7-7-2016 12:00 AM

Climbing Up Cloud Nine: Performance Enhancement Techniques for Cloud Computing Environments

Mohamed Abusharkh
The University of Western Ontario

Supervisor
Abdallah Shami
The University of Western Ontario Joint Supervisor
Abdelkader Ouda
The University of Western Ontario

Graduate Program in Electrical and Computer Engineering
A thesis submitted in partial fulfillment of the requirements for the degree in Doctor of Philosophy

Abstract

With the transformation of cloud computing technologies from an attractive trend to a business reality, the need is more pressing than ever for efficient cloud service management tools and techniques. As cloud technologies continue to mature, the service model, resource allocation methodologies, energy efficiency models and general service management schemes are not yet saturated. The burden of making this all tick perfectly falls on cloud providers. Surely, economy of scale revenues and leveraging existing infrastructure and giant workforce are there as positives, but it is far from straightforward operation from that point. Performance and service delivery will still depend on the providers' algorithms and policies which affect all operational areas.

With that in mind, this thesis tackles a set of the more critical challenges faced by cloud providers with the purpose of enhancing cloud service performance and saving on providers' cost. This is done by exploring innovative resource allocation techniques and developing novel tools and methodologies in the context of cloud resource management, power efficiency, high availability and solution evaluation.

Optimal and suboptimal solutions to the resource allocation problem in cloud data centers from both the computational and the network sides are proposed. Next, a deep dive into the energy efficiency challenge in cloud data centers is presented. Consolidation-based and non-consolidation-based solutions containing a novel dynamic virtual machine idleness prediction technique are proposed and evaluated. An investigation of the problem of simulating cloud environments follows. Available simulation solutions are comprehensively evaluated and a novel design framework for cloud simulators covering multiple

variations of the problem is presented. Moreover, the challenge of evaluating cloud resource management solutions performance in terms of high availability is addressed. An extensive framework is introduced to design high availability-aware cloud simulators and a prominent cloud simulator (GreenCloud) is extended to implement it. Finally, real cloud application scenarios evaluation is demonstrated using the new tool.

The primary argument made in this thesis is that the proposed resource allocation and simulation techniques can serve as basis for effective solutions that mitigate performance and cost challenges faced by cloud providers pertaining to resource utilization, energy efficiency, and client satisfaction.

Keywords: Cloud Computing, Resource allocation, Analytical models, Systems simulation, Virtualization, Network and systems monitoring and measurements,Cloud simulators, Scalability, Data Centers, Clouds,Web and Internet services, Energy efficiency, High Availability.

To my father Fathi Ahmed Abu Sharkh and to my mother Zainab Ibrahim Al-Halabi: My intelligent teachers, my humble role models and my patient guides. May Allah unite us with our loved ones in Heaven accompanying prophet Mohamed peace be upon him.

Acknowledgements

As modest as this work is, it gives me a chance to thank Allah -glorified and exalted is He- for His countless blessings. I pray that I can serve Him, as He guides me to, and be a true follower of Prophet Mohammed Who taught us to strive for knowledge, peace and helping others.

I would like to express my unlimited gratitude to my supervisor Prof. Abdallah Shami. His continuing concern with the students' futures and his amazing efforts to secure the best possible working environment are really commendable. In my case, his continuous guidance and accommodating patience throughout this journey made it a great experience and made this thesis possible. I am also very grateful to my co-supervisor Dr. Abdelkader Ouda for always granting me his confidence and continuous support in any way he could.

Many thanks go to Western and everybody at Western engineering. You are the unsung heroes of this educational process. Special thanks to Chris Mariott, Stephanie Tigert, and Georgea Giakoumatos for their efforts and smiles. I am deeply blessed with and grateful to my beloved family. My brothers and sisters have been the best ever and have always been there for me. I would like to thank Ahmed, Ala'a, Wafa'a, Faten, Basma and my nephews and nieces. And a big thank you to Maha and Mona for their contribution. You guys are always in my heart. Special thanks to my father and mother in law, uncle Afdal and Aunt Hala for their continuous support and kindness and for always treating me like a son. My deep gratitude goes to my dear uncle Taysir and his family for making us feel at home in London . Special thanks to my cousin, Ayman Abu Sharkh. Ayman's leadership talent is only matched by his kindness and true friendship.

Special thanks go to my friends and colleagues at Western. Your friendship and unwavering support turned the nice moments into glorious memories and turned each tough moment into a push forward. Special thanks to this colorful road's companion Mohamed Kalil for being a true friend and a relentless engineer's engineer. Hassan Hawilo & Manar Jammal, Karim Hammad, Peng Hao, Khaled Al Hazmi, Mohamed Hussein, Emad Aqili, Abdallah Mobyed, Maysam Mirahmdi, Dan Wallace, Oscar Fillio, Ahmed AlShaer, Ahmed Hamade, Nojan Madineh, Rejaul Choudry, Aidin Reyhani-Masouleh, Omar Sallam, Jay Nadeu, Ajmal Khan, Eric Southern, Siamack Gademi, Mohamed Noor, Khaled Abu Sharkh, Ahmed Abu sharkh, Fuad Shamieh, Wilson Higashino, Abdulfattah Noorwali, Anas Saci and all my friends and colleagues at Western; You have all contributed greatly to my life and to this work and I will always remember you fondly.

I would like to thank Ericsson research and Dr. Ali Kanso for partly supporting my research. Dr Ali's knowledge and guidance helped to push this work to the standard it is at now. I will always be indebted to my old friends who have been there for me through

the years and were always there for advice, support and companionship. Many thanks go to Ahmad Haidar, Sharif Oteafy, Omar Alnaggar, Zaid alshaweesh, Mohamed Maslouh, Mohamed AlSayyed, Omar AlTerkawi, Ali Krboj, Ali Alnaggar, Ammar Alkhateeb, Mohamed AlMbayid, and Anas Nayfeh.

Many thanks to all the great teachers who taught me through the years; special thanks to Mr. Mohammed Ghoneim, Dr. Bader Al Bdaiwi, and Dr. Ziad Najem. I would like to specially thank my M.Sc. supervisor, Dr. Hazem Raafat, for his support. I would like to thank anyone who gave away the least bit of their time to contribute to this project.

Last but far from least, this work would not have been possible without the love and support of my amazing wife Doaa and lovely daughter Hana. I remember reading my first paper when Hana was just 2 weeks old when I started this program and now she is a little girl holding this thesis in her hands. You guys are a true blessing. May Allah grant us happiness in this life and the other and make our dreams true. Amen.

Table of Contents

List of Tables

List of Figures

Acronyms

BW	*Bandwidth*
CC	*Cloud Computing*
CPU	*Central Processing Unit*
DC	*Data center*
DCN	*Data center networks*
DIP	*Dynamic VM idleness prediction technique*
DP	*Duration Priority Technique*
DVFS	*Dynamic Voltage and Frequency Scaling*
EC2	*Amazon Elastic Compute Cloud*
ED	*Equal Time Distribution Technique*
FF	*First Fit*
GA	*Greedy Algorithm*
GUI	*Graphical User Interface*
HA	*High availability , Highly available*
IaaS	*Infrastructure As a Service*
MBFD	*Modified Best First Search*
MIPs	*Million Instructions Per Second*
NaaS	*Network As a Service*
ND	*Node Distance Distribution Technique*
OPL	*Optimization Programming Language- used in CPLEX*
PaaS	*Platform As a Service*
PCA	*Phased Communication Application*
QoS	*Quality of Service*
RA	*Resource Allocation*
RB	*Resource Based Distribution Technique*
RST	*Requested start Time*

SaaS	*Software As a Service*
SDN	*Software Defined Networking*
SMB	*Small and Medium Businesses*
SLA	*Service-Level Agreement*
SVOP	*Smart Overprovision VM placement Technique*
VM	*Virtual Machines*
VPN	*Virtual Private Networks*

Chapter 1

Introduction

The substantial weight cloud computing holds in the current information technology marketplace nowadays makes it hard to remember that cloud computing started gaining this momentum in the not-so-distant past. In 2008, a figure as big as Oracle's CEO Larry Ellison stated that: "The interesting thing about Cloud Computing is that we've redefined Cloud Computing to include everything that we already do...I don't understand what we would do differently in the light of Cloud Computing other than change the wording of some of our ads". This was "the" Larry Ellison, quoted in the Wall Street Journal, on September 26, 2008.[4].

Fast forward to today, and cloud computing is not the future but the present of computing. Cloud technology adoption rates show that clearly. 78% of U.S. small businesses will have fully adopted cloud computing by 20020, more than doubling the current 37% [5]. The percentage grows to 90% when looking at large businesses (larger than 1000 employees) [6]. The U.S. Small and Medium Business (SMB) cloud computing and services market will grow from 43 billion dollars in 2015 to 55 billion dollars in 2016 [5]. This trend is consistent in Europe as well. The percentages of small, medium and large businesses adopting cloud technologies in UK are 46, 63 and 82% respectively. In Germany, the percentages are 50, 65 and 86%.

There is proven potential for client demand growth as well. In a survey conducted by the rightscale.com team, 68% of enterprises indicated they run less than a fifth of their application portfolios in the cloud. 55% of enterprises report that a significant portion of their existing application portfolios are not in cloud, but are built with cloud-friendly architectures [7]. This is used to serve major functions like data protection (backup), business continuity (replication, disaster recovery), archiving and file services and office enablement (sharing, synchronization, collaboration).

The appeal of robust cloud computing services for clients is understandable. Short term capital savings are a major attraction point. The flexibility offered by the pay-as-you-go conditions relieves the clients from the burden of extensive infrastructure planning in advance. Virtualization comes as a layer of protection for the clients against implementation complexities. Merging into the cloud is becoming a business reality for clients and for a good reason.

However, the burden of making this all tick perfectly falls on cloud providers. Surely, economy of scale revenues and leveraging existing infrastructure, giant workforces and expertise are there as positives, but it is far from straightforward operation from that point. As much as cloud computing as a paradigm redefines the service model and client priorities, it still requires a major shake up on the cloud provider side as well. Performance and service delivery will still depend on the providers' policies and algorithms that affect all operational areas. This covers a wide spectrum of aspects ranging from security, application portability to communication efficiency and reaching energy efficiency and resource allocation in a data center.

Cloud data center resource allocation and management represents a critical area in need of continuous attention. Resource allocation has a direct impact on two sides that are at the core of why the cloud is financially effective: resource utilization and energy efficiency. Simply put, the better resource utilization is, the more client requests the data center (cloud) can serve and the better the performance can be. As for energy efficiency, it is a pressing issue for cloud providers. Power costs represent between 25% and 40% of the operational expenses of a data center [8]. The carbon footprint of a data center is a political and environmental pain point for cloud providers as well.

With that in mind, this thesis tackles a set of the more critical challenges faced by cloud providers with the purpose of enhancing cloud service performance and saving on providers' cost. This is done by exploring innovative resource allocation techniques and developing novel tools and methodologies in the context of cloud resource management, power efficiency, high availability and solution evaluation.

A detailed discussion of some of the external and internal challenges faced by an network-aware resource allocation methodology in a cloud data center is introduced next.

This discussion aims at providing the reader with an elaborate context for this lively field. After that, the contributions of this thesis are stated chapter by chapter.

1.1 Resource Allocation in a Network Based Cloud Computing Environment: Design Challenges

Several providers have cloud computing (CC) solutions available, where a pool of virtualized and dynamically scalable computing power, storage, platforms, and services are delivered on demand to clients over the Internet in a pay as you go manner. This is implemented using large data centers (DCs) where thousands of servers reside. Clients have the choice between using private clouds which are DCs specialized for the internal needs of a certain business organization and public clouds which are open over the Internet to the public for use. Services are offered under several deployment models including Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) and Network as a Service (NaaS). Each provider offers a unique service portfolio with a range of options that include Virtual Machines (VMs) instance configuration, nature of network services, degree of control over the rented machine, supporting software/hardware security services, additional storage, etc. To move to the cloud, clients demand guarantees with regards to achieving the required improvements in scale, cost control and reliability of operations. Despite its importance, providing computation power alone is not sufficient as a competitive advantage. Other factors have gained more weight recently such as the networking solution offerings. The network performance and resource availability could be the tightest bottleneck for any cloud. This is seen as an opportunity for network service providers who are building their own clouds using distributed cloud architecture.

Here, there is a need for a comprehensive resource allocation (RA) and scheduling system for CC data center networks (DCNs). This system would handle all the resources in the cloud providers' DCNs and would manage client requests, dictate RA, ensure satisfaction of network QoS conditions, and eliminate performance hiccups while minimizing the service provider cost and controlling the level of consumed energy. The resource

---

The contents of this section (1.1) have been published in [9]

management of the DCs' servers and the network resources while scheduling and serving thousands of client requests on virtual machines (VMs) residing on DC servers, is a critical success factor. First, it is a main revenue source to the service provider as excess resources translate directly to revenue. Second, it is a key point that will make or break potential clients' decision to move fully to a particular cloud.

1.1.1 A Comprehensive Solution for Network Processing RA

Provisioning for cloud services in a comprehensive way is crucial to any RA model. Any model should consider both computational resources and network resources to accurately represent practical needs. First, excluding the computational resources during the design of the RA model deprives the model of the main cloud service. Cloud DCs are built first and foremost as ways to outsource computational tasks. Any model that optimizes DC resources should include answers to questions like: How are VMS allocated? How are processing resources modeled? What is the resource portfolio that is being promoted to clients? How the DC resources are distributed physically? The other side of the coin is networking services. As clients ask for tasks to be processed in the DC, they need networking service with adequate QoS standards to send and receive their application data. As reported in [10], only 54% of the IT professionals surveyed about their use of cloud services indicated that they involve network operations' personnel, down from 62% recorded in a similar poll in 2009. This directly affects the use of network best practices and the attention to the health of overall traffic delivery. Also in [10], 28% of survey respondents believed that monitoring and troubleshooting packet traces between VMs is required. In addition, 32% believed that monitoring and troubleshooting traffic data from virtual switches is required. Bandwidth costs deeply affect the cloud clients' financial structure. The trend shows that network resources require more attention. A study performed by the authors of [11], shows that a client who downloads a relatively small amount of 10 GB per day would be charged 30$ /month when using MS Azure. This is equal to 16$/Mb while the market price per Mb is around 8$ [11]. Therefore, optimizing the bandwidth cost represents an opportunity of profit for providers and an opportunity of saving for clients. The network resources weight in the cloud market has alerted network service providers to build their

own distributed DCs with a vision to enter the CC market. They envision replacing a large DC with multiple smaller DCs to be closer to the clients. This setup turns the network infrastructure into a distributed cloud. That in turn helps in controlling costs and increasing service differentiation. A cloud service provider caters network services to clients to support one of three functions:

1. Connecting the clients' private cloud (or headquarters) to VMs the client reserved in the DCs; using Internet or VPNs as shown in Fig. 1.3.
2. Connecting the VMs on different public clouds to facilitate data exchange between two VMs reserved by the same client.
3. Connecting VMs on the same public cloud together

It is no use to the clients if their application is producing the results needed in the required time if these results cannot be delivered to them through a stable network connection. In [4], data transfer bottlenecks are stated as one of the main obstacles cloud client growth is facing. The authors show that when moving large amounts of data in a distributed DC environment, the network service performance will be a critical point for the whole process. In the example mentioned, the authors reached the conclusion that the data transmission tardiness can cause the client to prefer sending data disks with a courier (FedEx, for example).

Targeting a network-aware RA system brings to the front multiple challenges that face the CC community. Addressing those issues would be of utmost importance to form a complete solution. These design challenges can be classified into external challenges which are enforced by factors outside the RA process and internal challenges that are related to the way the RA is done.

1.1.2 External Challenges

1.1.2.1 Regulative and Geographical Challenges:

In the virtualization model used in cloud offerings, the client does not manage the physical location of data. Also, there is no guarantee given by the provider as for the data physical location in a certain moment [4]. In fact, it is a common practice to distribute client data over multiple geographically distant DCs. Splitting the data will enhance fault tolerance,

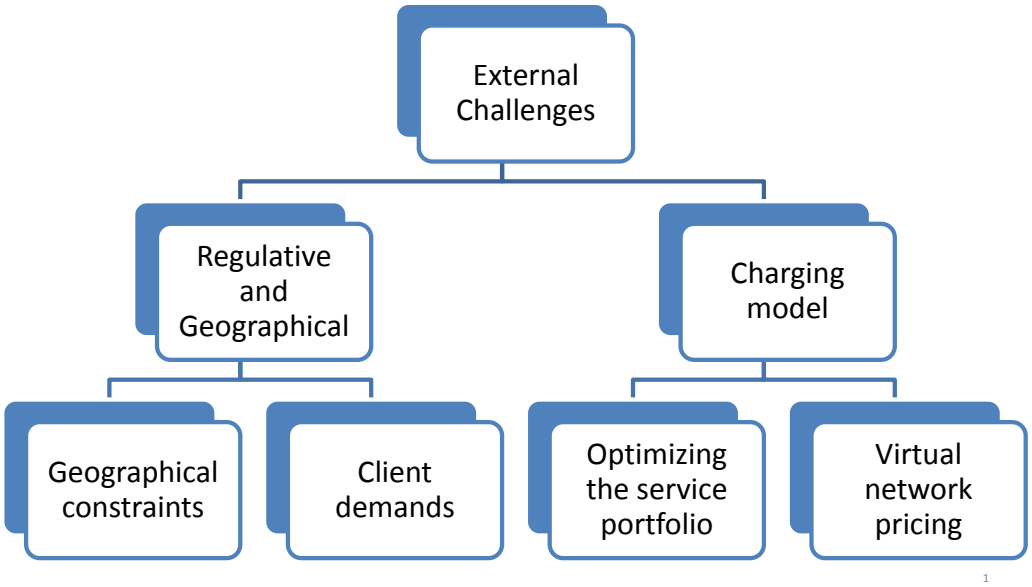

Figure 1.1: Network-aware Resource Allocation: Design Challenges-External Challenges

but it presents regulative and security challenges. An example would be the regulative obligation of complying with the U.S. Health Information Portability and Accountability Act (HIPAA) (the Health Information Protection Act (HIPA) in Canada). HIPAA does not apply directly to third party service providers, it is imperative that health care organizations require the third-party providers to sign contracts which require them to handle all patient data in adherence with HIPAA standards. This raises some constraints to handling and storing data:

1. Geographical constraints: HIPAA requires that patient data does not leave US soil. This constraint limits the choice of DCs to allocate a VM to and limits data movement maneuvers while trying to optimize performance. Additionally, when data is stored in the cloud, it is necessary to know the physical location of the data, the number of data copies, data modification details, or data deletion details.
2. Client actions: To get more assurance about data security, clients may require guarantees like instant data wiping (writing over byte by byte) instead of deletion. They might also require storing encrypted data on the cloud. This would pose extra pressure on the performance and will make it harder to comply with QoS requirements.

3. Under HIPPA, patients have a right to access any information stored about them. A careful study of the locations of the patients and the usage distribution of these patients is crucial for the RA system. Considering this factor when placing the data would minimize the distance patient data will travel in the network. Making a decision where the data is located has a direct effect on minimizing the cost.

1.1.2.2 Charging Model Issues:

The resources management system should incorporate the clients charging model. For example, when using Amazon EC2, a client can pay for the instances completely on demand, reserve an instance for term contract or choose spot Instances that enable him to bid for unused Amazon EC2 capacity. Issues to be considered here include:

1. Finding the service portfolio offering that maximizes the revenue weight of excess resources in the DC. Examining the options available in the market, it is clear that cost is not calculated based on static consumptions.
2. Finding the best way to integrate the virtual network usage into the cost analysis. Challenges would arise because a virtual link length/distance (and in turn cost) varies from link to link. A virtual link could even change to use another physical path on the substrate network based on the methodology used.

1.1.3 Internal challenges

1.1.3.1 Data Locality

There is a need for systems to implement data locality features "the right way". This means how to combine the management of compute (processing) and data (network) resources using data locality features to minimize the amount of data movement and in turn improve application performance/scalability while meeting end users security concerns. It is important to schedule computational tasks close to the data, and to understand the cost of moving the work as opposed to moving the data.

To have a full view of how to use data locality these issues need to be considered:

1. A data aware-scheduler is critical in achieving good scalability and performance. A more specific perspective needs to be reached. This includes answering questions

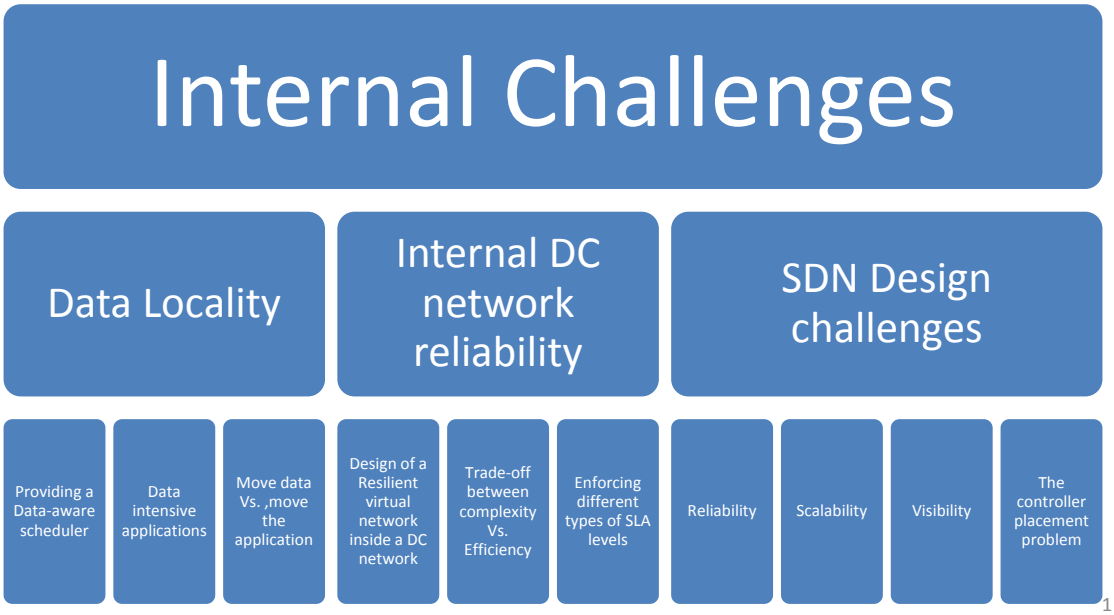

Figure 1.2: Network-aware Resource Allocation: Design Challenges-Internal Challenges

like: What information is needed by the scheduler when making a decision? What are the policies and decision criteria for moving data? What data integration policies should be enforced?

2. Analyzing the behaviour of data intensive applications is a very good starting point on the road to understanding data locality and data movement patterns. This is needed to determine data integration policies for specific data locality and movement patterns. This allows the study of meta-policies that switch data integration policies as patterns change.
3. Moving the application itself to servers in the data center where data is needed. Many factors have to be considered, such as:
    - The availability of servers in the data center
    - Policy/ algorithm specifications on when to move considering future demand might need data stored in the original location
    - iii) Decision criteria as to migrate the whole VM or just move the concerned application

1.1.3.2 Reliability of Network Resources Inside a DC

The DC internal network affects the performance deeply. The DC internal network design decisions affect performance and reliability of the DC resources. These decisions relate to factors like network topology, traffic routing, flow optimization, bandwidth allocation policies and network virtualization options.

1.1.3.3 SDN Design Challenges inside the DCs

SDN is a networking paradigm in which the forwarding behavior of a network element is determined by a software control plane decoupled from the data plane. This paradigm can enable many advantages if it is coupled with an efficient RA model. SDN leads to many benefits such as increasing network and service customizability, supporting improved operations and increased performance. The software control plane can be implemented using a central network controller which can handle the task of RA in the DCN by directing all the client requests to it. This controller will execute the RA algorithms then send the allocation commands across the network. Since it is a relatively new paradigm, the community has to tackle the following SDN challenges:

1. Reliability - Using centralized SDN controller affects reliability. Although solutions like stand by controllers or using multiple controllers for the network are suggested, practical investigation is needed to reveal the problems and analyze the trade-offs of using such solutions.
2. Scalability - When the network scales up in the number of switches and the number of end hosts, the SDN controller becomes a key bottleneck. For example, [12] estimates that a large DC consisting of 2 million virtual machines may generate 20 million flows per second. The current controllers can support about 10^5 flows per second in the optimal case [13]. Extensive scalability results in losing visibility of the network traffic, making troubleshooting nearly impossible.
3. Visibility - Prior to SDN, a network team could quickly spot, for example, that a backup was slowing the network. The solution would then be to simply reschedule it

The contents of 1.1.3.3 are coauthored by Manar Jamal

to after hours. Unfortunately with SDN, only a tunnel source and a tunnel endpoint with UDP traffic are visible. One cannot see who is using the tunnel. There is no way of determining the origin of the problem. The true top talker is shielded from view by the UDP tunnels, which means that when traffic slows and users complain, pinpointing the problem area in the network is not possible. With the loss of visibility, troubleshooting is hindered, scalability is decreased and a delay in resolution could be quite detrimental to the business.

4. The controller's placement problem influences every aspect of a decoupled control plane, from state distribution options to fault tolerance to performance metrics. This problem includes placement of controllers with respect to the available topology in the network and the number of needed controllers. The placement is related to certain metrics defined by clients like latency, increasing number of nodes, etc. According to [14], random placement for a small value of k medians will result in an average latency between 1.4x and 1.7x larger than that of the optimal placement. As a result, cloud clients will see network service specification as decisive factor in their choice to move to the cloud or to choose their cloud provider. Factors like bandwidth options, port speed, number of IP addresses, load balancing options and availability of VPN access should be considered by any comprehensive model.

1.1.3.4 Fault Tolerance and High availability

Fault tolerance requirements are essential to clients with large sets of data. A fault tolerance strategy affects how clients VMs are distributed across the fault domains. This distribution often contradicts performance. The challenge here is to find the fault domain definitions and VM distribution that complies with fault tolerance constraints without compromising the performance.

1.1.3.5 Portability and vendor lock in

This issue is a concern for cloud clients. Clients require guarantees of the applications being portable and easily movable to other cloud providers. This affects VM deployment design and raises a concern for cloud providers regarding the optimal procedure when a

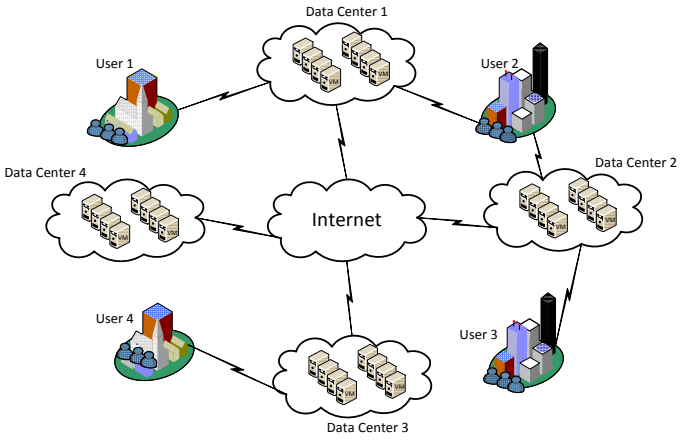

Figure 1.3: A sample network of private and public clouds connected through Internet or VPNs

certain client leaves. Which RA adjustments are made and how? Here, designing efficient procedure is a big performance booster.

1.2 Thesis Outline and Contributions

After laying the base for this work by discussing this new field's background, the focus here is defining which specific challenges within the cloud computing ecosystem are to be tackled in this thesis. The remaining chapters of this thesis discuss the following. Chapter 2 Offers optimal and suboptimal solutions to the resource allocation problem in cloud data centers from both the computational and the network sides. Chapter 3 dives deep into the energy efficiency challenge in cloud data centers. Consolidation-based and non-consolidation-based solutions constructed upon a novel dynamic virtual machine idleness prediction technique are proposed and evaluated. Next, an investigation of the problem of simulating cloud environments follows in Chapter 4. Available simulation solutions are comprehensively evaluated and a novel design framework for cloud simulators that covers multiple variations of the problem is presented. In Chapter 5, the challenge of evaluating cloud resource management solutions performance in terms of high availability is addressed. An extensive framework is introduced to design high availability-aware cloud simulators and a prominent cloud simulator (GreenCloud) is extended to implement and

evaluate this framework. Moreover, real application scenarios evaluation is demonstrated using the new tool. Finally, Chapter 6 concludes this thesis by envisioning the future steps for the aforementioned research projects.

1.3 Contributions of the Thesis

The major contributions of the thesis are summarized as follows.

1.3.1 Contributions of Chapter 2

This chapter aims to tackle the problem of allocating client VM reservation and connection scheduling requests to corresponding data center resources while achieving the cloud provider's objectives. This chapter's main contributions include the following:

.

1. The resource allocation problem for cloud data centers is formulated in order to obtain the optimal solution. This formulation takes into consideration the computational resource requirements at a practical granularity while considering the virtualization scenario common in the cloud. It also considers conditions posed by the connection requests (request lifetime/deadline, bandwidth requirements and routing) at the same time. An important advantage of this approach over approaches used in previous efforts is considering both sets of resource requirements simultaneously before making the scheduling decision. This formulation is looked at from the providers' perspective and aims at maximizing performance.
2. The formulation is constructed in a generic way that it does not restrict itself to the limited environment of one data center's internal network. The connection requests received can come from one of many geographically distributed private or public clouds. Moreover, the scheduler is given the flexibility to place the VMs in any of the cloud provider's data centers that are located in multiple cities. These data centers (clouds) represent the network communicating nodes. The complete problem is solved using IBM ILOG CPLEX optimization library[15].

3. Multiple heuristic methods are introduced to preform the two phases of the scheduling process. Three methods are tested for the VM reservation step. Two methods are tested for scheduling connections. The performance of these methods is investigated and then compared to some of the currently available methods.
4. A suboptimal method based on relaxing the optimal solution is introduced to solve the same problem for large scale cases. This method exploits the technique of decomposing the original problem into two separate sub-problems. The first one is referred to as the master problem which performs the assignment of VMs to data center servers based on a novel VM-node relation function. The second one, termed as the subproblem, performs the scheduling of connection requests assigned by master problem. This suboptimal method achieves better results than the heuristic methods while getting these results in more feasible time periods in contrast to the optimal formulation.

1.3.2 Contributions of Chapter 3

This chapter aims at offering a more complete solution for energy efficiency in cloud data centers. As seen in TABLE 3.1, each one of the solutions surveyed, despite covering a significant flavor of the problem, does not offer a comprehensive vision. To achieve an outstanding solution for such a layered problem, the following contributions are offered:

1. The problem is formulated as a mathematical optimization problem with the objective of minimizing the energy consumption.
2. A new technique called Dynamic Idleness Prediction (DIP) is introduced where the future demands for Virtual machines (VMs) are considered when placing/scheduling the VM on a host. This technique is based on using an artificial intelligence classifier (in our case REPtree classifier) to predict the nature of the load every VM will receive in a pre-specified future period. A novel scheduling/placement technique was constructed by combining DIP and the resource based scheduling technique we introduced in Chapter 2. In this technique, DIP dictates VM initial placement and VM migration in order to reach more efficient consolidation-based energy efficiency as opposed to traditional methods like first fit, round robin or greedy methods.

3. A novel technique for energy efficient VM management that does not depend on consolidation (VM Migration) is introduced. The technique is called Smart VM Overprovision (SVOP). This technique exploits the DIP technique described earlier to dictate the overprovision of VMs by choosing the mostly idle VMS to be switched off and in turn minimize lost requests.
4. A data set published by of Google (data center traces [16]) is used to evaluate the two proposed methods. The performance of these methods is compared to common scheduling algorithms in terms of critical energy efficiency metrics including: energy used per server, energy used per served request, service rate, and number of migrations performed.

1.3.3 Contributions of Chapter 4

This chapter strives to offer a timely and comprehensive view of the cloud simulators as of today. The contributions of this chapter can be summarized as follows.

1. This chapter examines the major cloud simulators available to researchers and industry engineers and compares them in terms of the main simulated components, application model, network model, and architecture. This includes systematically covering general purpose simulators as well specialized simulators or limited scale simulators. A discussion of the merits of building a new simulator as opposed to extending an already existing one is presented.
2. The limitations found in each simulator are presented using an approach that depicts what it is and what it is not. The ground on which the current simulators stand is illustrated in terms of environment assumptions they were based on. This helps in the following step which is constructing a framework for the cloud simulator design process which would ideally cover the industry and research community needs. This cloud simulator design framework can serve as an elaborate design checklist for newly initiated projects.
3. A deep analysis of the open research challenges related to this topic is offered. Challenges covered include realistic user application patterns, cloud deployment

and pricing, reliability and high availability, challenges originated outside the data center and Big data considerations among others.

1.3.4 Contributions of Chapter 5

Building on the discussion from Chapter 4, this chapter addresses the need for a cloud simulator that enables high availability(HA) algorithm testing in order to reach a HA scheduling technique that does not sacrifice other performance metrics. To the best of our knowledge, there is not a simulator that provides the detailed functionality that enables measuring HA metrics, testing HA algorithms and producing the results in a way that can serve academia and the industry. The aim of this chapter is to try and do just that.

1. A framework to amend cloud simulators with HA features is introduced. This framework would enhance the cloud simulator of choice with required features in order to turn it into an HA-aware simulator. The framework includes features related to component failure and recovery, synchronization and functional dependency as well as work flow features like dynamic scheduling of multi-phased tasks in addition to a higher granularity modeling of applications.
2. GreenCloud is taken as an example of a major simulator with a direct focus on green computing and these features are implemented as an additional measurement layer. This is illustrated using the specifications of a phased communication application (abbreviated henceforth as PCA).

Chapter 2
Wind Driven Clouds: Optimal and Suboptimal Resource Allocation Techniques in Cloud Computing Data Centers

2.1 Introduction

The appeal of cloud computing for clients comes from the promise of transforming computing infrastructure into a commodity or a service that organizations can pay for exactly as much as they use. This idea is an IT corporation executive's dream. As Gartner analyst Daryl Plummer puts it *"Line-of-business leaders everywhere are bypassing IT departments to get applications from the cloud .. and paying for them like they would a magazine subscription. And when the service is no longer required, they can cancel that subscription with no equipment left unused in the corner"* [18]. The idea that centralized computing over the network is the future, was clear to industry leaders as early as 1997. None other than Steve Jobs said:*"I don't need a hard disk in my computer if I can get to the server faster .. carrying around these non-connected computers is byzantine by comparison"* [18]. This applies to organizations purchasing and planning large data centers as well.

However, performance remains the critical factor. If at any point doubts are cast over a provider's ability to deliver the service according to the Service Level Agreements (SLAs) signed, clients will consider moving to other providers. They might even consider going back to the buy-and-maintain model. Providers are under constant pressure to improve

---

A version of this chapter has been submitted for publication in [17].

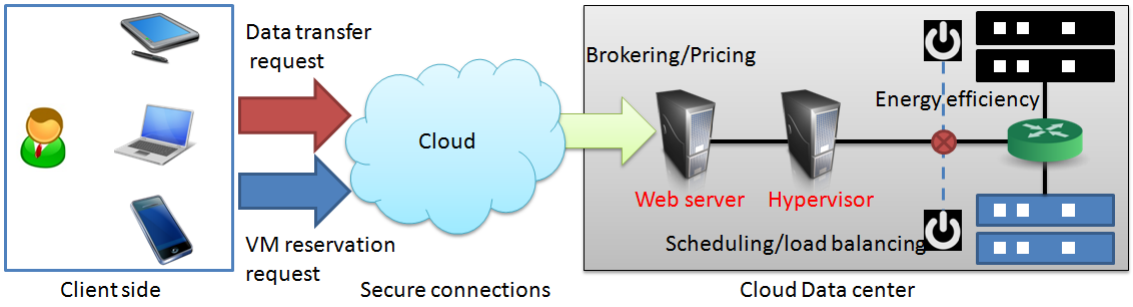

Figure 2.1: Cloud simulated environment and components

performance, offer more diverse resource deployment options, improve service usability, and enhance application portability. A main weapon here is an efficient resource allocation system. In the cloud scenario, clients are able to rent Virtual Machines (VMs) from cloud providers. Providers offer several deployment models where VM configuration differs in computing power, memory, storage capacity and platform to name a few factors. During the rental period, clients require network capabilities. Clients will exchange data between client headquarters (or private clouds) and VMs or between two client VMs. The task here is to schedule VM reservation requests and connection requests in the fastest possible way while using the data center resources optimally.

This task is getting even harder with the emergence of the big data concepts. IBM summarized big data challenges into 4 different dimensions referred to as the 4 Vs: Volume, Velocity, Variety, and Veracity [19]. With most companies owning at least 100 TB of data stored and with 18.6 billion network connections expected to exist in 2016 [19], resource allocation efficiency has never been so important.

When faced by the task of designing a resource allocation methodology, many external and internal challenges should be considered. An attempt to summarize these challenges can be found in [9]. External challenges include regulative and geographical challenges as well as client demands related to data warehousing and handling. These limitations result in constraints on the location of the reserved VMs and restrictions to the data location and movements. External challenges also include optimizing the charging model in such a way that generates maximum revenue. Internal challenges discussed in [9] also include data locality issues. The nature of the application in terms of being data intensive should be considered while placing the VMs and scheduling connections related to this

application.

To achieve these performance and cost objectives, cloud computing providers need a comprehensive resource allocation system that manages both computational and network resources. Such an efficient system would have a major financial impact as excess resources translate directly into revenues.

The following sections are organized as follows: a discussion of the related research efforts is introduced in Section 2.2 leading to this chapter's contribution. Detailed model description is given in Section 2.3. Section 2.4 presents the mathematical formulation of the problem. The heuristic methods are presented in Section 2.5. The suboptimal solution is presented in Section 2.6. Results are shown and analyzed in 2.7. Finally, Section 2.8 concludes the chapter.

2.2 Related Work

Previous attempts were made to optimize a diverse set of cloud resources. In [20], Dastjerdi and Buyya propose a framework to simplify cloud service composition. Their proposed technique optimizes the service composition on the basis of deployment time, cost and reliability preferred by users. The authors exploit a combination of evolutionary algorithms and fuzzy logic composition optimization with the objective of minimizing the effort of users while expressing their preferences. Despite including a wide range of user requirements in the problem modeling and providing an optimization formulation along with a fuzzy logic heuristic, [4] tackles the problem from the user's prospective rather than the provider's. The main goal is to provide the best possible service composition which gives the problem a brokering direction instead of the focus on cloud data center performance. SLA conditions are considered an input guaranteed by the cloud provider regardless of how they are achieved.

Wei *et al.* [21] address Quality of Service (QoS) constrained resource allocation problem for cloud computing services. They present a game-theoretic method to find an approximate solution of this problem. Their proposed solution executes in two steps: (i) Step 1: Solving the independent optimization for each participant of game theory;(ii) Step 2: Modifying the multiplexed strategies of the initial solution of different participants of

Step 1 taking optimization and fairness into consideration. The model in [21] represents a problem of competition for resources in a cloud environment. Each system/node/machine represents a resource that has a corresponding cost and execution time for each task. More granularity is needed in terms of considering the multiple degrees of computational and network resources when scheduling. Memory, storage, computational powers and bandwidth (at least) should be considered separately in an ideal model. Moreover, network resource impact is not considered thoroughly in [5]. Also, no detailed discussion for Virtualization scenarios was given.

In [22], Beloglazov *et al.* define an architectural framework in addition to resource allocation principles for energy efficient cloud computing. They develop algorithms for energy efficient mapping of VMs to suitable physical nodes. They propose scheduling algorithms which take into account QoS expectations and power usage characteristics of data center resources. This includes, first, allocating VMs using modified best fit decreasing method and then optimizing the current VM allocation using VM migration. Considering challenges migration might cause in terms of performance hiccups caused by copying and moving delays and scheduling challenges along with provider vulnerability for SLA violations, a solution that minimizes the need for VM migration is a preferable one. No deadline for tasks is considered.

Duan *et al.* [23] formulate the scheduling problem for large-scale parallel workflow applications in hybrid clouds as a sequential cooperative game. They propose a communication and storage-aware multi-objective algorithm that optimizes execution time and economic cost while fulfilling network bandwidth and storage requirements constraints. Here, the computation time is modeled as a direct function of the computation site location and the task instead of using a unified unit for task size. Memory was not used as a resource. Task deadlines are not considered. The goal is to complete a set of tasks that represent a specific application. This model is closer to job execution on the grid rather than the model more common in the cloud which is reserving a VM with specific resource requirements and then running tasks on them. Moreover, the assumption presented is that data exchange requests can run concurrently with the computation without any dependency.

One more variation can be seen in [24] in which two scheduling algorithms were tested, namely , green scheduling and round robin. The focus was energy efficiency again

but the model offered contains detailed network modeling as it was based on NS-2 network simulator. The user requests are modeled as tasks. Tasks are modeled as unit requests that contain resource specification in the form of computational resource requirements (MIPs, memory and storage) in addition to data exchange requirements (task size variable representing the process files to be sent to the host the task scheduled on before execution, data sent to other servers during execution and output data sent after execution). There was no optimization model offered.

When looking at the solutions available in the literature, it is evident that each experiment focuses on a few aspects of the resource allocation challenges faced in the area. We try to summarize the different aspects in Table 2.1.

An ideal solution would combine the features/parameters in Table 2.1 to build a complete solution. This would include an optimization formulation that covers computational and network resources at a practical granularity level. Dealing with bandwidth as fixed commodity is not enough. Routing details of each request are required to reflect the hot spots in the network. That applies for computational resources as well. CPU, memory and storage requirements constitute a minimum of what should be considered. Moreover, A number of previous efforts concentrate on processing resources while some focus on networking resources. The question arising here is: How can we process client VM reservation requests keeping in mind their data exchange needs? The common approach is to perform the VM placement and the connection scheduling separately or in two different consecutive steps. This jeopardizes the QoS conditions and forces the provider to take mitigation steps when the VM's computational and network demands start colliding. These steps include either over provisioning as a precaution or VM migration and connection preemption after issues like network bottlenecks start escalating. Minimizing VM migration incidents is a major performance goal. Off-line VM migration, however fast or efficient it may be, means there is a downtime for clients. This does not really comply with a demanding client environment where five 9's availability (99.999% of the time availability) is becoming an expectation. As for online migration, it pauses a load with more copying/redundancy required. These challenges associated with VM migration cause cloud computing solution architects to welcome any solution that does not include migration at all.

This shortcoming calls for a resource allocation solution that considers both sides at

Table 2.1: A comparison of cloud resource allocation efforts

Feature/ Reference	[4]	[5]	[6]	[7]	[8]	Proposed solution
Optimization model offered	Yes	Yes	No	Yes	No	Yes
Perspective	User/broker	Provider	Provider	Provider	Provider	Provider
Computation resources	VM types (generic)	Computational node (no memory)	CPU, Memory and Storage	Computational node (no memory)	CPU, Memory and Storage	CPU, Memory and Storage
Network resources	amounts of data	Does not affect execution time	No	BW & amounts of data	BW, source & destination	BW, source & destination
Scheduling considers both network & computational resources	No	No	No	Yes	No	Yes
Request deadline/ lifetime	No	Yes	No	No	Yes	Yes
VM modeling	VMs & VM appliances offered	No	VM placement & migration considered	No	VM placement & migration considered	VM placement considered

the same time. This solution would consider the VM future communication demands along with computational demands before placing the VM. In this case, the network demands include not only the bandwidth requirements as a flat or a changing number, but also the location of the source/destination of the requested connection. This means the nodes/VMs that will (most probably) exchange data with the VM. As these closely tied VMs are scheduled relatively near each other, network stress is minimized and the need to optimize the VM location is decreased dramatically.

In this work, we aim to tackle the problem of allocating client VM reservation and connection scheduling requests to corresponding data center resources while achieving the cloud provider's objectives. Our main contributions include the following:

1- Formulate the resource allocation problem for cloud data centers in order to obtain the optimal solution. This formulation takes into consideration the computational resource requirements at a practical granularity while considering the virtualization scenario common in the cloud. It also considers conditions posed by the connection requests (request lifetime/deadline, bandwidth requirements and routing)at the same time. An important advantage of this approach over approaches used in previous efforts is considering both sets of resource requirements simultaneously before making the scheduling decision. This formulation is taken from the providers' perspective and aims at maximizing performance.

2- Make the formulation generic in a way that it does not restrict itself to the limited environment of one data center internal network. The connection requests received can come from one of many geographically distributed private or public clouds. Moreover, the scheduler is given the flexibility to place the VMs in any of the cloud provider's data centers that are located in multiple cities. These data centers (clouds) represent the network communicating nodes. The complete problem is solved using IBM ILOG CPLEX optimization library[15].

3- *Introduce multiple heuristic methods to preform the two phases of the scheduling process.* Three methods are tested for the VM reservation step. Two methods are tested for scheduling connections. The performance of these methods is investigated and then compared to some of the currently available methods mentioned earlier.

4- *Introduce a suboptimal method to solve the same problem for large scale cases.* This method is based on a technique of decomposing the original problem into two separate

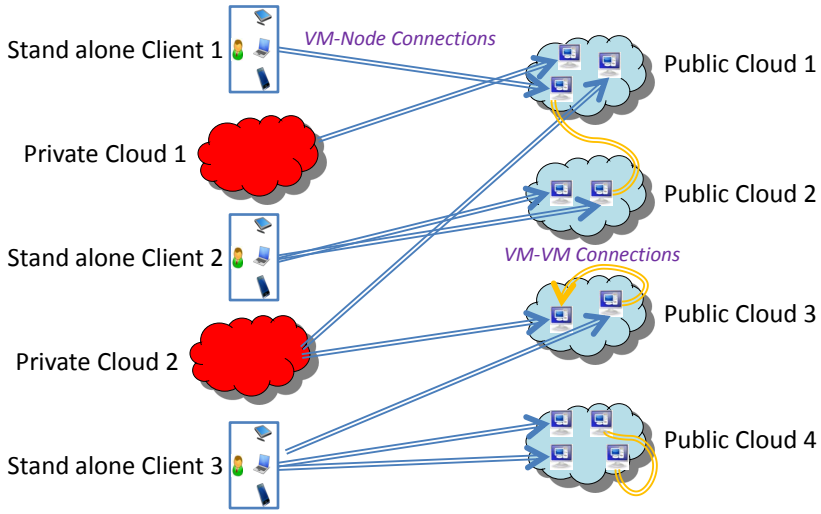

Figure 2.2: An example of a cloud provider-client network : clients can connect from their private clouds, their headquarters or from a singular machine through the Internet. The provider data centers represent public clouds

sub-problems. The first one is referred to as master problem which performs the assignment of VMs to data center servers based on a VM-node relation function. The second one, termed as subproblem, performs the scheduling of connection requests assigned by master problem. This suboptimal method achieves better results than the heuristic methods while getting these results in more feasible time periods in contrast to the optimal formulation.

2.3 Model description

We introduce a model to tackle the resource allocation problem for a group of cloud user requests. This includes the provisioning of both computational and network resources of data centers. The model consists of a network of data centers nodes (public clouds) and client nodes (private clouds). These nodes are located in varying cities or geographic points as in Fig. 2.2 They are connected using a network of bidirectional links. Every link in this network is divided into a number of equal lines (flows). It is assumed that this granularity factor of the links can be controlled. We also assume that each data center contains a number of servers connected through Ethernet connections. Each server will have a fixed amount of memory, computing units and storage space. As an initial step, when clients

Table 2.2: An example of a set of Resource Allocation requests

Client	Request	Type	Start	Duration	Source	Destination
C-1	Res VM1	High-CPU	T=10	125	-	-
C-2	Res VM2	High-Storage	T=15	400	-	-
C-1	Res VM3	Standard	T=20	150	-	-
C-2	Res VM4	High-Memory	T=10	70	-	-
C-1	Req con	VM-VM	T=15	10	VM1	VM3
C-1	Req con	VM-C	T=18	20	VM3	C1
C-2	Req con	VM-VM	T=25	8	VM4	VM2
C-2	Req con	VM-C	T=30	30	VM4	C2

require cloud hosting, they send requests to reserve a number of VMs. All of these VMs can be of the same type or of different types. Each cloud provider offers multiple types of VMs for their clients to choose from. These types vary in the specification of each computing resource like memory, CPU units and storage. We will use these three types of resources in our experiment. Consequently, each of the requested VMs is allocated on a server in one of the data centers. Also, the client sends a number of requests to reserve a connection. There are two types of connection requests: 1- A request to connect a VM to another VM where both VMs were previously allocated space on a server in one of the data centers (public clouds). 2- A request to connect a VM to a client node. Here, the VM located in a data center node connects to the client headquarters or private cloud. The cloud provider-client network is illustrated in Fig. 2.2 For every request, the client defines the source, the destination, the start time and the duration of the connection. Thus, an initial objective is to minimize the average tardiness of all connection requests. A sample of client requests is shown in Table 2.2. Requests labeled Res indicate a VM reservation. Requests labeled Req con indicate a connection request between a VM and a client node or between 2 VMs. An example of the VM configuration is shown in Table 2.3 [1].

2.4 Mathematical Formulation

To solve the problem of resource scheduling in cloud computing environment, we introduce an analytical model where we formulate the problem as a mixed integer linear problem. We model the optimization problem of minimizing the average tardiness of all reservation

connection requests while satisfying the requirements for virtual connection requests of different clients. This model is solved using IBM ILOG CPLEX software for a small set of requests.

2.4.1 Notations

Environment and network parameters are described below. A set of VMs and a set of servers are represented by VM and Q respectively. M_{qm} represents the amount of resources (e.g. memory) available on a server where $q \in Q$ and $m \in \{memory(mem), CPUunit(cu), storage(sg)\}$ such that $M_{qm} = 30$ indicates that available memory on server q is 30 GB assuming that m denotes a specific type of required resource, i.e., memory on a server. K_{vm} is used to represent the amount of resources needed for every requested VM such that $K_{vm} = 7$ indicates that the VM $v \in VM$ requires 7 GB of memory assuming that m denotes memory resource on a server. The set of network paths and a set of links are represented by P and L respectively. a_{lp} is a binary parameter such that $a_{lp} = 1$ if link $l \in L$ is on path $p \in P$; 0 otherwise. In our formulation, fixed alternate routing method is used with a fixed size set of paths available between a node and any other node. These paths represent the alternate paths a request could be scheduled on when moving from a server residing in a node a to a server in any other node. b_{qcp} is a binary parameter such that $b_{qcp} = 1$ if path $p \in P$ is one of the alternate paths from server, $q \in Q$ to server, $c \in Q$; 0 otherwise. I represents a set of connection requests. Every connection request, $i \in I$ is specified by a source (s_i), a destination (d_i), requested start time (r_i) and connection duration (t_i). $TARD$ represents the allowed tardiness (accepted delay) for each connection request. The formulation covers scenarios in which networks can divide a link into shares or streams to allow more flexibility with the formulation and cover a wide set of scenarios. The set of shares (wavelengths in the case of an optical network) could contain any number of wavelengths based on the problem itself. The set λ is the set of all available wavelengths in the network. The parameter h used in constraint 6 indicates a large number that helps to ensure the solution is derived according to the conditions in the constraint. In addition, the binary parameter W_{ij} indicates if request i is scheduled before request j. Using this parameter ensures constraint 6 is tested only once for each pair of requests.

2.4.2 Decision Variables

F_i is an integer decision variable which represents the scheduled starting time for connection request, $i \in I$. X_{vq} is a binary decision variable such that $X_{vq} = 1$ if $v \in VM$ is scheduled on server $q \in Q$. Y_{ipw} is a binary decision variable such that $Y_{ipw} = 1$ if request, $i \in I$ scheduled on path, $p \in P$ and wavelength, $w \in \lambda$

2.4.3 Objective Function

The problem is formulated as a mixed integer linear programming (MILP) problem. The objective of the MILP is minimizing the average tardiness of client connection requests to and from VMs. Tardiness here is calculated as the difference between the requested start time by the client (represented by r_i) and the scheduled start time by the provider (represented by F_i). The solver looks for the solution that satisfies clients in the best way while not harming other clients' connections. The solution works under the assumption that all clients requests have the same weight/importance to the provider. The objective function of the problem is as follows:

$$MIN \sum_{i} (F_i - r_i) \qquad i \in I, \tag{2.1}$$

2.4.4 Constraints

The objective function is subjected to the following constraints:

$$\sum_{q \in Q} X_{vq} = 1, \qquad v \in VM, \tag{2.2}$$

$$\sum_{p \in P} \sum_{w \in \lambda} Y_{ipw} = 1, \qquad i \in I, \tag{2.3}$$

$$\sum_{v \in VM} X_{vq} \times K_{vm} <= M_{qm}, \qquad q \in Q, m \in \{m, c, s\}, \tag{2.4}$$

$$Y_{ipw} + (X_{s_iq} + X_{d_ic} - 3b_{qcp}) <= 2, \quad i \in I, q \in Q, c \in Q, p \in P, w \in \lambda, \tag{2.5}$$

$$\sum_{p \in P} [(t_i \times a_{lp} \times Y_{ipw}) + (h \times a_{lp} \times Y_{ipw}) + (h \times a_{lp} \times Y_{jpw})] + F_i - F_j + h \times W_{ij} <= 3h, \quad i, j \in I, l \in L, w \in \lambda, \tag{2.6}$$

$$W_{ij} + W_{ji} = 1, \qquad i, j \in I, \tag{2.7}$$

$$X_{vq}, Y_{ipw}, W_{ij} \in \{0, 1\}, \tag{2.8}$$

$$F_i - r_i >= 0, \qquad i \in I, \tag{2.9}$$

$$F_i - r_i <= TARD, \qquad i \in I, \tag{2.10}$$

$$F_i, r_i >= 0, \qquad i \in I. \tag{2.11}$$

In Constraint 2.2, we ensure that a VM will be assigned exactly to one server. In Constraint 2.3, we ensure that a connection request will be assigned exactly on one physical path and one wavelength(stream/share of a link). In Constraint 2.4, we guarantee that VM will be allocated on servers with enough capacity of the computational resources required by the VMs. In Constraint 2.5, we ensure that a connection is established only on one of the alternate legitimate paths between a VM and the communicating partner (another VM or client node). In Constraint 2.6,we ensure that at most one request can be scheduled on a certain link at a time on each wavelength and that no other requests will be scheduled on the same link and wavelength until the duration is finished. Constraint 2.7 ensures Constraint 2.6 will only be tested once for each pair of requests. It indicates that request *i* will start before request *j*. In Constraints 2.9 and 2.10, we ensure that the scheduled time for a request is within the tardiness window allowed in this experiment.

Table 2.3: VM configuration for the 3 instance (VM) types used in the experiment as offered by Amazon EC2 [1].

Instance Type	Standard Extra large (SXL)	High Memory Extra Large (MXL)	High CPU Extra Large (CXL)
Memory	15 GB	17 GB	7 GB
CPU (EC2 units)	8	6.5	20
Storage	1690 GB	490 GB	1690 GB

2.5 Heuristic solution

2.5.1 Heuristic model

The proposed model in this paper tackles the resource allocation challenges faced when provisioning computational resources (CPU, memory and storage) and network resources. A central controller manages these requests with the objective of minimizing average tardiness and request blocking. The solution aims at solving the provider's cost challenges and the cloud applications performance issues.

For every request, the client defines the source, destination, start time and duration of the connection. Thus, this problem falls under the advance reservation category of problems.

The central controller (could be a Software Defined Networking controller (SDN) for example) keeps the data tables of the available network paths, available server resources and connection expiration times in order to handle newly arriving requests. The controller then allocates the requested VMs on servers according to the method or policy used. It updates the resource availability tables accordingly. After that, the controller schedules and routes connection requests to satisfy the client requirements. Network path availability tables are also updated. As an initial objective, the controller aims at minimizing the average tardiness of all the advance reservation connection requests. Also, a second objective is minimizing the number of the blocked requests. This objective is to be reached regardless of what path is used. Heuristic policies/techniques proposed aim at getting good, although not mathematically optimal, performance metrics while providing this feasible solution within acceptable amounts of time.

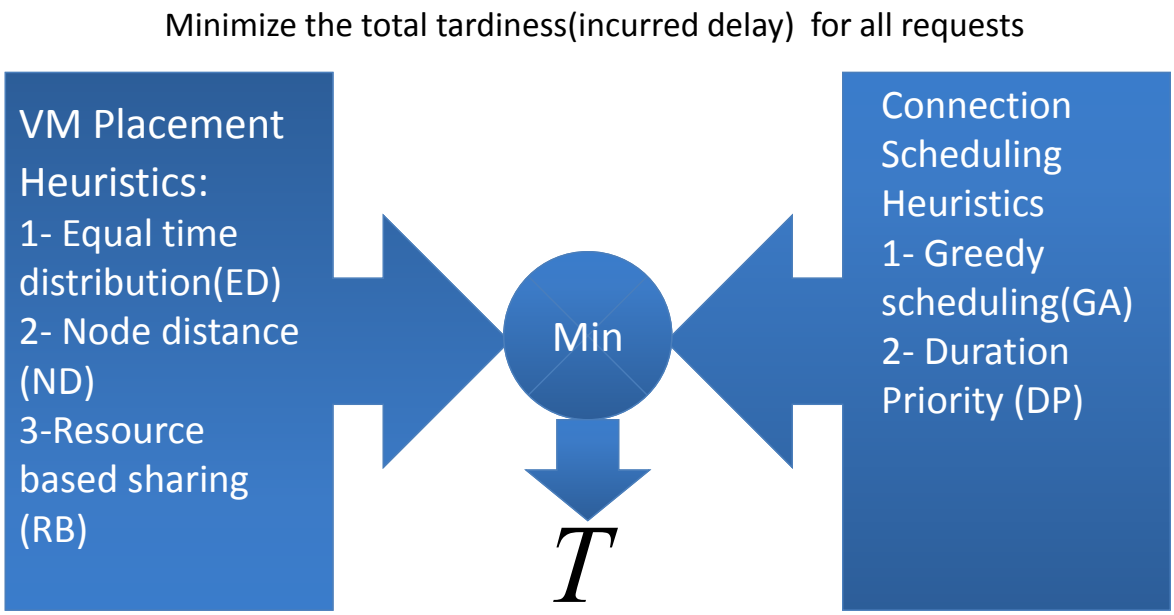

Figure 2.3: The allocation process using heuristic techniques

2.5.2 Heuristic techniques for minimizing tardiness

The allocation process is divided into two consecutive steps as in 2.3.

1- Allocation of VMs on data center servers. Here, all the VM reservation requests are served based on server resource availability before any connection request is served.

2- Scheduling of connection requests on the available network paths. This happens after all VMs have been allocated resources and started operation on the servers.

For the first subproblem, three heuristic techniques were evaluated. For the second step (subproblem), two heuristic techniques were tested. For a complete experiment, one heuristic for each subproblem is used. These heuristics are divided as follows:

2.5.2.1 VM reservation heuristic techniques

a) Equal Time Distribution Technique (ED):

In this heuristic, TM_i is the total time reserved by connection requests from the virtual machine VM_i (sum of the connection durations). Next, the share of one server is calculated by dividing the total time units all the VMs have requested by

Input: $Virtual\ machine\ set\ VM,\ Server\ set\ Q,\ connection\ request\ set\ R$
Output: $Allocation\ of\ VMs\ on\ servers,\ TM_i\ has\ the\ total\ connection\ time\ requested\ by\ VM_i$
for $TM_i \in TM$ **do**
  $TM_i = 0$
end for
for $VM_i \in VM$ **do**
  for $R_j \in R$ **do**
    if $R_j.source = VM_i\ OR\ R_j.dest = VM_i$ **then**
      $TM_i = TM_i + R_j.duration$
    end if
  end for
end for
$TM_{total} = \sum_i TM_i$
$ServerShare = TM_{total}\ /|Q|$
$i = 0$
for $S_j \in Q$ **do**
  $ThisServerShare = ServerShare$
  while $S_j\ isNotFull\ AND\ ThisServerShare > TM_i$ **do**
    $Schedule\ VM_i\ on\ S_j$
    $ThisServerShare = ThisServerShare - TM_i$
    $i = i + 1$
  end while
end for

Figure 2.4: Equal time distribution heuristic technique

the number of servers. This is based on the assumption that all servers have the same capacity (for computational and network resources). Then, for each server, VMs are allocated computation resources on the corresponding servers one by one. When the server is allocated a number of VMs that cover/consume the calculated server share, the next VM is allocated resources on the following server and the previous steps are repeated. The algorithm is described in pseudo code in Fig. 2.4.

b) Node Distance Technique (ND):

First, the average distance between each two nodes is calculated. The two nodes furthest from each other (with maximum distance) are chosen. Then, the maximum number of VMs is allocated on the servers of these two nodes. Next, the remaining nodes are evaluated, the node with maximum average distance to the previous two nodes is chosen. The same process is repeated until all the VMS are scheduled. The algorithm is described in pseudo code in Fig. 2.5. *fillNode* here is a function that basically tries to schedule as many VMs as possible on the called node until the node's resource are exhausted. *fillNode* is illustrated in Fig. 2.6

c) Resource Based Distribution Technique (RB):

In this heuristic, the choice of the server is based on the type of VM requested. As shown in Table III, three types of VMs are used in the experiment: i) High Memory Extra Large (MXL) has high memory configuration; ii) High CPU Extra Large (CXL) has a high computing power; iii) Standard Extra large (SXL) is more suited to typical applications or the ones that need a lot of storage space. Depending on the type of VM requested by the client, the heuristic picks the server with the highest amount of the available resources. The VM then is allocated resources on that server. This causes the distribution to be more balanced.

2.5.2.2 Connection reservation heuristic techniques

a) Duration Priority Technique (DP):

In this heuristic, connections with the shortest duration are given the priority. First, connection requests are sorted based on the requested duration. The following step is to pick the connection with the shortest duration and schedule it on the shortest path available. This step is repeated until all connection requests are served. The

Input: $Virtual\ machine\ set\ VM,\ Server$
$set\ Q,\ Node\ set\ N,\ Path\ set\ P,$
$where\ P_{ijk}\ is\ path\ k\ between\ nodes$
$i\ and\ j,\ NP\ is\ a fixed\ Number\ of\ paths$
$between\ node\ i\ and\ node\ j$
Output: $Allocation\ of\ VMs\ on\ servers$
for $N_i \in N$ **do**
  for $N_j \in N$ **do**
    $A[i][j] = \sum_k^{NP} P_{ijk}.Length/NP$
  end for
end for
$Pick\ 2\ nodes\ x,\ y\ with\ max\ A[x][y]$
$U = \{x, y\}$
$RemVMs = |VM|$
$RemVMs = fillNode(x, RemVMs)$
$RemVMs = fillNode(y, RemVMs)$
while $U \neq N\ AND\ RemVMs > 0$ **do**
  $maxDist = 0$
  for $N_i \in N\ AND\ N_i \notin U$ **do**
    $avgDist = 0$
    for $B_j \in U$ **do**
      $avgDist = avgDist + A[B_j][N_i]$
    end for
    if $avgDist > maxDist$ **then**
      $maxDist = avgDist$
      $NextNode = N_i$
    end if
  end for
  $RemVMs = fillNode(NextNode, RemVMs)$
  $U = U \cup \{NextNode\}$
end while

Figure 2.5: Node distance heuristic technique

Function :fillNode
Input: $Virtual\ machine\ set\ VM,\ Node\ x,$
$RequestedVMs,\ server\ set\ Q$
Output: $servers\ in\ node\ x\ filled\ with\ max$
$VMs\ possible$
$i = |VM| - RemVMs$
for $S_j \in Q\ and\ S_j\ residing\ in\ Node\ x$ **do**
  while $S_j\ isNotFull\ AND\ i < |VM|$ **do**
    $Schedule\ VM_i\ on\ S_j$
    $i = i + 1$
  end while
end for
$return\ i$

Figure 2.6: Function :fillNode

Input: $Path\ set\ P\ where\ P_{xyk}\ is\ path\ k$
$between\ nodes\ x\ and\ y,$
$and\ connection\ request\ \ set\ R$
$T\ is\ the\ allowed\ tardines\ per\ request$
Output: $Scheduling\ of\ network\ connection$
$requests\ on\ network\ paths$
$Sort\ R\ in\ descending\ order\ based\ on R_i.duration$
for $R_i \in R$ **do**
  for $t = R_i.RST\ to\ R_i.RST + T$ **do**
    $Pick\ shortest\ path\ P_{xyk}\ where\ R_i.source = x\ and\ R_i.destination = y$
    if $P_{xyk}\ isAvailable(t, R_i.duration)$ **then**
      $Schedule\ R_i\ on\ P_{xyk}\ at\ time\ unit\ t$
      $Move\ to\ next\ request$
    end if
  end for
end for

Figure 2.7: Duration priority heuristic technique

Input: $Path\ set\ P$
$where\ P_{xyk}\ is\ path\ k\ between\ nodes$
$x\ and\ y\ is\ \ Server\ set\ Q,\ connection$
$request\ set\ R,\ NP\ is\ Number\ of\ paths$
$between\ node\ x\ and\ node\ y$
$T\ is\ the\ allowed\ tardines\ per\ request$
Output: $Scheduling\ of\ network\ connection$
$requests\ on\ network\ paths$
$Sort\ R\ in\ descending\ order\ based\ on\ R_i.RST$
$(requested\ start\ time)$
for $R_i \in R$ **do**
  for $t = R_i.RST\ to\ R_i.RST + T$ **do**
    $x = R_i.source$
    $y = R_i.destination$
    for $k = 0\ to\ NP$ **do**
      if $P_{xyk} isAvailable(t, R_i.duration)$ **then**
        $Schedule\ R_i\ on\ P_{xyk}\ at\ time\ unit\ t$
        $Move\ to\ next\ request$
      end if
    end for
  end for
end for

Figure 2.8: Greedy heuristic technique

algorithm is described in pseudo code in Fig. 2.7.

b) Greedy Algorithm (GA):

In this heuristic, scheduling is based on the connection Requested Start Time (RST). Connection requests with earlier RST are scheduled on the first path available regardless of the path length.The algorithm is described in pseudo code in Fig. 2.8.

2.5.2.3 Complexity analysis of the heuristic solutions

The resource allocation problem in a cloud data center is a variation of the well known knapsack problem. The knapsack problem has two forms. In the decision form which is considered less difficult as it is NP-Complete- the question is: Can an objective value of at least K be achieved without exceeding a specific weight W? The optimization form of the problem which is the form we try to solve in this thesis- tries to optimize the possible objective value. The optimization form is NP-Hard. This means it is at least as hard as all the NP problems. There is no current solution in polynomial time for this form.

This motivated the introduction of the heuristic algorithms. It might be of interest to the reader to visit the complexity of the introduced heuristic algorithms.

First, we revisit the variables covered in this analysis. VM represent the VM set, N represent the set of nodes, S is the set of servers, R is the set of connection requests, T is the allowed tardiness per request and D is the average duration of a connection. This analysis is offered with the sole purpose of being an approximation of the time complexity to show that these algorithms run within polynomial time and in turn- can be practically used by large scale cloud networks. Looking at the introduced algorithms one by one, we find that Equal time distribution has a complexity of O($|VM||R| + |S|$). Node distance algorithms runs in O($|N^3| + |S| + |V|$). Resource based distribution runs in O($|V||S|$) which constitutes the quickest among the 3 VM placement algorithms we introduced. As for connection scheduling heuristics algorithms, Duration priority runs in O($|R|.lg|R| + |R|.T.D$) or O($|R|.(1+lg|R|.+T.D$). Finally, the greedy connection scheduling algorithm run in O($R.T.D$). Therefore, all the mentioned algorithms run in polynomial times and can yield a result for large scale problem in practical time periods.

2.6 Suboptimal Solution

Although an optimal solution can be obtained using the formulation in Section IV, this is only feasible for small scale problems. Even when using a 5-node network with 4 servers and 7 links connecting them, the number of optimization variables can be as big as 5000 variables when scheduling 50 requests that belong to 5 VMs. On the other hand, heuristic methods achieve feasible solution in relatively quick times but the solution quality cannot be proven. This motivates us to move to the next step which is finding a method that achieves a suboptimal solution. The method introduced here is based on a decomposition technique. We illustrate the method in the Fig. 2.9 The steps go as follows:

1- In Step 1, a set of known connection requests are preprocessed to generate interdependency measurements. This is figured out by calculating the frequency of communications between each two points in the network. To be more specific, the frequency of the connection requests between each VM_i and VM_j is calculated as well as the frequency of connection requests between VM_i and $node_k$ which represent a private cloud. This gives us an indication of which direction most of the VM's connections go. This is closely correlated with the dependencies this VM has and should ideally affect where it is scheduled.

2- In the second step, a utility function is constructed based on the connection frequency values generated in step 1. The utility function serves as the objective function of the master problem that allocates VMs on hosts.

3- Next, a master problem in which we handle the assignment of VMs to servers and connections to specific paths without scheduling them is generated. In other words, we solve for the decision variable X_{vq} without considering any scheduling constraints. This produces a feasible assignment for VMs that aims at scheduling interdependent VMs close to each other.

4- After getting the VM assignment locations, A subproblem in which we try to find the optimal scheduling for the input connections under these specific VM assignment conditions. In other words, we solve for the decision variable Y_{ipw}, F_i in the subproblem. The minimum tardiness produced from the subproblem is the objective value we are looking for. As in any decomposition based optimization, the success of the decomposition tech-

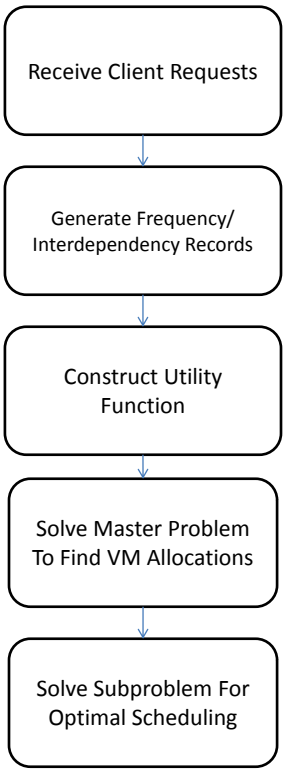

Figure 2.9: The suboptimal method step by step

nique depends on the way the solution of the master problem is chosen. We formulate the master problem and the subproblem as follows.

2.6.1 Master Problem Formulation

We first introduce Distance function. It represents the distance between two nodes measured by the number of links in the shortest path between them. A frequency function based on connection duration is also added. This is a function where the connection duration is preferred as dominant factor. The frequency function is a value that will represent interdependency between two VMs or between a VM and a private cloud (client node). Another alternative here is depending on the number of connections requested between these two points rather than the total amount of connection time. Once we calculate the frequency function values, the utility function is constructed as:

$$MIN \sum_{v \in VM} \sum_{u \in VM} \sum_{s \in Q} \sum_{q \in Q} (Freq_{vu} \times Distance_{sq} \times X_{vs} \times X_{uq}), \tag{2.12}$$

Subject to

$$2.2, 2.4. \tag{2.13}$$

The master problem finds the VM allocation that maximizes the value of point to point interdependency.

2.6.2 Subproblem

As the subproblem focuses on scheduling, its objective function is the same as in the optimal form, i.e., minimizing the average connection tardiness. In this case, the final value of the relaxed objective will come directly from the solution of the subproblem. The difference is that the subproblem already knows where the VMs are allocated and is scheduling connections accordingly. The objective of the sub-problem is as follows:

Table 2.4: Experiment parameter configuration.

Parameter	Value
Total number of servers	132
Servers/ data center	44
VM reservation requests	200
Connection requests	10000
RST distribution	Poisson with Lambda = 10
Connection duration distribution	Normal with mean = 200 time units
Source and destination distribution	Uniform
Allowed tardiness per request	ranging from 1 to 500 time units
Total experiment time	70,000 time units

$$MIN \sum_{i} (F_i - r_i) \qquad i \in I, \tag{2.14}$$

$$2.3, 2.5 to 2.11 \tag{2.15}$$

2.7 Results

2.7.1 Simulation Environment

The problem is simulated using a discrete event based simulation program and solved on a more practical scale using the heuristic search techniques discussed in the previous sections. The network used for the experiment is the NSF network. It consists of 14 nodes of which 3 are data center nodes and the rest are considered client nodes. Nodes are connected using a high speed network with link granularity chosen goes up to 3 lines (flows) per link. Fixed alternate routing method is used with 3 paths available between a node and any other node. Server configuration and request data parameters are detailed in Table 2.4. Preemption of connection requests is not allowed in this experiment.

2.7.2 Heuristics

As explained in the previous sections, every experiment includes two phases and hence two heuristics are needed: one to schedule VMs on servers and the other to schedule connection

requests. The five techniques explained earlier yield 6 possible combinations. However, We chose to show the results from the best 4 combinations (best 4 full-solutions). This is due to space constraints. The 4 chosen combinations cover all the 5 heuristics. The simulation scenarios and combined heuristics used for the two subproblems are as follows:

1-ED-GA: Equal Time Distribution technique and Greedy algorithm.

2-RB-DP: Resource Based Distribution technique and Duration Priority technique.

3-ED-DP: Equal Time Distribution technique and Duration Priority technique.

4-ND-DP: Node Distance technique and Greedy algorithm.

In figures 2.10 and 2.11, the results for blocking percentage are shown for the four methods. Fig. 2.10 shows a comparison between the blocked requests percentage produced when using each one of the four methods where the allowed tardiness is very small (1 time unit). Fig. 2.11 shows the same comparison when the allowed tardiness per request is large (30000 time units). It can be seen from both figures that ED-DP and RB-DP methods have shown a clear advantage. This indicates a clear advantage of using DP over GA when scheduling connection requests in tight or real time conditions. As it shows in Fig. 2.13, RB-DP has shown a decent advantage over ED-DP in terms of blocking percentage.

As for the average tardiness per request, the measurements are shown in Fig. 2.12 The figure shows a comparison between the average tardiness per request produced when using each one of the four methods, where the allowed tardiness is small (25 time unit). Once more, ED-DP and RB-DP methods have shown a clear advantage. Also, it is noticed from the figure the ED-DP produces slightly better results (less average tardiness) than RB-DP. Therefore, using RB-DP method is more suitable to scenarios where there is an emphasis on serving the largest number of requests. On the other hand, using ED-DP is more suitable to scenarios where the individual request performance or service level is prioritized over serving more requests.

2.7.3 Relaxed Solution Results

With regards to the network we tested on, a 5-node network was used in these tests with 2 as data center nodes and the rest are client nodes (private clouds). 4 servers were used in the tests with 2 servers in each data center. To connect the nodes in the substrate network, 7

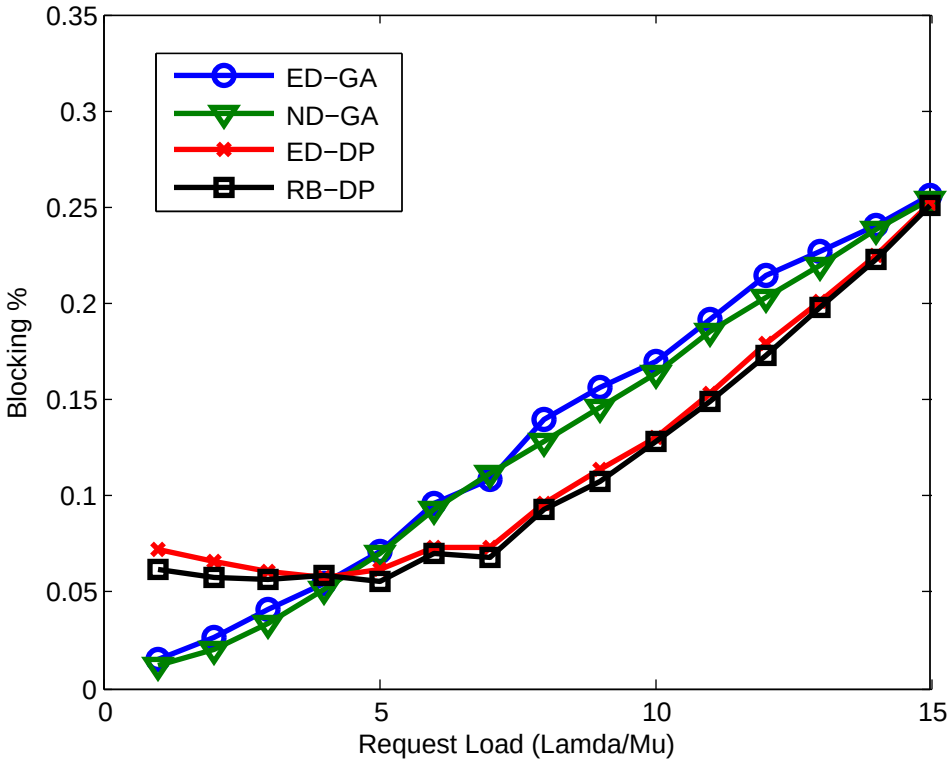

Figure 2.10: Request Blocking results for scheduling methods (allowed tardiness/request =1 time units

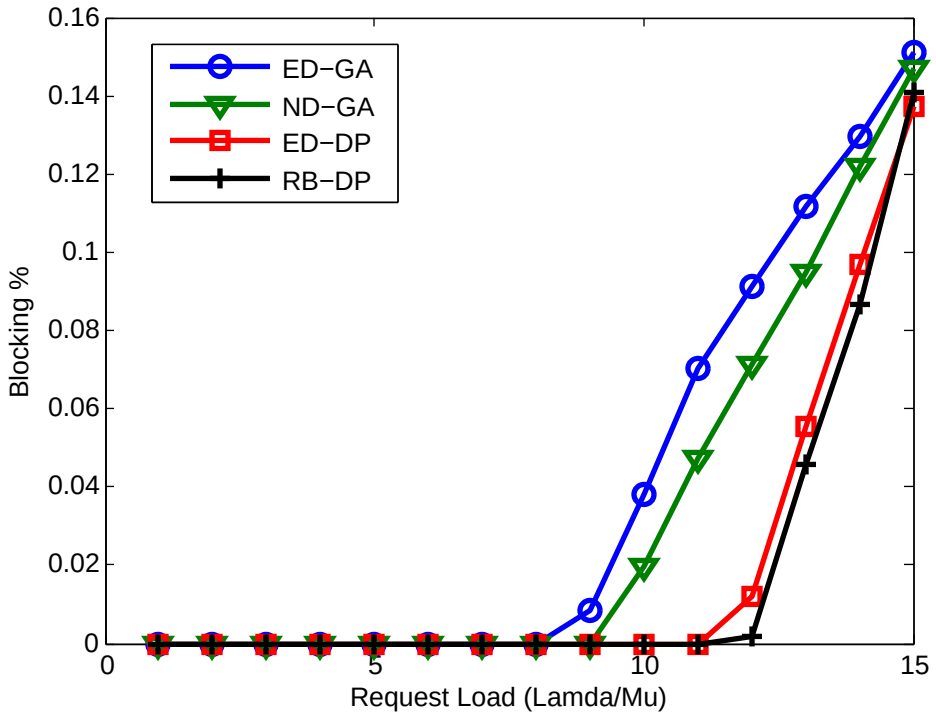

Figure 2.11: Request Blocking results for scheduling methods (allowed tardiness/request =30000 time units

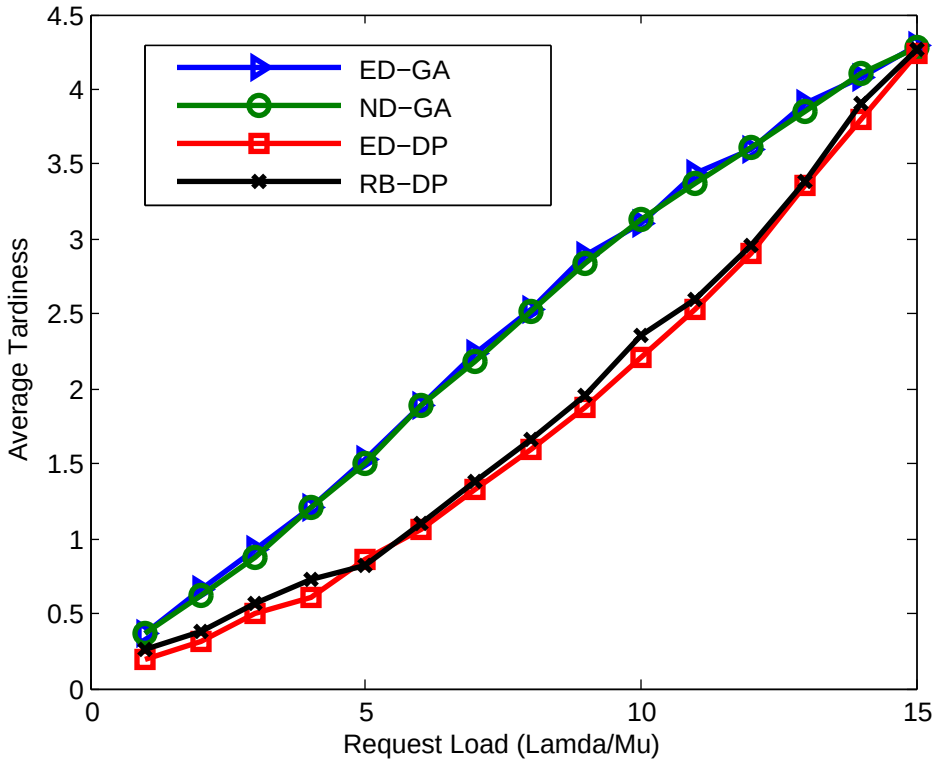

Figure 2.12: Request average tardiness results for scheduling methods (allowed tardiness/request =25 time units

Table 2.5: Optimal Vs. Relaxed solution values and execution times

Number of Requests	Network Load	Optimal Solution		Relaxed Solution	
		Average Tardiness Value	Execution Time	Average Tardiness Value	Execution Time
30	0.86	0	3 Sec	5.73	8.14 Sec
50	0.86	0	7 Sec	10.12	9 Sec
200	0.86	0	2 M 24 Sec	10.785	1 M 2 Sec

links were used and 20 different paths were defined. Two alternate routing paths were defined for each couple of nodes. The input contained data corresponding to 5 VM instances. The choice of this network is due to two factors. First, condensing requests in an architecture with limited resource puts the network under high load to eliminate the effect the network capacity would have on the result. This would allow more control by eliminating any factors related to network design or node distribution that might ease the pressure on the scheduling algorithm. This way, the problem size is controlled directly using only the parameters we are testing for which are the number of requests, their specification and their distribution. Second, it makes it easier to compare results and execution times of relaxed solution to those of the optimal solution.

For the connection requests coming from the clients, as in [25], their arrival rate values were set according to a Poisson process. The connection request lifetime (duration) was normally distributed with an average of 100 time units and the total number of connection requests was gradually increased from 20 up to 3000 requests. Every connection request is associated with a source, a destination, a requested start time and a duration. The source nodes/VMs were uniformly distributed.

To evaluate the optimal and relaxed solutions, we used the IBM ILOG CPLEX optimization studio v12.4. Both the optimal and relaxed solution were programmed using Optimization Programming Language (OPL) and multiple testing rounds were performed. Both solutions were tested for multiple values of normalized network load.

Table 2.5 shows a comparison between the objective values obtained using the optimal scheme vs. the values obtained from the relaxed (decomposed) scheme for small scale problems (up to 200 requests). While the optimal solution was able to schedule all requests

Table 2.6: Execution times & average tardiness for connection requests for large scale problems when increasing the average connection duration to 100 Time units

Network load	Heuristic Solution (RB-DP) average tardiness (percentage of duration/lifetime)	Relaxed Solution average tardiness	Relaxed Solution execution time
0.86	19.81%	2.88%	8 Min 21 Sec
0.93	21.18%	6.36%	8 Min 54 Sec
1	22.54%	9.08%	11 Min 31 Sec

without any delay (tardiness), the relaxed solution achieved an acceptable average tardiness in comparison. As noticed from the table, the execution times for the optimal scheme are slightly better for small data sets, but as the number of requests grows, the difference in execution times becomes evident. This goes on until the optimal solution becomes infeasible while the relaxed solution still executes in a relatively short period. The maximum number of requests the optimal solution is able to solve depends on the machines used and the network load parameters used to generate the input data.

Concerning large scale problems, the experimental results shown in Table 2.6 illustrate that the relaxed solution has achieved an acceptable average tardiness in comparison to the optimal solution. The effect of increasing the problem size on the value of average tardiness when using the relaxed solution is evident. The average tardiness achieved is less than 10% of the average request duration (lifetime). This is well within the bound set in [26] for acceptable connection tardiness which is half (50%) the lifetime or requested duration of the connection. This is also a considerable improvement over the performance of the heuristic solution which is shown in the same table (average tardiness values around 20% of request lifetime). The table also shows an increase in the average tardiness when increasing the number of requests (problem size). This is due to the fact that tardiness accumulates as priority is given to the request arriving earlier. In terms of execution time, as the number of requests grow, the difference in execution time between the optimal and relaxed solutions becomes evident. The optimal solution becomes infeasible while the relaxed solution still executes in a relatively short period scheduling 3000 requests in around in a period between 8-11 minutes depending on the network load. This -of course - is an example of offline execution on a personal computer which is we mainly use here for

Table 2.7: Connection Requests Acceptance rate for different network loads using the Relxed solution

Allowed tardiness (percentage of request lifetime or duration)	50%	200 %	1000%
Acceptance rate	86%	87%	100%
Average tardiness for accepted requests	1.98%	16.72%	219.767%

benchmarking purposes.

To illustrate the impact of the allowed tardiness parameter on the request acceptance ration, the results in Table 2.7 are presented. Using the heuristic solution with the combination RB-DP, the table shows the increase in the acceptance ration as we increase the allowed tardiness per request for a specific network load.

To measure the acceptance ratio, we introduced a maximum waiting period parameter. This parameter represents the period of time a connection request will wait to be served before it is considered blocked. For that, an ideal value is the same value used in [26], namely, half the request lifetime. In other words, If the connection waited for more than 50% of its duration and it was not scheduled then it is blocked or not served. Table 2.6 shows the acceptance ratio and the average tardiness for requests with an average duration of 100 time units.

Considering this scenario where requests with high tardiness are blocked presents a trade off between average connection tardiness and the percentage (or number) of blocked connections. It is noticed that the average tardiness decreases as we remove the requests with high tardiness and consider them blocked. An average tardiness of less than 2% of a request lifetime can be guaranteed if we are willing to sacrifice 13% of the requests as blocked.

Deciding weather to use this scenario or not is up to the cloud solution architects. This depends on the client sensitivity to the precision/quality vs. the speed of achieving results.

2.7.4 Comparison with Previous Solutions

When planning the comparison between the proposed solution and solutions available in the literature, we are faced with a challenge. As discussed in detail in Section II, the avail-

able solutions are diverse in terms of the parameters considered and the covered sides of the cloud resource allocation problem. This limits the number of solutions that can realistically be used to solve this particular flavor fo the problem. However, we were able to use the algorithms implemented in [22] (Modified best fit decreasing method) and [3] (GREEN scheduling) to solve the same problem and compare them to method we developed. The focus was the network capacity (minimizing blocking percentage) and performance(when blocking is not an issue, minimizing the average tardiness per served request). this comparison was performed for a smaller network first, in order to explore the stress effect on a cloud network. Then, the same comparison is performed for a larger network scenario. as in the previous experiments, the tests were performed for different problem sizes and various levels of allowed tardiness per requests.

2.7.4.1 Small network results

Fig. 2.13 shows the results of the request blocking percentage for the three algorithms as the allowed tardiness level increases. The figure show that out technique(RB-DP) first performs consistently better the Green scheduling algorithm while performing at the same level as MBFD before showing clear advantage for hig allowed tardiness. In terms of average tardiness, Fig. 2.14 shows that RB-DP starts by performing on the same level to the other two algorithms and while we increase the allowed tardiness level for requests, RB-DP shows clear advantage. The effect of increasing the allowed tardiness is basically eliminating the need to block requests in the experiments and instead focusing the experiment on showing the algorithm that can serve/schedule requests in the most efficient way and this, in turn, decreases the average tardiness per requests.

2.7.4.2 Large network results

The same trends carry on while testing on large scale networks. In Fig. 2.15, blocking percentage for the three algorithms for different problem sizes (represented by the number of requests submitted to the central controller per cycle). These results are shown for allowed tardiness level= 5 time units (very low level) which adds extra pressure to serve requests within a short period of their arrival and focuses the algorithms work on serving

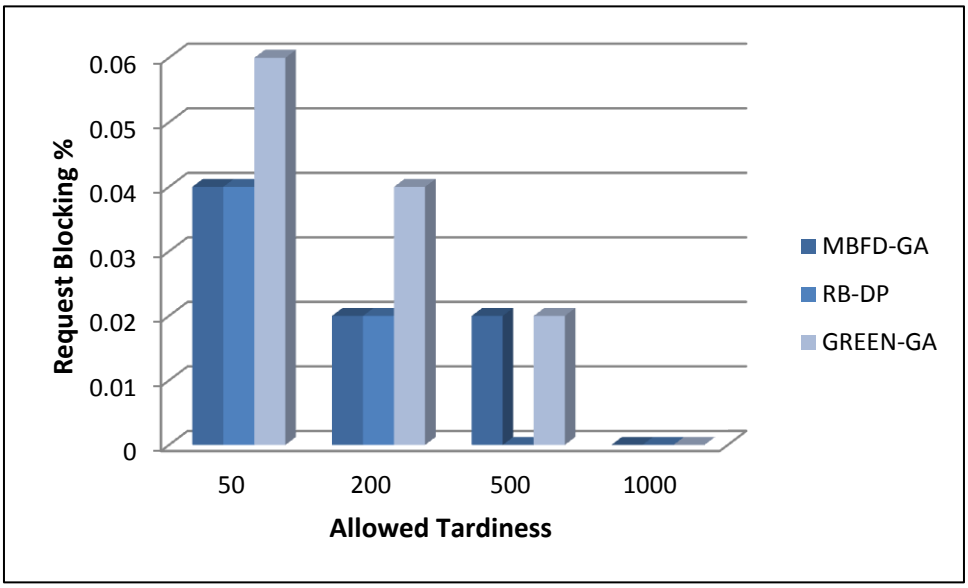

Figure 2.13: Request blocking percentage results for scheduling methods when allowed tardiness changes

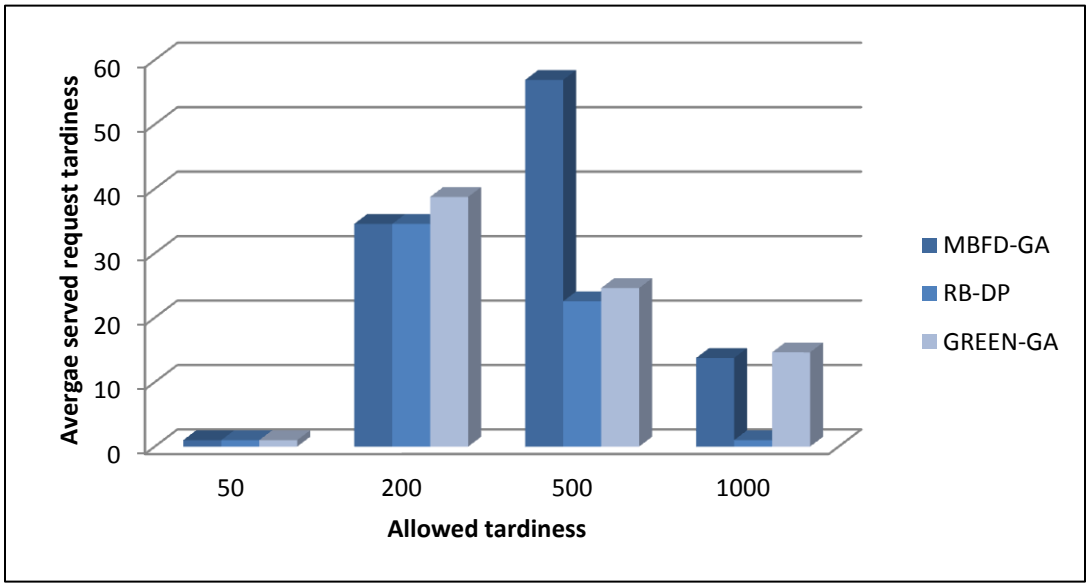

Figure 2.14: Request average tardiness results for scheduling methods when allowed tardiness changes

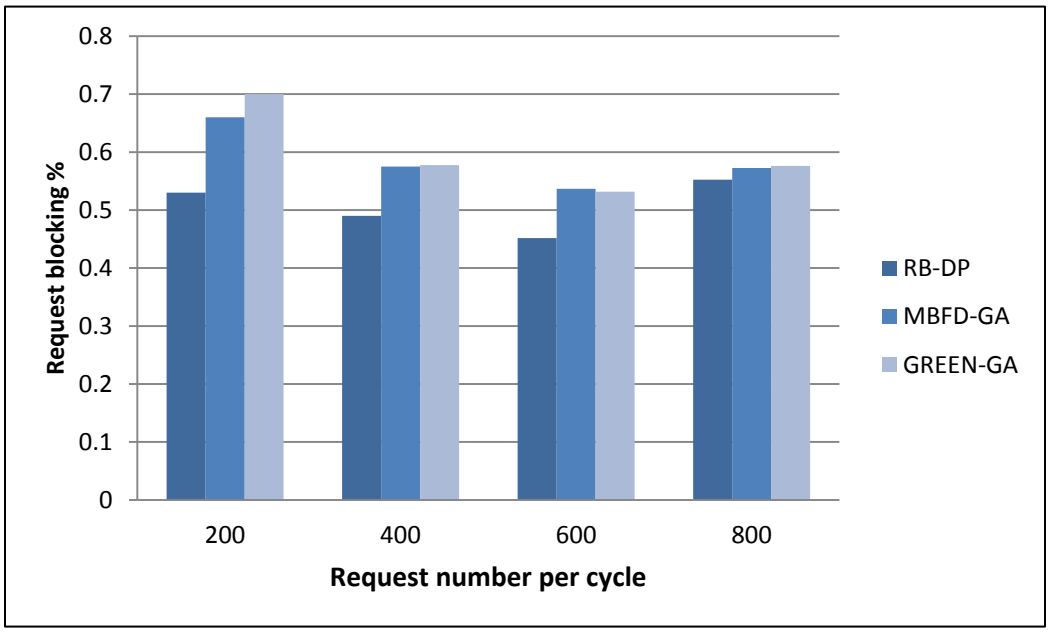

Figure 2.15: Request blocking percentage results for scheduling methods with allowed tardiness=5 units(very low)

the highest number of requests without focusing on tardiness levels. The figure shows that out technique(RB-DP) performs consistently better than the other two algorithms under high loads.

Fig. 2.16 explores the performance of the algorithms under high allowed tardiness levels. RB-DP offers clear advantage in terms of the blocking percentage metric for various allowed tardiness levels.

Moving to the second metric, Fig. 2.17 show the performance of the three algorithms in terms of average request tardiness while changing the allowed tardiness levels (or request lifetime) RB-DP performs on a comparable level to the other two algorithms for small allowed tardiness levels and then exceeds the performance of MBFD starting medium levels of request lifetimes and then clearly exceeds both algorithms with the higher levels starting 400 time units.

These results prove the potential our solution has in terms achieving better performance in both blocking percentage (more accepted connection requests and less network congestion) and average tardiness (better Quality of service conditions for cloud users).

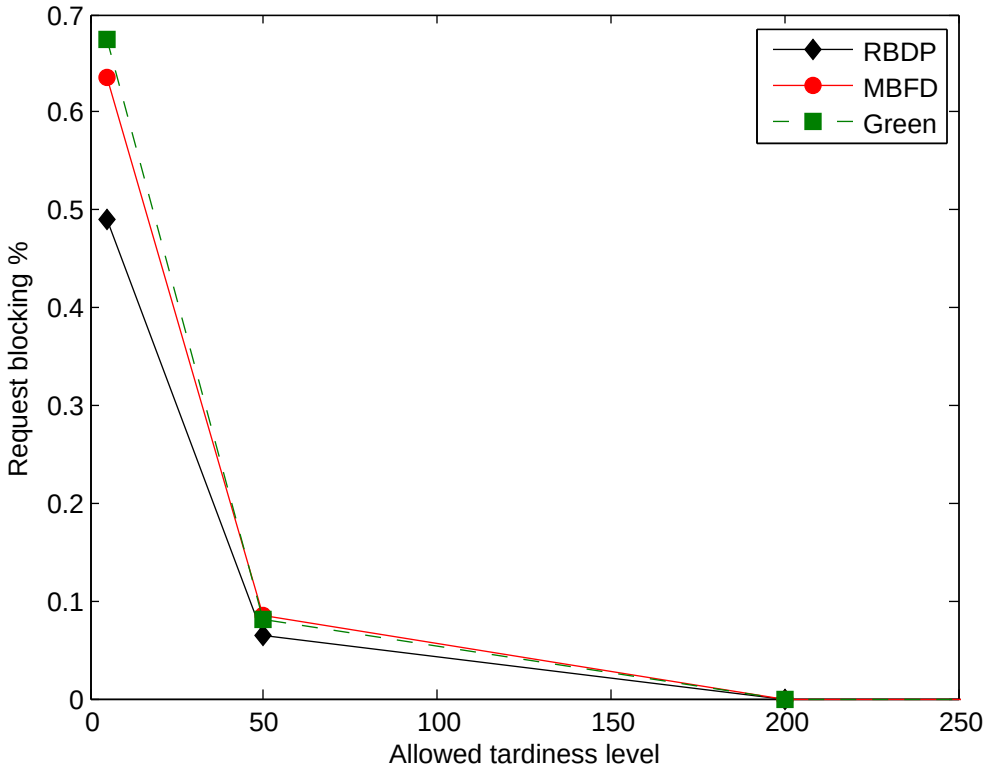

Figure 2.16: Request blocking percentage results for scheduling methods while changing allowed tardiness levels

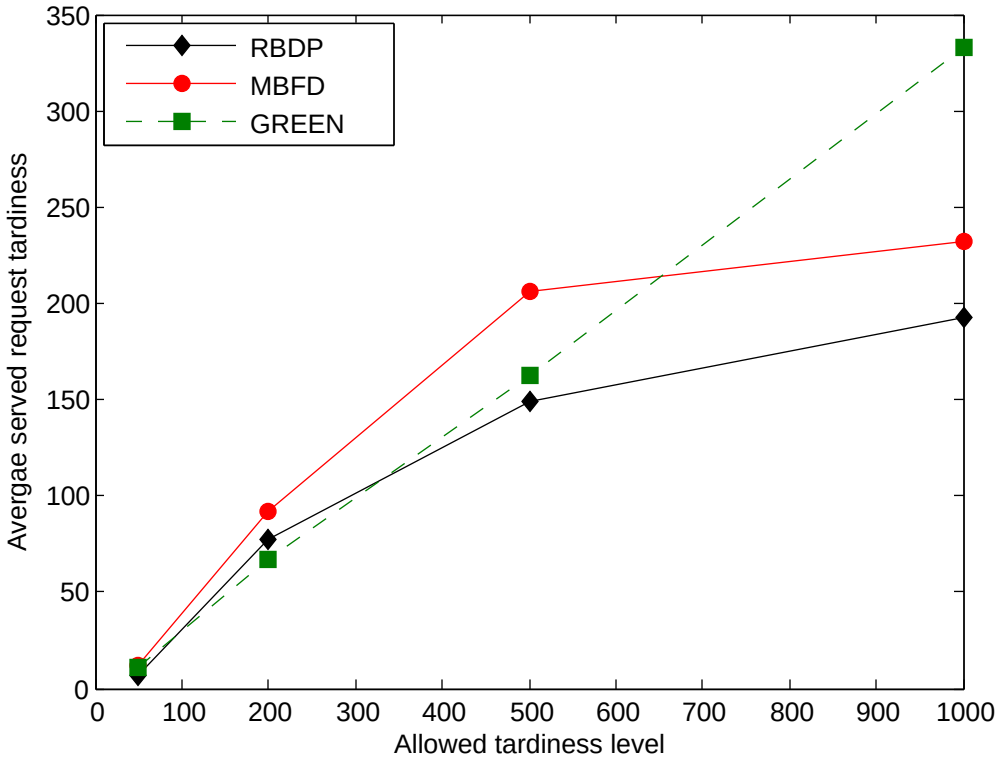

Figure 2.17: Request average tardiness results for scheduling methods (allowed tardiness/request =25 time units

2.8 Chapter Summary

We introduced a comprehensive solution to tackle the problem of resource allocation in a cloud computing data center. First, the problem was formulated as a mixed integer linear model. This formulation was solved using an optimization library for a small data set. However, finding the optimal solution for larger more practical scenarios is not feasible using the optimal mathematical formulation. Therefore, we introduced 5 heuristic methods to tackle the two sides of the problem, namely VM reservation and connection scheduling. The performance of these techniques was analyzed and compared. Although the solution scale issue is solved, a heuristic solution does not offer optimality guarantees. This constituted the motivation to introduce a suboptimal solution. The solution contained 4 steps that exploited the VM interdependency as a dominant factor in the VM allocation process. This allows us to solve the scheduling phase optimally in the following step which causes the solution to improve considerably. The relaxed solution achieved results matching with parameters preset in the literature for average connection tardiness. The results were also shown for the scenario where request blocking is allowed. Results were achieved without sacrificing the computational feasibility which shows our method to be a valid solution for reaching acceptable connection tardiness levels. Furthermore, the proposed solution was compared to two of the prominent algorithms in the literature. The proposed solution was shown to be advantageous in terms of minimizing both average request tardiness and blocking percentage for multiple cloud network scenarios. This makes it a strong candidate to be used in cloud scenarios where the focus is on metrics like more accepted connection requests and less network congestion or request average tardiness (better quality of service conditions for cloud users.

Chapter 3
An Evergreen Cloud: Optimizing Energy Efficiency in Green Cloud Computing Environments Based on Virtual Machine State Prediction

3.1 Introduction

To adapt to the surge in cloud technology demand levels, cloud providers are expected to implement more innovative and effective solutions for a list of long standing challenges faced by the industry. The aim here for a cloud provider is to serve the high volume of request which are received continuously from a diverse set of constantly changing (and moving) devices. As illustrated in Fig. 3.1, This is done by successfully receiving the request data from client device and then scheduling these requests to the corresponding virtual machine in the cloud data center based on the functionality or application required. From there, the computational part is done and then the results are sent back to the client. A major challenge in this scenario is how to serve these requests with the required performance while minimizing the energy the cloud data center users.

Energy efficiency in the cloud data center (DC) is one of the more pressing issues near the top of that list. As DCs expand, so does their energy consumption. U.S. data centers are on track to consume roughly 140 billion kilowatt-hours of electricity annually by

---

A version of this chapter has been submitted for publication in [27].

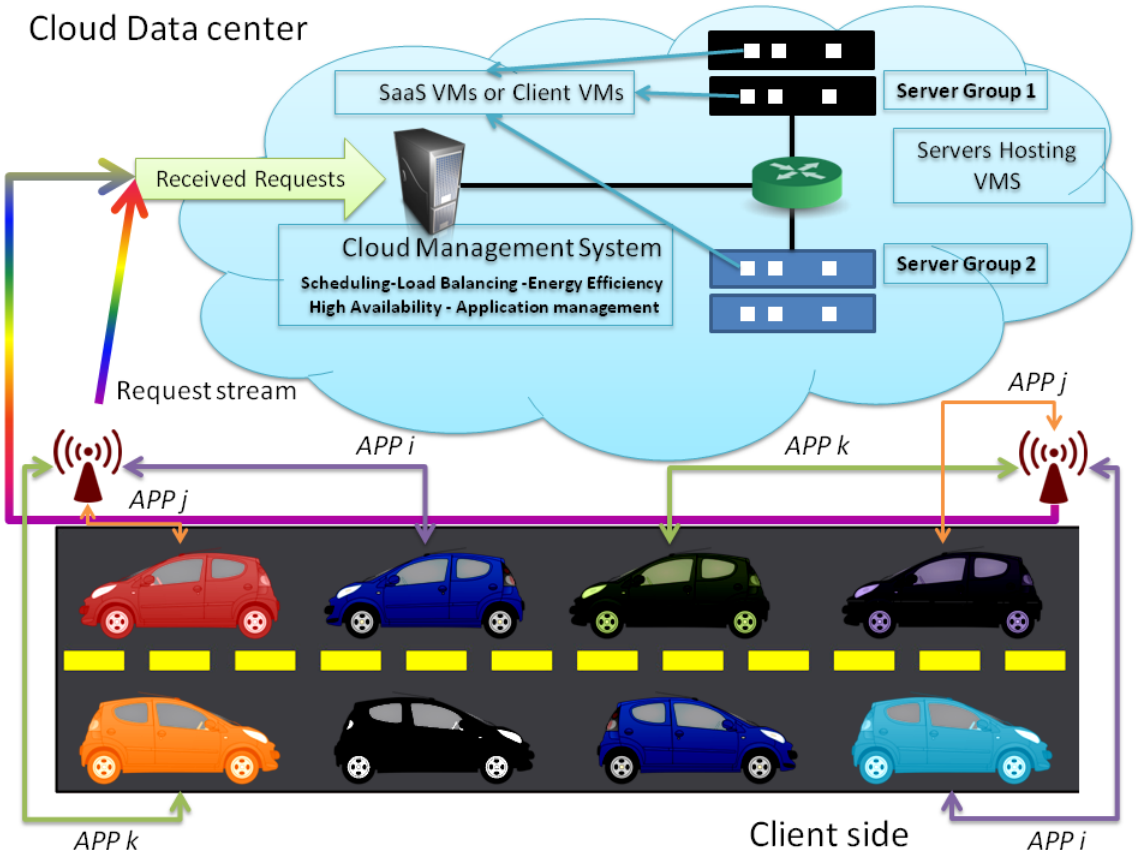

Figure 3.1: Heterogeneous network clients sending diverse computational requests to the public cloud

2020, equivalent to the output of 50 large power plants (each with 500 megawatts capacity)" [28]. This is not only due to the increasing amount of servers per DC, but also the individual server consumption of energy has increased too. The increase in energy consumption is a major concern to the data center owners because of its effect on the operational cost. It is also a major concern for governments because of the increase in data centers' carbon footprint.

Cloud technology adoption rates are another factor to look out for. 78% of U.S. small businesses will have fully adopted cloud computing by 2020, more than doubling the current 37%. [5] The percentage grows to 90% when looking at large businesses (larger than 1000 employees) [6]. The U.S. Small and Medium Business (SMB) cloud computing and services market will grow from 43 billion dollars in 2015 to 55 billion dollars in 2016. [5] This trend is consistent in Europe as well. The percentages of small, medium and large businesses adopting cloud technologies in UK is 46, 63 and 82% respectively. In Germany, the percentages are 50, 65 and 86%.

There is proven potential for client demand growth as well. In a survey conducted by the rightscale.com team, 68% of enterprises indicated they run less than a fifth of their application portfolios in the cloud. 55% of enterprises report that a significant portion of their existing application portfolios are not in cloud, but are built with cloud-friendly architectures [7]. This is used to serve major functions like data protection (backup), business continuity (replication, disaster recovery), archiving and file services and office enablement (sharing, synchronization, collaboration).

Power consumption in cloud data centers is a pressing issue for cloud providers. Power costs represent between 25% and 40% of the operational expenses of a data center. [8] The Natural Resources Defense Council (NDRC) published a data center efficiency assessment in Aug. 2014, as an attempt to depict the scale of data centers the world over. [28] The study mentions that . "If worldwide data centers were a country, they would be the globe's 12th-largest consumer of electricity. Another fact here is their assessment of energy efficiency. "An analysis by the NRDC in partnership with Anthesis finds that up to 30% of servers are obsolete or not needed and no longer needed, other machines are grossly underutilized." Persistent issues obscuring efficiency include:

- Peak provisioning.

- Limited deployment of virtualization technology.
- Failure to power down unused servers,
- Challenges with efficiency incentive programs.
- Competing priorities : (keeping costs low and maintaining high levels of security, reliability, and uptime for their clients.)

These are 5 out of the 8 main factors cited that affect power efficiency and stand in the way of a staggering 40% potential improvement in power consumption. These 5 factors are all largely affected by data center load planning and management. An efficient scheduling energy- aware algorithm is in need. This algorithm should exploit the benefits that come from virtualization technologies, optimal demand driven provisioning, and efficient load modeling.

In this work, we explore the possible venues to reach this efficient energy efficiency solution for cloud environments. This includes introducing a mathematical optimization model for the problem, a novel VM consolidation-based method and a novel non-consolidation based method. These methods are evaluated in terms of multiple critical energy efficiency metrics.

In the following Sections, we start by visiting the related work and defining our main contributions in Section 3.2. Section 3.3 presents the system model and then we proceed to present the mathematical formulation for the energy efficiency problem in virtualized cloud data centers in Section 3.4. In Section 3.5, a consolidation-based energy efficiency solution is proposed and details are put forward. Section 3.6 presents our novel non-consolidation-based energy efficiency solution: Smart VM Over Provision (SVOP). We describe the experimental setup in Section 3.7 then proceed to present and analyze the experimental results in Section 3.8. Section 3.9 concludes the chapter.

3.2 Related work

3.2.1 Server Consolidation

The classic solutions to the energy efficiency problem in the cloud -if we disregard the cooling process- all stem from two major ideas: consolidating the loads on fewer servers

(hosts) or using variations of dynamic voltage and frequency scaling (DVFS). The latter includes algorithms that exploit dynamic power management in servers. Server computational power/speed can be toned down and thus energy consumption decreases. Server consolidation can be seen in early papers like [29]. The algorithm proposed in [29] executes the consolidation of different applications on cloud computing data center servers. The idea is to consolidate VMs on the least amount of servers and then switch the unused servers off or to an idle state. That problem is modeled as a bin-packing problem with the assumption that the servers are the bins and they are full when their resources reach a predefined optimal utilization level. This utilization level is calculated and set beforehand. The optimal utilization is a level where the a balance is reached between the resource utilization and performance degradation caused by pressuring the resources. The issues faced by over utilization are cache contentions, conflicts of CPU functional units, desk scheduling and desk write buffer issues and that is only from the computational side of things.

The final objective is to minimize energy consumption per transaction. Resources used are processor and disk space. The heuristic algorithm is then used to allocate workloads to servers or bins. This heuristic tries to maximize the Euclidean distance between the current allocations of the servers and the optimal utilization point of each server. There were no comparisons to the optimal solution. Also, power consumption by network components is not considered. Another issue here is that it is debatable whether finding an optimal point for each server is only based on utilization without considering other factors like the type of the application.

Another approach can be seen in [30], where methods for live migration of VMs according to the current utilization of resources are introduced. Each node has a CPU, which can be multi core, with performance defined in Millions of Instructions Per Second (MIPS). Besides that, a node is characterized by the amount of RAM and network bandwidth. The aim is to prevent service level agreement (SLA) violations which occur when a VM cannot get the requested amount of resource, which may happen due to VM consolidation. A decentralized resource allocation system is offered containing a dispatcher, global and local managers.

Only utilization of CPU is considered. “The main idea of the policies is to set upper and lower utilization thresholds and keep total utilization of CPU created by VMs sharing

the same node between these thresholds" using live migration. A challenge here is to determine values of the utilization limits (thresholds).

In [31], the authors offer a migration algorithm that depends on copying the VMs and dividing the original CPU allocation among the copied VMs while keeping the memory allocated at the same level. This would cause an additional resource consumption. Allocation is done using dynamic programming and a local search is run after to find opportunities for consolidation. This algorithm is run periodically to improve performance. The period the algorithm is run can have a great impact on the success of the method. The balance between the running cost and the running gain along with the workload change period should be investigated.

The work in [32] tackles the power management problem for minimizing the total electricity cost. This paper aims at minimizing the total electricity cost under multiple electricity markets environment while guaranteeing quality of service geared to the location diversity and time diversity of electricity price. The problem is modeled as a constrained mixed-integer programming and an efficient solution is proposed. Extensive evaluations based on real-life electricity price data for multiple cloud data center locations to illustrate the efficiency and efficacy of our approach.

The work in [33] focuses on the live migration strategy of multiple virtual machines with different resource reservation methods. A live migration framework of multiple virtual machines with resource reservation technology is introduced. Then a series of experiments to investigate the impacts of different resource reservation methods on the performance of live migration in both source machine and target machine are introduced. Additionally, the efficiency of parallel migration strategy and workload-aware migration strategy is analyzed. The metrics such as downtime, total migration time, and workload performance overheads are measured. Based on the observed results, corresponding optimization methods to improve the migration efficiency are offered.

The authors of [34] present formulations and solutions for Green Cloud Environments (GCE) to minimize its energy consumption under new models by considering static and dynamic portions of cloud components, to reduce environmental impacts. Energy consumption patterns are investigated including measurable metrics based on runtime tasks to

rationally compare the relation existing between energy consumption, cloud workload and computational tasks, as well as system performance.

Often, Cloud distributed systems leverage commodity server hardware in mass quantity, similar in theory to many of the fastest Supercomputers in existence today while also exploiting the dynamic frequency and voltage scaling techniques. The authors of [35] present the design and implementation of an efficient scheduling algorithm to allocate virtual machines in a DVFS-enabled cluster by dynamically scaling the supplied voltages. The algorithm is studied via simulation and implementation in a multi-core cluster. Test results and performance discussion justify the design and implementation of the scheduling algorithm.

3.2.2 Adaptive Allocation

Consolidation is far from a standalone solution though. For changing workload patterns, the cost of moving VMs in and out of a server in terms of performance and power could be worse than keeping them where they are. A careful consideration of the amortization period of the migration costs is required for a successful decision. The authors of [36] consider this when presenting their solution. They propose a central controller (termed Mistral) that balances steady state performance and power with the dynamic adaptation costs under changing workloads.

The authors assume workloads from multi-tiered applications are being scheduled on cloud VMs. Each application type is associated with a set of transaction types through which users access its services. Each transaction type correspond to a unique call graph of some of the application types. “The workload for each application is then defined as a vector of the mean request rates for each transaction type”. Mistral controllers are called periodically to impact the VM locations and their CPU allocations.

“Costs of these adaptation actions are measured experimentally offline for different workloads and VM placements, and are stored in tables used at runtime”. This includes adaptation duration, change in response time for the application being adapted as well as co-located applications, and change in power consumption during the adaptation. A model is introduced to predict workloads of each component and define the stability interval

following the current adaptation. No details are offered as for the structure or operation of the workload predictor. Experimental results were shown for data centers of up to just 8 servers. Also, calculating the costs offline sacrifices precision and often would not take interactions into account as it assumes that every time the decision will cost the same in terms of power and performance for example.

3.2.3 Other Efforts

We can see a detailed discussion of the network resource energy consumption in the cloud data centers in sources like [37]. VMs are assigned to servers with the objective of reducing the amount of traffic and generating favorable conditions for traffic engineering. Moreover, the number of active switches and balance traffic flows is decreased depending on the relation between power consumption and routing, to achieve energy conservation.

There are a few existing schemes that transition a CPU into various low-power and sleep states to reduce its idle power. One of the more recent efforts using this approach is in [38]. The paper offers a method to predict which CPUs will be idle based on analyzing the reading of each CPU hardware parameters. This helps the decision making process in order to achieve intelligent sleep states. This means that a more accurate prediction of the period a CPU might be idle reflects on the choice of which sleep state the CPU is moved into. the CPU performance metrics monitored are IPC, cache miss rates, structure occupancies, branch predictor statistics, and others. These readings are used as an input to an expert system based on classifiers like boosted regression trees. The output is the length of the CPU idle interval. The cost of monitoring here could be a decisive factor. Decision based CPU readings are on different level of speed to processes like running a VM on a server or switching off the server.

As one of the most detailed cloud simulators available, Greencloud arises as a powerful tool to evaluate energy efficiency in cloud environments. GreenCloud was developed as a simulator with a focus on energy efficiency and fine grained networking capabilities. The prime purpose cited for building GreenCloud is mitigating overprovision issues [3]. Overprovision happens in a data center due to the loads constantly changing on the computational and communication resources. The average load can be as low as 30% of the

data center server and network capacity [3]. This, in turn, causes the data center to systematically use more power than the optimal value. In GreenCloud, solutions implementing server consolidation and dynamic server power management are simulated with an option to expand to a hybrid solution containing both.

GreenCloud offers simulation capabilities including multiple topology choices (2 layers and 3 layers) and it offers communication through packets using the underlying NS-2 simulator features. GreenCloud also offers the choice of scheduling tasks (user requests) on hosts directly or on virtual machines which reside on hosts. Tasks are modeled as unit requests that contain resource specification in the form of computational resource requirements (MIPs, memory and storage) in addition to data exchange requirements (task size variable representing the process files to be sent to the host the task scheduled on before execution, data sent to other servers during execution and output data sent after execution).

3.2.4 Contribution

Upon reviewing the available solutions, a need arises for a more complete solution for energy efficiency in cloud data centers. As seen in Table 3.1, each one of the solutions surveyed, despite covering a significant flavor of the problem, does not offer a comprehensive vision. A mathematical model would be crucial to provide the theoretical base for the solution. To achieve an outstanding solution for such a layered problem, we offer the following contributions:

1- We first formulate the problem as mathematical optimization problem with the objective of minimizing the energy consumption.

2- A new technique called Dynamic Idleness Prediction (DIP) is introduced where the future demands for VMs are considered when placing/scheduling the VM on a host. This technique is based on using an artificial intelligence classifier (in our case REPtree) to predict the nature of the load every VM will receive in a pre-specified future period. A novel scheduling/placement technique was constructed by combining DIP and the resource based scheduling technique we introduced in an earlier publication [25]. In this technique, DIP dictates VM initial placement and VM migration in order to reach more

efficient consolidation-based energy efficiency as opposed to traditional methods like first fit, round robin or greedy methods.

3- We introduce a novel technique for energy efficient VM management that does not depend on Migration. The technique is called Smart VM Overprovision (SVOP). This technique depends on the DIP technique described earlier to dictate the overprovision of VMs by choosing the mostly idle VMS to be switched off and in turn minimize lost requests.

4- We use a data set published by of Google (data center traces [16]) to evaluate the two proposed methods. The performance of these methods is compared to common scheduling algorithms in terms of critical energy efficiency metrics including: energy used per server, energy used per served request, service rate, and number of migrations performed.

3.3 System Implementation Scenario

A typical cloud data center contains hosts that are available for clients with multiple lease terms on offer. Every client profile is tailored based on their budget, applications and general portfolio. For every rented VM, a client specifies the time scale (or at least the start time), the resource requirements in terms of computational and network resources, and operating systems in some of the cases. While the VM is alive in the data center, a continuous stream of requests arrive at the data center. Requests (tasks or Cloudlets) Tasks are modeled as unit requests that contain resource specification in the form of computational resource requirements (MIPs, memory and storage) in addition (sometimes) to data exchange requirements.

The power consumption in the data center amounts to the total power consumption of all the resources residing in it. As the computational resources -specifically the hosts- consume most of the data center power, we will focus our optimization efforts on optimizing the power consumed by hosts not the network resources like switches, routers, etc. Power calculation models used in the literature are mostly utilization based [3] [29] [36]. We will introduce the formula in detail in the following section. Therefore, workload distribution on available hosts and VMs has a prime impact on the energy efficiency. This includes, VM initial placement, migration decision management in terms of migration frequency and the

Table 3.1: A comparison of energy efficiency virtualization based efforts in cloud environments- Part1

Technique	Offer Optimization model	Scheduling considers network & computational resources	VM modeling	Application layer	network model
[9]	No	No	CPU and storage requirements	No	No
[10]	No	No	MIPs,ram,BW, VMs can be resized	No	BW
[11]	Not full	No	VM host applications of different types and up to one replica per VM	Yes	No
[12]	Yes	No	CPU & memory requirements (replicas of a VM share CPU), 1 client per VM	Yes	No
[13]	Yes	Yes	Fixed scheduling, no migration	No	Full
[14]	No	No	N/A	N/A	N/A
[15]	No	No	CPU, storage , memory, BW,communication demands	Tasks request Comp+ network resource	BW +fixed source and destination network
This Solution	Yes	No	Detailed resources;User & workload profile	Yes	No

Table 3.2: A comparison of energy efficiency virtualization based efforts in cloud environments-Part 2

Computational resources	Live migration (major method used)	Guaranteed reservation	Centralized / decentralized	Scheduling algorithm
CPU (numerical) & storage Counting by usage not absolute	No migration, Best allocation at a desired utilization level vs. degradation considered	Yes	C	Bin packing modified +maximize euclidean distance
CPU (multicore) ram	Live migration according to current utilization of resources	SLA violation	D	Multi dimensional bin packing
CPU	Yes (for replicas of Apps and for VMS)	Adaptable (affects performance)	C	Multi layer Bin packing Looking for to satisfy power-performance while minimizing the number of hosts
CPU & memory	No migration, Dynamic programming	Yes	C	Dynamic algorithm to place VMs and local search to check servers to be consolidated
No	No	Yes	C	Focused on traffic engineering (network optimization in the cloud)
CPU	No	N/A	C	No scheduling algorithm, CPU idle intervals dynamically predicted
CPU, Mem, storage	Enabled enabling DVFS Using multiple power models	Yes	C	(green-scheduling and round robin scheduling
CPU, Memory, storage, user defined resources	Migrated VM chosen via VM idleness estimator	Yes	C or D	First fit / Round Robin/ Resource Based Scheduling + Dynamics prediction

choice of the migrated VM and also the scheduling of Cloudlets (requests) on the corresponding VMs. Regardless weather the Cloudlets belong to one client or multiple clients, efficient scheduling that focuses on energy efficiency is a demand. In the following section, we offer a substantiation of this problem in the form of mathematical formulation. This is a preceding step to discussing a more scalable solution in Section 3.5.

3.4 System Model

To solve the problem of scheduling tasks in a cloud computing environment while minimizing the energy efficiency, we introduce an analytical model where we formulate the problem as a mixed integer linear problem. The optimization problem is modeled focusing on two objectives, namely, minimizing the required power to serve a specific load of tasks and minimizing the load that needs to be migrated/recovered in case of a data center component failure. This model's purpose is to demonstrate the major constraints imposed on the problem and the potential search space size.

3.4.1 Notations

Environment parameters are described below. A set of resource providers (servers or VMs) are represented by S. CLT is a set of tasks (Cloudlets) sent by cloud clients. These tasks demand specific amounts of resources to run as per the scenarios discussed in earlier sections. CAP_{sm} represents the amount of resources (e.g. memory) available on a server(resource provider) where $s \in S$ and $m \in \{memory(me), CPUunit(c), storage(st)\}$ such that $CAP_{sMem} = 30$ indicates that available memory on server s or memory capacity is 30 GB. DEM is used to represent the demand matrix or the amount of resources needed for every requested task(Cloudlet). Memory and storage requirements are measured by GB while CPU requirements are measured by the task size shown in Million instructions(MI) or million instruction per second (MIPs) and duration. They could alternatively measured by the fraction of processor power required. Moreover, the same model could be applied when using other common metrics for processing demand like (the amount of employee data processed per hour (employee/hour or Java server side operations per second(JOPs).

$DEM_{CletCPU} = 700$ indicates that the task (Cloudlet) $Clet \in CLT$ demands computational power to run 700 Million instructions, $v \in V$ requires 7 GB of memory assuming that m denotes memory resource on a server.

The matrix D contains the deadline of every Cloudlet (denoting request lifetime). $D_Clet = t$ means that Cloudlet $Clet \in CLT$ has to be served before time unit t. All the specification of a Cloudlet could in the same way applied to VMs depending on the problem in terms of if the problem is just scheduling Cloudlets on VMs(servers) or that it is scheduling both VMs on servers and Cloudlets on VMs. The parameters $Pidle_s$ and $Pmax_s$ indicate the amount of power consumed by server s at the idle state and at maximum utilization respectively.

3.4.2 Decision Variables

Y_{sClet} is a binary decision variable such that $Y_{sClet} = 1$ if Cloudlet $Clet \in CLT$ is scheduled on server(resource provider) s and 0 otherwise. X_{tClet} is a binary decision variable such that $X_{tClet} = 1$ if Cloudlet $Clet \in CLT$ is served at time unit t and 0 otherwise.

3.4.3 Objective Function

The problem is formulated as a mixed integer linear programming (MILP) problem with two possible objectives. The first one is to minimize power consumption needed to perform a specific load (minimize the kilo watt hours required to run a specific set of tasks(requests)). The second objective is more high availability-oriented. The goal is to minimize the potential migration load in case of a failure to any server (or by extension to any component in the data center hierarchy). Scheduling the tasks in a way that minimizes the migration load affects the performance positively as it decreases the performance hiccup induced by failures and aids in reaching a more seamless failure event handling. This decreases the amount of resources dedicated to maintain high availability as the work load is less.

3.4.3.1 Power Consumption

Formulating power consumption in severs is a standing challenge in the literature. Some of the commonly used efforts can be seen in [3] [29] where linear models are proposed. These models are based on the assumption that servers consume minimal power when idle and that level of consumption increases linearly while the computational capacity is increases. Thus, power consumption at a specific processor utilization percentage *u* is calculated as:

$$P_u = P_{idle} + (P_{max} - P_{idle}) \times u \tag{3.1}$$

Hence, power consumption can be minimized by directly minimizing the average utilization.

$$MIN \qquad Z1 \tag{3.2}$$

$$\begin{aligned} Z1 = &\textstyle\sum_{t \in T} \sum_{s \in S} (Pidle_s + (Pmax_s - Pidle_s) \times \\ &(\textstyle\sum_{Clet \in CLT} X_{tClet} Y_{sClet} DEM_{CletCpu}) \div CAP_{sCPU})) \end{aligned} \tag{3.3}$$

3.4.3.2 Migration Load

It is desirable to minimize the load to be migrated in case of component failures. That translates to the data load on the server at the moment of failure. To minimize that we use a min max approach where we try to minimize the maximum computational load on any of the servers in the data centers. This is calculated based on the computational capacity of Cloudlets (tasks) scheduled on servers. However, it can be easily adjusted to accommodate other components (VMs, racks, etc) or to take into consideration other resources when calculating the load.(like memory, storage, bandwidth).

$$MINMAX \quad Z2 \tag{3.4}$$

$$\begin{aligned} &Z2 = \textstyle\sum_{Clet \in CLT} X_{tClet} Y_{sClet} DEM_{CletCpu} \\ &\forall s \in S \; \forall t \in T \end{aligned} \tag{3.5}$$

This model assumes an equal Mean Time To Fail(MTTF) for all servers in the data center. A weighted function based on the MTTF would be added in case of using varying values for servers.

3.4.4 Constraints

The objective function is subjected to the following constraints:

$$\sum_{t \in T} X_{tClet} >= DEM_{CletCpu}/Cap_{CletCpu}, \quad \forall Clet \in CLT \tag{3.6}$$

$$\sum_{s \in S} Y_{sClet} = 1, \quad \forall Clet \in CLT \tag{3.7}$$

$$\sum_{Clet \in CLT} X_{tClet} Y_{sClet} DEM_{Cletm} <= CAP_{sm}, \quad \forall t \in T, s \in S, m \in \{me, c, st\} \tag{3.8}$$

$$X_{tClet}.t <= D_{Clet} \quad \forall Clet \in CLT, \forall t \in T \tag{3.9}$$

$$X_{tClet}, Y_{sClet}, \in \{0, 1\} \tag{3.10}$$

In Constraint 3.7 , we ensure that a Cloudlet (request) will be assigned exactly to one server(service provider). In Constraint 3.6, we ensure that a Cloudlet is scheduled for enough time units to satisfy its computational demand given the server computational capacity. In Constraint 3.8, we guarantee that Cloudlets will be allocated on servers with enough capacity of the computational resources required by the Cloudlets. In Constraint 3.9, we ensure that a Cloudlet is served before its deadline. Constraint 3.10 guarantees the binary constraints of the problems.

3.5 Consolidation-based energy efficiency

Moving to more practical solutions, we start by proposing a consolidation-based solution. Then, we move to discussing the impacts of consolidation-based solutions and live migra-

tion leading to the proposal of a novel non consolidation-based solution.

3.5.1 VM Dynamic Idleness Prediction (DIP)

The challenge of choosing which VM to move or migrate is central to any consolidation technique and therefore crucial to energy efficiency policies. a core activity of consolidation-based energy efficiency is migrating VMs in order to empty a machine so it can be switched off or moved to an idle state. This decision affects the performance highly, first by specifying the amount of data to be moved and the nature/amount of resources required at the destination host. In addition, regardless of how much algorithms are able to minimize the live migration time, there will always be a certain amount of delay or performance degradation. This means that depending on the load expected of the VM and SLA agreement, an SLA violation is highly possible. This way, the priority should be given to VMs with more strict SLAs, and critically, with less activity when the migrated VMs are chosen. We propose to tackle this challenge by introducing a scoring system for the VMs to decide which ones to shut down and move based on the use of an expert system. The hypothesis here is that with the successful prediction of which VMs will be idle (or least active) and for how long, we will have a clear advantage in terms of migration decision management and the consolidation process in general. This step, which we termed dynamic idleness prediction (DIP) will make an instant impact in terms of the total power consumed by the data center with all the other factors unchanged.

3.5.2 Classification Parameters

The classification parameters are the parameters used by the system to predict the state of the VM for a preset future period. They include variables that would affect the system's expectation of the VM future behavior. Some of these variables would affect the VM activity directly (for example: redundancy models specifies the frequency of backup/redundancy activities). Some affect the VM indirectly and contribute to behavior patterns than are not specified explicitly (for example: User location or the type of application served by the VM). The more parameter values that can be collected for the VM, the more reflective the

profile built by the classifier will be. In turn, the results will be more reflective of the VM activity level. Examples of parameters that can be used to build a profile for the VM are:

- User ID
- User location
- User VMs
- Type of contract (rental term)
- VM reserved resources
- VM start time/reserved time
- Redundancy model
- Redundancy activity frequency
- Component type
- Request types and frequency
- Communication/data exchange request
- Dependencies
- Response time required

3.5.3 How DIP Works

First, the classifier is fed a list of records containing the parameter readings or values for the VMs and their resulting states for a certain time period. This would be the training data set Then the classifier uses this training data set to build behavioral models for the VMs in question. Alternatively, cross validation method can be used. These behavioral models would depend on the classification method used (for example: decision trees, Naive Bayes or Support vector machines). From now on, the Classifier would be able to predict (classify) the number of requests sent to the VMs in a fixed future period. Next, this information can be used by the scheduling component (centralized to for the whole data center or decentralized) to rank the VMs and either:

A- Choose the VM with predicted least received requests in time period t2-t1

B- Set a cutoff threshold (CO) such that VMs with incoming future request in a pre-specified period less than CO are considered idle.

Once the idle VMs list is generated, the list is used in the consolidation step as explained in the algorithm in Fig. 3.2.

3.5.4 The Proposed Consolidation-Based Algorithm

The proposed method is illustrated in the flowchart in Fig. 3.2 First, the algorithm starts by initializing the major parameters including parameters in Table 3.3. VMs are placed based on one of the following 3 methods: first fit scheduling, round robin or the resource based scheduling technique with the variation proposed in [25]. Then, as the requests for resources keep arriving, these request are scheduled based on the availability of host and VM resources. Periodically, and based on a preset consolidation trigger parameter, the VM shuffle process starts. This trigger defines the period or the frequency of revisiting the VM placement and running the shuffle process and the consolidation function. The shuffle process starts by constructing a list of VMs to be switched off. These VMs are chosen based on VM choice method parameter (or in other words, idle VMlist construction technique). We chose 4 options to test in our experiment. These options are choosing the VMs randomly, choosing the VMs based on a greedy method that chooses VMs hosted on the least used server(s), exploiting the proposed technique DIP as explained in the previous subsection and finally, we set the fourth option to be keeping the VMs running without any switching off. This would aid us in calculating how much of an improvement these techniques are offering as opposed to not using any technique. Next, after constructing the Idle VMlist, these VMs are switched off and the data center loads are consolidated into fewer servers. This shuffle process is repeated on a periodical basis depending on the trigger parameter mentioned earlier. Another step that is done periodically is swapping the VMs that are switched off in order not to starve a certain VM and to ensure fairness. In the following section, we propose a solution based a technique using DIP that is not dependent of live migration. Then, we present the experimentation results for both migration-based and non-migration based techniques. Fig. 3.3 shows the pseudo code for the Idle VMlist construction function.

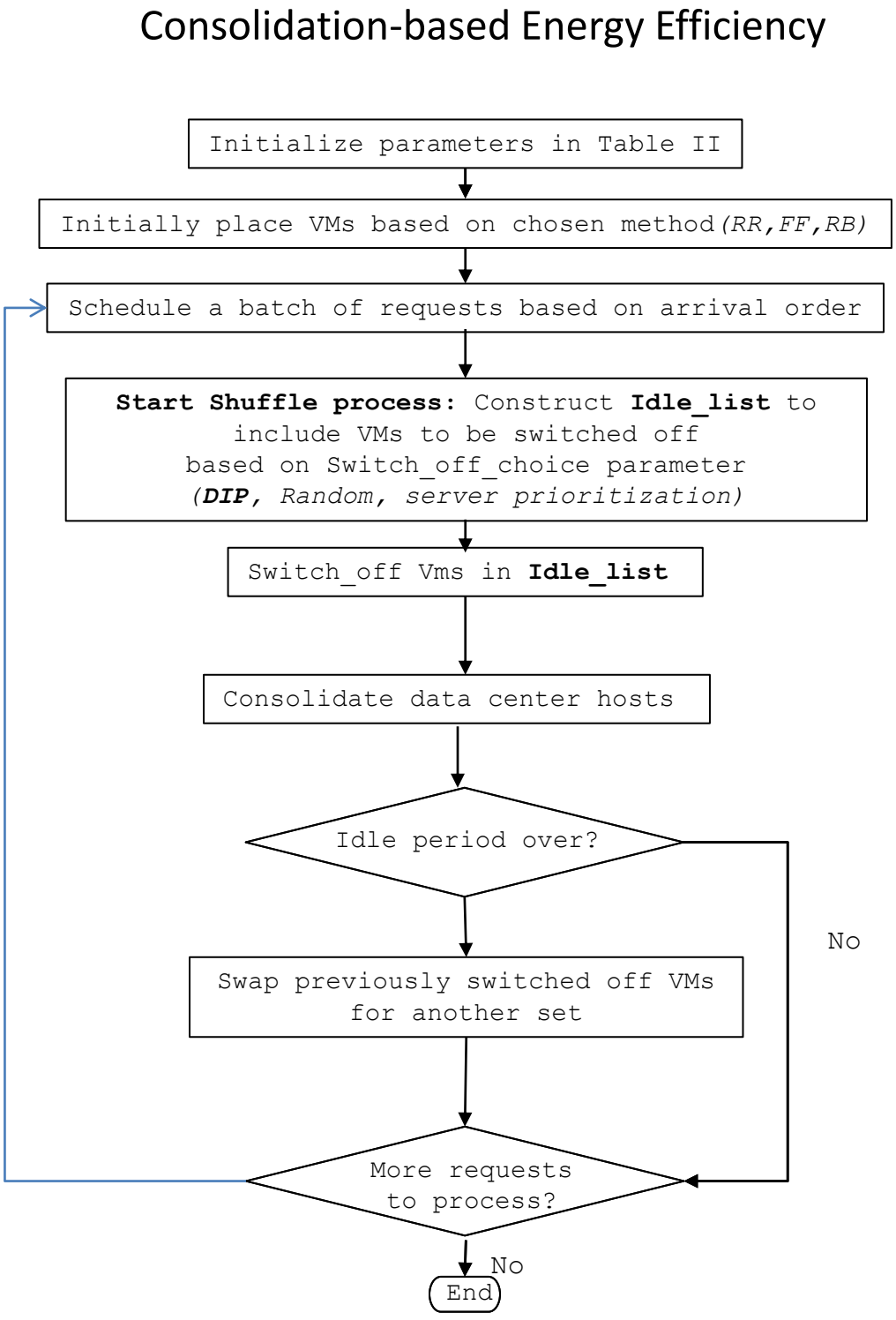

Figure 3.2: Consolidation-based energy efficiency flowchart

Table 3.3: Energy Efficiency Problem - Major Parameters

Parameter	Description
Placement Method	How are VMs initially scheduled on hosts round robin, First fit or Resource based scheduling proposed [25]
Switched off(consolidated) VM choice method (Idle VMlist construction technique)	no switch off, DIP, random, greedy-servers
cutoff limit (CO)	the amount of requests in the future below which the VM is considered idle
Consolidation Trigger	the frequency of revisiting the VM placement and running consolidation function
Migration allowed	if migration is used for this experiment or fixed VM placement is imposed

Function: $constructIdleList()$
Input: $Virtual\ machine\ set\ VM$,
$where\ VM_i.state\ is\ either\ 'idle'\ or\ 'running'$,
$getClassifierPredictedValue(v, FP)\ retrieves$
$the\ number\ of\ incoming\ requests\ for$
$VM\ v\ for\ a\ pre\ specified\ future$
$period\ (FP)\ as\ predicted\ by\ the\ classifier$,
$CO\ is\ the\ cutoff\ limit\ to\ consider\ a\ VM\ idle$
Output: $VM\ state\ values\ assigned\ correctly$
for $VM_i \in VM$ **do**
  if ($getClassifierPredictedValue(v, FP) <= CO$) **then**
    $VM_i.PredictedReqs = idle$
  end if
end for

Figure 3.3: Idle VMlist construction function using Dynamic Idleness Prediction technique (DIP)

3.6 Non-Consolidation-based energy efficiency

3.6.1 Live Migration: Why Not?

Taking migration as automatic solution is far from agreed upon. Migration typically imposes performance degradation to the extent of having an off time which is not welcome by the clients. This time could span through an unexpected range based on the efficiency of the process and the network bottlenecks in the data center at the time and the amount data included. VMware, for example, offers live migration as major feature introduced in vMotion where a VM can be moved from a host to another without having to shut it down. Larger providers like Amazon, for example , do not depend on this. Reasons, cited in, [39], include the fact that AWS (and Rackspace as well) keep the VM data in the local disk. This makes it harder to send all the data across the network. "Evacuating a given host, particularly one at capacity can take hours". This casts doubts over the practicality of using live migration in principle. More so, it casts doubts on using migration with the freedom and frequency suggested in some of the energy efficiency solutions, where VMs are to be consolidated periodically in fewer servers. Finding a solution that does not depend on live migration or at least minimizing the number of migrations performed is a pressing requirement.

3.6.2 Smart VM Overprovision(SVOP)

We introduce a novel technique for energy efficient VM management that does not depend on migration. The technique is called Smart VM Overprovision (SVOP). This technique depends on the DIP technique described earlier to dictate the overprovision of VMs by choosing the mostly idle VMS to be switched off and in turn minimize lost requests. SVOP is illustrated in the flowchart Fig. 3.4 This method works in two phases. First, the initial VM profile building phase. Then, the regular VM operation phase. The first phase starts by the initializing the parameters then scheduling the VMs on the corresponding host based on RB scheduling explained the previous section. Next, a test batch of requests is scheduled to build each VM's profile. These requests can be real requests demanded by the VM or client. They can alternatively be a training set of request constructed based on the client

and VM profile in order to be used for the following steps. After that, the classifier builds a profile for the VM and a predicted value of the future demand is calculated. Then the overbook-shuffle process is started. An idle VMlist is constructed using the DIP technique discussed previously. The list of idle VMs are separated from the active VMs. They are scheduled on a separate set of hosts where the concept of overbooking is applied. An overbooking factor is used to define the overbooked load on each of these overbooked hosts. For example, if the host's capacity is 4 VMs and the overbooking factor is 150%, then the Overbook-idle-list function will book up to 6 VMs on this host(assuming all VMs have requested equal amounts of resources in that case). From here, these VMs can alternatively share the overbooked resources (which is the concept we used in our implementation). Another technique that can alternatively be used here is dividing resources between the VMs in a way that each one would have reduced capacity. In the second phase of the SVOP technique, the operation phase, the incoming requests are served based on the setup in the first phase. Another step that is done periodically is swapping the VMs that are switched off in order not to starve a certain VM and to ensure fairness. Fig. 3.5 contains the pseudo code for Idle VMlist overbooking function included in the SVOP method.

3.7 Experimental Setup

3.7.1 Data Set

To perform the experiment, we used a data set taken from Google's cluster workload traces. These are traces of workloads running on Google compute cells. The dataset provides traces from a Borg cell that were taken over a 7 hour period. The workload consists of a set of tasks, where each task runs on a single machine. Tasks consume memory and one or more cores (in fractional units). Each task belongs to a single parent; a parent may have multiple tasks (e.g., mappers and reducers). In our work, the parent is represented by the VM the task belongs to. "The data have been anonymized in several ways: there are no task or job names, just numeric identifiers; timestamps are relative to the start of data collection; the consumption of CPU and memory is obscured using a linear transformation. The data

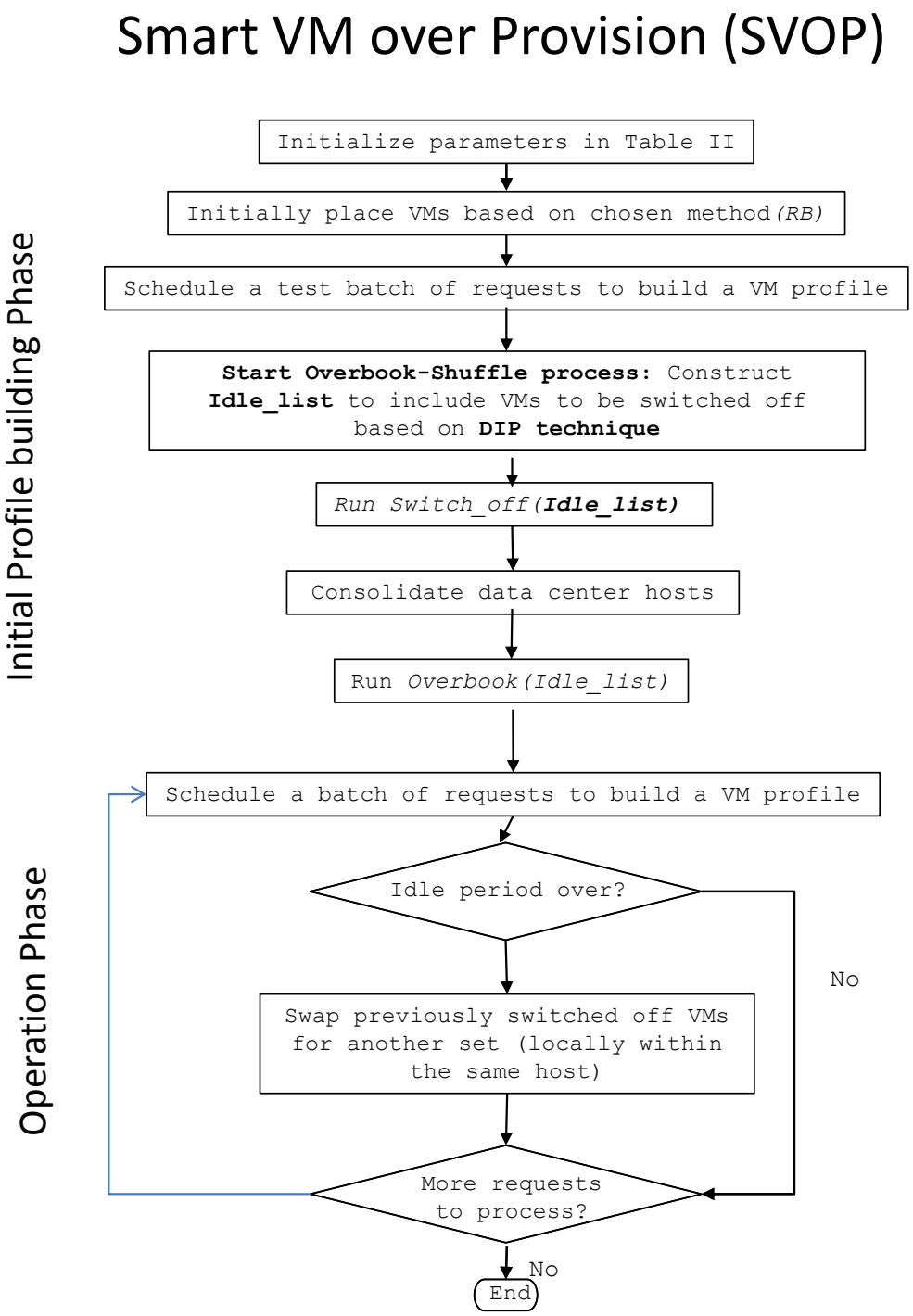

Figure 3.4: Consolidation-based Energy efficiency flowchart

2: **Function:** $overbookIdleList()$
3: **Input:** $Virtual\ machine\ set\ VM,$
4: $where\ VM_i.state\ is\ either\ 'idle'\ or\ 'running'$
5: $with\ states\ assigned\ based\ on$
6: $Construct_idle_list function()$
7: $(OBF)\ as\ overbooking\ factor$
8: **Output:**$VMs\ with\ idle\ state\ are$
9: $scheduled\ using\ SVOP\ method$
10: $hostS = leastUsedServer()$
11: **for** $VM_i \in VM$ **do**
12: **if** ($VM_i.state = idle$) **then**
13: $ScheduleVM(i, S, OBF)$
14: **if** ($isHostfull(S, OBF)$ **then**
15: $S = leastUsedServer()$
16: **end if**
17: **end if**
18: **end for**

Figure 3.5: Idle VMlist overbooking function

are structured as blank-separated columns. Each row reports on the execution of a single task during a five minute period.

- Time (int) - time in seconds since the start of data collection
- parentID (int) - Unique identifier of the job to which this task belongs (may be called ParentID)
- TaskID (int) - Unique identifier of the executing task
- Type (0, 1, 2, 3) - class of job (a categorization of work)
- Normalized Task Cores (float) - normalized value of the average number of cores used by the task
- Normalized Task Memory (float) - normalized value of the average memory consumed by the task Using classifiers

3.7.2 Classifier and Classification Tool

Machine learning (ML) classifiers automatically analyze a large data set composed of several attributes and decide what information is most relevant. This builds the classifier's ability to predict the values of a specific preselected attribute. This value (which could be

Table 3.4: Classifier Prediction Precision Comparison

Classifier(as named in Weka)	Relative Absolute Error
Decision Table	18.6%
MSRules	19.0%
Conjunctive Rule	55.0%
Gaussian Processes	52.0%
Multi Layer perception	34.9%
IBK	17.3%
KStar	11.9%
LWL	44.8%
meta bagging	10.0%
Random sub space	16.2%
Regression by discretization	13.0%
MSP tree	17.2%
REPtree	7.8%

qualitative or quantitative) is the classification. Classifiers are used in many application fields. A commonly used tool that has a variety of the most common classifiers readily implemented is Weka [40].

The tool has "incorporated several standard ML techniques into a software "work-bench" called Weka"[40]. Once the data is formatted in the format readable by Weka (.arff format) which defines what is the relation name, the attributes and their possible values and the data rows themselves, the tool can pre-process and classify. The relation defined for this work to predict the number of future requests is VM-predictor. We have tested multiple classifiers to find the classifier most suitable to our DIP technique and the energy efficiency problem. Table 3.4 contains the classifier names as in Weka and the classification precision measured using root absolute error. It is seen from the table that classifiers differ in their achieved precision. The highest performing classifiers for this specific case is REP-tree with a root absolute error of 7.8% and then meta bagging and KStar classifiers. Fig. 3.6 show a sample fo the visual results gained for individual prediction using the REPtree classifier. Most of the values lie in or around the line which has a slope of 1. This indicates the equality of the predicted and the actual values of the number of future requests.

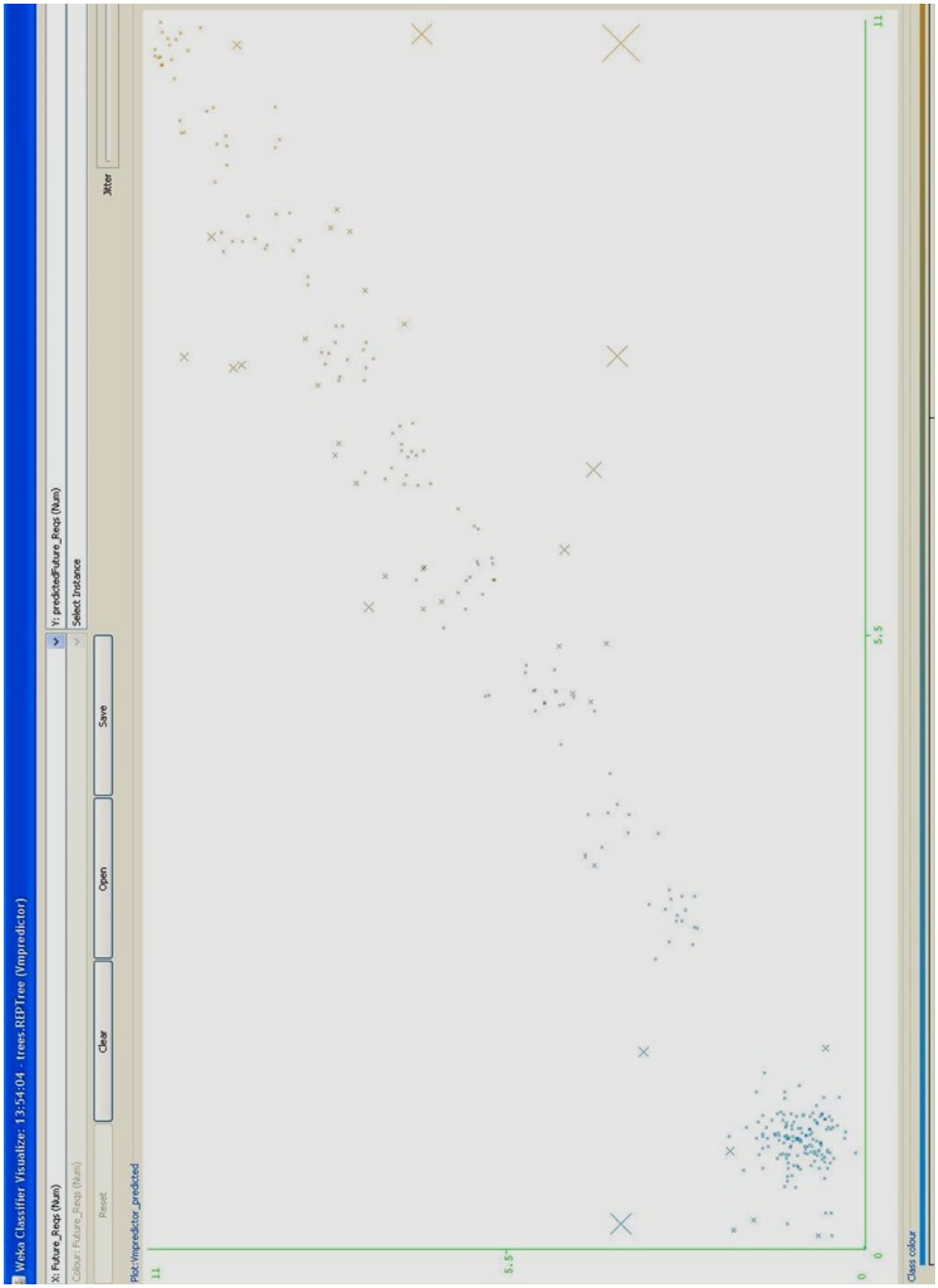

Figure 3.6: VM prediction results using REPtree classifier (the number of received requests on X-Axis vs. the number predicted requests on Y-axis)

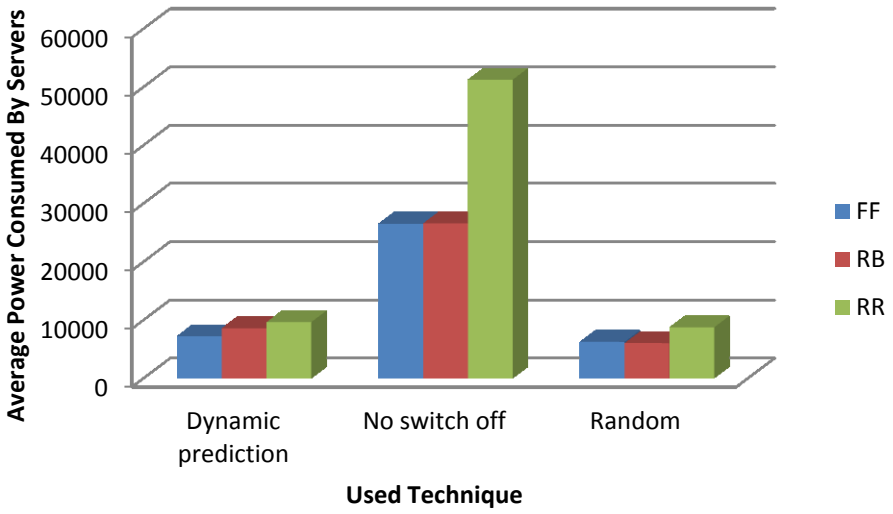

Figure 3.7: Comparing request acceptance rate for different placement methods

3.7.3 Simulation Parameters

A discrete event simulator was built in C++ to evaluate and compare the aforementioned techniques. As for the VM resource specification, we based it on some of the offered VMs by Amazon AWS.[1]. The simulated time reached 6500 time units. The power calculation model used is a linear power consumption model as in [3]. This could easily be swapped with any other model as per the cloud provider's preference. Each request was considered a fixed-duration request for the corresponding memory and CPU values that are taken from the Google data trace. This helps in eliminating any distortion caused by the request duration distribution and increasing the dominance of the evaluated parameters over the results. More details on the specifications of each of the evaluated techniques in the following section.

3.8 Result Analysis

3.8.1 Comparing Placement Methods

We start by a comparison of the placement methods used in the VM initial placement step. A look at Fig. 3.7 shows the major advantage each of FF and RB methods has over RR when either of these methods are combined with any of the idle list construction (switch off factor) methods. This is due to the fact round robin mainly focuses on distributing the load on as many hosts as possible. This means starting many "unnecessary" hosts and while this method has advantages in terms of high availability and minimizing network bottlenecks, it is not really suitable for energy efficiency purposes. Moreover, the other two techniques perform comparatively mainly because of their tendency to fill the hosts before looking at using new ones. This happens in a greedy way (FF) or in resource oriented way (RB).

3.8.2 Evaluated Methods (Energy Efficiency Solutions)

Next, we evaluate the aforementioned solutions in terms of a number of critical metrics, namely, energy used per server, energy used per served request, request acceptance rate, and number of migrations performed. It can be seen that the multiple factors considered in this problem and the possible methods employed can yield a high number of solution permutations. Due to space constraints, we will show the results for the 9 methods with high performing or significant results for any of these metrics. Table 3.5 Explains each method in terms of nature and the techniques used in it. The table specifies if the method is consolidation-based or not (depends on migration or not), which placement method is used for the initial placement, how the idle VM list (mentioned in flowcharts 1 and 2) is constructed, if the VM switch off act is permanent (until the end of the experiment) or temporary and interchangeable between VMs as explained in the previous section and finally, it discusses the frequency the VM consolidation technique is called whenever it is used. The last factor was added to show the effect of increasing the frequency of calling the VM consolidation method on the evaluated metrics. From this point on, we will refer to each of the evaluated methods with the abbreviated name used in Table 3.5.

Table 3.5: Evaluated methods (Energy Efficiency Solutions)

Method	Consolidation-Based?	Placement Method	IdleList construction technique	Switch off duration	VM Consolidation frequency
FF-DIP-PermSW	Yes	FF	DIP	Perm	Normal
RB-DIP-TmpSW	Yes	RB	DIP	Temp	Normal
RB-DIP-TmpSW-Hfreq	Yes	RB	DIP	Temp	High Frequency
RB-Greedy-PermSW	Yes	RB	Server greedy	Perm	Normal
RB-Greedy-TmpSW	Yes	RB	Server greedy	Temp	Normal
RB-Rnd	Yes	RB	Random	Temp	Normal
RR-NoSW	No	RR	No Switch off	-	Normal
FF-NoSW	No	FF	No Switch off	-	Normal
SVOP	No	RB	DIP	Temp	Normal

3.8.3 Consolidation Frequency

Consolidation frequency defines how often the consolidation function is called to look for space on the hosts to be saved. The effect of increasing the consolidation frequency can be seen by comparing the metric readings for RB-DIP-TmpSW and RB-DIP-TmpSW-Hfreq in figures 3.8 to 3.12. It can be inferred that increasing the consolidation increases the acceptance rate significantly. However, this increase is paid in the form of system load. RB-DIP-TmpSW-Hfreq scores highest in terms of the number of migration per VM (in Fig. 3.11) and in terms of the total number of servers used for a fixed load (in Fig. 3.12). A balanced level of frequency needs to be reached for each specific case to reach a trade off between the load caused by the high number of migrations might cause the gain in accepted requests caused by updating the system to reflect the momentarily loads states of the VMs.

3.8.4 Permanent Vs. Temporary Switch Off

As seen in figures 3.9 and 3.10, using permanent switch off for inactive VMs will yield significant power savings. This applies regardless of the Idle list construction technique. In Fig. 3.10 for example, FF-DIP-PermSW consumes 7251.6 compared to the 18811 consumed by RB-DIP-TmpSW while RB-Greedy-PermSW consumes 21672 compared to the 26616 consumed by RB-Greedy-TmpSW. However, the acceptance rate losses caused by the permanent switch off of idle VMs are very high. As in Fig. 3.8, both FF-DIP-PermSW and RB-Greedy-PremSW gain largely discounted acceptance rates (58.35% and 80.80%) compared to their counter part methods (79.39 % and 100%). This confirms the notion that using permanent switch off even for the most idle VMs is not effective in terms of scheduling fairness and general acceptance rate. Therefore, using permanent switch of should be saved only for cases where the data center cannot serve the load for all the VMs requested at a certain moment.

3.8.5 DIP technique's Impact on the Consolidation based Techniques

Looking at DIP's impact when it is introduced as the technique of choice to construct the Idle List during any consolidation based technique, it is found that this impact is significant.

In Fig. 3.10, we notice that FF-DIP-PermSW and RB-DIP-TmpSW have a clear advantage in terms of power consumed per server specially compared to the other consolidation-based techniques. This is supported by an advantage in terms of power consumed per request where these two methods ranked 1 and 2 again. RB-DIP-TmpSW specifically performs favorably in terms of energy efficiency and acceptance rate. (81% of request). However, when looking at the number of migrations per VM, we notice that this technique requires a relatively high number. In the cases where migration is not a preferred option, there is a critical need for another method which performs comparatively to RB-DIP-TmpSW and that does not depend on migrations.

3.8.6 Smart VM Over Provision(SVOP) as a Method that is Not Dependent on Migration

Two methods which serve as a benchmark for our solution are the No switch off methods (RR-NoSW and FF-NoSW). In these two methods, the initial placement of the VMs is the only step performed. All VMs are given high priority for the resource allocation. No Vm is switched off or migrated. Naturally, this means that most or even all requests are accepted. However, the energy efficiency is far from optimal. Also, the initial placement method is the dominant factor that affects the method performance. A look at Figures 3.9 and 3.10 shows that RR-NoSW method has the highest value for power consumed per request and power consumed per server metrics and by a distance. Fig. 3.12 (as discussed earlier) shows that the same method used a higher percentage of the data center servers even than some of the methods that use migration. This leaves as with FF-NoSW. When comparing our proposed method SVOP with the best performing non-consolidation-based method (which is FF-NoSW), encouraging results are seen. Although SVOP does not quite reach 100% acceptance rate, SVOP consumes lower power per server than FF-NoSW and comes third for that metric only after FF-DIP-Perm and RB-DIP-TmpSW. Both of those methods are consolidation-based and both achieved lower acceptance rates than SVOP. As for the power consumed per request metric, SVOP comes in third and consumes lower power than all methods with 80% acceptance rate or higher. SVOP consumed power per request is close to the value achieved by the best consolidation-based technique RB-DIP-

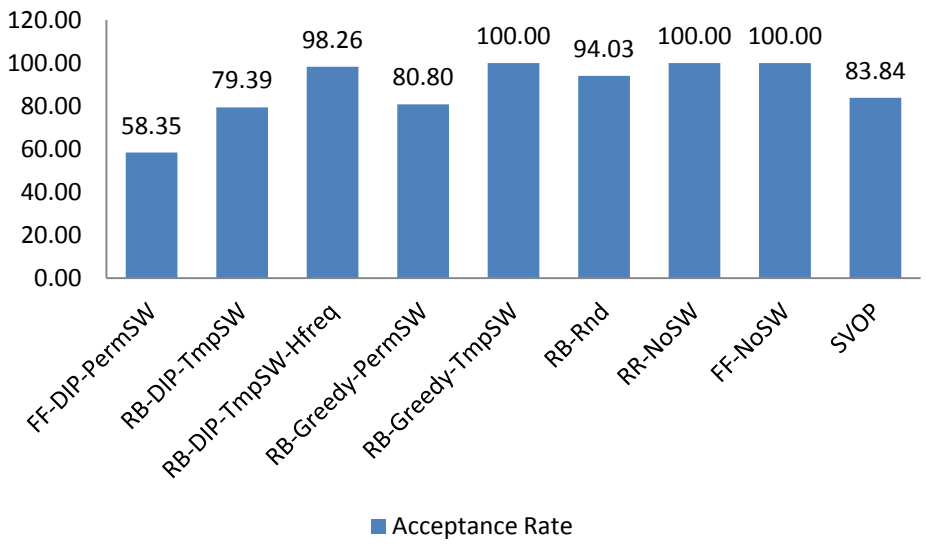

Figure 3.8: Request acceptance rate for different energy efficiency methods

TmpSW. SVOP also offers the advantage of not needing any migrations in the operation phase and using considerably lower number of servers on average than RB-DIP-TmpSW.

Therefore, from combining the previous results, the consolidation-based method RB-DIP-TmpSW (which depends on the proposed DIP technique) is the best performing method in terms of energy efficiency with a viable acceptance rate. However, the proposed non consolidation-based method SVOP comes very close in terms of energy efficiency while offering the added advantage of less/no migration load.

3.9 Chapter Summary

Energy efficiency in cloud data centers is one of the more pressing issues cloud providers are faced by. Cloud clients require certain levels of performance in aspects like high availability, request acceptance rate and deployment options. To satisfy those demands, cloud providers are in constant pursuit of a system that satisfies client demands for resources, maximizes availability and other service level agreement metrics while minimizing energy consumption and, in turn, minimizing cloud providers' cost.

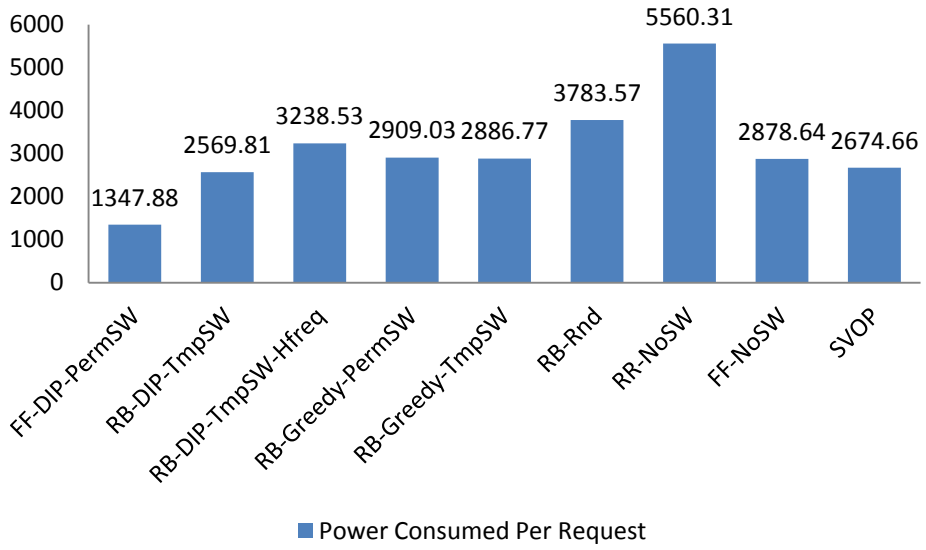

Figure 3.9: Power consumed per request for different energy efficiency methods

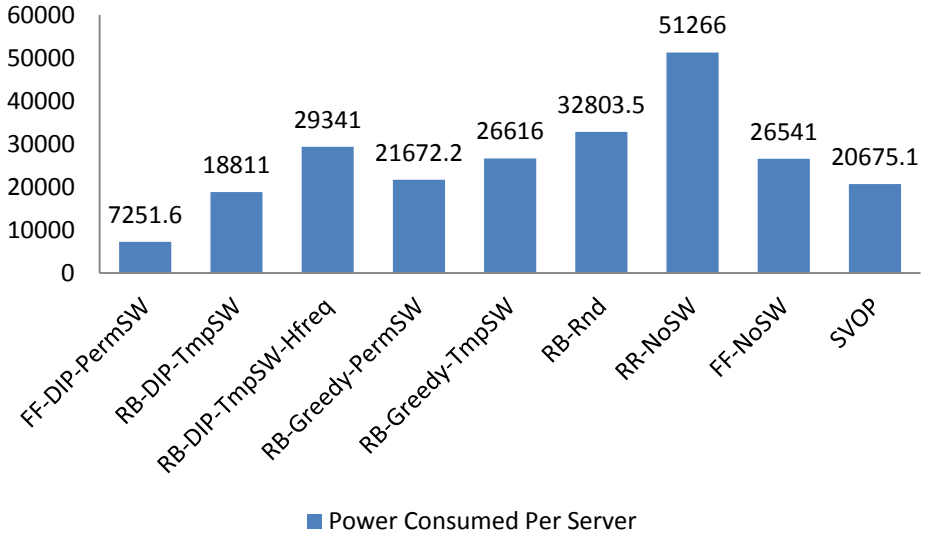

Figure 3.10: Power consumed per for different energy efficiency methods

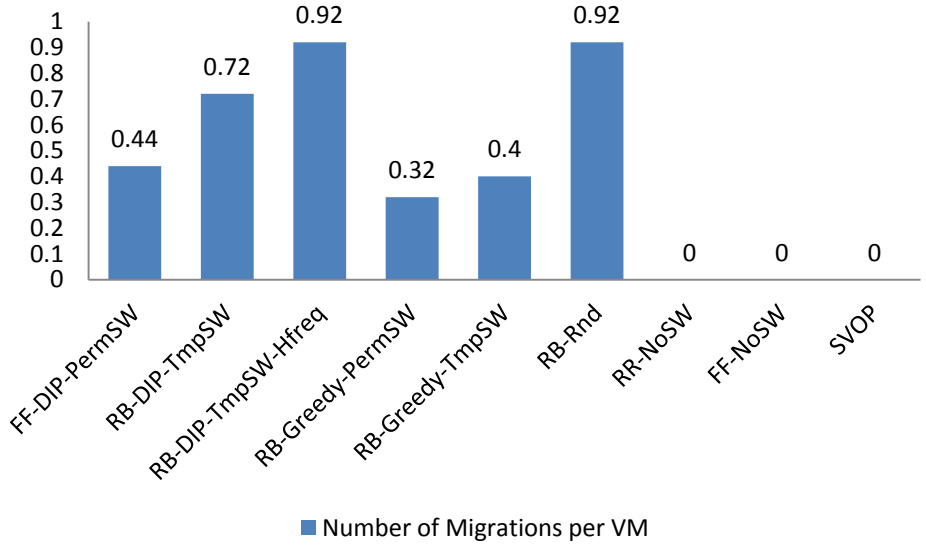

Figure 3.11: Number of migrations per VM for different energy efficiency methods

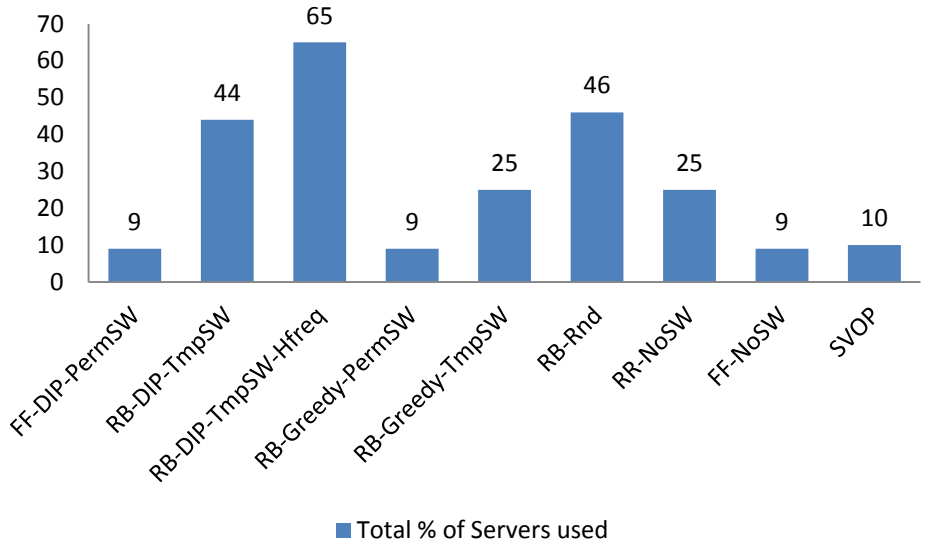

Figure 3.12: Percentage of used servers for different energy efficiency methods

We introduced a novel mathematical optimization model to solve the problem of energy efficiency in a cloud data center. Next, We offered a solution based on VM migration that tackles this problem and minimizes energy efficiency in comparison to other common solutions. This solution includes a novel proposed technique to be integrated in any consolidation-based energy efficiency solution. This technique depends on dynamic idleness prediction (DIP) using machine learning classifiers. Potential classifiers were evaluated and a recommendation with regards to the most suitable classifiers was made. Moreover, a robust and efficient energy efficiency scheduling solution that does not depend on VM consolidation or live migration. This method, termed Smart VM Over Provision(SVOP), offers a major advantage to cloud providers in the cases where live migration of VMs is not preferred. 9 candidate solutions with multiple energy efficiency techniques were evaluated for a number of critical metrics, namely, energy used per server, energy used per served request, acceptance rate, and number of migrations performed.

The experimental results gained from testing these methods on data taken from the Google trace data set showed that the consolidation-based method RB-DIP-TmpSW (which depends on the proposed DIP technique)was the best performing method in terms of energy efficiency with a viable acceptance rate. However, the proposed non consolidation-based method SVOP came very close in terms of energy efficiency while offering the added advantage of less/no migration load.

Chapter 4
Building a Cloud on Earth: A Study of Cloud Computing Data Center Simulators

4.1 Introduction

As the cloud computing client base grows, the cloud service providers face the challenge of adapting to the needs of this growth in both the technical and business dimensions. Cloud providers should maintain this gradual enhancement without losing focus on delivering the level of service their clients demand. The dynamic and unpredictable nature of cloud computing adds extra layers of complexity to the providers' tasks. This challenge is magnified by the rise of Big Data concepts that are pushing a new wave of solutions and use case scenarios. Newly developed cloud solutions should be able to handle the scale client data have reached. It is expected of the available infrastructure and software stack to serve the 2.5 Quintillion Bytes (2.3 trillion Gigabytes) that are created every day [19]. IBM estimates that there will be 18.9 billion network connections by 2016. This covers all types of connections especially the ones originated at smartphones carried by any one of the 6 billion expected carriers all over the world.

In [43], the author presents an example of this change."With machine to machine (M2M), though, this paradigm changes. Systems generate vast amounts of data independent of human business processes –collecting, analyzing and then deploying this informa-

---

The contents of this chapter have been published in [41] and [42]

This work is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC-STPGP 447230) and Ericsson Research.

tion for use represents a net-new activity for most organizations, in both IT and business operations."

When you add to this mix the number of technical/performance related parameters involved in forming a cloud solution, the task looks increasingly challenging. Providers need efficient tools to support solution design decisions related to deployment models, resource allocation, scheduling, and performance adjustments. Evaluating solutions directly and from the beginning of the solution development process using the real infrastructure is not always practical due to cost factors. The idea of benchmarking using a subset of the infrastructure (like a set of servers with the same configuration and the same topology of the data center for example) would not guarantee a wholesome vision of scalability issues. Scalability is a key player in this scenario and it (along with the cost) constitutes a challenge when using real testbeds. Solution evaluation using analytical methods is rendered infeasible due to the increasing complexity as the scale of the problem grows. Simulation arises here as a tool that - while is not enough alone to handle the whole cloud solution evaluation process- can rather play major roles before a solution is deployed on real hardware. These roles can be considerably less expensive and with less risk. Cloud simulation involves modeling a real or a proposed cloud system using computer software. It is notably useful when changes to the actual system are either difficult to implement, involve high costs, or are impractical.

Several attempts have been made to develop a competitive cloud simulator. Each cloud simulator differs in vision, the focal points and the resulting features. In this work, we examine the major cloud simulators available to researchers and industry engineers and compare them in terms of the main simulated components, application model, network model, and architecture. Previous attempts to survey cloud simulators can be seen in [44], [45], and [46]. In this work, we strive to offer an updated and more comprehensive view of this topic. We present the limitations found in each simulator using an approach that depicts what it is and what it is not. We also aim at illustrating the ground on which the current simulators stand. This helps us to construct a framework for the cloud simulator design process which would ideally cover the industry and research community needs.

Moreover, a deep analysis of the open research challenges related to this topic is offered. Challenges covered include realistic user application patterns, cloud deployment

and pricing, reliability and high availability, challenges originated outside the data center and Big data considerations.

The coming sections are divided as follows:

Section 4.2 details the specific roles expected of a cloud simulator. Section 4.3 is a discussion of the design decisions included in the process of developing a cloud simulator in terms of visions for hardware, applications and network sides. Next, Section 4.4 traverses a chosen set of common cloud simulators, giving significant attention to a few simulators that contain interesting implementation concepts. This simulator comparison is then consolidated in Section 4.5. We then proceed to discuss pros and cons of building a new simulator as opposed to extending a current one. Finally, an illustration of some of the research challenges and future work is presented in Section 4.6 and the chapter is concluded.

4.2 The Roles of a Cloud Simulator

To better understand expectations of the stakeholders, a clear perspective of the cloud simulator roles needs to be defined. This can be done by depicting the potential cloud service planning activities a simulator can serve in a cloud environment. Each activity can be matched with a use case where using a cloud simulator can achieve the required impact. These activities can be summarized in the following [2, 9, 47, 48, 49, 50]:

1. **Define**: Develop greater understanding of process details. Service deployment options, green data center policies and high availability policies are examples of aspects that are affected by several conditions that work simultaneously. Understanding their interaction on the lowest level is a must if efficient resource management is to be achieved.

2. **Pinpoint**: Identify problem areas or bottlenecks in a process that affect execution speed or increase the solution cost.

3. **Maneuver**: Test different "What if?" scenarios to better predict how a real life problem evolves under specific conditions.

4. **Analyze**: Evaluate the effect of system or process changes such as demand, supply, resources specifications, and constraints

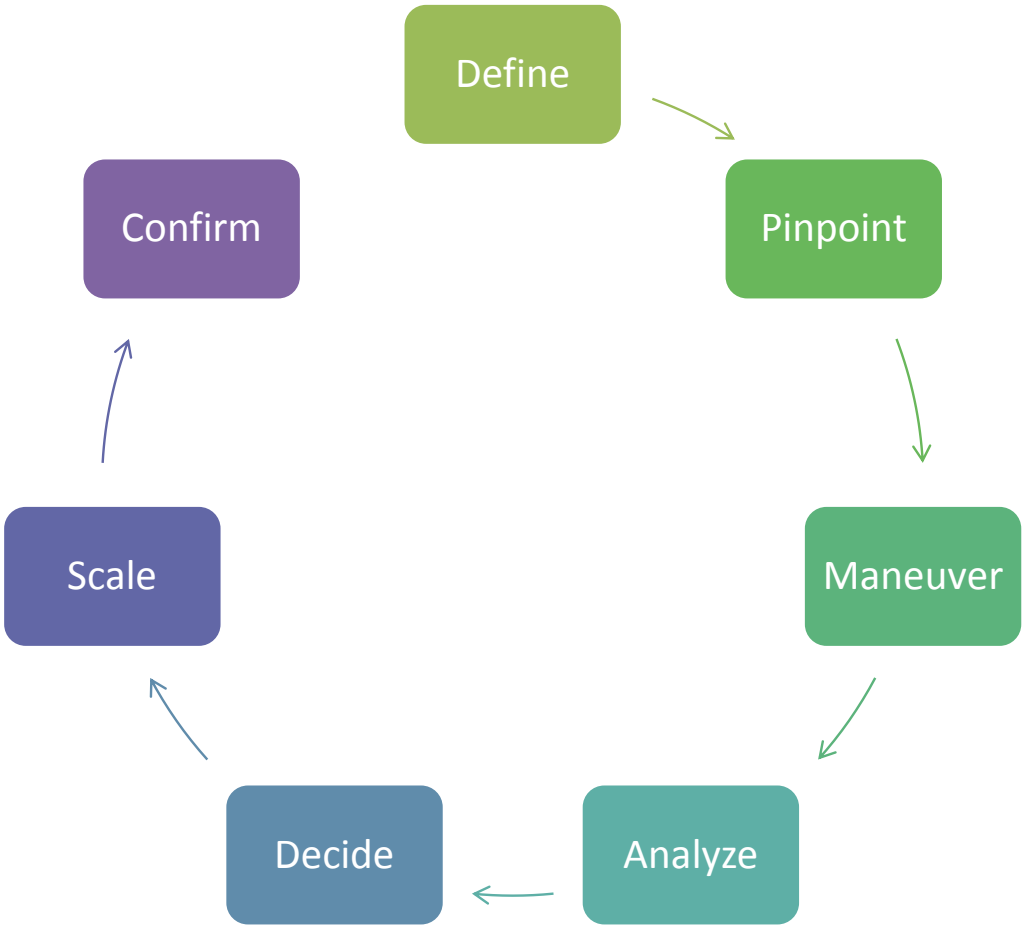

Figure 4.1: The expected roles a cloud simulator may play

5. **Decide**: Compare the impact of alternative policies to determine their points of strength. A simulator is an accurate way to quantify the advantages and disadvantages of newly developed policies.

6. **Scale**: Run real life scenarios on different scales as needed and repeat them as many times as the verification and validation process requires.

7. **Confirm**: Use it as a last step to confirm the behavior of proposed solutions when a failed solution has high risk or high cost associations. This includes tuning performance bottlenecks before deployment. The simulator roles are illustrated in Fig. 4.1.

An example of a use case in which the simulator would play these roles is a scenario where the architects are developing a multi layered resource allocation policy for a cloud data center. This use case constitutes developing a solution that performs the resource allocation for the cloud in order to minimize user request latency to comply with service level agreements (SLAs). The cloud simulator first would help us define the problem by answering questions like: Which elements are involved in this experiment (environment)? Which resources are affected? When do we apply the resource allocation algorithm? Then, it would help us pinpoint potential bottlenecks that would cause this solution to underperform: Is it the VM placement? Is it the resource allocation for the VMs? Is it the chosen topology? Is it the network resources? Then, it would help us maneuver by testing multiple "what if?" scenarios in terms of testing different data sets or use cases or edge cases. Now, we analyze the effect of every factor on the resource allocation process. For example, we might notice that the algorithm performs with sparse heavy requests better than numerous light requests or that it performs better with high-CPU VMs. Based on that, we can gauge and decide which factors have more importance and give them more weight in our scheduling policy. Next step is to scale by testing our scheduling algorithm for a very large data center, large internal network, heavy communication and a large number of requests and see where/when it breaks. Finally, it can be used for final tuning before the algorithm is tested on a real test bed to schedule real requests.

The potential users taking advantage of these simulator roles include:

1. Cloud providers and solution architects: Naturally, cloud providers represent the main stakeholder as they would use this to develop, evaluate and improve their solutions.

2. Cloud clients: Typical cloud client list include large companies that have the capabilities to run their own clouds. A cloud simulator would be beneficial to compare different providers, evaluate currently deployed solutions or for studies of the client workload and to support decisions regarding private vs. public clouds.

3. The research community: simulators are a critical preliminary step for researchers who are developing new cloud technology before testing on a real setup.

4. Other external players: Simulators can be a useful supportive for any parties concerned of evaluating cloud solutions. This could involve auditing and consulting teams or government teams investigating the energy efficiency or carbon print of a cloud for example.

4.3 Comparisons methodology

4.3.1 Cloud Simulator Design Decisions

The process of designing a simulator includes informed decisions related to: components being produced, served, or acted upon by the simulation process. Typical components in the cloud include servers, racks, switches, links, applications, and users. Furthermore, the layout should cover the simulation process flow, its associated resources and events or process steps. A special attention should be given to event frequency and duration. The choice of which probability distributions better characterize execution uncertainties and process variations is crucial. Table 4.1 summarizes the list of elements to be considered when designing the ideal cloud simulator. The first prototype of the simulator can be generated based on these elements.

4.3.2 Cloud Simulator Ingredients

A major objective of this work is to examine each cloud simulator and discuss the fundamental aspects presented by each one of them. We start by introducing a framework of design components for simulators. This cloud simulator design framework is illustrated in Fig. 4.2.

Table 4.1: Simulator basic elements as an initial set of design decisions

Design Element	**Examples**
Entities/components	Servers, VMs, racks, data centers, clouds, switches (access, aggregation, etc), links. users (clients)
Entity attributes	Capacity, power consumption
Simulation scheme	Stochastic, deterministic
Events	Client arrival, new task, new application, task completed
Event frequency and duration	Covering traffic models and client request generation process (Exponentially distributed, normally distributed, uniformly random, deterministic frequency)
Activities	Schedule a VM, schedule a task, migrate a VM
Process sequence	Relations defined between events,activities and outcomes

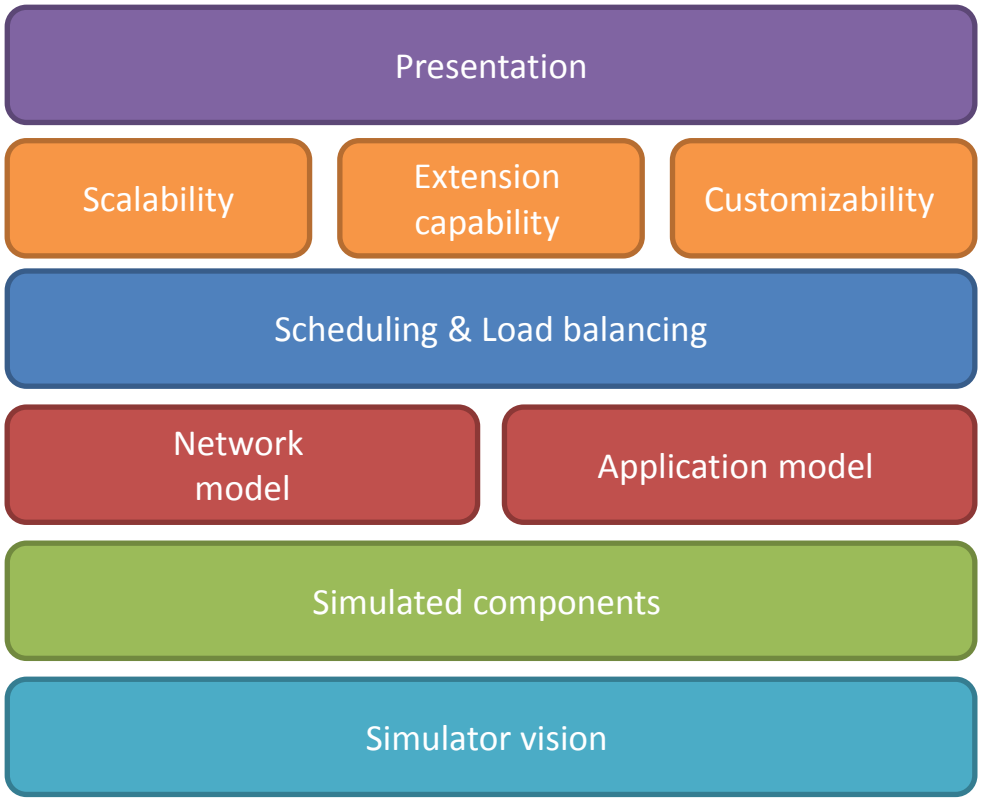

Figure 4.2: Cloud simulator design framework

4.3.2.1 Simulator motivation and vision

The motivating purpose of the developing team when constructing the simulator drives the construction process and built-in features. For example, when the team aims to test cloud computing deployment solutions, a special consideration is given to user request patterns, geographical distributions, types of resources requested and pricing packages. On the other hand, if the prime purpose is performance enhancement, the focus is shifted to request scheduling algorithms, data center network topology options, and general virtualization efficiency. Furthermore, a cloud simulator created to test power consumption methodologies, will contain extensive policies for power consumption recording, variety of power saving techniques revolving around server consolidation or hardware solutions.

4.3.2.2 Simulated physical components and architecture

As discussed previously, the choice of simulated components included affects the simulator architecture and general organization. First, a cloud simulator by definition will contain components representing servers, server resources, and at least symbolic network representation. This is especially important as most of the simulators focus on the Infrastructure as a service (IaaS) aspects of the cloud offerings not Platform as a Service (PaaS) or Software as a Service (SaaS). The choice of server resources to include differs from a simulator to another. Memory, storage capacity, processing units are almost always there. Some configurations add more resources like chips with a certain purpose, for example. The number and capacity of data centers come into play here. Moreover, the vision of distributing clients in terms of different locations and varying request probability distributions is a point to discuss. Virtual machine (VM) offerings and resource allocation are other critical factors. Some simulators will go with a predefined set of VMs with fixed resource amounts allocated to these VMs. This gives the user a choice of VM types or models to choose from [1]. Another alternative is assigning the VM a chunk of resources and then scaling this amount up or down based on the VM real time usage. This will require a more elaborate adaptive resource scheduling and allocation despite it seeming like a more efficient method.

4.3.2.3 Application model

Cloud simulators vary in how they represent user requests and execution components. A subset of the simulators go for the direct method of representing user requests as a list of resource specifications (required processing time or Million Instructions per Seconds (MIPs), required memory, required storage, preferred start time, duration, and/or a deadline). An enhancement of this method can been seen when requests are abstracted into applications or application components. This scenario, to represent reality more accurately, would have to include request inter-dependability and input/output control. In addition, effects on request scheduling will have to be considered. Moreover, dynamic scalability is one of the defining attributes of the cloud. the ability to scale in and out should be accounted for in the application model. This includes increasing the number of components serving a specific application and the increasing the capabilities. Sometimes, these two options are differentiated by calling the former scale out and the latter scale up.

4.3.2.4 Network model

a) Topology representation

Laying out the data center network includes multiple aspects. The topology used arises as a critical issue. When a simulator supports multiple topologies and becomes open for adding new ones, it inherently supports a much larger set of experiments. Tree based topologies are common in cloud data center designs. Server centric topologies like DCell [51] and BCube [52] and switch centric topologies like jellyfish [53] and fat-tree [54], [55] are all possible solutions.

b) Network request representation

Another issue is choosing how to represent data communication (network) requests. The alternatives here are the packet model or the flow model [47]. In the flow model, network requests are dealt with as flows from point A to point B and the aim is to find links with available bandwidth capacity. Some simulators go with an even simpler model that considers bandwidth a commodified resource in a similar manner to memory or storage. In

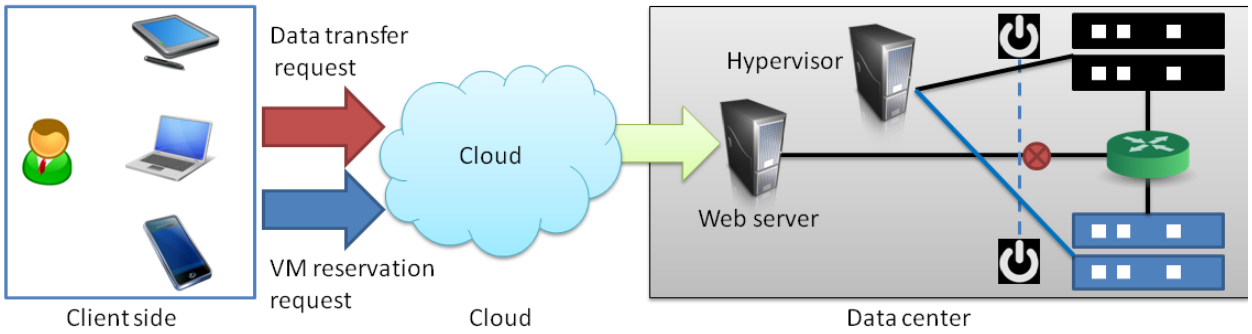

Figure 4.3: Cloud simulated environment and components

this method, the source and destination of a network request are not looked at. Each host is assigned a bandwidth capacity and requests demand is deducted from this capacity during the request lifetime.

c) Other network model concerns

Many challenges that are faced when designing a cloud computing system still face software architects when they try to produce a cloud simulator. Deciding which techniques to be used for routing of a network request (both initially and rerouting) is one of them. Reducing tardiness here is the main objective. More elaborate simulators could even introduce alternatives for traffic decision similar to what is available in cloud data centers. These alternatives include: implementing traffic decision by the relay switch, implementing traffic by a central controller (similar to techniques used in Software Defined Networking(SDN) controllers) or even letting the user make the decision. The last alternative is being promoted now using the term Routing as a Service [56].

The choice between fixed and flexible bandwidth allocation is another decision. However, this issue can be mitigated much easier in a software environment as the switch between these two methods does not prove costly or complicated.

4.3.2.5 Resource scheduling, allocation and load balancing

When faced by the task of designing a resource allocation methodology, many external and internal challenges should be considered. An attempt to summarize these challenges can

be found in [9]. External challenges discussed include regulative and geographical challenges as well as client demands. This results in constraints on the location of the reserved VMs and restrictions impacting the data location and movements. External challenges also include optimizing the charging model in such a way that generates maximum revenue. Internal challenges also include data locality issues. The nature of the application in terms of being data intensive should be considered while placing the VMs and scheduling connections related to this application. All these factors should be put into consideration when choosing the resource scheduling policies. Supporting multiple preconfigured policies is a strong point for any simulator. Moreover, allowing the user to plug in their own policies is another plus that enables users to evaluate their policies precisely. Resource scheduling has a critical role affecting power efficiency, availability, and general data center performance. Fig.4.3 encompasses the elements in play in the resource allocation process within a cloud environment.

4.3.2.6 Extension capability and customizability

A successful cloud simulator, like any software product, gains more client market penetration when the cost to customize is less. Changing the software to meet clients' function automation can take one of two forms:

a) Configuration: where the cloud simulator has the required capabilities and what remains is selecting the correct setup options (configuration) or adding minor GUI components.

b) Customization: where additional functionality or features that did not exist before are to be added.

The chief concerns regarding customizability that arise for clients requiring cloud simulators are:

a) Lack of wide vision:

It is found that many simulators are built to serve a specific research purpose or model. These simulators are then extended/presented as a general cloud simulator when they achieve success. This leads to this simulator being beneficial to researchers/industry parties working in a similar topic. This topic could be power consumption, scheduling

efficiency or cloud pricing options. However, for parties working on a different topic, the customization process proves non trivial. This could lead to increasing the customization cost to exceed the value of building new software components.

b) Availability issues and other typical open source software challenges:

For open source cloud simulators, lack of documentation and support is a consistent issue. Besides, simulator reliability is something to consider.

4.3.2.7 Scalability

The ability of the simulator to scale up to realistic use cases is an important factor when considering whether to build a new simulator from scratch or extend an already existing one. A simulator should have the ability to handle complex topologies and a number of requests large enough to represent real cloud environments. The used algorithms' space and time complexity is a prime indicator here. The type of environments/programming languages a simulator is based on may add scalability challenges as well. Some cloud simulators are based on simulation engines that govern event creation/succession and status updates. This base engine might impose limitations on the size of the solved problem, the number of nodes or the request arrival rates. A simulator that might solve a problem in an acceptable amount of time might not be able to do the same when faced with a problem of larger scale. Determining the acceptable period of time to get a solution for an instance of the problem depends mainly on the client tolerance and the problem urgency.

4.3.2.8 Presentation issues

Well designed appearance and detailed GUI are desirable features in a simulator. Although a new GUI can be built for basically any simulator if it is deemed to have the required core capabilities, having that in place saves time and effort. An issue cloud simulator architects should consider here is including all the simulator critical input values in the input forms/screens. This would save effort for any teams working to extend the simulator in the future.

4.4 Cloud Simulator Detailed review

In this section, a select set of simulators are put under a magnifying glass. The objective is to take a closer look at the Why, What, and How of their implementation. More specifically, a focus on the motivation, distinct features not available elsewhere, strong points and limitations. This survey does not, by any means, target exhausting every single cloud simulator. The aim, instead, is to focus on simulators with high impact and interesting design concepts in order to cover a high percentage of the variations of available simulators. The simulators covered include, CloudSim, NetworkCloudSim, GreenCloud, iCanCloud, TeachCloud, GroudSim, CloudAnalyst, CDOSim, MDCSim, GDCSim, SPECI, and BigHouse. In the interest of saving space and not repeating the content in each simulator, we have explored the ingredients of the first three simulators in more detail. Then, for simulators towards the end of the section, we included the first ingredient (motivation for building the simulator). Moreover, in the second subsection (main features), we focused on the unique features and major additions appearing in each simulator and not repeated in the previous simulators.

4.4.1 CloudSim

4.4.1.1 Description and motivation

CloudSim is one of the most commonly used cloud simulators. CloudSim is a simulation toolkit and an application that enables modeling of single clouds or cloud networks. The designers of the system cited the existing distributed system simulators inapplicability to evaluate: “the performance of cloud provisioning policies, application workload models, and resources performance models in a repeatable manner under varying system and user configurations and requirements” [2], as their leading motive to develop CloudSim. CloudSim is not a standalone fixed-scenario simulator. CloudSim users have the ability to develop the cloud scenario that most fits their needs, design the input parameters and evaluate the output patterns. CloudSim uses robotics simulator Gazebo and is based on an

underlying toolkit called SimJava [57]. SimJava uses a discrete event simulator. It includes facilities for representing simulation objects as animated icons on screen.

4.4.1.2 Simulated physical components and architecture

Through CloudSim, large cloud data centers can be simulated. This includes hosts, virtual components, federated cloud scenarios. Moreover, CloudSim enables users to define and control resource allocation and provisioning policies, virtualization techniques, and energy consumption management techniques. On the operational side, CloudSim supports adding elements dynamically and pausing/resuming the simulation.

a) CloudSim architecture

Studying CloudSim design would give potential architects the insight they need in terms of the required components and layers they will need to build. In [2], the authors explain the layered organization of the CloudSim software framework in detail. SimJava is the discrete event simulation engine that administers tasks like queuing and event processing, creates the system components, and manages the clock. GridSim toolkit is above SimJava. It performs two roles: (i) implementing infrastructure components similar to the ones used in grid applications like networks and traffic distributions;(ii) critical operational components like resource types, data sets, and workload traces. CloudSim is implemented at the highest level containing core cloud functionalities related to data center design and virtualized cloud resources. Finally, on top of the simulation stack comes the user code. The user defines parameters related to resource configuration and the number of servers, number of users, and cloud deployment in terms of brokering options.

b) Simulated physical components

The model used by CloudSim employs a set of classes to represent basic cloud functionality. We will consider two of the critical classes mentioned in [2] as an example. First, the DataCenter class is a core class to the simulator functionality. A similar class is expected to be seen in every major cloud simulator. Server sets can be homogeneous or heterogeneous in terms of resource types and available allocations. Moreover, a DataCenter object instantiates a generalized resource provisioning component that implements a set of resource allocation policies. Attributes of a data center include: Architecture, Operating

system, List of machines, Allocation policy, Time or space-shared, and Resource price per time unit(which represents cost of memory, storage and bandwidth(BW).

Second, is the DatacenterBroker class. The broker's responsibility is standing between service providers and cloud clients. The purpose is to find the most suitable solution package to the client's required resources and Quality of Service (QoS) conditions.

4.4.1.3 Application model

User application in CloudSim is represented through Cloudlet class. Application size (complexity) is represented based on computational demands (instruction length). This is translated into two numbers: instruction length and amount of data transfer (both pre and post fetches totaled). CloudSim does not specify any VM dependencies or data exchange requirements apart from that. Properties of a Cloudlet include: *(i)* length (in Million instructions - MI); *(ii)* file Size; *(iii)* output Size; *(iv)* number of CPUs.

4.4.1.4 Network model

CloudSim reads the data center topology using an input file in the BRITE format. For every node, the file contains attributes specifying x and y-axis coordinates, in-degree and out-degree of the node. For each edge, it includes specifications of the source, destination, Euclidean length, propagation delay, bandwidth and type.

Despite reading the topology details, this information was not mainly used in the CloudSim until NetworkCloudSim was added. The network inside the data center is not explored in detail either.

4.4.1.5 Power consumption features

CloudSim contains basic energy consumption recording statistics (energy consumed, CPU utilization, etc.). The CloudSim team members have performed a detailed implementation

of multiple energy conservation policies. These policies are mostly based on two energy conservation concepts:

(i) DVFS: A simulation of a heterogeneous power-aware data center that only applied Dynamic voltage and frequency scaling (DVFS), but no dynamic optimization of the VM allocation; and

(ii) Server consolidation/VM migration: Simulations of heterogeneous power-aware data centers testing multiple VM allocation policies and VM selection policies (migrated VM selection method) The goal is to arrive at the combination that performs best in terms of energy consumption.

These used policies include: *(i)* VM allocation techniques (like Inter Quartile Range (IQR), Local Regression (LR), Local Regression Robust (LRR), Median Absolute Deviation (MAD) and Static Threshold (THR)); *(ii)* VM selection policies like (Maximum Correlation (MC), Minimum Migration Time (MMT), Minimum Utilization (MU) and Random Selection (RS)). There was no consideration for different energy power sources in CloudSim.

4.4.1.6 Scalability testing results

We tested CloudSim on our machine that has 8 cores and 64 GB of memory. Parameters are included in Table 4.3.

a) Execution times

We scaled the problem up by changing the number of VMs, Cloudlets and hosts. Apart from the problem size, many factors may affect the speed of the execution. For example, a considerable increase in the time delay is noticed when the VM/application load is more than the hosts could handle. This might be due to limitations in the VM placement algorithms. It is noticed that if the simulator cannot allocate the VM, it will go through every single host and report that they do not have enough space to allocate it.

b) Testing different types of loads/applications

To further specify the factors that affect CloudSim runtime, we performed additional tests on a load of 20K VMs, 40K Cloudlets and 10K hosts with 4 processors per host. This time, the controlled parameters included: *(i)* simulated data center host processing

Table 4.2: A summary of CLoudSim features

Feature	**Available?**	**Details**
Ability to model user requests	Partially	Applications represented by a workload object that contains user workloads (by MIPS)
Ability to model inter-VM dependency	No	-
Ability to model multiple DCs	Yes	-
Ability to model servers	Yes	With fixed set of attributes
Ability to model network elements	limited	Data center network is modeled using BRITE" format but not used. Internal network not represented.
Ability to model VMs	Yes	Resource configuration and placement
Ability to model inter-VM connectivity	No	BW required by a VM is treated as a fixed commodity
Ability to model failures/recoveries	No	-
Ability to model energy power sources mix (24 hour source types	No	-
Ability to model power usage per VM, Server, Facility	Yes	Multiple power management methods are implemented Basic power usage statistics are available
Ability to model security measures (attacks, firewalls)?	No	-
Ability to model network flows?	No	-

Table 4.3: CloudSim test parameters

Data Center Parameter Values	**VM parameters**
a-system architecture= "x86" b-operating system= "Linux" c- processing resource cost= 3.0 d- memory resource cost = 0.05 e- storage resource cost = 0.1 f- BW resource cost = 0.1	a-image size = 10000 MB b-VM memory = 512 MB c- MIPS = 1000 d-BW = 1000 e-number of CPUs = 1 f-VMM name = "Xen"
Host parameters	Cloudlet(application) parameters
a-host memory (2048 MB) b-host storage = 1000000 c-BW= 10000	a-length = 1000 MI b-file size = 300 c-output size = 300 d-required CPUs = 1

Table 4.4: CloudSim Execution times for different load amounts

Number of VMs	**Number of Apps (cloud lets)**	**Number of Hosts**	**Execution time**
20	40	10	3 Sec
2K	4k	1k	4 Sec
10 K	20 K	5 K	1 Min.44 Sec
20 K	40 K	10 K	8 Min. 43 Sec
100K	200K	30K	267 Min

Table 4.5: CloudSim Execution times for different load types when for 10K hosts, 20K VMs,40K Cloudlets with 4 processors per hosts.(the changed parameter in each line is put in bold).

VM capacity (in MIPS)	**Cloudlet size (in mi)**	**Host processing capacity (in MIPS)**	**Average execution time**
1000	1000	2000	8.33 Sec
2000	1000	2000	8.23 Sec
100	1000	2000	8.30 Sec
1000	10000	2000	8.29 Sec
1000	100	2000	8.21 Sec
1000	1000	5000	8.28 Sec

capacity; *(ii)* simulated VM assigned processing speed (in Million instructions Per Second (MIPS)); *(iii)* simulated requested CLoudlet size (in MI). The results in Table 4.5 show that the load type does not have a significant effect on CloudSim runtime if the load amount does not change. If we look at one of the major cloud providers, we can find that "Amazon data centers house between 50,000 and 80,000 servers, with a power capacity of between 25 and 30 megawatts." [58] In comparison, Google's major data centers are supported by at least 50 megawatts of electric power, with some estimates ranging as high as 103 megawatts [59] which supports the 2013 estimates in [60] which put their server count at 900000 in 13 data centers at the time. (Averaging around 70000 servers per data center). When Rackspace reached 70000 servers, it was only the sixth company to reach this amount (in total servers operated).[61] This number is in all of its data centers in 6 cities around the world.[62] Therefore, the largest experiment we have conducted on CloudSim (30000 servers) is fairly close to the biggest data centers on the planet and would cover most of the other data centers.

4.4.2 NetworkCloudSim

4.4.2.1 Description and motivation

NetworkCloudSim is an extension of CloudSim that focuses on network capabilities. The major motive is enhancing CloudSim with complex application models such as message passing applications and workflows in addition to supporting a scalable network model for cloud data centers. As for the first point, other simulators like CloudSim and MDCsim model application requirements in the form of instruction length or other variations of the computational resource requirements. A more detailed definition of the client applications is required to cover tasks like parallel applications and workflows.

As an extension to CloudSim, it is similar in operation to it. NetworkCloudSim is capable of simulating Data center networks (DCNs) and applications of communicating tasks such as an MPI. The authors have designed a network flow model for cloud data centers utilizing bandwidth sharing and latencies to enable scalable and fast simulations [47].

4.4.2.2 Application model

In typical simulators, you can send a request to define a task on more than one processor. What really happens is they are scaled to the computational time of one processor which is not the real scenario. Tasks in a real cloud computing scenario have to communicate. NetworkCloudSim introduced the NetworkCloudlet class to represent a task executing in several phases/stages of communication and computation. CloudSim Architecture diagram [2] shows how the areas/classes in CloudSim architecture are affected by the changes/extensions of NetworkCloudSim. To model the application precisely, a class called AppCloudlet is introduced. Each application contains several communicating elements (NetworkCloudlets). Each element runs in a VM and consists of stages where it either performs data exchange tasks (communicating) or computing tasks. These stages are: execute, send data, receive data or finished. Computing stages can be defined by MIPS while data transfer tasks are characterized by the amount of data. When in the send stage,

the VM scheduler submits a packet to the send-packet-queue of the VMs. After the execution stage of each NetworkCloudlet, VM scheduler forwards theses packets to VMs on the same host or to switches. In NetworkCloudSim's implementation, no messages will be blocked even if the destination is not ready to receive the message. The receiver VM has the option either to process other tasks or to be blocked until the message arrives. The authors summarize: "This communication model allows the Simulation of the non-blocking message passing paradigm (such as MPI Isend() and MPI Irecv()), which is a common practice in parallel applications" [47].

4.4.2.3 Network model

Quality of Service conditions required by a cloud client often take a hit because of network request latency. NetworkCloudSim aims at modeling realistic network requests in terms of topology, request size, and hierarchy. CloudSim lacks the ability to facilitate intra–data center communication. Bandwidth sharing on network links is not modeled. This is an obstacle to modeling features like VM migration. There are two questions posed while extending network capabilities. These question where addressed in detail in [47]. First, designers should choose if they want to go with flow model or packet model. Flow model has the advantage of lower computational overhead despite having much less details. The second issue is VM interconnection topology. A fully connected model is not common in real data centers. NetworkCloudSim tackled this issue by adding root, aggregate and edge (access) level switches. This gives users the freedom of configuring switches and ports according to their use cases. A bandwidth allocation algorithm should be included here in the case multiple simultaneous flows use the same link. The simulator adds present latency based on the link length as well.

To model a network within the data center, Switch class has been introduced as a network entity (switch or a router) that can also model forwarding latency. Moreover, NetworkPacket and HostPacket classes represent data flow out of the VM. A HostPacket is transferred using the virtual network. A NetworkPacket goes from a server to another through the physical network. Being an extension of CloudSim, the other sides of Network-

Table 4.6: A summary of NetworkCloudSim features

Feature	**Available?**	**Details**
Ability to model user requests	Yes	Can model communicating/dependent applications
Ability to model inter-VM dependency	Yes	-
Ability to model multiple DCs	Yes	-
Ability to model servers	Yes	With fixed set of attributes
Ability to model network elements	Yes	Internal DC network is modeled including switches
Ability to model VMs	Yes	Resource config and placement
Ability to model inter-VM connectivity	Yes	-
Ability to model failures/recoveries	No	-
Ability to model energy power sources mix (24 hour source types	No	-
Ability to model power usage per VM, Server, Facility	Yes	Multiple power management methods are implemented, basic power usage statistics are available
Ability to model security measures (attacks, firewalls)?	No	-
Ability to model network flows?	Yes	The designers chose flow model over packet model to send/receive data

CloudSim are largely similar to the original simulator. We will not traverse these sides in the interest of saving space.

4.4.3 GreenCloud

4.4.3.1 Description and motivation

GreenCloud is a cloud simulator with a focus on energy efficiency and enhanced capabilities for network communications. The prime purpose cited for building GreenCloud is mitigating overprovision issues. Overprovision happens in a data center due to the changing loads on its computational and network resources. The average load can be as low as 30% of the data center server and network capacity [3]. This, in turn, causes the data center to systematically use more power than the optimal value.

GreenCloud is an open source extension of the network simulator NS-2. Data transfer processes are modeled on a packet level. Most of the code (80%) is in C++ and the rest is in Tool Command Language (TCL). The simulator records energy consumption for servers, switches, and links as well as having workload patterns. On the other hand, a challenge faced by GreenCloud is the limited scalability to small data centers due to very large simulation time and high memory requirements.

4.4.3.2 GreenCloud architecture

The increasing scale to which data centers are growing (tens of thousands of hosts) and the increase in the percentage of internal communications (70% of all communications performed by data center components) [3] call for more attention to the data center architecture robustness and efficiency.

GreenCloud offers three options for the possible data center topology:

a) Two-tier data center architecture:

Racks of servers form the tier-one of the network. Network Layer-3 (L3) switches facilitate full mesh connectivity using 10 GE links. Two-tier data centers in Greencloud may support up to 5500 nodes.

b) Three-tier data center architectures:

Access, aggregation, and core layers are included in this architecture which increases supported number of servers to 10000. This architecture supports an 8-way equal cost multi path routing(ECMP) with 10 GE Line Aggregation Groups (LAGs). This gives the client the ability to address several links with a single MAC address. However, LAGs are known to limit network flexibility and performance. As the authors put it: “LAGs make it difficult to plan the capacity for large flows and make it unpredictable in case of a link failure. In addition, several types of traffic patterns, such as ICMP and broadcast are usually routed through a single link only” [3].

c) Three-tier high-speed data center architecture:

This architecture adds 100 GE links (IEEE 802.3ba) between aggregation and core switches. This leads to reduction in the core switches and avoiding the disadvantages of LAGs as well as cabling reduction and supporting data center scale ups.

4.4.3.3 Application model

Host objects represent servers containing typical data center resources. Scheduling techniques include round robin and energy aware scheduling. Tasks can either be scheduled on the hosts directly or on the VMs residing on the hosts. Users can control transmission rates to support power saving with several options available (for GE links, 10 Mb/s, 100 Mb/s, and 1 Gb/s are available).

a) Task and user request specification

A workload object consists of a computational part that is measured by MIPS and a network request part. The authors set the task network demands to consist of three parts. First, the task size which is transmitted from the root node to the hosting server. This represents the task code or instructions along with the input data. Second, the data the task communicates with other servers in the data center. Third, the server transmits the task output to the client (represented by sending it to the root node). GreenCloud enables choosing a distribution to configure the task arrival pattern (like exponential or Pareto) or even generating data from trace log files.

b) Types of tasks:

User applications are modeled in objects called tasks. Three types of tasks or workloads are available in GreenCloud: Computationally Intensive Workloads (CIWs), Data-Intensive Workloads (DIWs) and Balanced Workloads (BWs) [48]. The comparison of the varying types in Table 4.8 is especially helpful when faced by the task of setting up user workloads.

c) Power consumption features

Power consumption efficiency is GreenCloud's primary target. Basic statistics are available for all data center components power consumption (computational and network resources). The equation used to calculate server power consumption is taken from [63]. It calculates the power consumed as a function of the resource utilization of each resource type. The equation introduced to calculate the power consumed by a switch is as follows:

$$P_{switch} = P_{chassis} + n_{linecards} + P_{linecard} + \sum_{i \in R} n_{ports,r} + P_r \qquad (4.1)$$

Table 4.7: Types of workloads/tasks at GreenCloud

Features/Application type	**Computationally Intensive Workloads (CIWs)**	**Data-Intensive Workloads (DIWs)**	**Balanced Workloads (BWs)**
Real life Application Modeled	High-Performance Computing (HPC) applications that solve advanced computational problems	Applications like video sharing (for each request there is a streaming process.)	Applications with computing communication requirements (geographic information systems)
Computational vs. network load	high computing load, low network load	no computing load, high network load	load the computing servers and communication links proportionally.
Scheduling central point	Server energy efficiency is critical (server consolidation)	No network congestions	Network becomes the bottleneck Constant exchanged feedback is a must-Both computational and network portions.

P_r represents power consumed by a port running at a rate valued r. GreenCloud implements three energy efficiency techniques:

(i) Dynamic voltage and frequency scaling (DVFS); *(ii)* Dynamic power management (DPM); *(iii)* A hybrid scheme of both.

d) Testing Results

We tested GreenCloud on our machine that has 8 cores and 64GB of memory. GreenCloud works on Linux Ubuntu. The results shown in Fig.4.4 are for a relatively small problem (144 servers) and it took around a minute (65 seconds). When moving from the testing architecture (3-tier-debug) to the two larger architectures (3-tier and 3-tier-high-speed) that contain 1536 servers while keeping the other factors without change, the simulator times increase vividly to around 20 minutes. Execution times are shown in Table 4.9. Clearly, GreenCloud commands more execution time than CloudSim. This is due to multiple design factors including NS-2 dependency and using the packet network model. A detailed representation of the multiple power consumption metrics is displayed as soon as the simulation is over. A sample output is shown in Fig. 4.4.

Due to the lack of space, we will henceforth conduct a high level analysis of the

Table 4.8: A summary of GreenCloud features

Feature	**Available?**	**Details**
Ability to model user requests	yes	CIWs,DIWs,BWs
Ability to model inter-VM dependency	no	Can configure Tasks to communicate with a random server
Ability to model multiple DCs	no	-
Ability to model servers	Yes	With fixed set of attributes
Ability to model network elements	yes	3 different topologies available switches and links modeled packets sent over the network
Ability to model VMs	Yes	Resource configuration and placement
Ability to model inter-VM connectivity	yes	Can configure Tasks to communicate with a random server
Ability to model failures/recoveries	No	No mention of recovery in their documentation
Ability to model energy power sources mix (24 hour source types	No	
Ability to model power usage per VM, Server, Facility	Yes	Multiple power management methods are included Basic power usage statistics are available
Ability to model security measures (attacks, firewalls)?	No	-
Ability to model network flows?	Yes	It is based on NS2, TCP/IP enabled.

Table 4.9: A sample of GreenCloud execution times for different architectures with a data center load of 70 per cent

Architecture	**Server count**	**Execution time**
3-tier-debug	144	56 Sec
3-tier-hi-speed	1536	19 Min 8 Sec
3-tier (default)	1536	19 Min 20 Sec

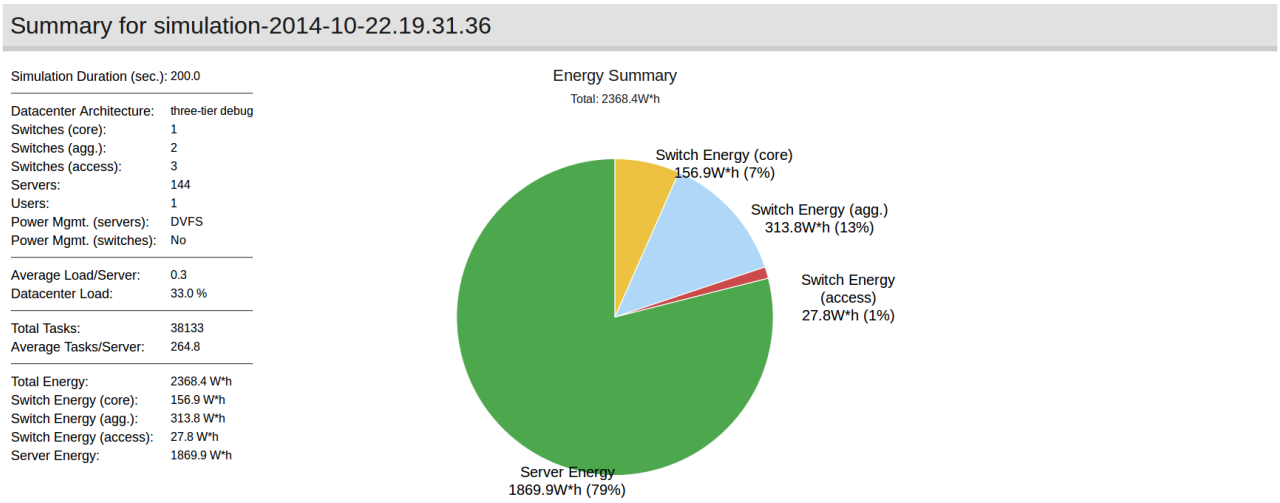

Figure 4.4: GreenCloud output

remaining simulators. We will illustrate the standing out aspect of each simulator in terms of purpose or functionality. We will also take a look at the scalability of the rest of the simulators at the end of Section 4.4.

4.4.4 iCanCloud

4.4.4.1 Description and motivation

The designers of iCanCloud used Simcan framework to build their cloud simulator [64]. They have built a full GUI where users can configure, manage and run VMs, data centers and experiments for different cloud scenarios. Distributed experiments are also supported as iCanCloud can run an experiment on multiple machines. The challenge of predicting and structuring cost in cloud data centers is well studied.

4.4.4.2 Main features

Apart from the basic cloud simulation, iCanCloud combines a set of features that stand out among cloud simulators. iCanCloud simulates the hypervisor module used by cloud

providers. The hypervisor module handles brokering between cloud clients who send jobs and cloud data centers where these jobs can be served. Data centers represent a set of VMs, each one configured with pre-defined features such as CPU, storage, memory, and network. Cloud clients in iCanCloud represent entities that submit a set of jobs to be executed on specific VM instances. Those submissions arrive directly to the hypervisor module. The hypervisor manages the brokering process by insuring efficient distribution of the jobs to the data centers and VMs inside them. This is done using preset brokering policies which can be amended by customization. The available brokering policies are cost based policies, execution time-based policies and resource allocation-based policies. The hypervisor also handles managing VMs and the execution of the jobs on VMs, and defining cost policies for each VM instance type. This aids in integrating and testing brokering policies.

Moreover, features like flexibility in modeling architectures and VMs that represent single or multi-core hosts are supported. Storage gets a special attention in iCanCloud as it supports models for local or remote storage systems as well as parallel storage and RAID designs. In regards to the development environment, POSIX API and an adapted MPI library is available to build and run simulation experiments. In their article [64], the authors explain that they experimented on an application named Phobos where processors are used to calculate the trajectories of Phobos, the Martian moon, over a tracing interval. This is done by dividing the overall tracing interval in identical subintervals, each of them executed by a different task. The same task load distribution was simulated on iCanCloud and produced the same results in terms of the calculated performance cost.

4.4.5 TeachCloud

4.4.5.1 Description and motivation

TeachCloud was built for a specific purpose, namely education [65]. TeachCloud shows a GUI through which researchers or industrial engineers can configure and perform experiments based on cloud scenarios.

4.4.5.2 Main features

TeachCloud is an extension of CloudSim. On top of the CloudSim Framework, the designers added a GUI and a module that generates cloud related workload. Architectures like VL2, BCube, Portland and Dcell are supported. The simulator contains modules that monitor data center components, show the impact on system effectiveness and allow reconfiguration of the experiments.

4.4.6 GroudSim

4.4.6.1 Description and motivation

GroudSim is a discrete event simulator that runs simulations specific to scientific applications on grid and cloud environments. The simulation core processes are performed by SimEngine. Groups of events can be created using features of the SimEventReference [66].

4.4.6.2 Main features

GroudSim can be integrated into ASKALON environment [67] and that enables users to import experiments that represent real applications from that environment. Although GroudSim's vision involves executing jobs for the grid, it still includes basic cloud computing features like task implementation, cost estimation and resource loading and scheduling. Failures of grid sites and data transfer between cloud resources are features that can be seen in GroudSim.

However, not many bases are covered on the network side. In addition, performance issues arise when the scale of the experiment reaches hundreds of nodes.

4.4.7 CloudAnalyst

4.4.7.1 Description and motivation

CloudAnalyst is another extension of CloudSim. The motive behind CloudAnalyst is analyzing and evaluating geographically distributed user workloads [68]. Requests that come from distant locations with heterogeneous distributions and sizes constitute a challenge when cloud providers plan their cloud deployments solutions. These scenario is also challenging when choosing the resource allocation policies for the data centers. In this case, the quality of service received by multiple users distributed all over the map is evaluated while varying the experiment parameters.

4.4.7.2 Main features

Through CloudAnalyst, experiment conditions can be collected. The use of Java Swing to extend CloudSim makes it easier to extend CloudAnalyst. Additionally, CloudAnalyst provides a GUI where users specify the experiment parameters, the data center distribution and user location along with the network setup.

4.4.7.3 Configurable parameters

CloudAnalyst offers a wide range of configurable parameters. All CloudSim parameters are included in Table 4.10 . In addition, Internet characteristics, simulation configuration, Data center configuration and user distribution/load parameters are supported. The output from CloudAnalyst compare the response times/user experience from different geographically distributed bases.

Table 4.10: CloudAnalyst test parameters

Internet characteristics	**User parameters**
1-Region/region delay matrix : transmission delay between regions in (milliseconds) 2-Region/region bandwidth matrix (available bandwidth between regions)	1-User grouping factors in user bases (how many simultaneous users from the same base) 2-User grouping factors in data centers (how many simultaneous users on the same host) 3-Instruction length/ request (bytes) 4-Load balancing policies a-round robin b-equally spread current execution load c-throttled
Simulation configuration	Data center configuration
1-Simulation time 2-User bases: a-average number of users at peak time b-average number of users at off-peak time c-region d-request/user/hour and request size. e-daily peak hours 3-Broker service policy: a-closest data center b-optimize response time c-reconfigure dynamically	1-Region 2-Architecture 3-OS 4-VMM 5-VM cost 6-Storage cost 7-Data transfer cost 8-Physical hardware units 9-Server configuration: a-memory b-storage c-BW d-number of processors e-processor speed f-VM policy (time shared or space shared)

4.4.8 CDOSim

4.4.8.1 Description and motivation

Cloud Deployment Simulator (CDOSim)'s purpose is, as the name indicates, optimizing cloud deployment options. This includes measuring delays, Service Level Agreement violations and the subsequent costs based on a specific cloud deployment option from a client perspective. CDOSim provides features that support client decision making in terms of choosing the cloud provider, the runtime deployment policies and VM resource configuration.

4.4.8.2 Main features

CDOSim enables cloud clients to compare the cost and effectiveness of a specific cloud solution with those of other solutions. This aims at mitigating the cloud clients' lack of knowledge over cloud platform options. Features like real life user trace integration and independence of programming languages are available [69].

4.4.9 MDCSim

4.4.9.1 Description and motivation

Multi-tier Data Center Simulator (MDCSim) was built to constitute a scalable platform to conduct system evaluation and power consumption measurements for cloud environments. The authors demonstrated the abilities of the simulator using studies that included three types of applications [70].

4.4.9.2 Main features

As a general purpose cloud simulator, MDCSim support data center component definition

and configuration. Moreover, it includes features related to analyzing and evaluating of Infiniband Architecture (IBA) and 10GigE performance under varying cluster sizes, network load, and with different tier configurations. It also supports basic power measurement and configuration features. Still, more work is required to enhance the network model and strengthen power efficiency features for network component. In addition, the proprietary nature of MDCSim has affected its market penetration and extensibility.

4.4.10 GDCSim

4.4.10.1 Description and motivation

Green Data Center Simulator (GDCSim) is another simulator whose central purpose is reaching the completely green cloud. The designers aim at creating a framework to evaluate new resource allocation policies or energy efficiency techniques. Conditions to be tested include topologies, workload distribution, platform power management schemes, and scheduling algorithms. In fact, both statements are correct. GDCSim contains a CFD component and its contribution and results are validated against traditional CFD simulators. The authors of GDCSIm clarify that GDCSim -when used in the context of cloud data center energy efficiency analysis- has four major advantages compared to a typical CFD simulator. Those are automated processing, online analysis, and more importantly, workload management and cyber physical interdependency. Compared to other cloud simulators, GSCSim is an example of a simulator with more focus on the thermal side of energy efficiency. Feedback on temperature and air flow patterns in the data center is used by the management algorithms.

To avoid the challenge of high complexity and slow execution of CFD simulation, the authors opted to use the CFD component GDCSim is built upon, BlueSim, only for a partial task. The CFD simulator module, BlueSim, was used to generate the thermal map of the data center and was not used in conjunction with the resource management module. Furthermore, GDCSim was validated against established Computational Fluid Dynamics (CFD) simulators like Flovent CFDSim [71].

4.4.10.2 Main features

GDCSim supports online analysis and adjustment as users can manage the simulation scenarios while the simulator is running. This gives users the chance to adjust and modify based on physical resource changes. GDCSim also supports thermal analysis features, in which the thermal conditions at a given moment are recorded and analyzed. Cooling policies can be tested in detail. Power modes can be controlled in terms of servers' status. An added feature here is considering "cyber-physical interdependency, which enables feedback of information on temperature and air flow patterns in the data center to the management algorithms and the closed loop operation of the servers and cooling units (CRAC) to achieve energy efficient operation" [71].

4.4.11 SPECI

4.4.11.1 Description and motivation

Simulation Program for Elastic Cloud Infrastructures (SPECI) is another cloud simulation tool that allows studying large data centers closely [72]. SPECI studies the performance of the data center nodes as a unit. It focuses on the efficiency of the middleware policies used in a cloud data center. The authors define middleware here as the layer of software that handles job scheduling, load-balancing, security, virtual networks, and resilience. It is basically the management layer of the data center. SPECI focuses on handling communication policies between the simulated hardware components inside a data center specially in the case the number of these components (or nodes) increases sharply. These nodes communicate with each other and therefore need to be aware of each other's states. Each of these nodes can be functioning ("alive") or not ("dead") and this state needs to be communicated to nodes that work with this node. SPECI offers an environment to simulate this process in order to evaluate communication protocols and data center load.

Basically, nodes (which can represent cloud hosts) are connected through a network with a configurable topology. Nodes are susceptible to random failures. Any component

which cooperates with a set of other components, is thus interested in the aliveness of each of these components and performs queries to find about their states. This state retrieval gets more complicated and it causes more data exchange over the network as the number of components (nodes) increases.

Several architectures of state or heartbeat retrieval can be configured. Examples include having a central node sharing all the states, a hierarchical design where data is shared or a P2P data sharing model. SPECI aims at observing the behavior of the whole system under different architectures, setup parameters and protocols especially as the number of nodes scales up.

4.4.11.2 Main features

SPECI is based on the SimKit framework [73] which offers the core simulation functionalities. The challenge faced in this scenario is scalability. The data center size increases and some middleware properties do not scale in a linear way with the number of components.

SPECI consists of two packages. The first one deals with data center architecture and topology. The second contains the modules dealing with simulation and measurements. Status updates are critical to the designers as they aim to evaluate the impact on component failure and policy changes. In terms of failure observance, the notions of hardware components being alive or dead are presented. These states are exchanged in a subscription based scheme. When the simulator is initialized, an object is created for each node and network link in the data center. Each node is subscribed to the states of a list of other nodes based on a predefined distribution. Next, the communication policy is loaded and the rest of the simulation is run by the event queue.

4.4.12 BigHouse

4.4.12.1 Description and motivation

BigHouse is another simulator with some unique features. BigHouse developing team introduced it as a simulator with a higher degree of abstraction. This is coupled with a focus

on the statistical view and execution time minimization. The motivation is to use simulation to approximate the performance of complex queuing models like G/G/1 or G/G/k queues (generalized inter-arrival and service time distribution with 1 or k servers).

4.4.12.2 Main features

In BigHouse, like most of the main simulators, Tasks are represented by random variables that describe their parameters including resource requirements, arrival and service time details. Examples of the workloads used include Departmental DNS and DHCP server under live traffic, Departmental POP and SMTP server under live traffic, Shell login server under live traffic, executing a variety of interactive tasks, Leaf node in a Google Web Search cluster, HTTP server under live traffic. Once workload distributions have been generated, a system model in terms of power consumption and performance distribution is specified and then estimate results can be derived.

BigHouse is based on the stochastic queuing simulation (SQS) methodology. This deals with the requests/tasks as high level components that run in time in the order of milliseconds instead of on an instruction by instruction level. Consequently, "BigHouse can simulate server systems in minutes rather than hours" [74]. Architecturally, BigHouse contains two main models. The first one handles constructing the data center components and the event sequences like any discrete event simulator. The second one mainly contains the reporting tools and handles the statistical modules. The authors have shown some results showing the parallel performance of BigHouse which is interesting in terms of scalability of the simulator [74]. Running time can be tweaked by changing the limit of accuracy or confidence in results.

A notable limitation to BigHouse is the network model. The examples previously mentioned model client-server scenarios. More realistic models (including a step to three tier topology for instance) are not included unless changes to the simulator are made.

4.5 Simulator Comparison

When looking at the previous efforts, we can divide them into two sets:

4.5.1 Simulators with a Specific Purpose or Working on a Limited Scale

This purpose could be educational like TeachCloud or related to a specific functionality like CDOSim. We note here that some of the simulators are not only developed by academic teams but also are meant to serve academic purposes mainly.

4.5.2 Simulators Operating on a More General Scale

This set includes simulators that include features in which users can test general cloud computing problems and solutions. The criteria here are: the simulator's extensibility, the simulator containing the critical scheduling and energy efficiency recording abilities and the simulator including a rich networking model. In Table 4.13 and Table 4.14, a comparison of the general features, strength and limitations of each of the simulators discussed previously can be found.

4.5.3 When Do I Need a New Simulator?

The major reasons that might push a research or an industrial team towards the choice of building a new cloud simulator include one of the following points.

4.5.3.1 Scalability

There is need for a simulator that is proven to work for large scales reaching data centers with a size in the order of 10000 to a 100000 nodes(hosts). Intense comprehensive testing is a key requirement. We have taken a look at the scalability properties of CloudSim and GreenCloud. More on the scalability of the rest of the surveyed simulators in the following subsection.

4.5.3.2 Portability

Many available simulators are subject to platform, environment, and hardware limitations. These limitations need to be examined and the new simulator should consider all the environments/conditions that may be called upon in the required future work.

4.5.3.3 Customizability

An in-house developed simulator offers advantages in terms of customization to the ever expanding needs of R&D departments.

4.5.3.4 Training and Knowledge transfer

An in-house simulator also offers an advantage in terms of training/knowledge transfer over other simulators. This covers even open sources simulators as documentation and support challenges usually arise.

4.5.4 A Look at Simulators' Scalability

by examining [47], [64], [65], [68], [66], [69], [70]. [71], [72], and [74], we gain a sense of the scale each simulator works on. For iCanCloud for example, the simulator was tested for experiments to run up to 250000 jobs on up to 5000 VMs. Those experiments used jobs whose input size is 5 MB, output size is 30 MB, and processing length is 1200000 MI. VMs computing capacity was up to 9500 MIPS, which simulate a standard small instance type provided by Amazon EC2. the largest experiment finished in about 10000 seconds (2 hours and 45 minutes).

For NetworkCloudSim, the experimental setup contained 4 servers, 8 VMs and up to 8 processes per VM. An MPI application was modeled, with the main process generating several random numbers and then send the data to all other processes. Each host has 2 Xen VMs, each one with 2 cores and 1.5 GB RAM. All the hosts are connected by a 100 megabits switch. varying number of communication messages up to 1000000 MPI INT elements transferred from one process to another. This took around 2.5 seconds. In terms

of experimenting on scheduling policies, the same topology was used to test scheduling requests with an arrival rate of 200 pr second.

For CloudAnalyst, the problem can be described as follows: Up to 3 data centers containing up to 75 VMs receive connection requests from users distributed along 6 user bases around the world. The total number of users can range from 180000 to 1800000 depending on peak hours. But that number is actually grouped by a factor of 1000. The VM size is 100 MB. VMs have 1 GB of RAM and have 10 MB of bandwidth. servers have 2 GB of RAM and 100 GB of storage. Each machine has 4 CPUs, and each CPU has a capacity power of 10000 MIPS. A time-shared policy is used to schedule resources to VMs. Each user request requires executing 250 instructions. Requests are grouped by a factor of 100.

For GroudSim, a number of jobs ranging from 5000 to 32000 was executed on a number of cloud instances from 8 to 32000. "The type of Cloud resources is not relevant to these experiments, as we are only interested in how fast a certain number of jobs on a certain amount of resources can be simulated, independent of their real execution time." [66] If we look at CDOSim, the problem representation is slightly different as it uses a workload intensity function origins from a service provider for digital photos producing 5000 requests (method calls) per minute on a data center ranging in sizes between 4 and 7 nodes. The experiment took around a day and a half in total.

For MDCSim, the case study major components included clients, WS tier, AS tier and DB tier. A simple round-robin web switch emulation was implemented to distribute client requests to the web server tier nodes. This included up to 128 nodes in the 3 tiers with a number of communicating clients up to 5600 clients. The exchanged message size was around 5 kb. In the GDCSim experiment, there were two rows of five industry standard 42U racks in each row, laid out in hot/aisle cold aisle configuration. These racks consisted of five 7U chassis each. The data center tends to be smaller due to GDCSim interest in the energy efficiency and thermal component of the data center operation.

As for SPECI, Scaling is easier in terms of the number of components as it focuses on the component livelihood status rather than execution of complex requests. Simulations data center networks of the size of 100000 hardware components are possible.

Lastly, when looking at the scalability of BigHouse, it is found that simulation of a 10 servers cluster takes less than a minute. The time increases approximately linearly until biggest recorded size which 10000 server data center where the simulation finishes in hours rather than days.

These results are summarized in Table 4.11 and Table 4.12 in addition to experimental results shown for CloudSim and GreenCloud in tables 4.4 and 4.5.

4.5.5 Previous Experiments and Problems Solved using Surveyed Cloud Simulators

The previously discussed simulators have been used variably for multiple projects. Some of these projects were attempts to solve a specific problem or demonstrate certain cloud scenario using the simulator. Some of these projects were done by the same team with the purpose of building upon the simulator and some was done by different teams who used the simulators mostly under the open source licensing schemes.

A look at the literature and the websites of active simulators would give us an idea of the sort of audience following these projects and the sort of extensions/solutions built upon them. In Table 4.15 we provide the list of topics and problems each simulator was used to evaluate or solve. This is another step towards helping the reader in the process of deciding which simulator is the most suitable to their project.

4.5.6 A Summary of Cloud Simulators Comparison

Finally, based on our analysis offered in this chapter and based on multiple factors including scalability, main features, environments, and previous experiments, we have provided a general recommendation that would help the reader directly choose the most suitable simulator(s) to start from based on the requirements or purpose. These are summarized in Fig. 4.5.

However, looking at the nature of the topic and vast list of variations between every cloud related problem and the other, this recommendation should be looked at as a general guideline to the reader instead of being a definite (right or wrong) fact. It should not be

Table 4.11: A comparison of the cloud simulator scalability using solved problem sizes

Simulator	**Number of Nodes**	**Experiment Parameters**	**Execution Time**
iCanCloud	5,000 VMs	250,000 jobs, input size is 5 MB, output size is 30 MB, processing length is 1,200,000 MI, VMs computing capacity up to 9,500 MIPS	10,000 seconds (2 hours and 45 minutes)
NetworkCloudSim	4 servers, 8 VMs and up to 8 processes per VM	Each host has 2 Xen VMs, VM has 2 cores, 1.5 GB RAM, Hosts have 100 Mbps switch, Communication messages up to 1,000,000 MPI elements, arrival rate of 200 requests per second	2.5 seconds
CloudAnalyst	Up to 3 data centers containing up to 75 VMs	Total number of users ranges from 108 to 1,080, users make a request every 5 simulation minutes and each 100 request are gathered together, VM size is 100 MB, VMs have 1 GB of RAM, VMs have 10 Mbps of bandwidth, Servers have 2 GB of RAM and 100 GB of storage, Each machine has 4 CPUs, Each CPU has a capacity power of 10,000 MIPS, Each user request requires executing 25,000 instructions. Simulated period is one day	Overall average response time for a combined request 121.07 milliseconds (A total of 2 minutes for all requests)
GroudSim	On a number of cloud instances from 8 to 32,000.	Jobs ranging from 5000 to 32,000, 1,000 MI per second (MIPS) for each CPU, 100,000 jobs, with minimal size	16 seconds for 1,000,000 jobs of minimal size, 50 seconds per job consisting of 3,500 activities (of fixed MI), 0.2-0.3 seconds per job for jobs consisting of 200 activities (of fixed MI)
CDOSim	A data center ranging in sizes between 4 and 7 nodes .	Problem representation is slightly different, uses a workload intensity function that originates from a service provider producing 5,000 requests (method calls) per minute	Around a day and a half in total

Table 4.12: A comparison of the cloud simulator scalability using solved problem sizes

Simulator	**Number of Nodes**	**Experiment Parameters**	**Execution Time**
MDCSim	Up to 128 nodes in 3 tiers, communicating clients up to 6,400 clients	Major components include clients, Web Server (WS) tier, Application Server (AS) tier and DB tier. Round-robin web switch emulation to distribute client requests to the web server tier nodes, message size 5 kb	Depending on the scenario the latency reaches around 0.9 seconds per client amounting to an experiment time of around an hour and a half
GDCSim	2 rows of 5 industry standard 42U racks in each row, racks consisted of 5 7U chassis each	GDCSim's main focus is on the structural side of the data center in terms of the heat dustribution and other thermal factors not the task/job scheduling side. Therefore, data centers tested on GDCSim tend to be smaller.	9,000 seconds experiment time (2 hours 30 minutes)
SPECI	100,000 nodes	Simple message exchange (state=alive/dead), Heartbeat retrieval events drawn from a uniform distribution with a delay between 0.8 and 1.2 seconds and reschedule themselves with a delay from the same distribution, number of subscriptions= square root of the number of nodes,run for 3600 simulation time seconds, SPECI-2 shows 5.5GB RAM JVM memory footprint when executed for similar experiments	hours
BigHouse	10-10,000 servers	number of simulated events takes from 0-6,000, the experiment varies both processor and memory settings, the performance setting space is 2-dimensional; for each variation the number of maximum queries (request) per second (QPS) is increased on a scale of percentages of a preset maximum load, the measured result is the 95th percentile latency when applying for the specific variation and QPS	10 servers cluster takes less than a minute, 10,000 server data center takes from hours up to a full day based on the scenario

Scenario/Simulator	CloudSim	GreenCloud	iCanCloud	NetworkCloudSim	CloudAnalyst	GroudSim	CDOSim	MDCSim	GDCSim	SPECI	BigHouse	TeachCloud
Simple State Sharing Problem/ High Number Of Nodes (>10k Nodes)	○					○				○	●	○
Energy Efficiency/ Energy-aware Scheduling		○	○									
Energy Efficiency/Cooling									○			
High Availability/ Fault Tolerance	○	○				○						
Network Modeling And Network-aware Scheduling		○		○				○		○		
Workload Planning /Evaluation	○	○			○			○			○	
Workflow Modeling	○	○				○						
Resource Allocation	○	○	○	○	○			○			○	○
Service Brokering	○		○		○		○					
Storage Modeling			○								○	○
GUI/ Easy Use			○		○					○		○
High request/job load (>10k requests)	○	○	○	○	○	○	○	○		○	○	○
MapReduce Application Modeling/Data Replication	○	○									○	○

Figure 4.5: Recommended Cloud Simulators depending on User Priorities

forgotten that the nature of software development process is that with enough time and effort, a capable team can extend any software to include many more features.

4.6 Open research challenges

As we weigh pros and cons of building a new simulator versus using a current one, research challenges, and learned lessons present themselves. We try to summarize these in the following points to serve as a blueprint of issues that need attention or open research topics.

4.6.1 Lose the Grid Perspective

It is common for simulators that were based on previous grid simulators to inherit much functionality. The point here is not inheriting the vision too. The grid model is principally

Table 4.13: A summary of the cloud simulators features: External view

Feature/ parameter	**Underlying toolkit/ platform**	**Programming language**	**Availability**	**Simulation time (estimate)**
CloudSim	SimJava	Java	Open Source	Seconds
GreenCloud	NS2	C++/TCL	Open Source	Minutes /tens of minutes
iCanCloud	OMNET, MPI	C++	Open Source	Seconds
MDCsim	CSIM	C++/Java	Commercial	Seconds
NetworkCloudSim	CloudSim	Java	Open source	Seconds
CloudAnalyst	CloudSim	Java	Open source	Seconds
GroudSim	SimEngine	Java	Academic	Seconds
TeachCloud	CloudSim	Java	Open source	Seconds
CDOSim	CloudSim	Java	Academic	Minutes
GDCSim	BlueSim	Java/XML	Academic	Minutes
SPECI	SimKit	Java	Open source	Minutes /tens of minutes
BigHouse	Mac/ Linux/ Windows	Java/Python	Open source	Minutes

different than the cloud. The job submission and waiting model does not apply to the cloud. The cloud has more of an interactive environment of leased machines. Simulators based on the grid should be revised to adopt new packaging that is based on unified commercialized packages of PC-like resource components. This is more accurate to the cloud client expectation than the distributed cluster/super computer assumptions of grid clients. This will generally affect the way resources are modeled, service packaging and pricing, user data modeling, and resource scheduling and allocation in the cloud simulator. The user experience in the cloud is more engaging; reliability and availability is a concern on the scale of a millisecond.

4.6.2 More Work towards Realistic User Application Patterns

Although multiple current simulators allow simulator users to test using any data sets they require, that does not seem to be enough. More work is required towards designing the user application models and data sets. We have seen efforts discussing the types of applications user request data specify in terms of being communication intensive or computationally in-

Table 4.14: A summary of the cloud simulators features:Strengths and limitations

Simulator / Parameter	**Focus/ strength**	**Limitations**
CloudSim	Most commonly used (many extension projects) Supports large scale problems Inclusive support for DCs, VMs and resource provisioning techniques.	Communication model on the packet level is not supported directly
GreenCloud	Focus on power management and energy consumption techniques testing Packet level communication supported	Runtime Detailed brokering model
iCanCloud	Trade-off between costs and performance. Full GUI Parallel execution of 1 experiment over several machines	Cost policies focus is on pay-as-you-go policies No focus on energy consumption No full network model
MDCsim	General purpose functionalities available	Commercial No full network model
NetworkCloud - Sim	Extension of CloudSim Extra communication features like message passing Full application model	No networking on packet level. Less focus on power/cost models Issues of CloudSim apply
CloudAnalyst	Has a full GUI Focus on geographical factors	Focus on Specific Purpose. No full network model
GroudSim	Works for Grid and cloud systems. Works for large scale problems (high number of requirements) General features available	Basic Network functionalities Basic power consumption optimization functionalities
TeachCloud	For Educational purposes. Simple to use GUI	Basic features Only for academic purposes
CDOSim	Focus on Deployment Options Extension of CloudSim	Unproven for general purposes and on a large scale
GDCSim	Focus on low level power saving methods and cooling	Unproven for general purposes and on large scales
SPECI	Focuses on component status update exchange and middleware policies	Does not cover computational resource reservation or variety of cloud applications, focused on the number of failures in a DC
BigHouse	showcases queuing models, parallel performance , result accuracy can be traded off with runtime	offered network model is limited

Table 4.15: Topics and problems cloud simulators were used to tackle

Cloud Simulator	**Topic/Problem**
CloudSim	Cloud workflow preparation and execution, Allocation based on Mixed-Integer Programming, VM dynamic performance change, Auction-based Services, Cloud provider migration planning, Web session modeling, MapReduce modeling simulation [75], Fault Tolerance [76]
GreenCloud	Power efficiency, Communication models and architecture evaluation[77, 78, 79, 80, 81, 82, 83], Data replication solutions [84], Network-As-A - Service (NaaS) implementation [85], Scheduling for opportunistic grids [86], High availability in the cloud [41]
iCanCloud	Efficient service brokering [87], Storage modeling [88], Live migration [89], Energy efficiency [90]
NetworkCloudSim	Cloud network modeling [91], Cloud resource allocation [92, 93],
CloudAnalyst	Service brokering [94, 95] , Load balancing polices [96, 97]
GroudSim	Efficient scheduling of scientific workflows [98, 99],Fault tolerant execution of scientific workflows [100]
CDOSim	Cloud deployment options [101] , Cloud provider migration [102]
MDCsim	Cloud resource allocation [103], Malicious activity detection through predictive modeling [104] ,
GDCSim	Energy efficiency and cooling inside a data center [71]
SPECI	Component subscription networks monitoring [105, 106], Hierarchical cloud network modeling [107],
BigHouse	Cloud workload modeling and planning [108, 109, 110] , Resource allocation [108, 110], MapReduce modeling simulation [111]
TeachCloud	Resource allocation [112, 113], MapReduce modeling simulation[114]

tensive. However, there is certainly a demand for a detailed application stack that represents interacting software components. The generic data exchange request or computational request does not represent the dynamic process that is a full user application.

For example, the four indicators of Big Data: Volume, Velocity, Variety and Veracity [19] have a big say in specifying the final form of the client application. The impact will not stop at the real client data though; it will spread to affect simulated use cases, request size, percentage of error and application components distribution. The simulator designers will have to put into consideration more complex scenarios where the degree of interdependency will reach a new level. Security considerations under Big data conditions are another topic that will influence this process heavily.

4.6.3 Cloud Deployment and Pricing

Deployment options and pricing packages are covered with the current set of simulators. Brokering is offered through CloudSim. The primary objective there is choosing the provider that satisfies client's resource request and service conditions with minimal price. Assuming a client can move between providers freely is not realistic in this case. Provider lock-in is a well documented issue that can cause client reluctance. The complications of moving between providers on the technical side (delay, portability, etc) and business side (release clauses, notice time constraints, price difference) should be considered in an ideal cloud simulator.

4.6.4 Deeper Look at Reliability and High Availability

Every hardware or software component is bound to fail sometime. Minimizing this time is a must in an environment where the 5 nines guarantee (99.999% of the time availability) is becoming a precondition sooner rather than later. This gains more importance as cloud providers aim at attracting telecommunication carriers (or when telecommunication carriers are themselves cloud providers). A deeper, more structured look at simulating component failure, repair, recovery and their potential effect on the applications/services is required. Providers will prefer any simulator with the opportunity to simulate and test redundancy policies as well. This does not only mean storage components, but also servers, racks,

VMs, and software components. Moreover, availability of network components should be included as part of a more inclusive simulation scenario.

4.6.5 Widen your Horizons: Outside the Data Center

It is noticed that most of cloud simulators do not consider the source of the cloud client data outside the data center. This is due to the fact that cloud clients access data center through the Internet. However, with the diverse set of connections seen in a cloud scenario, it would be interesting to study the effect of the connection method. Will the client connect from their private clouds? From a hand held device connected to a base station? From a PC? The client geographic distribution might have an effect as well. Enabling the simulator users to implement this sort of layout will add great value. The source of energy could have an effect on the consumption evaluation. It would be interesting to know the effects of alternating between sources of energy that differ in their "Greenness" as the amount of energy allowed to be used can change also.

4.6.6 Exploiting High Performance Computing features

Building a new simulator gives the chance to investigate HPC technology features. HPC offers robust and scalable high computing power methods including using a hybrid platform of CPUs/GPUs and modular design concepts. HPC technologies are not modeled in any of the aforementioned cloud simulators despite some of them supporting distributed architecture.

As seen in the previous sections, most of the simulators model the computing power either as:

- Reserved/not reserved processing unit (termed time shared)
- A million instruction per seconds (MIPs) commodity resource (termed space shared)

In the second method, requests or VMs reserve some of this capacity. This is understandable because it is easier to treat a resource as a simple integer.

However, a more sophisticated model of Processing units would be beneficial. It would be interesting to have a simulator that models a cloud facility that offers HPC capabilities to clients through the cloud. The major impact of implementing HPC in the cloud

Table 4.16: Research challenges to be tackled by future cloud simulators

Research challenge	**Areas most included/affected**
Lose the grid perspective	Resource modeling, service packaging, pricing, user data modeling, resource scheduling, reliability
More work towards realistic user application patterns	Application models, user data modeling, data exchange design, network model
Cloud deployment and pricing	Portability, brokering, user geographical distribution, cloud dynamic pricing, user SLAs
Deeper look at reliability and high availability	Component failure,repair and recovery, Redundancy, interdependency, user SLAs
Widen your horizons: outside the data center	Network model, user geographic distribution, power consumption, power sources, security
Exploiting High Performance Computing features	Data center topology, resource modeling, service pricing, user request modeling
Modeling Cloud Storage	Redundancy, storage distribution, resource modeling
Modeling Containers along with VMs	Resource modeling, performance monitoring

would come from the flexibility offered by virtualization. A cloud setup offers the ability to customize the virtual machine as per the scientists' specific needs offered by a cloud setting compared to the strictly-preserved system software in traditional HPC offerings. In the cloud simulator, the ability to model these differences would assist HPC system designers or supercomputer designers in tackling challenges like pricing, scheduling model changes from their traditional grid-like model (submit a job and wait) and other traditional cloud challenges customized to the specific environment of supercomputing (security, portability, etc).

[115] surveys some of the previous efforts tackling the challenges of coupling HPC and the cloud. Studying and solving these challenges would be assisted by a simulation tool that precisely represents HPC resources and typical applications. This should include modeling features of the application like the percentage of code that can be executed in a parallel setting.

4.6.7 Modeling Cloud Storage

Cloud storage in general has not gained enough attention by the surveyed simulators. In GreenCloud, for example, storage is considered to have fixed capacity for a resource

provider (host, or VM) and a fixed demand by the task (just a number). ”File size” is considered in CloudSim, while some simulators do not consider storage whatsoever. iCanCloud has a more detailed storage modeling among the current simulators as it supports models for local or remote storage systems as well as parallel storage and RAID designs. Missing a full network representation makes the model incomplete in this case. Cloud storage modeling should be included in a cloud simulator at large. This would cover the storage components distribution, redundancy and direct and indirect connection between computational resources like hosts and storage resources. User data storage is a critical part of the application performance and a solution cannot be evaluated without considering data locations and connectivity. The impact on pricing is also an area that is continuously drawing the attention of cloud architects and therefore should be included in a cloud simulator feature list. In addition, modeling the specific database solutions implemented in the cloud would be a valuable addition that would help understand solution performance.

4.6.8 Modeling Containers along with VMs

Containers are an abstraction performed at the operating system (OS) level that promises more efficient use of the hardware resources over VMs. Containers are gaining traction in the cloud providers' circles as a replacement for VMs either fully or for a specific set of applications and use cases. When using containers, the user space gets abstracted instead of the whole hardware stack like in the VM case. [116][117][118] The overhead produced by running the whole operating system is not incurred every time when using multiple containers so this saves on memory and CPU compared to VMs running the same workloads. Doubts over security and efficient management are still there for containers and a balance of when to use them vs. when to use VMs is still materializing. However, none of the currently available cloud simulators offer the option to model applications inside containers covering the resource allocation options along with performance comparison between both solutions.

4.7 Chapter Summary

Cloud providers are under constant pressure to deliver highly reliable and continuously inventive service. The cutting edge in this market will come from better performance metric values or newly added services that clients cannot find somewhere else. This requires a strong cloud simulation comprehensive solution. This cloud simulator would perform roles that range from defining the problems to pinpointing bottlenecks and from evaluating policies to testing the solution endurance and scalability. The process of building a simulator includes multiple design decisions and requires specifying the shape of many simulator ingredients. We have discussed these ingredients and introduced our vision for the simulator design framework. We then traversed a select set of common cloud simulators, stressing the main features and limitations within simulator environments.

Open research challenges and areas/topics that are in need of attention from the scientific community were compiled. Covering all these topics along with any new challenges related to security, and power consumption will enhance the cloud simulator position as a critical tool for cloud providers. As this area keeps growing, a cloud simulator with proven efficiency and effectiveness will continue to draw the attention both of the industry and academic parties.

Chapter 5
Simulating High Availability Scenarios in Cloud Data Centers : A Closer Look

5.1 Introduction

With the increased migration of business applications to the cloud, more familiar challenges related to service performance are arising. The Cloud tenants require certain levels of performance in aspects like high availability, pricing options and general reliability. On the other hand, Cloud providers strive to satisfy their clients' demands according, first, to their resource requirements and, second, to their service quality requirements. Cloud providers are in constant pursuit of the holy grail of cloud management systems. This translates to a system that satisfies client demands for resources, maximizes availability, minimizes power consumption and, in turn, minimizes cloud providers' cost.

A main challenge cloud providers face to achieve these goals is ensuring high availability (HA). Client are becoming more demanding in that aspect and the 5 nines requirement (guaranteeing that the data center service is available for 99.999% of the time) is becoming a reality. High availability includes the combined reliability for components of all categories including hardware and software components. This covers Network and processing resource sides. It also includes the availability of components of all layers (cloud, racks, servers, VMs, applications, application components).

HA received major attention as soon as the cloud solutions started being deployed. In [119] for example, we notice a solution depending on RAID-technology used to accomplish

---

The contents of this chapter have been published in [42]

the task of managing data across multiple cloud storage providers. Factors like geographic location, quality of service, providers' reputation, and budget preferences are taken into consideration. Early efforts up to 2012 are summarized in [120] which focused on HA techniques based on multi-core processing, virtualization, and distributed storage. In more recent work, the authors of [121] presented the HA constraints within the context of a multiple objective resource scheduling problem in the cloud environment. Their algorithm was tested on a real life social news application with synthetically generated costs and loads. Another variation is the one seen in [122], where a failover strategy is presented. The tested technique combines load balancing algorithms with multilevel checkpointing so as to decrease checkpointing overheads. In their paper published in 2014, the authors of [123] offer an architecture for automatic failover between multiple Platform-as-a-Service (PaaS) cloud providers.

Virtualized storage and redundancy are of prime interest as well. In addition, other works can be found discussing experiments on VM migration analysis, including security related aspects [124] and performance analysis of migration algorithms [125]. Finally, in [126], a solution is provided to ensure HA in cloud storage by decreasing virtual machine reboot time.

Cloud simulators play a critical role while developing the optimal cloud data center management System. A cloud simulator offers an environment to implement scheduling policies and operation scenarios with clear cost advantage and decreased risk. A cloud simulator serves as the first barrier that can examine resource allocation algorithms, energy efficiency techniques and HA-aware scheduling algorithms. A comprehensive survey and feature comparison of the available cloud simulators can be found in [41].

The most popular simulators tried to cover cloud functionality in a generic view that covers components, processing components and data center management as can be seen in efforts like CloudSim (along with the extensions) [2][47], GreenCloud [3] and MDCsim [70]. With all the efforts to propose HA solutions for the different elements composing the cloud, we still lack a Cloud simulator that can simulate the behavior of multi-tiered applications, while considering the different failures that can occur in the Cloud, and quantify their impact on the applications availability.

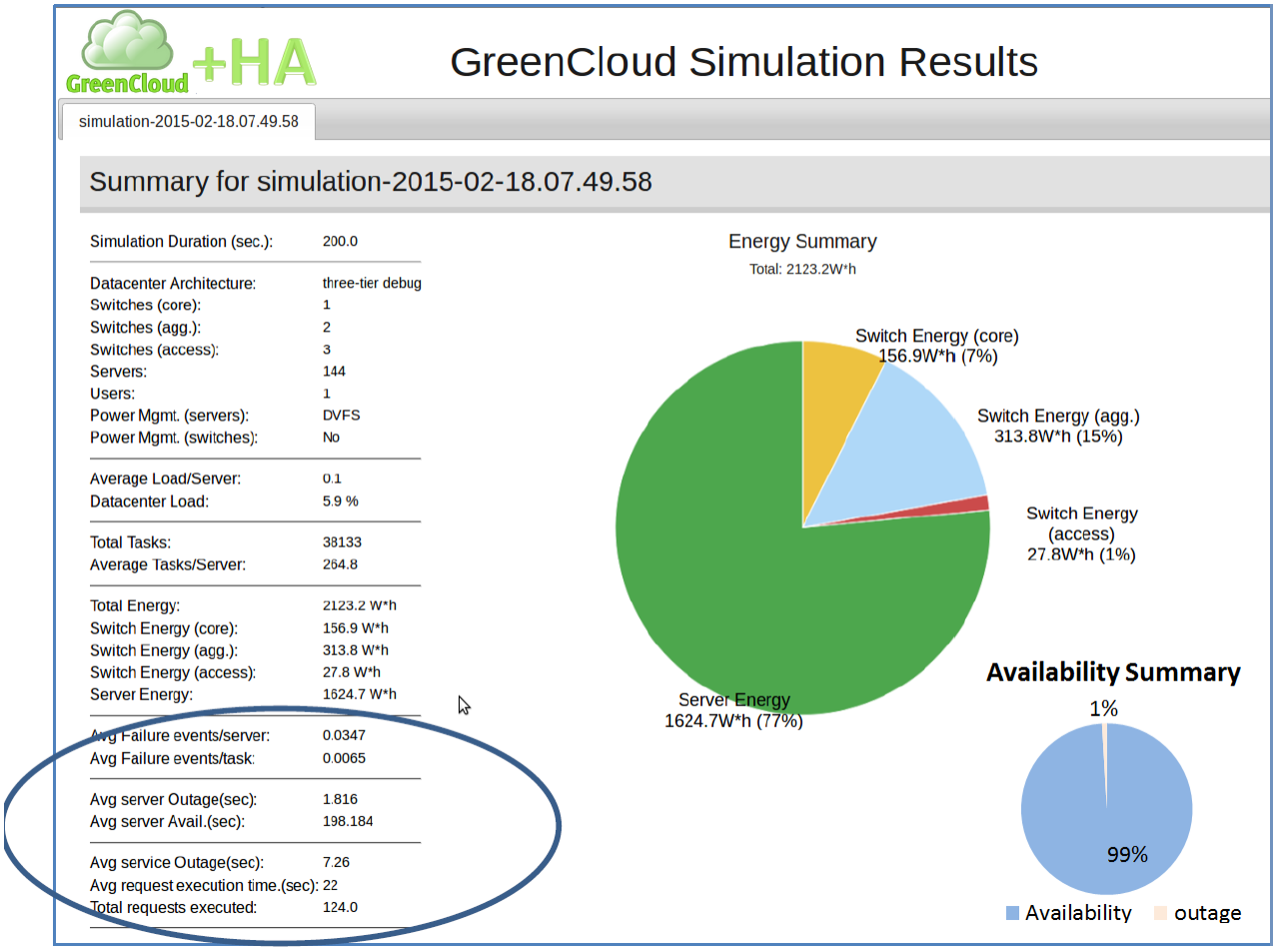

Figure 5.1: GreenCloud output

Therefore, there is a need for a cloud simulator that enables high availability algorithm testing in order to reach a HA scheduling technique that does not sacrifice energy efficiency. To the best of our knowledge, there is not a simulator that provides the detailed functionality that enables measuring HA metrics, testing HA algorithms and producing the results in a way that can serve academia and the industry.

In this work, we introduce a framework to amend cloud simulators with HA features. We take GreenCloud as an example of a major simulator with a direct focus on green computing and implement these features as an additional measurement layer. This is illustrated using the specifications of a phased communication application (abbreviated henceforth as PCA).

Section II introduces a framework to enhance the cloud simulator of choice with required features in order to turn it into an HA-aware simulator. Section III presents the enhanced GreenCloud architecture and added features. Section IV explains the experimental setup. Section V shows some of the testing results achieved after enhancing GreenCloud with HA-awareness and then we present conclusions in section VI.

5.2 A framework to implement HA-awareness in cloud simulators

The notion of guaranteeing a certain standard of high availability has a direct effect on performance and specially energy consumption in a cloud environment. Implementing redundancy which is a prerequisite to any highly available algorithm causes the amount of resources required to serve a certain load to increase linearly. The increase factor in this case will depend on the number of redundant hardware components. In addition, the activities required to implement redundancy require computational and network resources just like user requests. This will, in turn, increase the demand for data center resources and naturally increase energy consumption. Hence, a careful consideration of the effect HA algorithms have on the energy efficiency in the cloud is needed. This will assist cloud providers in making the decision as to which HA algorithm achieves the best trade-off for them in terms of availability and energy efficiency.

To enable any cloud simulator to measure availability and evaluate different HA algorithms in a cloud data center, it has to include the following features.

1. HA Features
   (a) Component Failure

   Components failure simulation means that simulator users would be able to inject failure events based on any specific time series or distribution that suits their input. Components covered should ideally include all GreenCloud components. This includes servers, racks, virtual machines, switches (all types) and other network components. This includes the case of the whole data center failing in case of a major power problem for example. Another issue here is the domino effect of failure. If a host fails, all VMs on it and tasks scheduled on them would have to suffer failure too. All these resource consumers (as termed in GreenCloud) would be rescheduled on other servers(resource providers).

   (b) Component Recovery

   In a real data center, failed components naturally come back to a running state after a period of time.

   (c) Synchronization Redundancy and Server Groups

   Based on the roles they can perform, servers are divided into groups . A task in phase I is only scheduled on a server in the corresponding(matching) group. As shown in Fig. 5.2 and the flowchart in Fig. 5.3, as the task in the first phase (for instance, PCA client phase) is executed, the following task is added to the scheduling queue matching the next phase(PCA). When that phase is over, the following task is added to the following queue(IMS-CSCF) until the last phase. That simulator also guarantees that if a task in the IMS-CSCF phase fails for any reason(server failure or VM failure for example), the task will only be scheduled on a server from the same server group(IMS-CSCF server group).

2. Workload Modeling and Scheduling
   (a) Higher Granularity Modeling of Application
      i. Adding User-defined Resources

A task as defined in GreenCloud includes the attributes seen in Table 5.1. The ability to add extra types of resources means more shades of problems can be represented. We can represent scenarios in which specific servers have resources other servers do not. For example, some servers would be able to offer DB service for a specific number of requests per second. This can be helpful in simulating scenarios where each server group performs a predefined role.

ii. Defining Tasks with User-defined Resource Requirements
Emphasizing on the previous feature, task resource specification has to be amended with the amount the task would request of the new resource (I.O percentage or number of HTTP requests the task contains for example).

iii. Defining Applications with Diverse Set of Tasks Representing Execution in Dynamically-defined Phases
A request in this case moves from being just a simple task disjoint from all other tasks and defined by only its computational resource requirements and data exchange requirements to something more inclusive. A request consists of a set of tagged tasks that have functional dependency between them. The first task would represent the first phase in implementing the complete request. The request would be fully executed when the task representing the last phase gets executed. As seen in the figure , the request consists of tasks being processed at the App front end component, app core , app back end then the app core again and the front end component once more before eventually sending the results to the client. Each one of these components could be scheduled on a distant VM on a separate server along the data center or event in other data centers.

(b) Functional Dependency and Dynamic Scheduling of Phases (based on resource requests)
Once the request is constructed of a set of tasks that differ in resource requirements, the scheduler is supposed to take that into consideration while scheduling. A task will only be scheduled on a server with the sufficient resources and also the server with the suitable functionality. Tasks in phase i

Table 5.1: tasks in GreenCloud- Defining attributes

Variable name	**Attribute**
task(size)	Input data to be sent to host the task is scheduled on
task(memory)	[Byte] of used RAM
task(storage)	[Byte] of disk space
task(duration)	computing deadline of tasks in seconds(can be set as parameter
task(duration)	computing deadline in seconds
task(outputsize)	standard- Size of output on task completion
task(outputsize)	Size of output on task completion
task(outputsize)	low comm- Size of output on task completion
task(intercom)	Size of inter-task communication

will not be processed or scheduled before tasks in phase i-1 are completed.

3. Monitoring and Reporting

   (a) Reporting Availability Status The measurements shown in figure 1 are something to start with. The metrics included are:

   - Failure cases/component (server, VM, etc)
   - Failure events/task
   - Total outage time and percentage.

   The availability of a complex component like a data center can be calculated factoring in the availability of its subcomponents (server, racks, switches , etc).

   (b) Reporting Service Total Exec Time & Outage Time
   The average completion time of a full request is shown in Fig. 5.1. This means the total time to complete all of its tasks (phases).

5.3 Enhancing GreenCloud with HA features

5.3.1 GreenCloud scheduling environment

As one of the most detailed cloud simulators available, Greencloud arises as a powerful tool to implement the proposed functionalities. GreenCloud was developed as simulator

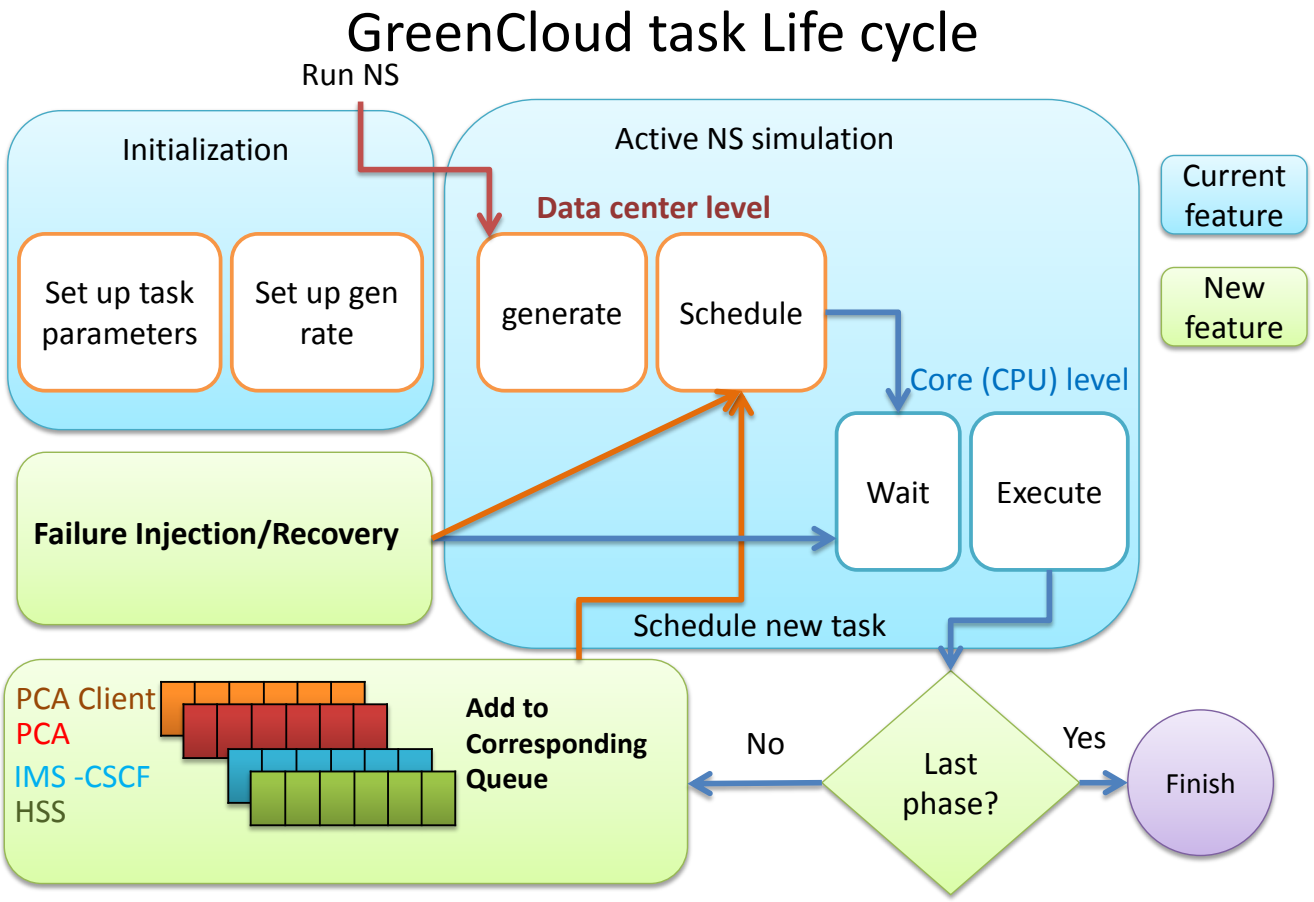

Figure 5.2: The enhanced task life cycle in GreenCloud

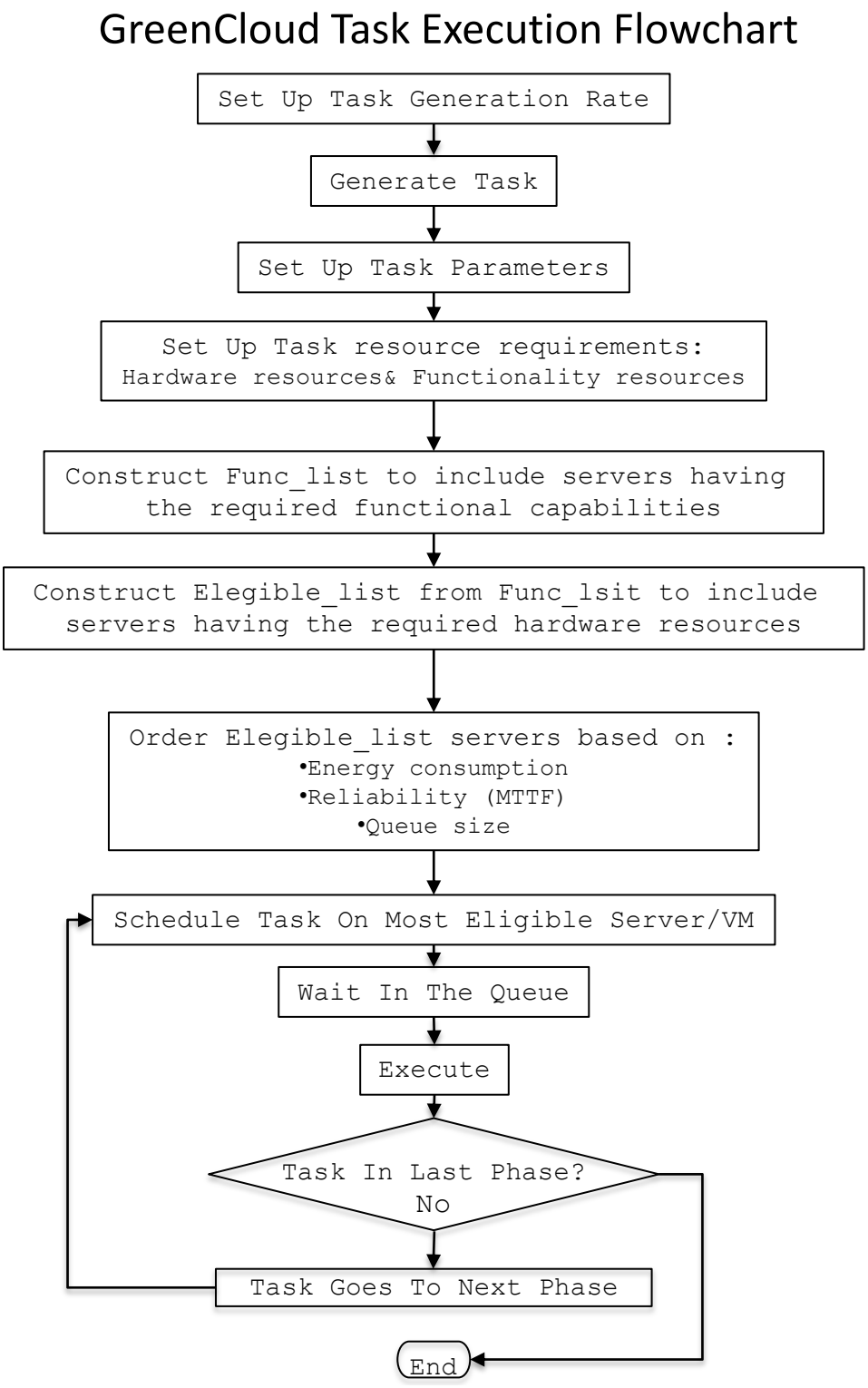

Figure 5.3: GreenCloud task execution flowchart

with a focus on energy efficiency and fine grained networking capabilities. The prime purpose cited for building GreenCloud is mitigating overprovision issues [3]. Overprovision happens in a data center due to the loads constantly changing on the computational and communication resources. The average load can be as low as 30% of the data center server and network capacity [3]. This, in turn, causes the data center to systematically use more power than the optimal value.

GreenCloud offers simulation capabilities including multiple topology choices (2 layers and 3 layers) and it offers communication through packets using the underlying NS-2 simulator features. GreenCloud also offers the choice of scheduling tasks(user requests) on hosts directly or on virtual machines which reside on hosts.

Tasks are modeled as unit requests that contain resource specification in the form of computational resource requirements (MIPs, memory and storage) in addition to data exchange requirements (task size variable representing the process files to be sent to the host the task scheduled on before execution, data sent to other servers during execution and output data sent after execution).

5.3.2 GreenCloud Scalability parameters

How can we measure GreenCloud scalability and solution scalability using GreenCloud? scalability of the GreenCloud input scenario is controlled using the following 4 parameters.

1. Data center Load which corresponds to the number of generated tasks. This can be seen in the following equation.

$$GenerationRate = TotalDCentermipsCapacity/Task(mips) * DataCenterTargetLoad \tag{5.1}$$

2. The data center chosen topology which decides the number of servers and network layers.
3. Task Specification: The resource requirements of a task can make it take longer to execute. the task size also affect the task scheduling in terms of the host availability.
4. Simulated time: The duration of the experiment is a critical factor too.

5.3.3 HA enhancements made to GreenCloud

We have implemented the HA features discussed in the previous section in GreenCloud. As illustrated in Fig. 5.1, failure injection feature was added to GreenCloud. Failures and recovery can be injected at preset times or taken from a file. Related recovery procedures were also added. Corresponding service and outage metrics are also available in the enhanced GreenCloud as seen in the figure. Functional dependency and synchronization dependency were implemented including implementing task tagging, server grouping and the related service metrics. Major changes to the GreenCloud Scheduler are illustrated in the flowchart in Fig. 5.3. For a start, we distinguish between hardware resource like RAM, CPU, disk, bandwidth and Functionality resource like the HTTP capabilities, the availability of DB implementation on the server and so on.,

The scheduler starts by filtering out the list of the all servers, leaving only those that do include the functionality recourse producing a Func_list. Next, the servers that do not satisfy the hardware resources from the Func_list are filtered out and hence an Eligible_list is produced. This list is ordered based on a combination of consumption of energy, reliability (MTTF) and queue size. The task is then scheduled on the most eligible server. This is done in every phase the task is in until the whole request is served.

In the following paragraphs, the scenario used to test these features is illustrated.

5.3.4 Phased Communication application(PCA) Scenario

The scenario we chose to evaluate the framework through is a case where the cloud client uses leased cloud resources to simulate the implementation of Phased Communication application(PCA). This application makes communication services accessible to developers using technologies like HTTP and WebRTC. Without loss of generality, this scenario is used as it demonstrates functional dependency, synchronization dependency and the tagged tasks phases.

5.3.4.1 Registration

In this scenario, a Web Access component allows HTTP-based clients to connect to an IMS network. The PCA receives HTTP/REST-based requests from a web client and sends

these requests to the IMS network. The PCA receives requests from the IMS network and sends them as events over an event channel to the web client. these requests allow clients to register to the IMS Core Network. The resource requirements for requests in this phase are as follows. (i) CPU: 1200 registrations per second where each registrations consumes 2 MIPS; (ii) Memory: 1 active registration, uses 13905 bytes of RAM for the duration of the registration; (iii) Network: 1 HTTP POST per registration (TCP); 2 SIP Register per registration (UDP); (iv) Duration of the registration is assumed to be 24 hours.

Once a registration is successful, PCA will automatically re-register the request at regular interval 3600 seconds (default).Re-registration consumes very little RAM.The resource requirements for requests in this phase are as follows. (i)CPU: 2400 re-registrations, each re-registration uses 1.625 MIP (ii)Network: 2 SIP Register per registration (UDP) This scenario is illustrated in the sequence diagram in Fig. 4.

5.3.4.2 Audio-Video Calls

A registered PCA Web Client can initiate an audio/video call to a remote user and the call is accepted/started. The resource requirements for requests in this phase are as follows. (i) CPU: 1000 calls per second where each call consumes 2.4 MIPS; (ii) Memory: 1 active call, uses 200210 bytes of RAM for the duration of the call; (iii) Network:2 HTTP POST per call (TCP); 2 HTTP GET;1 SIP INVITE (UDP);1 SIP ACK (UDP); 1 SIP BYE (UDP); (iv) Call Mean Hold Time is assumed to be 180 seconds. This scenario is illustrated in the sequence diagram in Fig. 5.

When implementing this scenario in the enhanced GreenCloud, phases of the full request can be defined by determining the steps where messages are exchanged and processing is required by the PCA and the components interacting with it.

For example, in the registration scenario, a request would go through the following phases and each phase would yield a task to be scheduled and served.

$PCA - App \rightarrow PCA \rightarrow IMS - CoreNet.$

In the next section, initial testing results of the HA enhanced GreenCloud are presented.

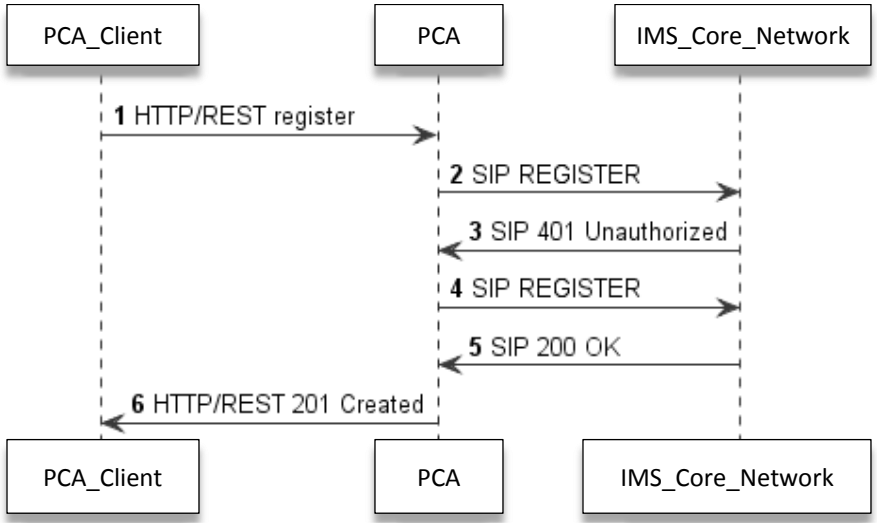

Figure 5.4: Sequence diagram of the PCA scenario-Register

Table 5.2: Simulation Parameters

Parameter	**Value/calculation Method**
Server count	tested for 144 and 1536 servers
Topology	3-tier topology
CPU configuration	HP ProLiant BL460c Gen8 Server Blade
Problem size	10 up to 100 requests/second
Average server outage time	$\sum_{i=1}^{n}$(average outage for servers in $group_i$)
Average service completion time	$\sum_{i=1}^{n}$(average execution time for tasks in $phase_i$)
Total simulated time	up to 3000 time units(seconds)
Average Experiment time	1-2 hours

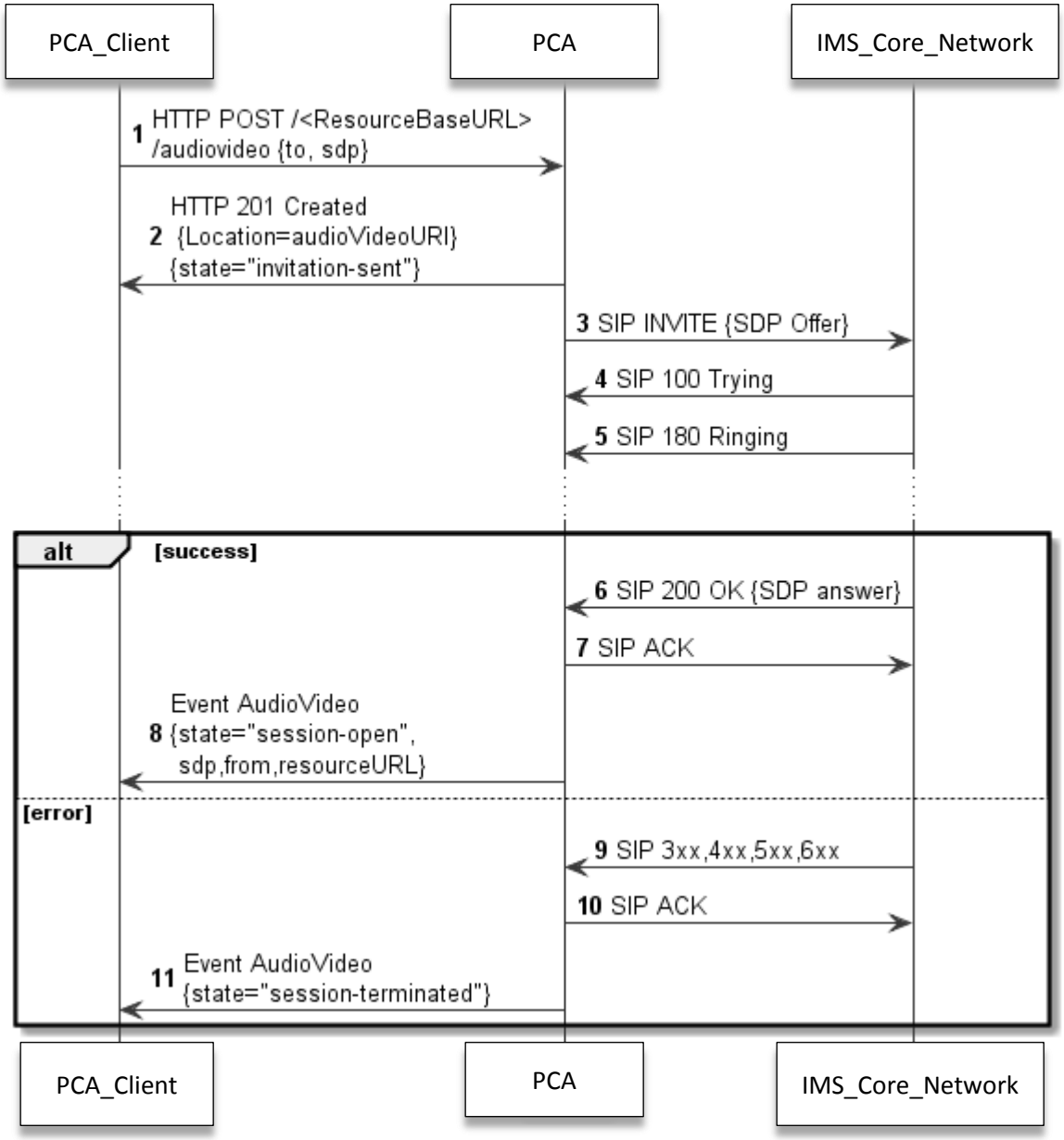

Figure 5.5: Sequence diagram of the PCA scenario-Audio-Video Call

5.4 Experimental Results

Major simulation parameters used include typical data center configuration parameters like the ones in Table 5.2. server count, topology and server resource configuration all go under that category. Simulator and input configuration parameters like input size, request resource specifications and total simulated time are also important here. In addition, the methods of calculating average service time and average outage time are also documented in Table 5.3.

As mentioned in the previous sections, we used the enhanced GreenCloud to test the implemented HA features and in order to implement PCA scenario which spans all these features. Moreover, we show the effect of some of the parameters on major metrics like the request average response time.

First, Table 5.3 shows some of the results for testing the PCA scenario while changing the problem size (represented by the number of request/second). The obvious trend here is an increase in the average response time as the problem size (arrival rate) grows. We used minimal number fo servers in order to stress the system as much as possible in this scenario.

Table 5.4 shows the impact of increasing the dequeuing rate of the phased request. As the request finished the first phase it goes into the scheduling queue to be executed for the following phase. this queue is dequeued with the rate shown in the table. As the rate grows the average execution time (response) for a request decreases of up to 37%. It keeps decreasing until a limit where it stabilizes and then starts increasing the average response time. This happens because the dequeuing process adds no more value and just causes an overhead for the simulator. the simulator runs the dequeuing process while the queue is empty.

Table 5.5 show the effect of changing the server group sizes on the average response time of a request. To show that and also to show that this scenario works for large data center, we tested the same set of features for a large data center of 1536 (the three-tier topology configuration in GreenCloud). A gain of up to 17% in the average response time can be reached by controlling the server group sizes even for the very small load of 10 requests per second. The lesson to be learned here is that the parameter configuration in terms of server group distribution and dequeuing rate can be have positive impact if they

Table 5.3: pca scenario simulation results- impact of increasing the arrival rate on average service time

Dequeuing Rate	**Total servers**	**Group 1 size**	**Group 2 size**	**Requests per sec**	**Average service time**
1 call/sec	2	1	1	10	0.36190456
1 call/sec	2	1	1	50	0.74810514
1 call/sec	2	1	1	100	1.24199910

Table 5.4: PCA scenario simulation results-impact of the dequeuing rate on the average service time

Dequeuing Rate	**Total servers**	**Group 1 size**	**Group 2 size**	**Requests per sec**	**Average service time**
1 call/sec	140	70	70	100	1.22678165
4 calls/sec	140	70	70	100	0.84924532
1000 calls/sec	140	70	70	100	0.77226364
1 call/sec	2	1	1	100	1.24199910
4 calls/sec	2	1	1	100	0.89261223
1000 calls/sec	2	1	1	100	0.79194872

were adjusted to the exact demands of the input request set. A closer look at the resource requirements and life time of each phase of the request has the potential to produce sizable gains in response time and power consumption. An effect on the power consumption metric was recorded as well. For the large data center setup (1500 servers), A gap of 14.5 kW*h was recorded between the two server group configurations used. This gap could grow as the load starts increasing to meet the data center capacity. The lesson to be learned here is that the parameter configuration in terms of server group distribution and dequeuing rate can be have positive impact if they were adjusted to the exact demands of the input request set. A closer look at the resource requirements and life time of each phase of the request has the potential to produce sizable gains in response time and power consumption.

Table 5.5: pca scenario simulation results-large data center

Group 1 size	**Group 2 size**	**Average service time**	**Power consumption**
750	750	0.26823299	17111.9 kW*h
1500	1	0.31653402	17095.4 kW*h

Table 5.6: sample research questions that can be investigated using the extended tools

Question	Corresponding feature in GreenCloud
A message shall be routed through the WCG system in $<$ 50ms in 85% of time when its average CPU $<=$ 75%	Service completion time
max processing delay that a HTTPS message should incur traversing the WCG shall be $<=$100 ms.	Average service time (or phase time)
support 99.999% availability	Outage/available percentage
handle up to 6M subscribers.	number of tasks handled
Single fault should not completely stop a system	Implementing dependencies
Should automatically reject calls/tasks in order to avoid an overload	Tasks do not get scheduled in case of Overload
Node shall generate a response to a service request within 750 ms in 95% of the cases	Time for a task to be scheduled?
At an offered load of 150%, there shall be at least a throughput of 85%	Load is measured & Number of successful tasks is measured
At an offered load of $>$ 150%, the throughput shall show a graceful decrease.	data center crashes can be measured
Reboot Time 150 sec	It can be measured

5.5 Chapter Summary

The cloud computing environment current state of affairs imposes the need for cloud solutions that enables high availability capabilities without sacrificing energy efficiency. To address this need, researchers need resilient comprehensive algorithms as well as sufficient tools that enables them to evaluate new techniques. We introduced a framework to amend GreenCloud cloud simulator with HA features. The sides covered were HA features, workload modeling and scheduling features,and reporting/monitoring. This was illustrated using the specifications of the PCA example.

After implementing the PCA scenario for different topologies and data center configurations, the results show that gains in the average response time can be achieved by controlling parameters including the server group distributions and dequeuing rate. This includes adjusting the scheduling decisions and the parameter configuration according to the demands of the input request set. A closer look at the resource requirements and life time of each phase of the request has the potential to produce sizable gains in response time and power consumption. Finally, a list of examples of real scenarios/questions that can be

answered oe evaluated using the extended tool can be seen in Table 5.6.

Chapter 6
Conclusion

With the transformation of cloud computing technologies from an attractive trend to a business reality for clients and providers, the need is more pressing than ever for efficient cloud service management tools and techniques. As cloud technologies continue to mature, the service model, resource allocation methodologies, energy efficiency models and general service management schemes are far from saturated.

It can be seen that as much as cloud computing as a paradigm redefines the service model and client priorities, cloud computing requires a major shake up on the cloud provider side as well. Performance and service delivery will still depend on the providers' policies/algorithms that affect all operational areas. This ranges from security, application portability to communication efficiency and reaching energy efficiency and resource allocation in a data center.

cloud Data center resource allocation/management represents a critical area in need of urgent attention. Resource allocation has a direct impact on two sides that are at the core of why the cloud is financially effective: resource utilization and energy efficiency. Simply put, the better resource utilization is, the more client requests the data center (cloud) can serve and the better the performance is. As for energy efficiency , it is a pressing issue for cloud providers. Power costs represent between 25% and 40% of the operational expenses of a data center. [8] The carbon footprint of a data center is a political and environmental challenge for cloud providers as well. The objective this thesis aimed for is to tackle a set of the more pressing challenges faced by cloud providers in order to enhance cloud service performance and save on providers' cost. This is done by exploring innovative resource allocation techniques and developing novel tools and methodologies in the context of cloud resource management, power efficiency, high availability and solution evaluation.

6.1 Thesis Summary and Future Work

6.1.1 Resource allocation in a network-aware cloud environment

Chapter 2 starts by offering a comprehensive solution to the resource allocation problem in a network-aware cloud data center. This offers a response to the wind of incoming request to the cloud provider in the form of VM reservation requests or communication requests from existing VMs. The optimal solutions tackles the constraints of computational resource scarcity, network resource scarcity and requests deadline(lifetime) in order to achieve to objectives. These objectives are minimizing blocked requests (or maximizing served requests) and minimizing average tardiness(service delay per requests. The Chapter tackles the problem complexity by , first, evaluating multiple heuristics. This heuristics are much faster than the optimal solution and are considered more practical specially for numerous requests (hundreds or requests or more). However, heuristics are (as the name indicated) offer no optimality(or sub-optimality) guarantees which motivates the third solution offered int he chapter. A suboptimal solution based on relaxing the optimal formulation is offered and evaluated compared to the previous solutions.

A logical future step is exposing the solutions to higher scrutiny to further evaluate their suitability to cloud environment. Despite initially proving the potential efficiency of this solution, the solution can be enhanced by further investigation into additional problem parameters.

Examples include:

1. the effect of different communication parameters on this scenario in terms of varying topologies, traffic models and distributions.
2. More layered computational resource models. VMs are not monopolizing virtualization at the moment as was the case earlier. Containers are gaining momentum and this could affect the application model in terms of resource separation and deeper virtualization imposed by containers and the impacts this could cause for performance and data security/
3. The resource model can be further structured to reflect the virtualized cloud technology. Resources like demanding a specific application or a specific service to be

running on the server a request is placed on can be part of the demand matrix. This cloud replace the conventional resource allocation model (X MIPs, Y GB, etc) with a more layered resource request matrix (a machine with X Http request per second capabilities for example) along with all challenges this would bring.

6.1.2 Energy efficiency resource allocation in cloud environments

Chapter 3 addressed the Second concern of resource allocation in the cloud, energy efficiency. After formulating the problem, A novel consolidation based technique is offered where VM state prediction is employed. the evaluation of this technique against conventional technique shows an encouraging potential in terms of metrics like acceptance rate (served requests) and power consumed per server and per served request. However, with the uncertainty around the live migration impact on performance a solution that offers the best possible energy efficiency without the need to depend on consolidation is important. Chapter 3 offers that technique and show its potential in terms of power consumed per server, power consumed per request, acceptance rate and the percentage of serves used at the data center.

future avenue for this work's expansion are detailed as follows.

1. The rise of the container as a VM replacement offers an interesting challenge. The more specialized and granular nature of containers means the current model of migrate and switch off energy efficiency algorithms employ cannot be left as is. A more complex view of the relation between neighbor containers is required here.
2. The nature of requests and distribution of request lifetime are under major transformation with the increasing penetration of Internet of Things applications. The traffic is going towards a model with more requests with less resource required. The impact this has on the performance bottlenecks is a deciding factor for future resource allocation algorithms.
3. Combining our methods with Dynamic Voltage and Frequency(DVFS) scaling is something to be explored. DVFS is a common method of controlling the power consumption of a server by adjusting the processing power to the utilized comput-

ing power. This could prove an interesting addition to the algorithms proposed in Chapter 3

6.1.3 Cloud simulators as a performance enhancement tools for cloud solutions

Chapter 4 and 5 aim to enhance the solution quality assurance process in cloud environments. This is done by investigating the cloud simulation tools in terms of the application model, architecture and performance. This is followed by stating the design challenges faced by cloud simulator architects and then offering a framework to dictate the cloud simulator design process. Chapter 5 follows by putting this process to the test. This is done by focusing on enhancing a specific performance aspect in a cloud data center which is high availability of cloud data center components. The required features were compiled and GreenCloud cloud simulator was extended with these features. Real application scenarios were implemented using the enhanced simulator as well to demonstrate the new scheme.

The challenges still to be addressed regarding cloud simulators topics were addressed in detail in Section 4.6. Each of these challenges would represent a possible avenue for this work's expansion. Finally, the primary argument made in this thesis is that the proposed resource allocation and simulation techniques can serve as basis for effective solutions to mitigate the performance and cost challenges faced by cloud providers pertaining to energy efficiency, user demands' satisfaction, and resources utilization.

References

[1] Amazon, "Amazon Elastic Compute Cloud (Amazon EC2)." [Online]. Available: http://aws.amazon.com/ec2/

[2] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[3] D. Kliazovich, P. Bouvry, and S. U. Khan, "Greencloud: a packet level simulator of energy-aware cloud computing data centers," *Journal of Supercomputing, special issue on Green Networks*, Jan. 2011.

[4] M. Armbrust *et al.*, *Above the Clouds: A Berkeley View of Cloud Computing*. Tech. Rep. UCB/EECS-2009-28, EECS Department, U.C. Berkeley, 2009.

[5] L. Columbus, "Roundup of small & medium business cloud computing forecasts and market estimates,." [Online]. Available: http://www.forbes.com/sites/louiscolumbus/2015/05/04/roundup-of-small-medium-business-cloud-computing-forecasts-and-market-estimates-2015/#5bc3cc621646

[6] P. Lamson, "A look at U.S. & European cloud adoption trends & forming lasting partnerships." [Online]. Available: http://www.slideshare.net/Carbonite/cloud-adoption-in-us-vs-eu

[7] K. Weins, "Cloud Computing Trends: 2015 State of the Cloud Survey." [Online]. Available: http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2015-state-cloud-survey

[8] C. Tulkoff, "Using Immersion Cooling Technology for Enhanced Efficiency & Reliability for Data Center Servers." [Online]. Available: http://www.slideshare.net/CherylTulkoff/df-r-webinar-immersion-cooling-technology

[9] M. Abu Sharkh, M. Jammal, A. Ouda, and A. Shami, "Resource Allocation In A Network-Based Cloud Computing Environment: Design Challenges," *IEEE Communication Magazine*, vol. 51, no. 11, pp. 46–52, Nov. 2013.

[10] J. Frey, "Network Management and the Responsible, Virtualized Cloud?,," 2011. [Online]. Available: http://www.enterprisemanagement.com/research/asset.php/1927/Network-Management-and-the-Responsible,-Virtualized-Cloud.

[11] A. Leinwand, "The Hidden Cost of the Cloud: Bandwidth Charges," 2009. [Online]. Available: http://gigaom.com/2009/07/17/the-hidden-cost-of-the-cloud-bandwidth-charges/

[12] A. Tavakoli *et al.*, "Applying nox to the datacenter," in *HotNets*, 2009.

[13] Pluribus Networks, "Of Controllers and Why Nicira Had to Do a Deal, Part III: SDN and Openflow Enabling Network Virtualization in the Cloud," 2012. [Online]. Available: http://pluribusnetworks.com/blog/

[14] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *In Proceedings of the First Workshop on Hot Topics in Software-defined Networks (HotSDN 2012),*, 2012, pp. 7–12.

[15] IBM, "IBM CPLEX Optimizer." [Online]. Available: http://http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/

[16] Google, "Google cluster data, trace version 1." [Online]. Available: https://github.com/google/cluster-data/blob/master/TraceVersion1.md

[17] M. Abu Sharkh, A. Ouda, and A. Shami, "Wind Driven Clouds:optimal and suboptimal resource allocation techniques in cloud computing data centers," *Submitted to IEEE Transactions on Cloud Computing*, 2015.

[18] J. Mckendrick, "12 Most Memorable Cloud Computing Quotes from 2013, Forbes Magazine." [Online]. Available: http://www.forbes.com/

[19] IBM's The big data and analytics hub, "The 4 Vs of the Big Data,IBM,"." [Online]. Available: http://www.ibmbigdatahub.com/infographic/four-vs.-big-data

[20] A. Dastjerdi and R. Buyya, "Compatibility-aware Cloud Service Composition Under Fuzzy Preferences of Users," *IEEE Transactions on Cloud Computing*, vol. 2, no. 1, pp. 1–13, Apr. 2014.

[21] G. Wei, A. Vasilakos, Y. Zheng, and N. Xiong, "A Game-theoretic Method of Fair Resource Allocation for Cloud Computing Services," *The Journal of Supercomputing*, vol. 54, no. 2, pp. 252–269, Nov. 2010.

[22] A. Beloglazov, J. Abawajy, and R. Buyya, "Energy-aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing," *Elsevier Future Generation Computer Systems Journal*, vol. 28, no. 5, pp. 755–768, May 2012.

[23] R. Duan, R. Prodan, and X. Li, "Multi-Objective Game Theoretic Scheduling of Bag-of-Tasks Workflows on Hybrid Clouds," *Elsevier Future Generation Computer Systems Journal*, vol. 2, no. 1, pp. 29–42, Jan. 2014.

[24] D. Guzek, M. Kliazovich and P. Bouvry, "A holistic model for resource representation in virtualized cloud computing data centers," in *CloudCom2013*, 2013, pp. 590–598.

[25] M. Abu Sharkh, A. Ouda, and A. Shami, "A resource scheduling model for cloud computing data centers," in *IWCMC2013*, 2013, pp. 213–218.

[26] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual Network Embedding Algorithms With Coordinated Node And Link Mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, Feb. 2012.

[27] M. Abu Sharkh and A. Shami, "An evergreen cloud: Optimizing energy efficiency in green cloud computing environments based on virtual machine state prediction," *Submitted to Elsevier computer networks journal*, 2016.

[28] National resources defense council, "America's Data Centers Are Wasting Huge Amounts of Energy." [Online]. Available: http://www.nrdc.org/energy/files/data-center-efficiency-assessment-IB.pdf.

[29] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy aware consolidation for cloud computing," in *the 2008 conference on Power aware computing and systems*, 2008, pp. 10–16.

[30] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, 2010.

[31] H. Goudarzi and M. Pedram, "Energy-efficient virtual machine replication and placement in a cloud computing system," in *IEEE Cloud*, Jun. 2012, pp. 750–757.

[32] L. Rao *et al.*, "Minimizing electricity cost: Optimization of distributed internet data centers in a multi-electricity-market environment," in *INFOCOM 2010*, Mar 2010, pp. 1–9.

[33] K. Ye *et al.*, "Live migration of multiple virtual machines with resource reservation in cloud computing environments," in *2011 IEEE International Conference on Cloud Computing (CLOUD)*, Jul 2011, pp. 267–274.

[34] K. L. A. Uchechukwu and Y. Shen, "Improving cloud computing energy efficiency," in *Cloud Computing Congress (APCloudCC), 2012 IEEE Asia Pacific*, Nov 2012, pp. 53–58.

[35] G. Laszewski *et al.*, "Power-aware scheduling of virtual machines in dvfs-enabled clusters," in *2009 IEEE International Conference on Cluster Computing and Workshops*, Aug 2009, pp. 1–10.

[36] G. Jung, M. Hiltunen, K. Joshi, R. Schlichting, and C. Pu, "Mistral: Dynamically managing power, performance, and adaptation cost in cloud infrastructures," in *IEEE 30th Intĺ Conf.Distributed Computing Systems (ICDCS)*, 2010.

[37] L. Wang *et al.*, "GreenDCN: A general framework for achieving energy efficiency in data center networks," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 1, pp. 4–15, Jan. 2014.

[38] L. Duan, Z. Dongyuan, and J. Hohnerlein, "Optimizing cloud data center energy efficiency via dynamic prediction of cpu idle intervals," in *2015 IEEE 8th International Conference Cloud Computing (CLOUD)*, 2015, pp. 985–988.

[39] B. Darrow, "is-live-migration-coming-to-amazon-web-services-smart-money-says-yes/." [Online]. Available: https://gigaom.com/2014/10/12/is-live-migration-coming-to-amazon-web-services-smart-money-says-yes/

[40] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations*, vol. 11, no. 1, 2009.

[41] M. Abu Sharkh, A. Kanso, A. Shami, and P. Öhlén, "Building a Cloud on Earth: A Study of Cloud Computing Data Center Simulators," *Submitted to Elsevier Computer Networks Journal*, 2015.

[42] M. Abu Sharkh, A. Shami, A. Ouda, and P. Öhlén, "Simulating high availability scenarios in cloud data centers : A closer look," in *Cloudcom2015*, 2015.

[43] M. O'Neil, "Big data facts, myths and eventualities: 8 insights and implications,." [Online]. Available: http://www.insightaas.com/big-data-facts-myths-and-eventualities-8-insights-and-implications

[44] W. Zhao, Y. Peng, F. Xie, and Z. Dai, "Modeling and simulation of cloud computing: A review," in *2012 IEEE Asia Pacific Cloud Computing Congress (APCloudCC)*, Nov. 2012, pp. 20–24.

[45] U. Sinha and M. Shekhar, "Comparison of Various Cloud Simulation tools available in Cloud Computing," *in International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 3, Mar. 2015.

[46] S. Mohana, M. Saroja, and M. Venkatachalam, "Analysis and Comparison of Simulators to Evaluate the Performance of Cloud Environments," *Journal of NanoScience and NanoTechnology*, vol. 2, no. 6, Feb. 2014.

[47] S. K. Garg and R. Buyya, "Networkcloudsim: modeling parallel applications in cloud simulations," in *4th IEEE International Conference on Utility and Cloud Computing*, 2011, pp. 105–113.

[48] D. Kliazovich, P. Bouvry, and S. U. Khan, *Simulation and performance analysis of data intensive and workload intensive cloud computing data centers in Optical Interconnects for Future Data Center Networks, C. Kachris, K. Bergman, and I. Tomkos, .* Springer-Verlag, New York, USA, 2013.

[49] H. Rajaei and J. Wappelhorst, "Clouds & grids: a network and simulation perspective," in *The 14th Communications and Networking Symposium*, 2011, pp. 143–150.

[50] C. Koo, H. Lee, and Y. Cheon, "Distributed simulator design by using of simnetwork to overcome speed limit on gensim," in *The 14th Communications and Networking Symposium*, 2011, pp. 430–435.

[51] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell: a scalable and fault-tolerant network structure for data centers," in *ACM SIGCOMM 2008 conference on Data communication*, 2008.

[52] C. Guo *et al.*, "BCube: a high performance, server-centric network architecture for modular data centers," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, Oct. 2009.

[53] A. Singla, C. Hong, L. Popa, and P. Brighten Godfrey, "Jellyfish: networking data centers, randomly," in *3rd USENIX conference on Hot topics in cloud computing*, 2011, p. 12.

[54] C. Leiserson, "Fat-trees: universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, vol. 34, no. 10, pp. 892–901, Oct. 1985.

[55] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable commodity data center network architecture," in *ACM SIGCOMM 2008 conference on Data communication*, 2008.

[56] B. Wang *et al.*, "A Survey on Data Center Networking in Cloud era," *Submitted to Communications Survey and Tutorials journal*, 2013.

[57] R. N. Calheiros, R. Ranjan, C. A. F. De Rose, and R. Buyya, "CloudSim: a novel framework for modeling and simulation of cloud computing infrastructure and services," *Technical Report, GRIDS-TR-2009-1, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia*, vol. 34, no. 10, pp. 892–901, Oct. 2009.

[58] R. Miller, "Inside amazon cloud computing infrastructure." [Online]. Available: http://datacenterfrontier.com/inside-amazon-cloud-computing-infrastructure/

[59] ——, "Google data center faq." [Online]. Available: http://www.datacenterknowledge.com/google-data-center-faq-part-3/

[60] ——, "Report: Google Uses About 900,000 Servers." [Online]. Available: http://www.datacenterknowledge.com/archives/2011/08/01/report-google-uses-about-900000-servers/

[61] ——, "Rackspace now has 70,000 servers." [Online]. Available: http://www.datacenterknowledge.com/archives/2011/05/10/rackspace-now-has-70000-servers/

[62] Rackspace, "Global infrastructure and uptime guarantee." [Online]. Available: http://www.rackspace.com/about/datacenters/

[63] S. Rivoire, M. Shah, P. Ranganatban, C. Kozyrakis, and J. Meza, "Models and metrics to enable energy-efficiency optimizations," *Technical Report, GRIDS-TR-2009-1, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia*, vol. 40, no. 12, pp. 39–48, 2007.

[64] A. Nunez, J. L. Vazquez-Poletti, A. C. Caminero, J. Carretero, and I. M. Llorente, "Design of a new cloud computing simulation platform," in *The International Conference on Computational Science and Its Applications*, 2011, pp. 582–593.

[65] Y. Jararweh, Z. Alshroa, M. Kharbutli, M. Jarrah, and M. Alsaleh, "Teachcloud: A cloud computing educational toolkit," in *ICA CON*, 2012.

[66] S. Ostermann, K. Plankensteiner, R. Prodan, and T. Fahringer, "Groudsim: an event-based simulation framework for computational grids and clouds," in *CoreGRID/ERCIM Workshop on Grids and Clouds.*, 2010.

[67] T. Fahringer *et al.*, "Askalon: a grid application development and computing environment," in *6th IEEE/ACM International Conference on Grid Computing*, 2005, pp. 122–131.

[68] B. Wickremasinghe and R. N. Calheiros, "Cloudanalyst: a cloudsim-based visual modeler for analyzing cloud computing environments and applications," in *24th International Conference on Advanced Information Networking and Application*, 2010, p. 446.

[69] F. Fittkau *et al.*, "Cdosim: Simulating cloud deployment options for software migration support," in *IEEE MESOCA'12*, 2012, pp. 37–46.

[70] S. H. Lim, B. Sharma, G. Nam, E. K. Kim, and C. R. Das, "Mdcsim: a multi-tier data center simulation, platform," in *IEEE International Conference on Cluster Computing and Workshops*, 2009, pp. 1–9.

[71] S. Gupta *et al.*, "Gdcsim: A tool for analyzing green data center design and resource management techniques," in *Green Computing Conference and Workshops (IGCC)*, 2011, pp. 1–8.

[72] I. Sriram, "Speci, a simulation tool exploring cloud-scale data centers," in *CloudCom09*, 2009, pp. 381–392.

[73] A. Buss, "Simkit: component based simulation modeling with simkit," in *34th Conference on Winter Simulation*, 2002, pp. 243–249.

[74] D. Meisner, W. Junjie, and T. Wenisch, "Bighouse: A simulation infrastructure for data center systems," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2012, pp. 35–45.

[75] T. C. Computing and U. o. M. Distributed Systems (CLOUDS) Laboratory, "Cloudsim: A framework for modeling and simulation of cloud computing infrastructures and services." [Online]. Available: http://www.cloudbus.org/cloudsim/

[76] A. Zhou, S. Wang, Q. Sun, H. Zou, and F. Yang, "Ftcloudsim: a simulation tool for cloud service reliability enhancement mechanisms," in *Proceedings Demo & Poster Track of ACM/IFIP/USENIX International Middleware Conference*, 2013, pp. 1–2.

[77] C. Fiandrino, D. Kliazovich, P. Bouvry, and A. Zomaya, "Performance and energy efficiency metrics for communication systems of cloud computing data centers," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–1, Apr. 2015.

[78] M. Guzek, D. Kliazovich, and P. Bouvry, "Heros: Energy-efficient load balancing for heterogeneous data centers," in *IEEE International Conference on Cloud Computing (CLOUD)*, Jun. 2015.

[79] D. Kliazovich, S. Arzo, F. Granelli, P. Bouvry, and S. U. Khan, "e-stab: Energy-efficient scheduling for cloud computing applications with traffic load balancing," in *IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing (GreenCom)*, 2013, pp. 7–13.

[80] D. Kliazovich, P. Bouvry, and S. Khan, "DENS: Data Center Energy-Efficient Network-Aware Scheduling," *Cluster Computing, special issue on Green Networks*, vol. 16, no. 1, pp. 65–75, 2013.

[81] D. Kliazovich, J. E. Pecero, A. Tchernykh, P. Bouvry, S. U. Khan, and A. Zomaya, "Ca-dag: Modeling communication-aware applications for scheduling in cloud computing," in *2013 IEEE Sixth International Conference on Cloud Computing (CLOUD)*, 2013, pp. 277–284.

[82] D. Kliazovich, S. T. Arzo, F. Granelli, P. Bouvry, and S. U. Khan, "Accounting for load variation in energy-efficient data centers," in *IEEE International Conference on Communications (ICC)*, 2013.

[83] X. Li, L. Nie, and S. Chen, "Approximate dynamic programming based data center resource dynamic scheduling for energy optimization," in *2014 IEEE International Conference on Internet of Things (iThings), and Green Computing and Communications (GreenCom)*, Sept. 2014, pp. 494–501.

[84] D. Boru, D. Kliazovich, F. Granelli, P. Bouvry, and A. Zomaya, "Energy-efficient data replication in cloud computing datacenters," *Springer Cluster Computing*, vol. 18, no. 1, pp. 385–402, 2015.

[85] J. Zhihua, "Greencloud for simulating qos-based naas in cloud computing," in *9th International Conference on Computational Intelligence and Security (CIS)*, 2013, pp. 766–770.

[86] I. D. Faria, M. Dantas, M. Capretz, and W. A. Higashino, "Energy-aware resource selection model for opportunistic grids," in *2014 IEEE 23rd International WETICE Conference (WETICE)*, Jun. 2014, pp. 167–172.

[87] A. Nunez *et al.*, "Design of a flexible and scalable hypervisor module for simulating cloud computing environments," in *2011 International Symposium on Performance Evaluation of Computer & Telecommunication Systems (SPECTS)*, 2011, pp. 265–270.

[88] C. Esposito, M. Ficco, F. Palmieri, and A. Castiglione, "Smart cloud storage service selection based on fuzzy logic, theory of evidence and game theory," *IEEE Transactions on Computers 2015*, vol. PP, no. 99, pp. 1–14, 2015.

[89] T. Hirofuchi, A. Lebre, and L. Pouilloux, "Adding a live migration model into simgrid: One more step toward the simulation of infrastructure-as-a-service concerns," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, 2013, pp. 96–103.

[90] G. Castanea, A. Nunez, P. Llopisa, and J. Carreteroa, "E-mc2- a formal framework for energy modelling in cloud computing," *Simulation Modelling Practice and Theory, S.I.Energy efficiency in grids and clouds*, vol. 39, no. pp, pp. 56–75, Dec. 2013.

[91] I. Moreno, P. Garraghan, P. Townend, and J. Xu, "An approach for characterizing workloads in google cloud to derive realistic resource utilization models," in *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, 2013, pp. 49–60.

[92] P. Samimia, Y. Teimourib, and M. Mukhtar, "A combinatorical double auction resource allocation model in cloud computing," *Information sciences*, Feb. 2014.

[93] I. S. Moreno, P. Garraghan, P. P. Townend, and J. Xu, "Analysis, modeling and simulation of workload patterns in a large-scale utility cloud," *IEEE Transactions on Cloud Computing,*, vol. 2, pp. 208–221, Apr. 2014.

[94] D. Limbani and B. Oza, "A proposed service broker strategy in cloudanalyst for cost effective data center selection," *International Journal of Engineering Research and Applications (IJERA)*, vol. 2, no. 1, pp. 793–797, Feb. 2012.

[95] R. Mishra and S. Bhukya, "Service broker algorithm for cloud analyst," *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, pp. 3957–3962, 2014.

[96] A. Ahmed and Y. Singh, "Analytic study of load balancing techniques using tool cloud analyst," *International Journal of Engineering Research and Applications (IJERA)*, vol. 2, no. 2, pp. 1027–1030, Mar. 2012.

[97] B. Mondal, K. Dasgupta, and P. Dutta, "Load balancing in cloud computing using stochastic hill climbing-a soft computing approach," in *2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012)*, 2012, pp. 783–789.

[98] G. Kecskemeti, S. Ostermann, and R. Prodan, "Fostering energy-awareness in simulations behind scientific workflow management systems," in *In Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, 2014, pp. 29–38.

[99] H. Fard, R. Prodan, J. Barrionuevo, and T. Fahringer, "A multi-objective approach for workflow scheduling in heterogeneous environments," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, 2012, pp. 300–309.

[100] K. Plankensteiner and R. Prodan, "Meeting soft deadlines in scientific workflows using resubmission impact," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 5, pp. 890–901, May 2012.

[101] S. Frey, F. Fittkau, and W. Hasselbring, "Optimizing the deployment of software in the cloud," in *the conference on software Engineering & management, 2015, Kollen Druck+Verlag*, Mar. 2015.

[102] S. Frey and W. Fittkau, F.and Hasselbring, "Search-based genetic optimization for deployment and reconfiguration of software in the clouds," in *2013 35th International Conference on Software Engineering (ICSE)*, 2013, pp. 512–521.

[103] S. Lim *et al.*, "A dynamic energy management in multi-tier data centers," in *2011 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2011, pp. 257–266.

[104] A. Almaatouq, A. Alabdulkareem, M. Nou, M. Alsaleh, and A. Alarifi, "A malicious activity detection system utilizing predictive modeling in complex environments," in *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, 2014, pp. 371–379.

[105] I. Sriram and D. Cliff, "Effects of component-subscription network topology on large-scale data centre performance scaling," in *2010 15th IEEE International Conference on Engineering of Complex Computer Systems*, 2010, pp. 72–81.

[106] ——, "Speci-2 - an open-source framework for predictive simulation of cloud-scale data-centres," in *Simulation and Modeling Methodologies , Technologies and Applications Conference (SIMULTECH11)*, 2011, pp. 418–421.

[107] ——, "Hybrid complex network topologies are preferred for component-subscription in large-scale data-centres," *Complex networks, Volume 116 of the series Communications in Computer and Information Science*, pp. 130–137.

[108] A. Banerjee, J. Banerjee, Z. Varsamopoulos, G. abd Abbasi, and S. Gupta, "Hybrid simulator for cyber-physical energy systems," in *Workshop on Modeling and Simulation of Cyber-Physical Energy Systems (MSCPES)*, 2013, pp. 1–6.

[109] C. Hsu *et al.*, "2015 ieee 21st international symposium on high performance computer architecture (hpca)," in *Adrenaline: Pinpointing and Reining in Tail Queries with Quick Voltage Boosting*, 2015, pp. 271–282.

[110] Z. Zhang, H. Yang, Z. Luan, and D. Qian, "Request squeezer: Mitigating tail latency through pruned request replication," in *2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC)*, 2015, pp. 357–362.

[111] P. Tandon, M. Cafarella, and T. Wenisch, "Minimizing remote accesses in mapreduce clusters," in *2013 IEEE 27th International Symposium on Parallel & Distributed Processing Workshops and PhD Forum*, 2013, pp. 1928–1936.

[112] M. Quwaider and Y. Jararweh, "Cloudlet-based efficient data collection in wireless body area networks," *Simulation Modelling Practice and Theory*, vol. 50, pp. 57–71, Jan. 2015.

[113] Y. Jararweh *et al.*, "Cloudexp: A comprehensive cloud computing experimental framework," *Simulation Modelling Practice and Theory*, vol. 49, pp. 180–192, Dec. 2014.

[114] Q. Althebyan, Y. ALQudah, O. Jararweh, and Q. Yaseen, "Multi-threading based map reduce tasks scheduling," in *2014 5th International Conference on Information and Communication Systems (ICICS)*, 2014, pp. 1–6.

[115] D. Milojicic, HP Labs, "High Performance Computing (HPC) in the Cloud," 2015. [Online]. Available: http://www.computer.org/web/computingnow/archive/september2012#sthash.n5vLzk7W.dpuf

[116] K. Townsend, "Containers: The pros and the cons of these VM alternatives," 2015. [Online]. Available: http://www.techrepublic.com/article/containers-the-pros-and-the-cons-of-these-vm-alternatives/

[117] S.J. Vaughan-Nichols, "Containers vs. virtual machines: How to tell which is the right choice for your enterprise," 2015. [Online]. Available: http://www.itworld.com/article/2915530/virtualization/containers-vs-virtual-machines-how-to-tell-which-is-the-right-choice-for-your-enterprise.html

[118] IBM, "Docker: Build, ship, run, an open platform for distributed applications for developers and sysadmins," 2015. [Online]. Available: https://www.docker.com/

[119] M. Schnjakin, R. Alnemr, and C. Meinel, "A security and high-availability layer for cloud storage," in *Grid Computing Environments Workshop*, 2008, pp. 1–10.

[120] C. Pham, P. Cao, Z. Kalbarczyk, and R. Iyer, "Toward a high availability cloud: Techniques and challenges," in *IEEE/IFIP 42nd International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 2012, pp. 1–6.

[121] M. Frincu and C. Craciun, "Multi-objective meta-heuristics for scheduling applications with high availability requirements and cost constraints in multi-cloud environments," in *IEEE International Conference on Utility and Cloud Computing (UCC)*, 2011.

[122] D. Singh, J. Singh, and A. Chhabra, "High availability of clouds: Failover strategies for cloud computing using integrated checkpointing algorithms," in *2012 International Conference on Communication Systems and Network Technologies(CSNT)*, 2012, pp. 698–703.

[123] I. Addo, S. Ahamed, and W. Chu, "A reference architecture for high-availability automatic failover between paas cloud providers," in *International Conference on Trustworthy Systems and their Applications (TSA)*, Jun. 2014, pp. 4–21.

[124] Y. Wang, J. Ma, D. Lu, X. Lu, and L. Zhang, "From high-availability to collapse: quantitative analysis of ?Cloud-Droplet-Freezing? attack threats to virtual machine migration in cloud computing," *Cluster Computing Journal*, vol. 17, no. 4, pp. 1369–1381, Dec. 2014.

[125] D. Prasad, "High Availability Based Migration Analysis to Cloud Computing for High Growth Businesses," *International Journal of Computer Networks (IJCN)*, vol. 4, no. 2, p. 35, Dec. 2012.

[126] S. Jaffer, M. Chitnis, and A. Usgaonkar, "Providing high availability in cloud storage by decreasing virtual machine reboot time," in *10th Workshop on Hot Topics in System Dependability (HotDep 14)*, 2014, pp. 7–12.

Curriculum Vitae

Name: Mohamed Fathi Abu Sharkh

Post-secondary Education and

2012-2016 Ph.D
Software Engineering
Western University
London, Ontario, Canada

2005-2009 M.Sc.
Computer Engineering
Kuwait University
Kuwait city, Kuwait

2001-2005 B.Sc.
Computer Science
Kuwait University
Kuwait city, Kuwait

Related Work Experience

2012-2016
Teaching Assistance
Multiple software engineering classes
Western University
London, Ontario, Canada

2014-2015
"Highly availability tools in cloud environment"-
Project researcher
Western University-Ericsson Research

2013-2014
"Resource allocation in cloud DCs"-
Project researcher
Western University-Samsung

2005-2011
Software Implementation Consultant
Beshara Group for Software Solutions

Honours and Awards

Western University Graduate Research Scholarship 2012-2015

Nominated by Western Engineering for the Ontario wide NSERC research funding competition (2013)

ACM local programming contest winning team member 2002, 2003, 2004

On Dean's Honor List 2002-2005

Awarded the 2015 volunteer appreciation award by IEEE London section for "engaging leadership of the YPs

Represented the electrical and computer engineering department at "The Big Data days @western-2014

Publications

[J1] M. Abu Sharkh, M. Jammal, A. Ouda, and A. Shami, "Resource Allocation In A Network-Based Cloud Computing Environment: Design Challenges," IEEE Communication Magazine, vol. 51, no. 11, pp. 4652, Nov. 2013.

[J2] M. Abu Sharkh, A. Kanso, A. Shami, and P. Öhlen, "Building a Cloud on Earth: A Study of Cloud Computing Data Center Simulators," Accepted for publication at Elsevier Computer Networks Journal, 2016.

[J3] K. Alhazmi, M. Abu Sharkh, and A. Shami, "Drawing the Cloud Map: Virtual Network Provisioning in Distributed Cloud Computing Data Centers", IEEE Systems Journal, vol. PP , no. 99, pp. 1-12 , Jan. 2016.

[J4] M. Abu Sharkh, A. Ouda, and A. Shami, "Wind Driven Clouds: Optimal and Suboptimal Resource Allocation Techniques in Cloud Computing Data Centers," Submitted to IEEE Transactions on Cloud Computing, 2016.

[J5] M. Abu Sharkh, A. Ouda, and A. Shami, "An Evergreen Cloud: Optimizing Energy Efficiency in Green Cloud Computing Environments Based on Virtual Machine State Prediction", Submitted to Elsevier

Vehicular Communication journal, 2016.

[C1] M. Abu Sharkh, A. Ouda, and A. Shami, "A resource scheduling model for cloud computing data centers, " The 9th International Wireless Communications & Mobile Computing Conference (IWCMC 2013), pp. 213218, Jul. 2013.

[C2] K. Alhazmi, M. Abu Sharkh, D. Ban and A. Shami "A Map of the Clouds: Virtual Network Mapping in Cloud Computing Data Centers," The 27th Annual Canadian Conference on Electrical and Computer Engineering, May. 2014.

[C3] M. Abu Sharkh, A. Shami, and A. Ouda, and P. Öhlen, "Simulating High Availability Scenarios in Cloud Data Centers: A Closer Look", CloudCom2015, Dec. 2015.