on x86-64 processors”, in *Proc. of International Conference on Green Computing*, Chicago, IL, USA: IEEE, August 2010, pp. 123–133, ISBN: 978-1-4244-7612-1. DOI: 10.1109/GREENCOMP.2010.5598316.
- [91] P. Kutch and B. Johnson, “SR-IOV for NFV Solutions Practical Considerations and Thoughts”, intel, Intel Technical Brief Rev1.0, February 2017.

- [92] (2013). Intel® Core™ i5-3230M Processor (3M Cache, up to 3.20 GHz) rPGA Product Specifications, [Online]. Available: <https://ark.intel.com/content/www/us/en/ark/products/72164/intel-core-i5-3230m-processor-3m-cache-up-to-3-20-ghz-rpga.html>.
- [93] *Mobile 3rd Generation Intel® Core™ Processor Family, Mobile Intel® Pentium® Processor Family and Mobile Intel® Celeron® Processor Family*, Datasheet, Volume 1 of 2, Jun. 2013.
- [94] (2018). Power Management Guide Red Hat Enterprise Linux 6, [Online]. Available: [https://access.redhat.com/documentation/en-us/red\\_hat\\_enterprise\\_linux/6/html/power\\_management\\_guide/index](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/6/html/power_management_guide/index).
- [95] H. Liu, H. Jin, C.-Z. Xu, and X. Liao, “Performance and energy modeling for live migration of virtual machines”, *Cluster Computing*, vol. 16, no. 2, pp. 249–264, Jun. 1, 2013, ISSN: 1573-7543. DOI: 10.1007/s10586-011-0194-3.
- [96] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, “Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation”, in *Cloud Computing*, M. G. Jaatun, G. Zhao, and C. Rong, Eds., ser. Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2009, pp. 254–265, ISBN: 978-3-642-10665-1. DOI: 10.1007/978-3-642-10665-1\_23.
- [97] *Network functions virtualisation; management and orchestration, ETSI Group Specification NFV-MAN 001 version 1.1.1*, December 2014.
- [98] M. Aldossary and K. Djemame, “Performance and Energy-based Cost Prediction of Virtual Machines Live Migration in Clouds:” in *Proc. of 8th International Conference on Cloud Computing and Services Science*, Funchal, Madeira, Portugal: Science and Technology Publications, 2018, pp. 384–391, ISBN: 978-989-758-295-0. DOI: 10.5220/0006682803840391.
- [99] Chongya Ma, Zhiying Jiang, Ke Zhang, Guangfei Zhang, Zhixiong Jiang, Chunyang Lu, and Yushan Cai, “Virtual machine power metering and its applications”, in *2013 IEEE Global High Tech Congress on Electronics*, Shenzhen, China: IEEE, November 2013, pp. 153–156, ISBN: 978-1-4799-3209-2. DOI: 10.1109/GHTCE.2013.6767262.
- [100] X. Wu, Y. Zeng, and G. Lin, “An Energy Efficient VM Migration Algorithm in Data Centers”, in *Proc. of 16th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES)*, Anyang, China, October 2017, pp. 27–30. DOI: 10.1109/DCABES.2017.14.
- [101] C. Wen and Y. Mu, “Power and Performance Management in Nonlinear Virtualized Computing Systems via Predictive Control”, *PLOS ONE*, vol. 10, no. 7, e0134017, Jul. 30, 2015, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0134017.
- [102] A. Carrega, G. Portomauro, M. Repetto, and G. Robino, “Boosting Energy Efficiency and Quality of Service through Orchestration Tools”, *IEEE Cloud Computing*, vol. 5, no. 6, pp. 38–47, November 2018, ISSN: 2325-6095, 2372-2568. DOI: 10.1109/MCC.2018.064181119.
- [103] Production-Grade Container Orchestration, [Online]. Available: <https://kubernetes.io/>.
- [104] Apache Mesos, [Online]. Available: <http://mesos.apache.org/>.

- [105] Swarm mode overview, [Online]. Available: <https://docs.docker.com/engine/swarm/>.
- [106] I. M. A. Jawarneh, P. Bellavista, F. Bosi, L. Foschini, G. Martuscelli, R. Montanari, and A. Palopoli, "Container Orchestration Engines: A Thorough Functional and Performance Comparison", in *Proc. of ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, Shanghai, China, May 2019, pp. 1–6. DOI: 10.1109/ICC.2019.8762053.
- [107] D. C. Devi and V. R. Uthariaraj, "Load Balancing in Cloud Computing Environment Using Improved Weighted Round Robin Algorithm for Nonpreemptive Dependent Tasks", *The Scientific World Journal*, vol. 2016, pp. 1–14, 2016, ISSN: 2356-6140, 1537-744X. DOI: 10.1155/2016/3896065.
- [108] R. Bertran, Y. Becerra, D. Carrera, V. Beltran, M. González, X. Martorell, N. Navarro, J. Torres, and E. Ayguadé, "Energy accounting for shared virtualized environments under DVFS using PMC-based power models", *Future Generation Computer Systems*, vol. 28, no. 2, pp. 457–468, February 1, 2012, ISSN: 0167-739X. DOI: 10.1016/j.future.2011.03.007.
- [109] R. Bertran, M. González, X. Martorell, N. Navarro, and E. Ayguadé, "Decomposable and responsive power models for multicore processors using performance counters", in *Proc. of International Conference on Supercomputing*, Tsukuba, Ibaraki, Japan, January 1, 2010, pp. 147–158. DOI: 10.1145/1810085.1810108.
- [110] Perfmon2, [Online]. Available: <http://perfmon2.sourceforge.net/>.
- [111] KVM, [Online]. Available: [https://www.linux-kvm.org/page/Main\\_Page](https://www.linux-kvm.org/page/Main_Page).
- [112] M. Hinz, G. P. Koslovski, C. C. Miers, L. L. Pilla, and M. A. Pillon, "A Cost Model for IaaS Clouds Based on Virtual Machine Energy Consumption", *Journal of Grid Computing*, vol. 16, no. 3, pp. 493–512, Sep. 2018, ISSN: 1570-7873, 1572-9184. DOI: 10.1007/s10723-018-9440-8.
- [113] D. Laganà, C. Mastroianni, M. Meo, and D. Renga, "Reducing the Operational Cost of Cloud Data Centers through Renewable Energy", *Algorithms*, vol. 11, no. 10, p. 145, Sep. 27, 2018, ISSN: 1999-4893. DOI: 10.3390/a11100145.
- [114] A. Forestiero, C. Mastroianni, M. Meo, G. Papuzzo, and M. Sheikhalishahi, "Hierarchical Approach for Efficient Workload Management in Geo-Distributed Data Centers", *IEEE Transactions on Green Communications and Networking*, vol. 1, no. 1, pp. 97–111, March 2017, ISSN: 2473-2400. DOI: 10.1109/TGCN.2016.2603586.
- [115] SPEC Benchmarks, [Online]. Available: <https://www.spec.org/benchmarks.html>.
- [116] G. Chen, X. Bai, X. Huang, M. Li, and L. Zhou, "Evaluating services on the cloud using ontology QoS model", in *Proc. of 2011 IEEE 6th International Symposium on Service Oriented System (SOSE)*, Irvine, CA, USA: IEEE, December 2011, pp. 312–317, ISBN: 978-1-4673-0412-2 978-1-4673-0411-5 978-1-4673-0410-8. DOI: 10.1109/SOSE.2011.6139122.

- [117] K. Hwang, X. Bai, Y. Shi, M. Li, W.-G. Chen, and Y. Wu, “Cloud Performance Modeling with Benchmark Evaluation of Elastic Scaling Strategies”, *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 1, pp. 130–143, January 1, 2016, ISSN: 1045-9219. DOI: 10.1109/TPDS.2015.2398438.
- [118] OProfile - a system profiler for linux (news), Oprofile, [Online]. Available: <https://oprofile.sourceforge.io/news/>.
- [119] G. Sebastien. (April 25, 2013). Sysstat/sysstat, [Online]. Available: <https://github.com/sysstat/sysstat>.
- [120] B. Gregg. (12-Jun-2018). Linux perf Examples, [Online]. Available: <http://www.brendangregg.com/perf.html>.
- [121] A. Z. Netto and R. S. Arnold. (Jun. 12, 2012). Evaluate performance for Linux on POWER, [Online]. Available: <http://www.ibm.com/developerworks/library/l-evaluatelinuxonpower/index.html>.
- [122] R. Bruschi, F. Davoli, P. Lago, and J. F. Pajo, “Joint Power Scaling of Processing Resources and Consolidation of Virtual Network Functions”, in *Proc. of IEEE International Conference on Cloud Networking (Cloudnet)*, Pisa, Italy, October 2016, pp. 70–75. DOI: 10.1109/CloudNet.2016.20.
- [123] A. Nouredine, A. Bourdon, R. Rouvoy, and L. Seinturier, “A preliminary study of the impact of software engineering on GreenIT”, in *Quantum Virtual Machine: A Scalable Model to Optimize Energy Savings and Resource Management*, vol. 3, Zurich, Switzerland: IEEE, Jun. 2012, pp. 21–27, ISBN: 978-1-4673-1832-7 978-1-4673-1833-4. DOI: 10.1109/GREENS.2012.6224251.
- [124] W. Wu, W. Lin, L. He, G. Wu, and C. Hsu, “A Power Consumption Model for Cloud Servers Based on Elman Neural Network”, *IEEE Transactions on Cloud Computing*, pp. 1–1, 2019. DOI: 10.1109/TCC.2019.2922379.
- [125] A. K. Singh, C. Leech, B. K. Reddy, B. M. Al-Hashimi, and G. V. Merrett, “Learning-Based Run-Time Power and Energy Management of Multi/Many-Core Systems: Current and Future Trends”, *Journal of Low Power Electronics*, vol. 13, no. 3, pp. 310–325, Sep. 1, 2017, ISSN: 1546-1998. DOI: 10.1166/jolpe.2017.1492.
- [126] C.-W. Tsai, C.-F. Lai, H.-C. Chao, and A. Vasilakos, “Big data analytics: A survey”, *Journal of Big Data*, vol. 2, December 1, 2015. DOI: 10.1186/s40537-015-0030-3.
- [127] S. S. Mangiafico, “P-values and r-square values for models”, in *Summary and Analysis of Extension Program Evaluation in R*, vol. 1.18.1, New Brunswick, NJ: Rutgers Cooperative Extension, 2016.
- [128] C. O. S. Sorzano, J. Vargas, and A. P. Montano, “A survey of dimensionality reduction techniques”, March 12, 2014. arXiv: 1403.2877 [stat].
- [129] J. Malmodin, D. Lundén, E. Research, and T. C. Ab, “The energy and carbon footprint of the global ICT and E&M sectors 2010-2015”, in *Proc. of International Conference on Information and Communication Technology for Sustainability*, vol. 52, Toronto, Canada, 2018, pp. 187–208.

- [130] R. Zamani and A. Afsahi, “A study of hardware performance monitoring counter selection in power modeling of computing systems”, in *Proc. of International Green Computing Conference (IGCC)*, San Jose, CA, Jun. 2012, pp. 1–10. DOI: 10.1109/IGCC.2012.6322289.
- [131] (August 29, 2018). vPMC – Virtual Performance Monitoring Counters – v(e)Xpertise, [Online]. Available: <http://vxpertise.net/2012/11/vpmc-virtual-performance-monitoring-counters/>.
- [132] (August 29, 2018). Kernel/Reference/stress-ng - Ubuntu Wiki, [Online]. Available: <https://wiki.ubuntu.com/Kernel/Reference/stress-ng>.
- [133] (2016). Intelligent rack power distribution-data sheet - px series, [Online]. Available: [http://www.mca-midwest.com/datasheets/dominion\\_px.pdf](http://www.mca-midwest.com/datasheets/dominion_px.pdf).
- [134] R. Rodrigues, A. Annamalai, I. Koren, and S. Kundu, “A Study on the Use of Performance Counters to Estimate Power in Microprocessors”, *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 12, pp. 882–886, December 2013. DOI: 10.1109/TCSII.2013.2285966.
- [135] L. F. Cupertino, G. Da Costa, and J.-M. Pierson, “Towards a generic power estimator”, *Computer Science - Research and Development*, vol. 30, no. 2, pp. 145–153, May 2015, ISSN: 1865-2034, 1865-2042. DOI: 10.1007/s00450-014-0264-x.
- [136] M. Dayarathna, Y. Wen, and R. Fan, “Data Center Energy Consumption Modeling: A Survey”, *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 732–794, 2016, ISSN: 1553-877X. DOI: 10.1109/COMST.2015.2481183.
- [137] T. Thein, M. M. Myo, S. Parvin, and A. Gawanmeh, “Reinforcement learning based methodology for energy-efficient resource allocation in cloud data centers”, *Journal of King Saud University - Computer and Information Sciences*, November 20, 2018, ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2018.11.005.
- [138] F. Bellosa, “The benefits of event: Driven energy accounting in power-sensitive systems”, in *Proc. of the 9th Workshop on ACM SIGOPS European Workshop beyond the PC: New Challenges for the Operating System - EW 9*, Kolding, Denmark: ACM Press, 2000, p. 37. DOI: 10.1145/566726.566736.
- [139] P. Xiao and D. Liu, “Improving Accuracy of Virtual Machine Power Model by Relative-PMC Based Heuristic Scheduling”, *Computing and Informatics*, vol. 35, pp. 241–258, 2016.
- [140] J. A. Baglivo, “Chapter 14 Linear Least Squares Analysis”, in *Mathematica Laboratories for Mathematical Statistics: Emphasizing Simulation and Computer Intensive Methods*, SIAM, January 1, 2005, ISBN: 978-0-89871-566-8.
- [141] M. Awad and R. Khanna, “Support Vector Regression”, in *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, M. Awad and R. Khanna, Eds., Berkeley, CA: Apress, 2015, pp. 67–80, ISBN: 978-1-4302-5990-9. DOI: 10.1007/978-1-4302-5990-9\_4.
- [142] S. E. N. Fernandes, A. L. Pilastrri, L. A. M. Pereira, R. G. Pires, and J. P. Papa, “Learning kernels for support vector machines with polynomial powers of sigmoid”, in *Proc. of 27th SIBGRAPI Conference on Graphics, Patterns and Images*, Brazil: IEEE, August 2014, pp. 259–265, ISBN: 978-1-4799-4258-9. DOI: 10.1109/SIBGRAPI.2014.36.

- [143] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, ser. Adaptive Computation and Machine Learning. Cambridge, Mass: MIT Press, 2006, 248 pp., ISBN: 978-0-262-18253-9.
- [144] T. Gunasegaran and Y.-N. Cheah, “Evolutionary cross validation”, in *Proc. of 8th International Conference on Information Technology (ICIT)*, Amman, Jordan: IEEE, May 2017, pp. 89–95, ISBN: 978-1-5090-6332-1. DOI: 10.1109/ICITECH.2017.8079960.
- [145] (2019). OpenStack docs: OpenStack compute (nova), Openstack, [Online]. Available: <https://docs.openstack.org/nova/latest/>.
- [146] Y. Li, Y. Wang, B. Yin, and L. Guan, “An Online Power Metering Model for Cloud Environment”, in *2012 IEEE 11th International Symposium on Network Computing and Applications*, Cambridge, MA, August 2012, pp. 175–180. DOI: 10.1109/NCA.2012.10.
- [147] S. Singh and I. Chana, “A Survey on Resource Scheduling in Cloud Computing: Issues and Challenges”, *Journal of Grid Computing*, vol. 14, no. 2, pp. 217–264, Jun. 2016, ISSN: 1570-7873, 1572-9184. DOI: 10.1007/s10723-015-9359-2.
- [148] S. Bansal and C. Hota, “Design and Development of Scheduling Algorithms for Grid Computing Systems”, Birla Institute of Technology & Science, Rajasthan, India, 2015.
- [149] (2010). Server virtualization: Decrease IT cost and data center space, Biztechmagazine. in collab. with, [Online]. Available: [https://biztechmagazine.com/sites/default/files/2\\_12\\_10%20Server%20Virt.pdf](https://biztechmagazine.com/sites/default/files/2_12_10%20Server%20Virt.pdf).
- [150] S. Varshney and S. Singh, “A Survey on Resource Scheduling Algorithms in Cloud Computing”, *International Journal of Applied Engineering Research.*, vol. 13, no. 9, pp. 6839–6845, 2018, ISSN: 0973-4562.
- [151] A. Ealiyas and S. P. J. Lovesum, “Resource Allocation and Scheduling Methods in Cloud- A Survey”, in *Proc. of Second International Conference on Computing Methodologies and Communication (ICCMC)*, Erode, India, February 2018, pp. 601–604. DOI: 10.1109/ICCMC.2018.8487967.
- [152] S. C. Nayak, S. Parida, C. Tripathy, and P. K. Pattnaik, “An enhanced deadline constraint based task scheduling mechanism for cloud environment”, *Journal of King Saud University - Computer and Information Sciences*, October 17, 2018, ISSN: 1319-1578. DOI: 10.1016/j.jksuci.2018.10.009.
- [153] A. Razaque, N. R. Vennapusa, N. Soni, G. S. Janapati, and k. R. Vangala, “Task scheduling in Cloud computing”, in *Proc. of IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, Farmingdale, NY, USA, April 2016, pp. 1–5. DOI: 10.1109/LISAT.2016.7494149.
- [154] L.-T. Lee, K.-Y. Liu, H.-Y. Huang, and C.-Y. Tseng, “A Dynamic Resource Management with Energy Saving Mechanism for Supporting Cloud Computing”, *International Journal of Grid and Distributed Computing*, vol. 6, no. 1, pp. 67–76, 2013.
- [155] M. B. Gawali and S. K. Shinde, “Task scheduling and resource allocation in cloud computing using a heuristic approach”, *Journal of Cloud Computing*, vol. 7, no. 1, p. 4, February 8, 2018, ISSN: 2192-113X. DOI: 10.1186/s13677-018-0105-8.

- [156] I. S. Moreno, R. Yang, J. Xu, and T. Wo, “Improved energy-efficiency in cloud datacenters with interference-aware virtual machine placement”, in *Proc. of IEEE Eleventh International Symposium on Autonomous Decentralized Systems (ISADS)*, Mexico City, Mexico, March 2013, pp. 1–8. DOI: 10.1109/ISADS.2013.6513411.
- [157] A.-p. Xiong and C.-x. Xu, “Energy Efficient Multiresource Allocation of Virtual Machine Based on PSO in Cloud Data Center”, *Mathematical Problems in Engineering*, vol. 2014, pp. 1–8, Jun. 12, 2014. DOI: 10.1155/2014/816518.
- [158] X. Zheng and L. Wang, “A Pareto based fruit fly optimization algorithm for task scheduling and resource allocation in cloud computing environment”, in *2016 IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, BC, Canada, Jul. 2016, pp. 3393–3400. DOI: 10.1109/CEC.2016.7744219.
- [159] N. Kumar, S. Agarwal, and B. Bhimrao, “An Analytical Model for Dynamic Resource Allocation Framework in Cloud Environment”, *Research Journal of Recent Sciences*, vol. 3, pp. 1–6, 2014.
- [160] P. Garraghan, Y. Al-Anii, J. Summers, H. Thompson, N. Kapur, and K. Djemame, “A Unified Model for Holistic Power Usage in Cloud Datacenter Servers”, in *Proc. of IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, Shanghai, China, December 2016, pp. 11–19.
- [161] B. Yemini. (Jul. 30, 2014). Is there an optimal CPU utilization?, [Online]. Available: <https://blog.turbonomic.com/blog/on-turbonomic/optimal-cpu-utilization-depends>.
- [162] H. Liu, “A Measurement Study of Server Utilization in Public Clouds”, in *Proc. of IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, Sydney, Australia: IEEE, December 2011, pp. 435–442, ISBN: 978-1-4673-0006-3 978-0-7695-4612-4. DOI: 10.1109/DASC.2011.87.
- [163] F. Al-Haidari, M. Sqalli, and K. Salah, “Impact of CPU Utilization Thresholds and Scaling Size on Autoscaling Cloud Resources”, in *Proc. of IEEE 5th International Conference on Cloud Computing Technology and Science*, Bristol, UK, 2013, pp. 256–261. DOI: 10.1109/CloudCom.2013.142.
- [164] N. Liu, Z. Dong, and R. Rojas-Cessa, “Task Scheduling and Server Provisioning for Energy-Efficient Cloud-Computing Data Centers”, in *Proc. of IEEE 33rd International Conference on Distributed Computing Systems Workshops*, Philadelphia, PA, USA, Jul. 2013, pp. 226–231. DOI: 10.1109/ICDCSW.2013.68.
- [165] V. S. Rathor, R. K. Pateriya, and R. K. Gupta, “An Efficient Virtual Machine Scheduling Technique in Cloud Computing Environment”, *International Journal of Modern Education and Computer Science*, vol. 7, no. 3, pp. 39–46, March 8, 2015, ISSN: 20750161, 2075017X. DOI: 10.5815/ijmecs.2015.03.06.
- [166] A. Kaur, V. Singh, and S. S. Gill, “The Future of Cloud Computing: Opportunities, Challenges and Research Trends”, in *Proc. of the Second International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC 2018)*, Palladam, India, India, February 28, 2019, pp. 213–219. DOI: 10.1109/I-SMAC.2018.8653731.



- [167] Lipika. (April 9, 2018). Why immersion cooling is imminent to power the next generation datacenters?, Web Hosting | Cloud Computing | Datacenter | Domain News, [Online]. Available: <https://www.dailyhostnews.com/why-immersion-cooling-is-imminent-to-power-the-next-generation-datacenters>.
- [168] A. Borah, D. Mucharary, S. Singh, and J. Borah, “Power Saving Strategies in Green Cloud Computing Systems”, *International Journal of Grid Distribution Computing*, vol. 8, pp. 299–306, February 28, 2015. DOI: 10.14257/ijgdc.2015.8.1.28.
- [169] S. Ismail, H. R. Hassen, and H. Zantout, “Open Challenges in Security of Cloud Computing”, in *Proc. of the International Conference on Big Data and Advanced Wireless Technologies - BDAW '16*, Blagoevgrad, Bulgaria: ACM Press, 2016, pp. 1–4, ISBN: 978-1-4503-4779-2. DOI: 10.1145/3010089.3016025.
- [170] N. Gonzalez, C. Miers, F. Redígolo, M. Simplício, T. Carvalho, M. Näslund, and M. Pourzandi, “A quantitative analysis of current security concerns and solutions for cloud computing”, *Journal of Cloud Computing: Advances, Systems and Applications*, vol. 1, no. 1, p. 11, Jul. 12, 2012, ISSN: 2192-113X. DOI: 10.1186/2192-113X-1-11.
- [171] M. R. Mesbahi, A. M. Rahmani, and M. Hosseinzadeh, “Reliability and high availability in cloud computing environments: A reference roadmap”, *Human-centric Computing and Information Sciences*, vol. 8, no. 1, p. 20, Jul. 6, 2018, ISSN: 2192-1962. DOI: 10.1186/s13673-018-0143-8.
- [172] L. Osembe, “A study of cloud computing technology adoption by small and medium enterprises (SMEs) in Gauteng Province.”, University of KwaZulu-Natal, Durban, 2015.
- [173] Hansen and P. Brinch, “Simulated Annealing”, Syracuse University, School of Computer and Information Science, New York, 1992, p. 170.
- [174] P. Brucker, *Scheduling Algorithms*. Springer Science & Business Media, March 20, 2013, 374 pp., ISBN: 978-3-540-24804-0. Google Books: 2\_v5BwAAQBAJ.
- [175] D. Meisner, B. T. Gold, and T. F. Wenisch, “PowerNap: Eliminating Server Idle Power”, in *Proc. of the 14th International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS XIV, Washington, DC, USA: ACM, 2009, pp. 205–216, ISBN: 978-1-60558-406-5. DOI: 10.1145/1508244.1508269.
- [176] R. L. Graham, “Bounds on Multiprocessing Timing Anomalies”, *SIAM Journal on Applied Mathematics*, vol. 17, no. 2, pp. 416–429, March 1969, ISSN: 0036-1399, 1095-712X. DOI: 10.1137/0117039.
- [177] H. Gibet Tani and C. EL Amrani, “Smarter Round Robin Scheduling Algorithm for Cloud Computing and Big Data”, *Journal of Data Mining and Digital Humanities*, vol. Special Issue on Scientific and Technological Strategic Intelligence (2016), January 2018.
- [178] L. Liu and Q. Fan, “Resource Allocation Optimization Based on Mixed Integer Linear Programming in the Multi-Cloudlet Environment”, *IEEE Access*, vol. 6, pp. 24 533–24 542, 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2830639.

## Bibliography

- [179] H. Madni, M. Latiff, S. Abdulhamid, and J. Ali, “Hybrid gradient descent cuckoo search (HGDCS) algorithm for resource scheduling in IaaS cloud computing environment”, *Cluster Computing*, vol. 22, March 1, 2019. DOI: 10.1007/s10586-018-2856-x.
- [180] M. Srinivas and L. Patnaik, “Genetic algorithms: A survey”, *Computer*, vol. 27, no. 6, pp. 17–26, Jun. 1994, ISSN: 0018-9162, 1558-0814. DOI: 10.1109/2.294849.
- [181] P. Ngatchou, A. Zarei, and A. El-Sharkawi, “Pareto Multi Objective Optimization”, in *Proc. of the 13th International Conference on, Intelligent Systems Application to Power Systems*, Arlington, VA, USA, November 2005, pp. 84–91. DOI: 10.1109/ISAP.2005.1599245.
- [182] J. Christensen and C. Bastien, “Chapter | seven - Heuristic and Meta-Heuristic Optimization Algorithms”, in *Nonlinear Optimization of Vehicle Safety Structures*, J. Christensen and C. Bastien, Eds., Oxford: Butterworth-Heinemann, January 1, 2016, pp. 277–314, ISBN: 978-0-12-804424-7. DOI: 10.1016/B978-0-12-417297-5.00007-9.