TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

LIST OF ABREVIATIONS

ACO	Ant Colony Optimization
CCS	Cloud Computing Systems
CP	Constraint Programming
DVFS	Dynamic Voltage and Frequency Scaling
FF	First Fit
FFD	First Fit Decreasing
GA	Genetic Algorithm
IaaS	Infrastructure as a Service
ICA	Imperialist Competitive Algorithm
IGA	Improved Genetic Algorithm
LP	Linear Programming
LR	Local Regression
MACO	Multi Objective Ant Colony Optimization
MGA	Multi Objective Genetic Algorithm
NIST	National Institute of Standards & Technology
OS	Operating System
OTF	Overload Time Fraction
PaaS	Platform as a Service
PM	Physical Machine
QoS	Quality of Service
SaaS	Software as a Service
SLA	Service Level Agreement

SP	Service Provider
ST	Single Threshold
VM	Virtual Machine
VMM	Virtual Machine Monitor
VN	Virtual Network

LIST OF SYMBOLS AND UNITS OF MEASUREMENTS

N	Number of virtual machines
M	Number of physical machines
R	The set of resources needed by each VM
y_j	If PM_j is active or not
x_{ij}	Whether VM_i is assigned to PM_j or not
x_{is}	Traffic of VM_i is transferred over NE_s
z_s	Whether NE_i is active or not
U_j^P	CPU utilization
P_j^{busy}	Average power values when the j-th PM is busy
P_j^{idle}	Average power values when the j-th PM is idle
W	The resource wastage of each PM
R^{PM}	The set of resources available in PM_j
$R_{i,1}^{VM}$	The set of CPU resources requested by VM_i
T	Traffic load communication between two VMs
S	Number of Network equipment
l	Number of resources available in a PM
K	Number of resources available in a VM
T^{VM}	Matrix of communication traffic load
$\tau_{v,p}$	Pheromone level of VM into PM
$n_{i,v}$	Heuristic information
C_p	Capacity of each PM

b_p	CPU load of each PM
r_v	The requested number of MIPS for a VM
E_j	Energy consumption of server j
E_{max}	Maximum energy consumption of each server
ρ	The pheromone evaporation parameter
$nAnts$	Number of ants
$nCycles$	Number of cycles
S_{cycle}	The best solution of that cycle
S_{best}	The best solution
t_a	The total time when a PM is being activated
t_o	The total time when the CPU utilization of an active PM is being overloaded
$SLAV$	Number of SLA violations
α	Parameter to give more power to the pheromone factor
β	Parameter to give more power to the pheromone factor
PEs	Number of processing elements

INTRODUCTION

In the past few years, cloud computing has evolved as a new computing paradigm for delivering services over the Internet using dynamic pool of virtualized resources. These services can be categorized as: Infrastructure as a Service, Platform as a Service (PaaS), and Software as a Service (SaaS) (Hai, Kun and Xuejie, 2010). The resources and services can be shared based on the pay-as-you-go model. However Cloud computing have to offer more flexible and high performance network and service infrastructures in order to provide reliable and efficient access to resources (hardware and software). Consequently, the more cloud infrastructure resources are used by service providers, the more energy is consumed. Studies have shown that the energy consumptions of the average data centers in the world are equal to twenty five thousand house consumptions (James M. Kaplan, 2008; W. F. James M. Kaplan, 2008). These consumptions have been increased by 56% from 2005 to 2010 (Koomey, 2011). IT equipment consumed 0.5% of total electricity produced by the world in 2005 (Koomey, 2008). The data centers have to minimize their carbon footprint by reducing their energy consumption Also reducing energy consumption can save a significant amount of money and help protect our environment by reducing CO_2emission (P. Johnson, 2009).

Now the question is what solutions are able to decrease the energy consumption of a data center while guaranteeing the appropriate service delivery. Implementing an energy aware computing is considered as a solution in order to solve the problem of energy consumption inefficiency.

Green cloud computing offers techniques and strategies to minimize energy consumption as well as optimize resource utilization (Rajkumar Buyya, 2010). As presented in (Fan, Weber and Barroso, 2007), the dynamic power range for network switches is 15%, for disk drivers is 25% and DRAM is 50%. It means some of the server components (e.g. Memory, CPU, Disk, PCI Slots …) can be switched off when the servers are under loaded. In addition even a physical server is not overloaded, it still needs more than 70% of power (Beloglazov, 2013).

To reduce power consumption and increase the performance of server resources, Virtualization technology is used. In this technology an abstraction layer is used between operating system and hardware. Then each physical resource is able to be split up into a number of logical pieces that conceptually called VMs. By using Virtualization technology multiple VM instances can be initialised on a physical machine and as a result the amounts of active hardware are reduced and the utilisations of physical resources are increased. So the appropriate mapping between VMs and PMs can improve the resource utilization of a total datacenter and as a result minimize the expected cost. For moving VMs between PMs, Live Migration techniques are used that it is considered as a well-known virtualization technique. The live migration capability can be provided through Virtualization technology and can be dynamically consolidated to keep the number of active PMs at the minimum level. So Dynamic consolidation of VMs includes two basic steps: 1) VMs migration from under-loaded PMs to minimize number of active PMs; 2) VMs load balancing between overloaded PMs to keep performance expectations to keep the QoS requirements (Beloglazov, 2013).

Dynamic consolidation of VMs is a method to optimize the energy consumption and resource utilization of VMs by using Virtualization technology. By using this technique, idle PMs are recognized and switched to a low-power mode in order to reduce power consumption. However if unexpected resource demands are increased, VM consolidation may lead to degrade performance. If the resource requirements are not met, the response times of application will be increased. So there is a trade-off between minimizing energy consumption, reducing costs and meeting QoS requirements. One of the main challenges of Cloud providers is how to deal with this trade-off between energy consumption and resource wastage while meeting QoS demands. Due to these reasons, this thesis focuses on multi-objective resource consolidation approach that allows minimizing energy consumption, minimizing resource wastage and minimizing SLA violation.

The development of dynamic VMs consolidation algorithm lies on the definition of an efficient VM placement algorithm. VM placement is a process that maps virtual machines

(VMs) to physical machines (PMs). In this thesis the following research problems are explored:

- How to propose an approach to take into consideration energy consumption and resource utilization.
- How to manage SLA violations and consider requirements of QoS in a distributed environment.
- Which resources should be migrated and also when these resources should be consolidated. In other words which VM should be placed to which PM in order to reduce to load of communications in Network.
- How to validate this approach with other proposed approaches
- Which simulation tools should be selected in order to have more realistic analysis

In order to save energy consumption of a datacenter, idle servers should be converted to a low-power mode. Two main energy-aware algorithms in a cloud management system have been proposed: VM placement and Resource consolidation. In the present thesis we categorized our main objectives as follows:

1) Develop a VM placement algorithm in order to initialize and place VMs among PMs.
2) Develop a VM consolidation algorithm in order to optimize the VM placement.
3) Test and analyze the performance of the proposed algorithms and compare them with existing approaches

The output of first algorithm is considered as input of second algorithm to take into consideration the mentioned objectives.

Our main contributions are: 1) Proposing a multi-objective optimization placement approach to minimize the total energy consumption of data center, resource wastage and energy communication cost. According to the position of PMs within a datacenter, VM communications might have different costs. However other multi-objective approaches have not taken into consideration communication network costs. 2) Comparison of the proposed placement approach with other meta-heuristic (MGA) approaches and other single-objective VM placement algorithms (FFD, DVFS, and LR). 3) Proposing a multi-objective consolidation approach to minimize the total energy consumption of a data center, minimize number of migrations, minimize number of PMs and reconfigure resources to satisfy the SLA. In particular, the new consolidation algorithm should take into consideration the VM-PM solution which have been provided by placement algorithm and tries to optimize it. Our multi-objective approach is integrated with Cloudsim tools in order to optimize the assignment of VMs and PMs. Like placement approach, the consolidation approach is compared with other meta-heuristic Genetic Algorithm (MGA) approaches and a Multi-Objective ACO which has been proposed by Feller.

The rest of the present thesis is organized as follows. Chapter 1 discusses related work. It presents the main concepts and definitions of cloud computing, virtualization, resource management in distributed systems. Chapter 2 gives the problem statement and presents a multi objective placement algorithm with specific three objectives. Chapter 3 presents a multi objective consolidation algorithm with four objectives and methodologies. Chapter 4 presents the results analysis based on the proposed approaches and discusses the experimental setup and simulation results. We conclude the last chapter with a conclusion of our approaches as well as discussions and suggestions for future works.

CHAPTER 1

STATE OF THE ART

1.1 Introduction

This chapter elaborates the main background information and concepts on Cloud computing, Virtualization and Resource management. The open issues of each concept are described at the end of each part.

1.2 Background and definition

1.2.1 Cloud computing overview

Due to the popularity of the Internet, cloud computing systems are using this infrastructure for sharing their resources through the Internet. The cloud computing is new paradigm that allows sharing services and resources over the communication networks. The cloud computing is based on virtualization technologies. These technologies enable deployment of different service and network architectures over shared physical infrastructures. These technologies allow efficient and flexible sharing of underlying resources. Cloud computing systems use Virtualization technology not only to distribute computing resources, but also use privacy protection and data security features of virtualization. However resource management is considered as a challenge in virtualization environment. In order to optimize resource usage we should consider multiple objectives which makes it more complicated (SOUALHIA, 2013). In Resource management, making selection of different strategies can effect on costs, energy usages and system efficiency. However allocation virtual machines (VMs) among physical machines are important issues in virtualization technology which will be controlled by Resource management. VM placement and VM consolidation are two main resource management strategies that help in addressing these challenges.

1.2.2 Definition

Different definitions have been provided for the concept of Cloud computing:
According to the definition of Buyya in (Buyya et al., 2009), a Cloud is a collection of virtualized computers and inter-connected devices that dynamically presented in a shared pool of resources based on SLA between customers and service providers.

In (Vaquero et al., 2008) Vaquero defined clouds as a large pool of resources that can be used easily and also can be accessible from consumers. According to the customized SLA, the resources can be reconfigured dynamically in order to adapt traffic load and thereby to satisfy the SLA. According to (Glauco Estácio Gonçalves, 2011), Clouds are a shared pool of virtualized resource that can be accessible with each other very easily. These resources are reconfigured dynamically in order to control traffic load.

The National Institute of Standards and Technology (NIST) specified four deployment models, three service models and five service attributes.

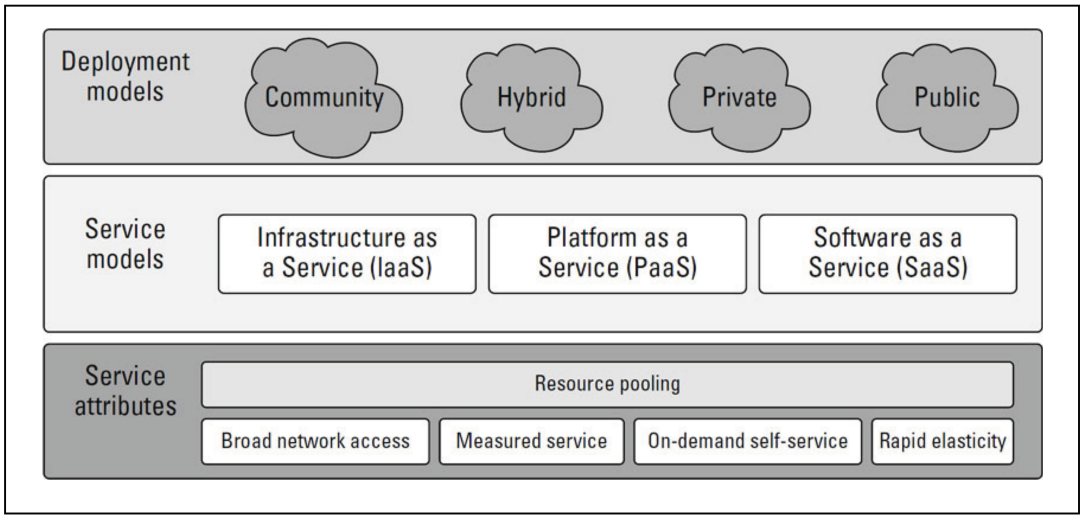


Figure 1.1 The NIST Cloud computing definitions
Taken From (Sosinsky, 2011)

1.2.3 Cloud computing deployment models

1.2.3.1 Public cloud

This cloud infrastructure is publically accessible over the Internet. End-users have access to different services and resources (such as: storages and cloud applications). It is owned by cloud service providers. The well-known public clouds are: Suncloud, Amazon Elastic compute cloud (EC2), Google APP Engine, IBM’s Blue cloud and windows Azure services platform (Sosinsky, 2011; Teng, 2012; Wikipedia).

1.2.3.2 Private cloud

This cloud infrastructure is designed for private usage of an organization or for specific market sector. The owner of this infrastructure or a third party usually manages this cloud. The services offered by these cloud providers are not accessible for everyone. We can mention HP Clouds start and eBay as an examples of this infrastructure (Sosinsky, 2011; Teng, 2012).

1.2.3.3 Community cloud

This infrastructure is designed to comply with specific purpose of a community that includes several organizations having the same objectives such as business security and regulations. These organisations or a third party will manage this cloud.

1.2.3.4 Hybrid cloud

Hybrid cloud includes different models of cloud: public, private and community. These clouds are tied together to form the same entity while keeping their own identities In order to give better business services, several IT organizations are using this model (e.g. IBM, HP, Oracle, and VMware) (Talia, 2012; Teng, 2012).

(Beloglazov and Buyya, 2013)

Figure 1.2 shows we can see different cloud computing deployment models.

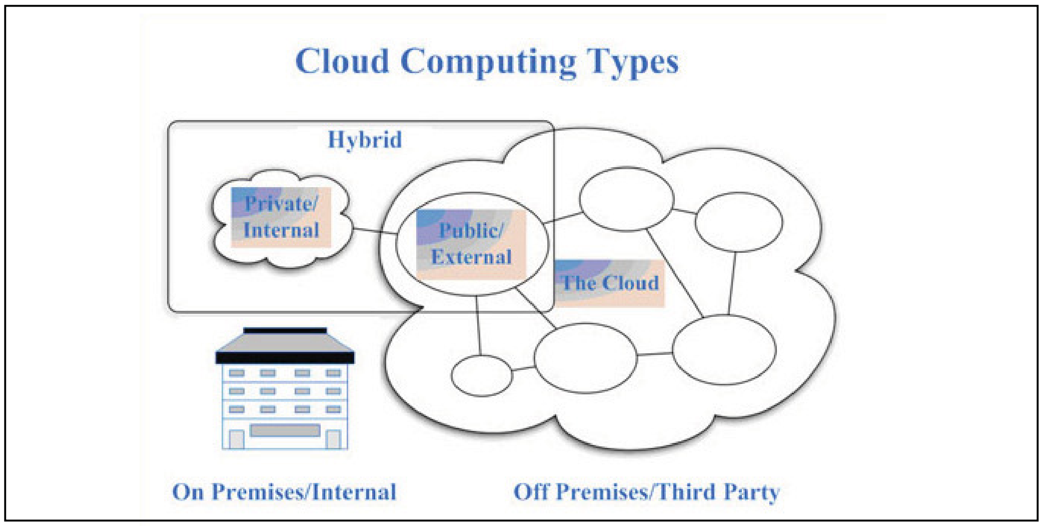


Figure 1.2 Different deployment models
Taken From (Feller, Rilling and Morin, 2011)

1.2.4 Cloud computing service models

1.2.4.1 Infrastructure as a Service (IaaS)

This model enables to offer virtual resources (such as computing, and storage) as services to customers. The entire infrastructure will be managed by the IaaS provider while the customer will be in charge of the system deployment.

1.2.4.2 Platform as a Service (PaaS)

Here the PaaS acts as an abstraction layer on top of the infrastructure. It provides API, operating and control services and tools that allow customers deploying their applications.

For instance Microsoft developed .Net as a platform for its customers. Installing and managing applications are under users' responsibility and cloud providers manage operations systems and enabling software at this service (Sosinsky, 2011; Talia, 2012).

1.2.4.3 Software as a Service (SaaS)

The SaaS acts by giving service to the providers which manage and control applications. For instance Web browser might be a good example of client interface for this service. This service considered as a complete operating environment with applications, management, and the user interface (Talia, 2012).

In order to understand the difference of the mentioned services, Figure 1.3 and Figure 1.4 show the order of these layers with some sample examples of services within each layer.

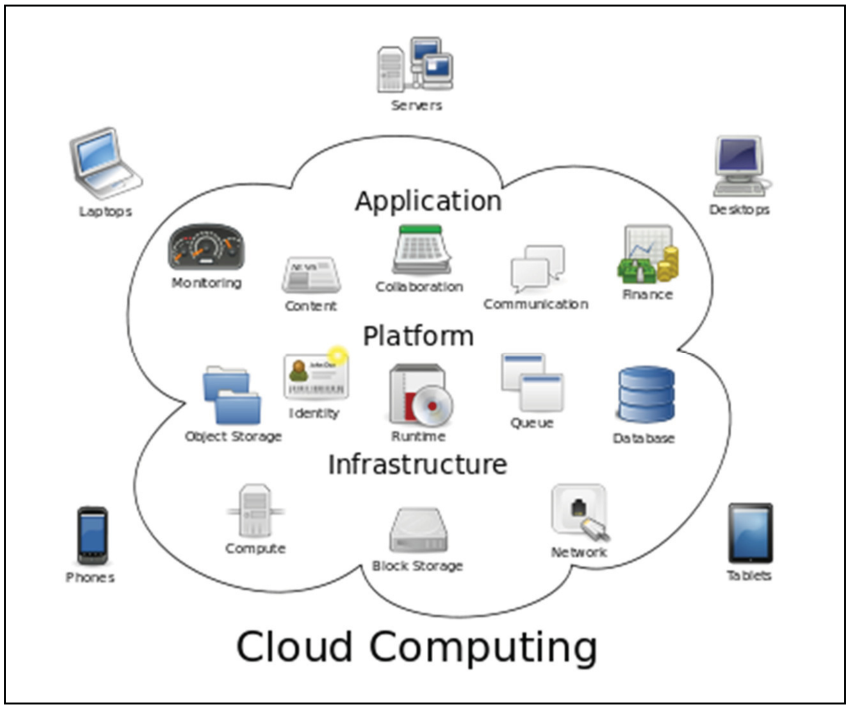


Figure 1.3 Cloud computing structure
Taken From (Rebecca Scudder, 2011)

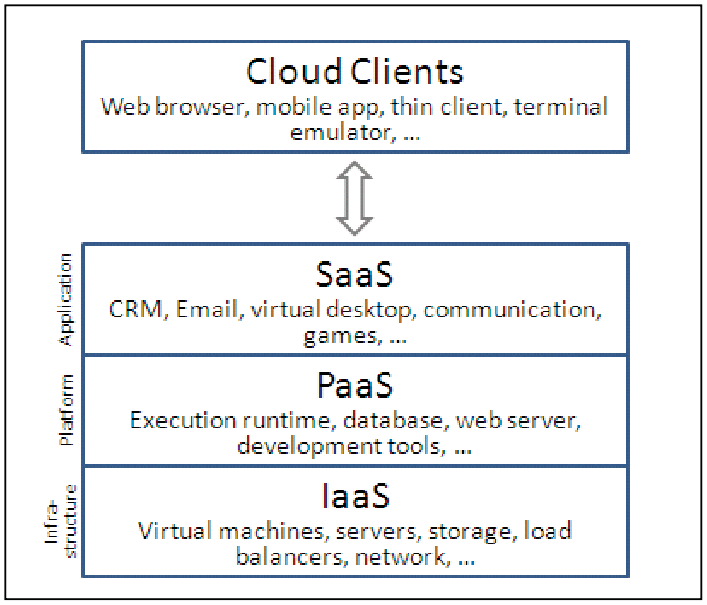

Cloud Clients
Web browser, mobile app, thin client, terminal emulator, ...
Application
SaaS
CRM, Email, virtual desktop, communication, games, ...
Platform
PaaS
Execution runtime, database, web server, development tools, ...
Infra-structure
IaaS
Virtual machines, servers, storage, load balancers, network, ...

Figure 1.4 Cloud computing services
Taken From (Bikeborg, 2012)

1.2.5 Comparison of cloud computing with similar technologies

In this section we elaborate the other technologies and infrastructures that have similar concepts of Cloud computing but with other specific usage.

1.2.5.1 Grid computing

Grid computing is a distributed architecture of different numbers of resources which connected together for a common purpose. The complex architecture of large distributed systems can cause frustration and decrease productivity. Frustration is considered as one of the main problems of these computing systems because each user has his/her environment configuration and also has different platforms and requests. The frustration problem in grid computing architecture can solve through virtualization technology. In addition, resource management is another problem in grid computing environments. In the peak usage time, users had to wait more than normal situation or the tasks took more time to complete and users could not able to assign any deadline for their jobs (Teng, 2012; Voorsluys, Broberg and Buyya, 2011).

1.2.5.2 Utility computing

In these systems, users can set different constraints as QoS (Quality of Service) for their jobs. For instance they can define deadlines, priority and users' satisfaction. Based on these QoS parameters, Service providers can have profit and they will try to optimize their strategies to get more benefit from market and also try to reduce their costs.

The utility idea has been used in another computing environments such as Grid and Cloud computing. HP Company has provided new utility computing for producing the IP services since last 90's (Teng, 2012; Voorsluys, Broberg and Buyya, 2011).

1.2.5.3 Autonomic computing

Due to the large number of electronic devices and large amount of data, maintenance and computation of these data became hard for humans. In 2001 one model is introduced by IBM which is called "Autonomic Computing". Autonomic computing systems are able to compute operations of different processes automatically without human involvement. It allows monitoring data through sensors and autonomic managers (Teng, 2012; Voorsluys, Broberg and Buyya, 2011).

Now we can conclude the relation of the mentioned computing models with each other. Utility Computing tries to use a meters service based on the users' requirements. Therefore it cannot be useful in Centralized and distributed systems. Grid computing concept is very similar to cloud computing definition but the entities are not economical. On the other hand Autonomic computing emphasizes on self-management computing model whereas this feature is considered as one of the features of cloud computing. To sum up we should say that we can have all the features of Autonomic computing, cloud computing and grid computing together in Cloud Computing systems.

1.3 Introduction to virtualization technology

In the previous subsection we have reviewed cloud computing concepts and different types of Cloud computing models including Grid computing, Utility computing and Autonomic Computing. Nowadays a lot of services and applications are being supported by cloud computing technology and lots of servers are used in order to response users' requests. Now the question is which platform can manage these servers. In addition what is the relationship between clouds computing and virtualization and why we should use virtualization technology in cloud computing environments?

Maintaining large mainframe computers was awkward in 1960s. In order to minimize maintenance overhead and increase the environment's efficiency, IBM Corporation Company introduced Virtualization technology to share resources (e.g. computing storage and network connectivity) among multiple processes running in parallel at the same time.

Virtualization technology introduces an abstraction layer between operating systems and hardware (Rossi, Beek and Walsh). This layer is called hypervisor or Virtual machine monitor (VMM) and controls hardware resources directly. Actually hardware resources have been hid from the OSs by the abstraction layer. Since operating resources are not controlled by operating systems, the same hardware could be run on multiple operating systems. Therefore the hardware platform can be divided into logical units called virtual machines (Sahoo, Mohapatra and Lath, 2010).

As a result, Virtualization enables deploying different logical units on a top of the same physical machine. Cloud computing systems use Virtualization technology not only to distribute computing resources, but also use privacy protection and data security features of virtualization.
Three various layers are considered in virtualization technology: 1) Physical Layer that supplies the physical resources 2) Service Layer which represents users' services 3) Mediation Layer that manages requirements of service layer and physical layer.

1.3.1 Types of virtualization

Virtualization technology has been divided into several categories such as: Hardware virtualization, Software virtualization, Server virtualization and Network virtualization. In this section we elaborate the mentioned main categories however there are other types of virtualization such as: Desktop virtualization, Service virtualization, Memory virtualization, Data and database virtualization.

1.3.1.1 Hardware virtualization

In hardware virtualization technology VMM runs on hardware layer directly and it controls access of the guest operating system to the hardware resources. In Figure 1.5 we can see different virtual machines with different OS hosting various users' softwares on top of a virtual machine monitor (Hypervisor) and a Hardware Platform (Voorsluys, Broberg and Buyya, 2011). The hypervisor enables controlling VM creation on different OS platforms and allocating resources to each VM based on VM request requirements.

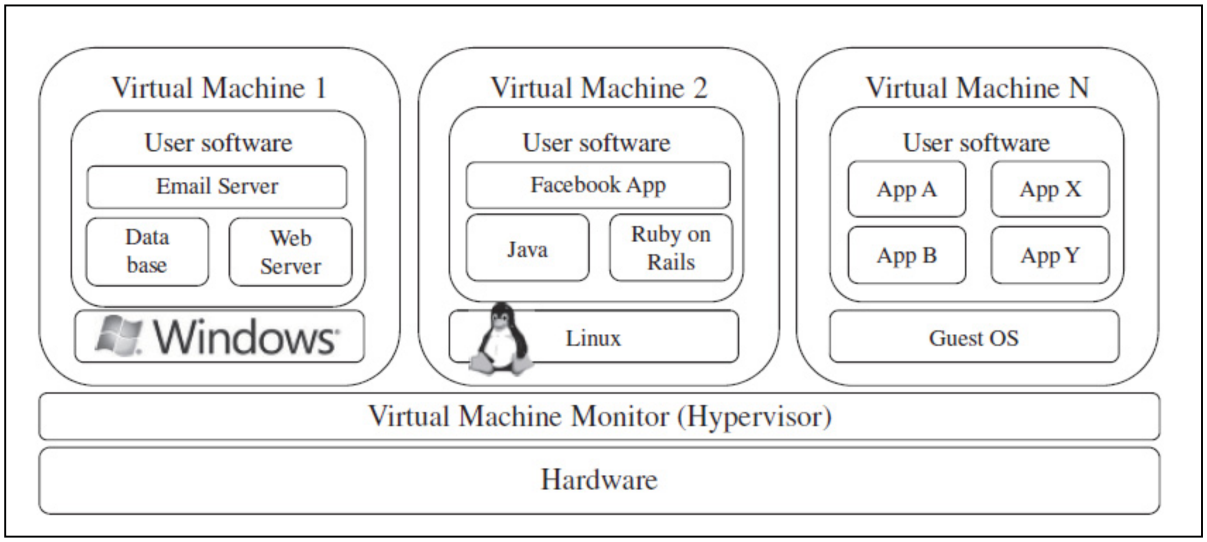


Figure 1.5 Three virtual machines with one hardware virtualized server
Taken From (Voorsluys, Broberg and Buyya, 2011)

1.3.1.2 Software virtualization

In this type of virtualization, applications do not need to be installed on their PCs and they can run local server application and local resources. In this method the requirement resources for executing application will be used and each user has a virtual application environment. This application environment acts as a layer between the host operating system and the application (Sahoo, Mohapatra and Lath, 2010). Figure 1.6 shows behavior of Software Virtualization in application layer.

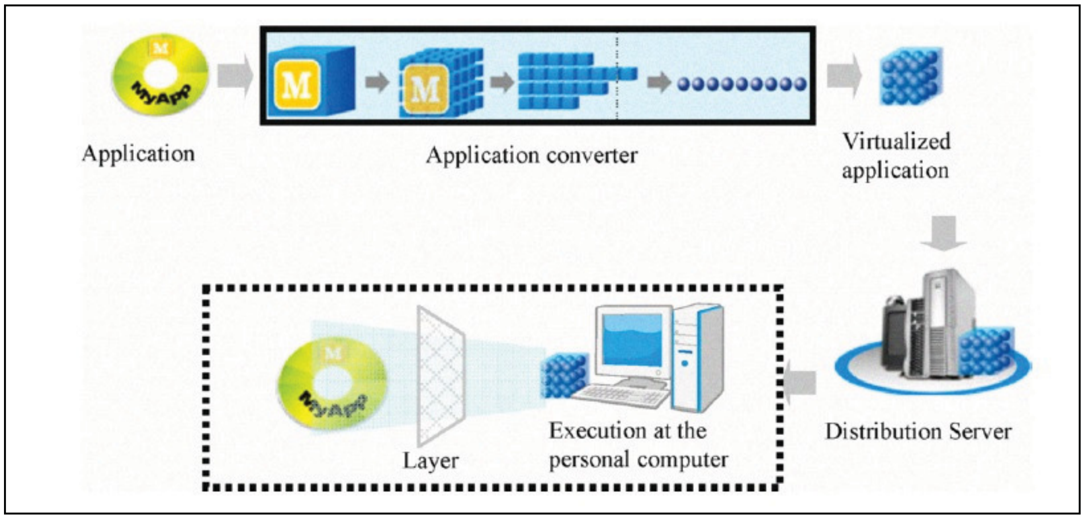


Figure 1.6 Behaviour of Software Virtualization
Taken From (Kyong et al., 2010)

1.3.1.3 Server virtualization

A physical server provides multiple virtual servers on a single platform. Through this method different virtual OS (Operating Systems) run individually on each physical machine. A Virtual Machine Monitor (VMM) is an abstraction layer between the operating systems and hardware. VMM is also called hypervisor. The hypervisors manage the allocation of resources to operating systems (Friedman, 2011).

1.3.1.4 Network virtualization

Monitoring and management of entire network structures as a single administrative entity is called Network virtualization. The physical network resources (such as nodes and links) are shared among virtual links and virtual nodes. In this method Service Provider (SP) creates Virtual Networks (VN) by using Infrastructure Providers' resources (Bo et al.) (Chowdhury and Boutaba, 2009).

1.3.1.5 Storage virtualization

Storage virtualization is used to abstract logical storage from physical storage. This technology provides a logical space for users and it handles mapping processes to actual physical location. The advantages of Storage virtualization are (Friedman, 2011):

- Make storage appear to users locally
- Reduce storage growth and improves utilization
- Reduce power and energy requirements
- Provide centralized data
- Eases data back up

1.4 Resource management in cloud computing

Customer could access to different applications and various services through cloud computing over the internet. With the rapid usage of cloud computing in business, academy and industrial environments, Cloud service providers give high storage space and computation ability to consumers through virtualization technology and cloud computing as a unified system. However allocating virtual machines (VMs) among physical machines are important issues in virtualization technology which will be controlled by Resource management. In Resource management, making selection of different strategies can effect on costs, energy usages and system efficiency. For example, if Resource management is not able to allocate resources among users in peak time traffic, the level of efficiency will be degraded. Therefore, deciding how to select appropriate strategy for managing resources is a demanding task which should be controlled by Resource management. In this section we focus on some important problems in Resource management and we review some researches which are related to these issues.

1.4.1 Resource management algorithms

1.4.1.1 Greedy algorithms

In order to solve resource management problem in cloud computing environments, the traditional greedy algorithms have been used. The aim of these algorithms is to find a local best solution and they can be a good candidate for solving VM placement and VM consolidation problems. However they are not able to necessarily find global optimal solution due to the local solution procedure. However these algorithms do not have complexity for implementation and have low polynomial time complexity (Feller, 2013). There are two kind of greedy algorithms: Offline and Online. The offline algorithms know all VM requests and they can make decision based on their requests. But the online algorithms do not have any knowledge about the whole VM requests and they allocate VMs to PMs as they receive new VMs. As an example First Fit Decreasing (FFD) is a well-known offline algorithm where the VMs are sorted in descending order (based on their request demands). These sorted VMs are allocated to PMs. On the other hand First Fit (FF) is an online greedy algorithm where the VMs are placed on the first available PMs with enough resource capacity. If the current PM does not have enough capacity, a new PM is activated to host the VM (Yue, 1991).

1.4.1.2 Mathematical programming algorithms

Constraint Programming (CP) (Rossi, Beek and Walsh, 2006) and Linear Programming (LP) (Schrijver, 1986) are examples of mathematical programming that are able to find optimal solution for VM placement and VM consolidation problems. However these algorithms need exponential time to find optimal solution. In addition we are not able to take into consideration different objectives in mathematical programming algorithms. So the execution time of these algorithms depends on number of VMs and PMs.

1.4.1.3 Meta-heuristic approaches

Meta-heuristic algorithms have also been proposed for resource management in cloud computing. These algorithms are able to find sub optimal solutions based on their probabilistic algorithms. Genetic algorithms (GA) (Goldberg, 1989) together with Ant colony optimization algorithms (ACO) (Dorigo, Caro and Gambardella, 1999), and Imperialist competitive algorithm (Eduardo Pinheiro) (Atashpaz-Gargari and Lucas, 2007) are few examples of meta-heuristic approaches. Compared to mathematical programming and greedy algorithms, the meta-heuristic algorithms allow defining a multi-objective approach. However these algorithms generate random data and due to this reason they cannot guarantee to find optimal solutions.

1.4.2 Energy-aware resource management

1.4.2.1 VM placement

Cloud computing model support a variety of applications on a shared hardware platforms. Due to the popularity of cloud computing, large-scale data centers need thousands of computing servers in order to cover customers' needs. The more servers are used in large data centers, the more energy is consumed. High performance has always been the main concern in deployment of data centers keeping a side the energy consumption and it impacts on the environment. According to Kaplan research in 2008 (James M. Kaplan, 2008), data centers consume as much energy as twenty five thousands house consumptions. Due to the large usage of data centers, Green cloud computing is introduced to minimize energy usage and achieve efficient management of cloud computing infrastructure (Rajkumar Buyya, 2010). In fact, Cloud providers need to reduce not only energy consumption but also to guarantee customer service delivery based on QoS constraints (Beloglazov, Abawajy and Buyya, 2012). In this section, we discuss one of the main issues in cloud computing: The Energy-aware resource management.

Pinheiro and Bianchini in (Eduardo Pinheiro 2001) have considered the issue of energy consumption in large clusters and PCs. Their approach is to develop systems for minimizing energy consumption in replication of cluster nodes and resources. In order to manage load balancing in the system efficiently, they used a technique to minimize cluster nodes and switching idle nodes off. They proposed a load distribution algorithm with cluster configuration under trade-off between performance (execution time & throughput) and power. Based on the performance expectation of the system, the algorithm monitors load of the resources and makes decision to turn on or turn off nodes dynamically for each cluster configuration. In comparison with static cluster configuration, authors claim that the proposed approach allows saving 43% and 86% of energy and power consumptions respectively. This system can be implemented in multi-application environment. However the algorithm executes on primary node and it may become a bottleneck of performance and create a single point of failure as well. On the other hand at a time one node has been added or has been removed by the algorithm whereas this method is not able to react immediately in large scale environments.

The issue of energy consumption in Internet hosting centers has been analyzed by Chase in (Chase et al., 2001). The main objective of this research work is to manage energy usage resource management frameworks in data centers. The authors propose resource management architecture for adaptive resource provisioning in data centers based on economic approach. This system is called Muse and it is based on executable utility function that allows measuring performance value of each service. The main challenge is how to find out the request of resources for each customer and also how to allocate these resources in an efficient way. In this approach system monitors load of resources and allocate resources based on their affection on service performance. In order to allocate resource efficiently, a greedy algorithm for resource allocation have been used to maximize profit by balancing the estimated revenue against resource unit. In order to solve the problem of a “noise” in loading web data and reduce the number of inefficient allocation, statistical flip flop filter has been used. One of the advantages of this system is that an active set of servers can be changed by converting idle server to sleep mode in order to save the power consumption. In this approach, authors propose to manage only CPU usage. For a typical and representative web workload, their experimental results display that the energy consumption can be minimized from 29% to78%.

Raghavendra and Ranganathan in (Raghavendra et al., 2008) work on problem of heat management and electricity consumption in data center environments. They focus on average power optimization and cooling in data centers. Their objectives are to represent and validate a power management solution based on coordination of various approaches. Various controller structures at different kinds of power management key points have: objective functions, actuators, multiple levels and time constants. The main features of the proposed solution are: control-theoretic core which enable system stability and overloading of the channels which reduce the number of interfaces. An important step toward their implementation is CPU utilization results have been analyzed based on CPU usage. The authors claim that their approach enables 64% reduction of power consumption. In addition they find out that local power optimization could be useful during high workloads. However in this method CPU utilization is assumed for implementation and other resources have not been considered.

Another approach for emergent aware resource management is proposed by Cardosa et al. (Cardosa, Korupolu and Singh, 2009). The authors shows the maximum and minimum of CPU usage of each physical resource which allocate to VMs. The authors have used min, max and shares features for VM placement in data centers. The main objectives of this paper are: 1) Set Minimum, Maximum, and Shares parameters for VM placement in consolidation of resources 2) Present a resource allocation technique with increasing degrees of effectiveness 3) Provide an experimental evaluation. The authors claim that the proposed technique can be practical in real situation however VM allocation does not work dynamically and like previous researches just CPU usage is considered for resource allocation.

1.4.2.2 VM consolidation

Dynamic VM consolidation is a method which is able to improve the usage of resources efficiently by placing under loaded resources into their suspend state or idle states. In this method the resources are reallocated repeatedly based on current status of requests in order to inactive the number of physical servers which are not used on that especial time. If the number of requests increases that physical resources should be activated. The approaches of dynamic VM consolidation can be categorized into three different strategies (Beloglazov and Buyya, 2012a): 1) dynamic allocation of VMs at each period of time 2) heuristic-based approaches 3) decision-making based on historical data. These strategies are reviewed below.

Schwan and Natuji (Schwan, 2007) are considered power management in enterprise systems as main problem. The main objective is to optimize the solutions of virtualization technology to support policies for efficient power management. The proposed VM placement approach called VirtualPower and has two main ideas: 1) support isolated guest Virtual machines on virtual platform 2) manage the energy consumption of these VMs on virtual platforms. In a VirtualPower architecture, a hypervisor and a controller, called Domain Zero (Dom0), are executed by each physical platform. In this approach the management of resources are categorized into global and local policies. In local policy, the requests of power management are captured for other VMs. Then this information is sent to components of power management software for making management decisions. Global policies are responsible for reallocating VMs via live migration. To evaluate the validity of proposed approach, the new power-efficient of Intel Core architecture has been used to show the benefits of power management through VirtualPower. The authors claim that the proposed approach enables 34% of power consumption without degrading system performance. However in this approach the global policies are not elaborated in details based on QoS parameters.

Gmach and Rolia in (Gmach et al., 2008) have proposed a trace-based dynamic VM consolidation approach to achieve better efficiency and effective application's QoS. In this approach, a VM placement controller is integrated with a reactive controller in order to balance the workloads of overloaded servers and switch off under loaded servers. To evaluate this approach, a host load environment is proposed that it is able to evaluate the combinations of controllers on different QoS parameters and also to evaluate the influence of management policies in long term usage. Moreover server and blade resource pool infrastructure are applied in this approach. Three months data for 138 SAP applications is used to compare this method with usage of each controller separately. Their result shows, CPU quality is increased to 20% by integrated controllers in blade pool environment (rather than separated controllers). In the server resource pool the penalty of hourly CPU quality was seven times better than the separated controllers. However, in this approach the main focus was on CPU usage and the impacts of other resources have not been analyzed. For instance, overloaded resources are assumed to have CPU usage between (85%, 95%) whereas other parameters should be considered as well (e.g. network bandwidth, memory and storage).

Gmach and Rolia have also proposed different VM migration and VM placement strategies (Gmach et al., 2009). They described an approach for analyzing the impact of policies for resource pool management by combination of migration controller to choose appropriate policy based on given resource pool strategy. In addition a reactive migration controller was proposed to detect overloaded and under loaded hosts. When the demands for resources exceed a certain threshold, the workload migration is initiated and in order to keep a balance of supply, servers are added or are removed dynamically. Like their previous paper they have uses 138 SAP application with three months data to evaluate their approach. The results showed that a proactive workload placement or reactive workload placement alone are not accurate for efficient resource pool management. However this approach cannot be applicable for various types of applications in IaaS environments because each work load type should be tuned in order to have efficient consolidation controller.

Zhu and Young (Zhu et al., 2009) worked on the issue of resource management for mapping virtual resource to physical resources. They have represented an automated workload management system for integrating different resource controllers into three various time scales and scopes. The architecture of their approach, called 1000 islands, is designed with three individual controllers: 1) “node controllers” which adjust resource allocation to VMs dynamically 2) “pod controller” which manage domain of workload migration with multiple nodes 3) “pod set controller” which analyze the resource consumptions based on historical data. This method allows integrating different islands through the workload management. The obtained result shows that this solution enables efficient resource usage in data centers and it can reduce the violations of services in important applications. With the usage of 27% more capacity in unified architecture, the quality of CPU and memory have been improved. Overall the result shows that the proposed approach could improve the performance to 32% over the fixed static allocation and 23% over separated controllers. However in this research one loos integration policy with static threshold is evaluated for different controllers. For instance utilization threshold of 85% for CPU is assumed as overloaded host whereas other major parameters are not considered.

In order to reduce power usage and numbers of SLA violations, an adaptive heuristic data based on the analysis of historical data has been proposed by Beloglazov in (Beloglazov and Buyya, 2012c). To solve the problems of dynamic consolidation and VM migration, various analysis have been proposed based on optimal online deterministic algorithms. In their approach, both static and dynamic amounts are considered as threshold for dynamic consolidation of resources. This approach is using upper and lower thresholds of utilization for physical servers. Consequently, if a CPU usage in one server is less than the lower threshold, all virtual machines are moved to another server to reduce energy consumption. On the other hand if the CPU utilization is more than the upper threshold, some virtual machines are migrated from the server in order to prevent SLA violation. The proposed algorithm is evaluated in a simulation environment, called Planet Lab, with more than a thousand VMs based on large-scale environments. This approach allows minimizing the number of VM migrations and preventing SLA violations. However in this method CPU utilization is considered as a threshold parameter. In addition this system has not been implemented in real environment and need further analysis with more complex workloads.

Belaglazov and Buyya (Beloglazov and Buyya, 2012c) have proposed an approach based on a Markov chain and multi size sliding window. In this method system administrator is able to set QoS target based on the independent QoS parameter called Overload Time Fraction (OTF). In order to handle a known stationary workload, Markov chain model detects overloaded host by maximizing the mean inter-migration time under QoS constraints. In order to control unknown workloads an estimation technique of “Multisize Sliding Window” is used based on heuristic adaption. For evaluating the performance of MHOD algorithm, an “optimal offline algorithm” is introduced in this research. This algorithm has been tested in PlanetLab with more than a thousand virtual machines. The result shows “optimal offline algorithm” has better performance (12%) than MHOD algorithm. Due to the usage of Markov chains some limitations and assumptions have been applied that it may not be practical for all types of workloads. For instance, a migration of a single VM and known workloads are assumed to implement the Markov chain model. In addition CPU usage is considered as a single metric for overload detection whereas there are other parameters which should be considered as well.

Jinhua et al. (Jinhua et al., 2010) have proposed a genetic algorithm for solving the problem of dynamical resource management and efficient resource allocation. The main objectives of this research are: reduce migration cost, achieve the best load balancing and introduce load variation rate. In this approach a VM resource scheduling approach is proposed based on genetic algorithm. After initialization of cloud computing environment, genetic algorithms look for the best solution in every scheduling. When VM resources increase the approach works based on the current state of system and analyze historical based on genetic algorithm to choose the best solution with the least impact on the system. In order to evaluate the proposed algorithm, the Platform ISF and OpenNebula platform have been chosen to show the performance of the algorithm. Different parameters have been used for analyzing algorithm effect based on migration cost and utilization rate. The authors claim that the result could achieve proper resource utilization and better load balancing. However there is no monitoring and analyzing mechanism in this approach. Because in real cloud computing environment, VMs might change dynamically and we should be able to control the system behavior to avoid any unpredicted incident.

Another heuristic approach based on improved genetic algorithm is introduced by Zhong (Hai, Kun and Xuejie, 2010). This research addresses the issue of resource scheduling in cloud computing environments. In order to minimize resources wastage in cloud environments and achieve an optimal VM allocation, Genetic Algorithm (GA) has been used on IaaS structures and an optimized scheduling algorithm is proposed through an improved genetic algorithm (IGA). This algorithm enables optimal VM allocation based on VM request and economic policy. The scheduling method is categorized into three parts: 1) While allocating of VMs happen, the list of available resources is updated by scheduler 2) IGA (Improved Genetic Algorithm) is applied to figure out an optimal fitness function 3) whenever the leasing time's up, VMs are suspended and cloud established the matched physical resources. A simulator has been developed by Eucalyptus, to compare this approach with Round robin and Greedy algorithms. In addition the result is compared with traditional GA algorithm and it shows that the result of IGA has better performance (almost twice) than traditional GA and authors claim the utilization rate of computing resources has been improved as well. However, in scheduling algorithms different parameters should be considered such as time, cost, scalability, availability, reliability, speed and resource utilization. In this approach, only the last two parameters (Speed & Resource Utilization) have been considered.

In (Dutta and Joshi, 2011) a resource scheduling approach based on genetic algorithm has been proposed for cloud computing environments. In this approach, different QoS factors have been considered. Their proposal model made based on five components: 1) A set of customers 2) Task classifier 3) Data Center executer 4) Data Center Manager 5) Job Scheduler. In order to provide a better solution for scheduling problem, some genetic cross over operators are applied in this method such as OX, CX, and PMX. The authors claim that both Cloud providers profit and QoS requirement of users are considered in this approach. However the job allocations are considered as independent, divisible and non-pre-emptive. Moreover, other limitations of real time situations are not assumed in this study (such as machine failure).

In (Xindong et al., 2009), the authors addressed the issue of resource allocation in distributed systems. The main objective of this research is to improve utilization of resources in large-scale data centers based on QoS constraints. A resource allocation strategy based on market (RAS-M) model is introduced in order to achieve better resource usage. Firstly, a QoS based function is defined based on different requirements of users' requests. Secondly, a GA algorithm is proposed for adjusting price to balance demand prices. Finally, Xen virtualization technology is used this approach (RAS-M) for allocation the weight of VMs. In order to validate their approach, authors simulated four different VMs with four types of workload on Xen. Different prices of CPU are initialized based Agent of CPU at different steps. The authors claim that the approach enables maximising the usage of all Consumer Agents. However this approach is only used CPU resource and implemented on the lowest level of Cloud computing and other resources (Storage, Memory, and Network bandwidth) have not been considered in this model.

1.4.3 Open issues of resource management in cloud computing

In this section, several open issues for management of resources in cloud computing environments have been discussed. Dynamic VM consolidation approaches allow minimizing resource wastage and reducing energy consumption by putting switching unused nodes to idle mode. However, reducing energy consumption by mean of resource consolidation may degrade system performance and violate SLAs. So the optimal resource management algorithm should balance the energy consumption with system performance.
Several resource allocation strategies focused on increasing performance and don't take into consideration energy consumption. Few of them focuses on saving energy. However, they

have some limitations. Indeed, if we turn off some physical servers for saving energy in cloud computing environment, some VMs cannot receive the required resource in peak time. As a result the reliability and availability of the approach will be reduced and we cannot cover the desired QoS. So it is very important that resource management approaches pay attention to energy consumption and high performance at the same time. In addition multi objective approaches should considered in these approaches. For instance some algorithms focus on time or scalability or speed but they did not evaluate other metrics such as: resource utilization, consolidation cost, reliability and availability. Moreover, in order to have approach applicable in real environments, different resource parameters should be selected such as CPU, Memory, Storage and Network bandwidth.

One of the traditional algorithms for solving VM placement problems is greedy algorithms. These algorithms are less complex and can also be implemented easily than meta-heuristics algorithms. However, these algorithms are highly centralized and hard to distribute (Feller, 2013). First Fit Decreasing (FFD) is one of the well-known greedy algorithms for the VM placement problem. The complexity of these algorithms are presented in (Coffman Jr et al., 2013). In (Stillwell et al., 2010) and (Stillwell, Vivien and Casanova, 2012) the results of well-known greedy algorithms (FFD, Permutation Pack, and Choose Pack) are compared with each other. According to their simulation results, Choose Pack is faster than FFD and Permutation Pack. Choose Pack and Permutation Pack are two greedy algorithms proposed by Leinberger and Karypis in (Leinberger, Karypis and Kumar, 1999). Beloglazov in (Beloglazov, Abawajy and Buyya, 2012) proposed Modified Best Fit Decreasing (MBFD) for the VM placement algorithm based on CPU utilization. This approach has been evaluated using Cloudsim based on energy consumption, SLA violations, and VM migrations. Beloglazov in (Beloglazov and Buyya, 2013) investigates the impact of overload detection algorithms on the quality of VM consolidation. Due to the usage of Markov chains, some limitations and assumptions have been applied that may not be practical for all types of workloads. For instance, a single VM migration and known workloads are assumed to implement the Markov chain in this model.

As VM placement problems are considered as NP-hard problems, meta-heuristic approaches are considered as good candidates for these problems. However meta-heuristic approaches work based on randomness and dynamic workloads. So we cannot expect them to find optimal solutions. An ACO-based approach is introduced based on a multi-dimensional bin-packing problem in (Feller, Rilling and Morin, 2011). In this approach, items are equivalent to VMs and bins are equivalent to PMs. To validate this approach, the authors compared their approach with CPLEX and FFD algorithm. The result shows that ACO-based approach enables better energy consumption than FFD. However, it is a single-objective algorithm.
Another ant colony algorithm for the VM placement problem is presented in (Gao et al., 2013) to reduce resource wastage and power consumption. The results have been compared with two other single-objective algorithms (SACO and FFD) and one multi-objective algorithm (MGGA). However, they didn't consider the communication cost between network elements nor describe the simulation tools that have been used in their evaluation. A single-objective genetic algorithm for VM placement in data centers is introduced in (Wu et al., 2012). Energy communication cost and power consumption are considered in their approach. According to their results, GA algorithm enables less energy consumption than FFD. However, in their objective function they have simple multiple-objective parameters in one function. According to their results, GA algorithm enables less energy consumption than FFD. They also used their own simulation tools (Mohammadhossein Malekloo, 2014).

1.5 Conclusion

Different strategies of resource management and dynamic resource consolidation have been discussed in this chapter. We analyzed different models for power and energy consumption management. We also reviewed two types of algorithms in cloud management systems which optimize the assignments of VM and PM: 1) VM placement algorithms and 2) VM consolidation algorithms. Both of these algorithms are categorized as NP-hard optimization problems.
To solve the VM placement problem, meta-heuristic algorithms (e.g., ACO and GA) are considered to find sub-optimal solutions. In (Feller, Rilling and Morin, 2011) an ACO-based

approach is presented. The aim of this approach relies on placing all items in the minimum number of bins. Then the result is evaluated by using FFD and CPLEX algorithms. The result shows better energy consumption than FFD does. However, this algorithm is a single-objective algorithm. The simulation java tools have not been elaborated clearly for use by other researchers. In (Gao et al., 2013) the authors proposed an ant colony algorithm for the VM placement problem based on two objectives: 1) Power consumption; 2) Resource wastage. The results have been evaluated with two other single-objective algorithms (SACO and FFD) and one multi-objective algorithm (MGGA). However, they have not considered the communication cost between network elements nor elaborated the simulation tools that have been used in their evaluations. In (Wu et al., 2012) the authors introduced a single-objective genetic algorithm for VM placement in data centers. They have considered power consumption and energy communication cost in their approach. According to their results, GA could reduce more energy consumption than FFD. However, in their objective function they have simple multiple-objective parameters in one function. They also used their own simulation tools (Mohammadhossein Malekloo, 2014).

Our main contributions are categorized in two main parts:

1) The proposal of a multi-objective optimization approach in order to minimize the total energy consumption of a data center, resource wastage and energy communication cost. Other multi-objective approaches have not taken into consideration communication network costs in the energy consumption of a data center. Our multi-objective algorithm considers this communication cost and it is also integrated with Cloudsim tools in order to emulate services and VMs as well as to map VMs to PMs. Another contribution is the comparison of the proposed approach with other meta-heuristic (MGA) approaches and other single-objective VM placement algorithms (FFD, DVFS, and LR). This approach is elaborated in Chapter 2 and in Chapter 4, the results have been analyzed.
2) Another contribution of this research is to extend ACO approach to consolidate VMs among PMs based on the proposed ACO placement algorithm. In particular, the new consolidation algorithm should take into consideration the VM-PM solution which

have been provided by placement algorithm and tries to optimize it. Our main contribution is the proposal of a multi-objective optimization approach to minimize the total energy consumption of a data center, minimize number of migrations, minimize SLA violations and minimize number of PMs. Our multi-objective approach is integrated with Cloudsim tools in order to optimize the assignment of VMs and PMs. Another contribution is the comparison of the proposed approach with other meta-heuristic Genetic Algorithm (MGA) approaches and a Multi-Objective ACO which has been proposed by Feller. Also the results have been compared with other single-objective VM consolidation algorithms (FFD and Single Threshold). In Chapter 3 we elaborate this approach and In Chapter 4, the results have been analyzed.

CHAPTER 2

MULTI-OBJECTIVE META-HEURISTIC VM PLACEMENT ALGORITHMS

2.1 Introduction

As discussed earlier, there are two key algorithms in cloud management systems which optimize the assignments of VM and PM: 1) VM placement algorithms and 2) VM consolidation algorithms. Both of them are time and resource consuming algorithms and are categorized as NP-hard optimization problems because these problems cannot be solved in specific time. In other words, these problems are able to be solved in polynomial time. In this chapter, we present a multi-objective Ant Colony Optimization (ACO) placement algorithm. This algorithm is compared with multi-objective Genetic Algorithm (GA).

This chapter is organized as follows. Section 2.2 presents the problem statement and assumptions. Section 2.3 gives the mathematical model. Section 2.4 presents the methodologies which have been used to analyze the proposed placement algorithm. Finally Section 2.5 presents the conclusion of this chapter.

2.2 Problem statement and assumptions

In (L. Minas 2009), the authors show that the main part of servers' power are consumed by CPUs. If server is in low-activity mode, CPUs might consume less than 30% of power and in high-activity mode CPUs can consume more than 70% of the power. Hence, in this report, we consider the CPU as the main resource metrics. However, this algorithm can be extended to support other resource metrics as well. In addition, we assume that VMs and PMs are heterogeneous and that VMs are independent from each other and no VM is associated with other VM. Moreover, in initialization phase, PMs can be either empty or pre-filled. In this approach we assume PMs do not host any VMs as starting point. Table 2.1 shows the notations that have been used.

Table 2.1 Notations used in VM placement algorithm

N	$Number\ of\ VMs, i = 1,2, \dots, N$
M	$Number\ of\ PMs, j = 1,2, \dots, M$
S	$Number\ of\ Network\ equipment, s = 1,2, \dots, S$
NE	$Network\ equipment$
R	$Set\ of\ resource\ needed$ $by\ VM_i \quad [R_i^{VM}]_{N \times K}$
K	$Number\ of\ resources\ available\ in\ a\ VM$
R^{PM}	$Set\ of\ resources$ $available\ in\ PM_j\ \left[R_{j,k}^{PM}\right]_{M \times K}$
R^{NE}	$Set\ of\ resources$ $available\ in\ NE\ (just\ for\ CPU)$
l	$Number\ of\ resources\ available\ in\ a\ PM$
T^{VM}	$Matrix\ of\ communication\ traffic\ load$ $\left[T_{i,h}\right]_{N \times N}\ between\ VM_i\ and\ VM_h$
$x_{i,j}$	$\begin{cases} 1\ if\ VM_i\ is\ assigned\ to\ PM_j \\ 0\ if\ VM_i\ is\ not\ assigned \end{cases}$
y_j	$\begin{cases} 1\ if\ PM_j\ is\ active \\ 0\ if\ PM_j\ is\ idle \end{cases}$
z_s	$\begin{cases} 1\ if\ NE_s\ is\ active \\ 0\ if\ NE_s\ is\ idle \end{cases}$
x_{is}	$traffic\ of\ VM_i\ is\ transferred\ over\ NE_s$

2.3 Objective function formulation

In a cloud computing environment, pool of resources in multiple physical machines is shared among different virtual machines that host different applications. The VM placement algorithm is defined to minimize energy consumption, minimize resource wastage and minimize communication cost of network elements within a data center. We formulate our objectives as follows.

2.3.1 Minimize energy consumption

We assume that N is the number of virtual machines and M is the number of physical machines. Also we assume that R represents the set of resources needed by each VM. The variable of y_j represents if PM_j is active or not and the variable x_{ij} indicates whether $VM\ i$ is assigned to PM_j or not. Our first objective is to minimize the energy consumption of a data center based on the formula illustrated in (2.1) and given in (Gao et al., 2013):

$$P_j = (P_j^{busy} - P_j^{idle}) \times U_j^P + P_j^{idle} \tag{2.1}$$

Where U_j^P is the CPU utilization ($U_j^P \epsilon [0,1]$) and P_j^{busy} and $P_j^{idle}$are the average power values when the $j-th$ PM is busy and idle, respectively. In our simulation experiments, both values have been fixed at 215 and 162 Watts (Feller, Rilling and Morin, 2011). The first formula for calculating the total energy consumption is presented in (2.2) and is described in (Gao et al., 2013). In this formula, P_j^{PM} is energy consumption of $j-th$ PM and $R_{i,1}^{VM}$ is a set of CPUs needed by VM_i :

$$Min \sum_{j=1}^{M} P_j^{PM} = Min \sum_{j=1}^{M} \left[y_j . \left[\left(P_j^{busy} - P_j^{idle}\right) \times \sum_{i=1}^{N} \left(x_{ij} . R_{i,1}^{VM}\right) + P_j^{idle} \right] \right] \tag{2.2}$$

2.3.2 Minimize resource wastage

Our second objective is to minimize resource wastage. We extended the formula that proposed in (Gao et al., 2013). If a PM has available resources but it's not used by any other VM, we consider that available resources of that PM are wasted. One of our objectives is to minimize these kinds of wastages in VM placement. In the follow equation, W is the resource wastage of each PM and R^{PM} is a set of resources available in PM_j. Also in (2.3), $R_{i,1}^{VM}$ represents the set of CPU resources requested by VM_i (Gao et al., 2013):

$$Min \sum_{j=1}^{M} W_j^{PM} = Min \sum_{j=1}^{M} \left[y_j . \left(\frac{R_{j,1}^{PM} - \sum_{i=1}^{N} \left(x_{ij} . R_{i,1}^{VM} \right)}{\sum_{i=1}^{N} \left(x_{ij} . R_{i,1}^{VM} \right)} \right) \right] \tag{2.3}$$

2.3.3 Minimize energy communication cost

The third objective is energy communication cost in order to consider this parameter for placement of VMs between PMs. In this report, we consider hierarchical topology between PMs, Routers, and Switches in a data center. In the following equation we assume T as traffic load communication between two VMs. In (2.4) we used the power model implemented in Greencloud simulator (Kliazovich et al., 2010). In this formula, P_s^{busy} and P_s^{idle} are the average power of Network equipment values when the s-th NE is busy and idle, respectively. $T_{i,h}^{NN}$ is the matrix of communication traffic load between VM_i and VM_h:

$$Min \sum_{s=1}^{S} P_s^{NE} = Min \sum_{s=1}^{S} \left[z_s . \left[\left(P_s^{busy} - P_s^{idle} \right) \times \sum_{i=1}^{N} \left(x_{is} . T_{i,h}^{NN} \right) + P_s^{idle} \right] \right] \tag{2.4}$$

We use k-shortest path algorithm to determine the number of NE between two VMs. Moreover, the follow constraints have been defined for this multi-objective optimization problem:

- $\sum_{j=1}^{M} x_{ij} = 1 \; each \; VM \; can \; be \; hosted \; in \; only \; one \; PM$
- $\sum_{i=1}^{N} R_{i,1}^{VM} . x_{ij} \le R_{j,1}^{PM} . y_j$

$(CPU \; resource \; consumed \; by \; VMs \; are \; less \; than \; resource \; allocated \; to \; PM \; hosting \; these \; VMs)$

- $z_s, y_j, x_{ij} \in \{0,1\}$

nCycles	Number of cycles
nAnts	Number of ants

2.4 Methodologies

2.4.1 Multi objective optimization

In order to apply the multi-objective method, we have used a population-based approach which finds Pareto solutions. Most of the current multi-objective algorithms use dominance concepts during their selection to find Pareto optimal solutions based on population based approach (Gao et al., 2013). A solution x_1 is considered as dominating another solution (for example x_2), if the conditions given below are true: 1) in all objectives, the solution x_1 is not worse than x_2; 2) in at least one objective, the solution x_1 is strictly better than x_2. Moreover, non-dominated solutions are all those that are not dominated by any other member of population. The solutions that are located on the non-domination front are not dominated by any other solution. Together, these solutions make up the Pareto optimal set. They are called Pareto optimal solutions.

2.4.2 Multi objective ACO placement (MACO)

The multi-objective ACO placement algorithm is based on the Pareto front method. Using Pareto front method, it is possible to obtain non-dominated resolutions which minimizes our objective functions (Gao et al., 2013). At each step this formula chooses a candidate based on the combination of pheromone factor and the heuristic factor that guides ants how to choose proper VM based on PM utilization. In (Feller, Rilling and Morin, 2011), the ACO-based approach is introduced as an instance of the multi-dimensional bin-packing problem. However, this method is modeled as a single-objective method for minimizing the number of physical machines. In our algorithm, we define multi-objective optimization and we propose to adapt the probabilistic decision rule and heuristic information formula given in (Feller, Rilling and Morin, 2011) to our problem. On the other hand we used the same objectives in Genetic algorithms. Genetic algorithms find results based on Chromosomes and ACO algorithms find results based on ant solutions. A set of encoded strings which represent solution, a decision vector, and assign a fitness value is called Chromosomes. Each chromosome composes of genetic strings and the number of genetic strings is determined by the type of virtual machine. To map the VM placement problem into a correspondent ant solution and chromosome, we represent it by string of naturel numbers. For instance, assume that an ant solution gives the string 1 2 2 3 4 4 3 1 as a final solution. This means that eight VMs are partitioned into four PMs. The first index of the solution shows that the VM1 is mapped to the PM1. The second index shows that the VM2 is mapped to the PM2; the third index shows that the VM3 is mapped to the PM2 and so on (see Figure 2.1).

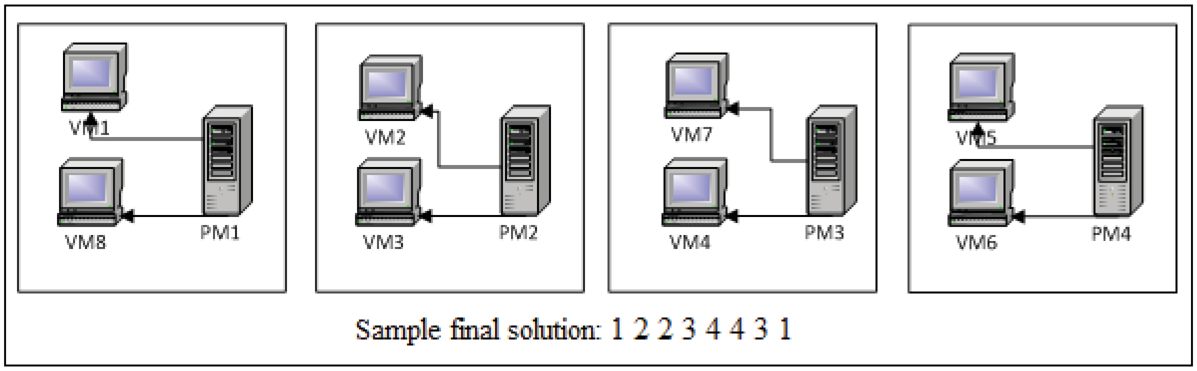


Figure 2.1 An example of corresponding ant solution in VM placement algorithm

In reality ants communicate with each other by depositing pheromone (a chemical substance) to find shortest path and find their foods. In our algorithm we used the same idea as pheromone factor. Also heuristic is used to push VMs and PMs to find better solution based on the objectives of algorithm. Moreover, probability decision rules are used to calculate probability for each VM and PM within a matrix and based on these matrix ants are able to choose the next VM and PM for completing placement process. At each step the ACO algorithm chooses a candidate based on pheromone factor and heuristic factor. Equation (2.5) shows the probabilistic formula used and which is described in (Feller, Rilling and Morin, 2011).

$$p_p^v := \frac{[\tau_{v,p}]^\alpha \times [n_{v,p}]^\beta}{\sum[\tau_{v,p}]^\alpha \times [n_{v,p}]^\beta} \tag{2.5}$$

In this equation $\tau_{v,p}$ shows the amount of pheromone in a set of VM and PM. This equation is a probability formula that shows the probability of VM v to be hosted into PM p. In this equation there is a heuristic factor $n_{i,v}$ that finds solutions with less resource wastage and less energy consumption (2.6).This factor applied in the probabilistic decision formula in order to build solutions. The current pheromone factor and a heuristic guide the ants to choose VM-PM based on their probability amounts.

Heuristic information is used to favor VMs which utilize better PMs. In (2.6), $n_{i,v}$ is a modified factor which presented in (Feller, Rilling and Morin, 2011). We defined inverse of scalar valued difference between resource wastage and energy consumption based on our objectives. It means VM-PM with less resource wastage and less energy consumption have higher amount of $n_{i,v}$. C_p is capacity of each PM and b_p presents the load of each PM based on CPU usage (1 means CPU). The r_v represents the requested number of MIPS for a VM. Also E_j is energy consumption of server j and E_{max} is maximum energy consumption of each server (We assumed 250 Watt as a fixed amount for this parameter).

$$n_{v,p} := \frac{1}{\left|C_p - (b_p - r_v)\right|_1} + \frac{1}{\sum_{j=1}^{p} \frac{E_j}{E_{max}}} \tag{2.6}$$

In addition, two parameters $\alpha, \beta \geq 0$ are used to emphasize more power to pheromone factor or heuristic factor (Feller, Rilling and Morin, 2011).

Different definition of pheromone trail can impact on optimization of ACO. In order to find sub optimal solution we find better solutions through updating pheromone trails in each cycle. The equation (2.7) is used to increase the learning curve of ants in order to adapt ants' solutions with changing environments (Feller, Rilling and Morin, 2011). The ρ is the pheromone evaporation parameter and $\tau_{v,p}$ is applied to simulate evaporation rate for finding sub optimal solutions using the energy consumption terms and resource wastage.

$$\tau_{v,p} \coloneqq (1-\rho) \times \tau_{v,p} + \frac{1}{f(S_{best})} \tag{2.7}$$

$$, f(S_{best}) = Min \sum_{j=1}^{M} P_j^{PM} \times Min \sum_{j=1}^{M} W_j^{PM} \times Min \sum_{s=1}^{S} P_s^{NE}$$

A set of proper parameters allow getting better results. Table 2.2 lists the parameters of ACO placement algorithm which we set in our ACO algorithm.

Table 2.2 Parameters of ACO placement algorithm

α	β	ρ	nCycles	nAnts
1	2	0.5	10	5

2.4.2.1 ACO pseudo code

The pseudo-code for the ACO algorithm presented in Algorithm 1, is a modified version of the algorithm given in (Gao et al., 2013). It takes the VM requests and PM requests as inputs. Then the algorithm iterates based on the number of times the nCycle performs. During iteration, ants try to find appropriate PMs and build their own solution until the current PM has the capacity to host the new VMs. This process continues until all of the VMs are assigned to the appropriate PMs (according to the probabilistic decision rule). When all ants find their solutions, we use the Pareto front to find non-dominant solutions as outputs of the algorithm. At each step this formula chooses a candidate based on the pheromone factor and the heuristic factor**.** In this equation, $\tau_{v,p}$ is the pheromone factor and $n_{i,v}$ is the heuristic factor. The formula in Line 9 calculates probability decision rule and heuristic information. In this line, C_p is the capacity of PMs' processors and b_p is the CPU usage of the PM. The r_v represents the requested number of MIPS for a VM. Also E_j is energy consumption of the server j and E_{max} is the maximum energy consumption of each server. The pheromone trail update formula is located in Line 26. The quality of the ACO implementation relies on the definition of the pheromone trail. In order to find the sub optimal solution we find better solutions by updating pheromone trails in each cycle.

Algorithm1 Multi-objective ACO

```
1.    Input: Initialize, $n$ (number of VMs), $m$ (number of PMs), $nAnts = 5$, $\alpha = 2$, $\beta = 1$, $\rho = 0.5$, $nCycles$ =10, $p = 1$ (number of allocated PMs), $b = 0$ (PM load), $\tau_0 = 1$, $S_{cycle} = [\,]$, $S_{best} = [\,]$
2.    Output: $S_{best}$ (A set of strings that shows which VM is assigned to which PM)
3.     for all t∈$(1..nCycles)$ do
4.      for all $k \in (1..nAnts)$ do
5.       $x =$ random positions of ants
6.       Update $b$ parameter based on the new loads on PMs
7.        while $n <> 0$
8.         $N =$ the location of VMs which can be hosted and have not been hosted
9.  if $N <> 0\ and\ r_v(x) \leq C_p$ then
10.
    $ant(k).p_p^v \coloneqq Call\ Probability\ function\ to\ calc\ \frac{[\tau_{v,p}]^\alpha \times [n_{v,p}]^\beta}{\sum_{u \epsilon N_p}[\tau_{v,p}]^\alpha \times [n_{v,p}]^\beta}$,
$\forall i \in N_p\ , n_{v,p} \coloneqq \frac{1}{|C_p-(b_p-r_v)|_1} + \frac{1}{\sum_{j=1}^{p}\frac{E_j}{E_{max}}}$
11.       ant(k).Tour🡐 call function Probability($ant(k).p_p^v$)
12.       if $r_i(ant(k).Tour) \leq C_p$ then
13.         $Put\ 1\ in\ solution\ matrix\ for\ element\ ant(k).Tour\ and\ v$
14.         $b = b + r_v(x)$ /*Update PM load*/
15.       else
16.          $p = p + 1$
17.       end if
18.      end if
19.    end while
20.       update PM capacity
```

```
21.     end for
22.     Calculate objective functions according to 2.2 and 2.4 for current ant population
23.     $ant(k).Sa \leftarrow calculate\ objective\ function\ f$
24.     If a solution is not dominated by any other solutions and the non-dominated
      solutions in Pareto set the solution is added to Pareto set.
25.      for each non-dominated solution of Pareto set do
26.                $\tau_{i,v} \coloneqq (1-\rho)\tau_{v,p} + \frac{1}{f(S_{best})}$
27.      end for
28.     end for
29.     return $S_{best}$
```

2.4.3 Multi objective GA placement (MGA)

We propose to another multi-objective approach based on Genetic Algorithm (GA), which is a stochastic search engine operating on a population of potential solutions in order to find more appropriate solution. Based on the fitness functions, a new set of approximations is created at each generation. GA algorithm is based on the following techniques described below.

1) Chromosomes

Chromosomes are a set of encoded strings (or parameters or data structures) which represent solution for the problem which should be solved.

2) Crossover

Through Crossover, chromosomes can vary from one generation to another generation. So Cross over is an operator which reproduces child chromosomes from parent chromosomes. Then we are able to find new solutions for our problem based on new generated child

chromosomes. There are different methods for selecting a chromosome as parent chromosome (Obitko, 2011).
In this approach we used Heuristic crossover method defined in Matlab (The MathWorks, 2014) which returns a descendent throughout two parents. The parent with the better fitness value has a small distance rather than the parents with the worst fitness value. For instance if parent1 finds better fitness result than parent2, the child will be generated as shown in (2.8):

$$Child = Parent2 + R \times (Parent1 - Parent2) \qquad (2.8)$$

In this equation, R is Ratio which can be row vector or a scalar value of length number of variables. $Parent1$ and $Parent2$ are two parents chromosomes.

3) Mutation

The purpose of mutation in genetic algorithms is introducing the total number of genetic characteristics, called genetic diversity. Genetic diversity is used to adapt population of chromosomes with changing environments. So mutation is an operator that keeps genetic diversity from one generation to another one (Obitko, 2011). According to our different tests of our algorithm with different parameters, we chose adaptive feasible mutation function of Matlab which creates the mutation children based on adaptive mutation and mutated genes should satisfy linear constraints.

2.4.3.1 Multi objective GA (MGA) pseudo code

MGA is a multi-objective optimization algorithm where the input arguments are 1) pop - Population size and 2) gen - Total number of generations. It allows finding the sub-optimal solution for various objectives i.e. Pareto front set. Initially enter only the population size, the stopping criteria and the total number of populations in order to stop algorithm automatically. Table 2.3 shows parameters that have been set in MGA placement algorithm. We run 10 cycles for each test and chose the best solution among 10 cycles.

Table 2.3 Parameters of GA placement algorithm

Cycle	10
Population Size	200
Pareto Fraction	0.7
Migration interval	20
Migration fraction	0.2
Crossover	Heuristic
Mutation	Adapt feasible

Algorithm2 Multi-objective GA

```
1.  Input: System state which includes of a set of VMs which already assigned to PMs,
    Initialize Population (Generation's number , Population Size, Crossover rate, and
    Mutation rate)
2.  Output: Global best system state
3.  for all $i \in (1..numberofgenerations)$ do
4.      for all $j \in (1..population\_size * crossover\_rate)$ do
5.              selection;
6.              crossover;
7.      end for
8.      for all $j \in (1..population\_size * mutation\_rate)$ do
9.              selection;
10.             mutation;
11.     end for
12.     system_state=Evaluation
13. end for
14. return system_state
```

2.5 Conclusion

In this chapter we proposed the multi-objective ACO placement algorithm aiming to minimize the energy consumption of PMs, minimize the resource wastage of PMs and minimize the energy communication cost between network elements of a data center. Moreover, we presented another multi objective genetic algorithm based on the Matlab optimization toolbox with the defined objectives function. The Pareto optimal approach has been applied to combine the various objectives and find a set of non-dominated solutions.

CHAPTER 3

MULTI-OBJECTIVE META-HEURISTIC CONSOLIDATION ALGORITHMS

3.1 Introduction

One of the main approaches for the reduction of energy consumption is to minimize resource wastage by turning off or to suspend unnecessary servers. As explained in Section 1.5, two main algorithms for minimizing resource wastage in a cloud management system have been proposed: VM placement and Resource consolidation. In this chapter, an algorithm for resource consolidation is presented. The main challenge is to decide which resources should be migrated and also the moment when these resources should be consolidated.

In this chapter, we extend ACO placement approach to be able to consolidate VMs among PMs. In particular, the new consolidation algorithm should take into consideration the VM-PM solution which has been provided by placement algorithm and tries to reassign them based on new objectives. Our main contributions are the proposal of a multi-objective optimization approach to minimize the total energy consumption of a data center, minimize number of migrations, minimize SLA violations and minimize number of active PMs.

Another contribution is the comparison of the proposed approach with other meta-heuristic algorithms such as Genetic Algorithm (MGA) approaches and a Multi-Objective ACO which has been proposed by Feller (Feller, Rilling and Morin, 2011). Moreover, the results have been compared with other single-objective VM consolidation algorithms (FFD and Single Threshold).The chapter is organized as follows: Section 3.2 gives the problem statement and assumptions. Section 3.3 gives the new objective functions for resource consolidation problem. Section 3.4 presents the methodologies used. Section 3.5 summarizes this chapter.

3.2 Problem statement and assumptions

We assume that PMs and VMs are heterogeneous and the VMs are independent from each other. We assume that different PMs hosting VMs are set in a data center using MACO placement approach described in section 2. Only CPU metrics is considered in this approach. Table 3.1 lists the notations used.

Table 3.1 Notations used in VM placement algorithm

N	$Number\ of\ VMs, i = 1,2,\dots,N$
M	$Number\ of\ PMs, j = 1,2,\dots,M$
R	$Matrix\ that\ describes\ the\ set\ of\ resource\ needed\ by\ VM_i \quad [R_i^{VM}]_{N\times K}$
K	$Number\ of\ resources\ available\ in\ a\ VM$
R^{PM}	$Matrix\ that\ describes\ the\ set\ of\ resources\ available\ in\ PM_j \ \left[R_{j,k}^{PM}\right]_{M\times K}$
$x_{i,j}$	$\begin{cases}1\ if\ VM_i\ is\ assigned\ to\ PM_j \\ 0\ if\ VM_i\ is\ not\ assigned\end{cases}$
y_j	$\begin{cases}1\ if\ PM_j\ is\ active \\ 0\ if\ PM_j\ is\ idle\end{cases}$

3.3 Objective function formulation

We propose four main objectives for VM consolidation algorithm: 1) Minimizing the energy consumption of PMs; 2) Minimizing SLA (Service Level Agreements) violations; 3) Minimizing number of VMs migrations and 4) Minimize number of active PMs.

$$Min\ f\left(M_{global}\right) = Min\ f(EnergyConsumption, SLAV, |MP|, ActivePMs) \tag{3.1}$$

In our algorithm we consider the four objectives together to get output.

3.3.1 Minimize energy consumption

We use equations 2.1 and 2.2 to calculate the consumed energy. Our first objective is to minimize the energy consumption of a data center based on these formulas described in *(Feller, Rilling and Morin, 2011)*

3.3.2 Minimize number of SLA violations

In order to meet QoS requirements, we propose the use of SLA metric given in (3.2) which have been defined by Beloglazov in (Beloglazov and Buyya, 2012b). If the CPU utilization of an active PM reaches 100%, the performance level of service will be reduced and the SLA will be violated.

$$OTF(u_t) = \frac{t_o(u_t)}{t_a} \tag{3.2}$$

In this equation 3.2, t_a is the total time when a PM is being activated, and t_o is the total time when the CPU utilization of an active PM is being overloaded.

3.3.3 Minimize number of migrations

The proposed approach allows minimizing the number of VMs migration. In order to get to new solution starting from VM-PM assignment, new various migrations are required. The more number of migrations consume more resources and decrease productivity. So the minimum number of migrations is another problem that we should consider.

3.3.4 Minimize number of active PMs

The PMs that already hosted any VM, called active PMs. In order to get more validated results and evaluate different objectives accurately, we implemented two consolidation algorithms. One of them is implemented with three objectives (minimize energy consumption, minimize number of SLA violations and minimize number of migrations) and another one is implemented with four objectives (with minimize number of active PMs) and different simulation setup of Cloudsim.

The following constraints applied to the optimization problem:

- $\sum_{j=1}^{M} x_{ij} = 1 \; each\; VM\; can\; be\; hosted\; in\; only\; one\; PM$
- $\sum_{i=1}^{N} R_{i,1}^{VM} . x_{ij} \leq$
  $R_{j,1}^{PM} . y_j \; CPU\; resource\; consumed\; by\; VMs\; are\; less\; than\; resource\; allocated\; to\; PM\; hosting\; these\; VM$
- $y_j, x_{ij} \in \{0,1\}$

3.4 Methodologies

3.4.1 Multi-objective ACO consolidation algorithm

We now present the multi-objective ACO consolidation algorithm by using the Pareto front method.

As described in section 2.4.2, the probability decision rule is used to calculate probability of each VM- PM within a matrix to select VM-PM with higher probability. At each cycle this formula recalculated to find higher amounts of VM-PM and build global solution.

As already explained in section 2.4.2, heuristic information ($n_{i,v}$) is another important factor in ACO in order to guide VMs which utilize better PMs. We defined inverse of scalar valued difference between resource wastage and energy consumption based on our objectives. It means VM-PM with less resource wastage and less energy consumption have higher amount of $n_{i,v}$.

Equation (3.3) shows the inverse of scalar valued difference between resource wastage and SLA violation (SLAV). It means VM-PM with less resource wastage and less number of violation have higher amount of $n_{i,v}$. In the follow formula C_p is capacity of each PM and b_p CPU usage of each PM. The r_v represents the requested number of MIPS for a VM. Just the different between this parameter and heuristic information of ACO placement is SLAV parameter which has been added to this formula. SLAV is the number of SLA violations.

$$n_{v,p} := \frac{1}{\left|C_p - \left(b_p - r_v\right)\right|_1 \times SLAV} \tag{3.3}$$

In our ACO consolidation algorithm we used the same formula which presented in (2.7). We have tested ACO algorithms with different parameters for the same input with 10 iterations for the same parameters. Table 3.2 shows the parameters of ACO consolidation algorithm that used.

Table 3.2 Parameters of ACO consolidation algorithm

α	**β**	**ρ**	**nCycles**	**nAnts**
0.1	0.9	0.1	2	5

3.4.1.1 ACO pseudo code

The pseudo-code for the ACO algorithm presented in Algorithm 1 is taking the mapping of VM-PM for the previous placement solution as inputs. During iteration, ants try to select VM and PM based on probability function. Then add (VM, PM) to the new matrix called Migration Solution, if the new location of VM is different with PM, then the number of migration will be increased. This process continues at most n (number of VMs) migrations. When all ants find their solutions, we use the Pareto front to find non-dominant solutions as outputs of the algorithm. At each step this formula chooses a candidate based on the pheromone factor and the heuristic factor.

Algorithm2: Multi objective ACO consolidation algorithm

```
/*Initialization*/
1-  Input: Global solution $(v, p)$ that provided by ACO placement algorithm, Available
    PMs CPU, VMs requests
2-  Output: Migration Global Solution $(M_{global})$
3-  Initialize, $n$ (number of VMs), $m$ (number of PMs), $nAnts = 5$, $\alpha = 0.1$, $\beta = 0.9$,
    $\rho = 0.1$, $nCycles$ =2, $nAnts$ =5, $p = 1$ (number of allocated PMs), $b = 0$ (PM load),
    $\tau_0 = 1$, $M_{cycle} = [\,]$, $M_{global} = [\,]$, $M_a = [\,]$, $MP = 0$
/*Iterative*/
4-  for all t∈$(1..nCycles)$ do
5-  for all $k \in (1..nAnts)$ do
6-  while $MP < n$
7-  $ant(k).p_p^v \coloneqq Call\ Probability\ function\ to\ calc\ \frac{[\tau_{v,p}]^\alpha \times [n_{v,p}]^\beta}{\sum_{u \epsilon N_p} [\tau_{v,p}]^\alpha \times [n_{v,p}]^\beta}$
$\forall i \in N_p, n_{v,p} \coloneqq \frac{1}{|C_p - (b_p - r_v)|_1 \times SLAV}$
8-  Add $(v, p)$ to the $M_a$
9-  $MP \coloneqq MP + 1$
10- Update PMs capacity
11- Add selected $(v, p)$ to $M_a$
12- end while
13- end for
/*Evaluation*/
14- compare $M_a$ and choose the best one based on objective function:
$Min\ f(M_{global}) = Min\ f(EnergyConsumption, SLAV, |MP|, ActivePMs)$
15- if $f(M_{local}) \leq f(M_{global})$ then
16- $M_{global} \coloneqq M_{local}$
17- end if
/*Pheromone trail update*/
18- for all $v, p$ do
```

```
19- $\tau_{i,v} := (1-\rho)\tau_{v,p} + f(S_{best})$
20- **end for**
21- **return** $M_{global}$
```

Like previous chapter, we have used multi objective genetic algorithm of Matlab with the same objectives function in order to compare two multi objective Meta heuristics approach with each other.

3.5 Conclusion

In this chapter a multi objective ACO consolidation algorithm has been proposed to minimize the total energy consumption of a data center, minimize number of migrations, minimize SLA violations and minimize number of PMs. The methodologies and Algorithm's pseudo code have been elaborated.

CHAPTER 4

RESULT ANALYSIS

4.1 Introduction

This chapter presents the evaluation of proposed ACO placement algorithm and ACO consolidation algorithm. As testbed, we used Cloudsim tool to simulate a virtualized environment (Calheiros et al., 2011) and hierarchical topology (Kliazovich et al., 2010) to simulate network connectivity inside a datacenter. The proposed ACO placement algorithm (MACO) has been compared with single objective algorithms FFD (Yue, 1991), DVFS (Guérout et al., 2013), LR (Beloglazov, Abawajy and Buyya, 2012) and ACO (Feller, Rilling and Morin, 2011)) and a multi-objective GA algorithm of Matlab Optimization tools.
Also we have tested our consolidation algorithms in Cloudsim with different configurations (explained in Table 4.2) to make sure the proposed algorithms are able to work in different configurations of a data center.

First, the Cloudsim setups for two different scenarios are detailed. Afterwards, the simulation results of placement algorithm are analyzed. Then the simulation results of consolidation algorithms with different objectives are discussed.

4.2 Setup of the simulation environment

The Cloudsim provides VM migration strategy that allows moving a single VM. In order to analyze our approaches with different scenarios, we simulated a data center with two different setup configurations in Cloudsim. According to Table 4.1 each PM has one CPU core of 1000, 2000, or 3000 MIPS, 1 TB of storage, and 8GB of RAM. In addition each VM needs 1 GB of storage, 128 MB of RAM, and one CPU core with 250, 500, 750 or 1000 MIPS. Also we consider the Linux x86 operating systems and Xen VMM as characteristics of our data center. We tested ACO algorithms ten times with different parameters for the same input in order to find appropriate parameters (ex. $nAnts$, $nCycles$, Population size …). As a result, we set α , β , ρ , $nCycles$ and $nAnts$ to 1, 2, 0.5, 10 and 5 respectively based on trial and error. Also for the Genetic Algorithm we set 10 as the number of cycles. We considered the population size as 200, the Pareto fraction as 0.7, the migration interval as 20 and the migration fraction as 0.2. The crossover method was set to "heuristic" and the mutation method was set to "adapt feasible". A population is a set of solutions that have been chosen as candidate to optimize problem.

Table 4.1 Configuration setup 1 of Cloudsim tools

Simulator:	Cloudsim
Version:	3.0.3
Datacenter Characteristics:	MIPS: 1000, 2000 or 3000 Storage: 1 TB RAM: 10 GB BW: 100 GB OS: Linux System Architecture : x86 VMM : Xen
VMs properties :	MIPS: 250, 500, 750, 1000. Storage : 1 GB RAM: 128 MB BW: 2500 MB Image Size: 2500 MB VMM: Xen
Cloudlet (Task) Properties:	Length: 15000 PEs (processing elements) number: 1 File Size: 300 Output Size:300

Unlike Table 4.1, we have changed Cloudsim configuration for another scenario to make sure the proposed algorithms are able to work in different configurations of data center. In Table 4.2, we have changed CPU configuration of data center, CPU requests of VM, size of RAM for VMs, Virtual machine monitor and number of PEs.

Table 4.2 Configuration setup 2 of Cloudsim tools

Simulator:	Cloudsim
Version:	3.0.3
Datacenter Characteristics:	MIPS: 4000, 1000, 3000, 2000 Storage: 1 TB RAM: 10 GB BW: 100 GB OS: Linux System Architecture : x86 VMM : KVM
VMs properties :	MIPS: 100, 400, 800, 1200 Storage : 1 GB RAM: 2048, 4096 MB BW: 2500 MB Image Size: 2500 MB VMM: KVM
Cloudlet (Task) Properties:	Length: 15000 PEs (processing elements) number: 2 File Size: 300 Output Size:300

4.3 Simulation results of placement algorithms

For first scenario we run the simulation for different numbers of VMs and PMs for up to 700 PMs, 1000 VMs and 1500 Cloudlets (tasks). We measured energy consumption, the number of active PMs and the amount of resource wastage for five placement algorithms: FFD, DVFS, LR, ACO and GA. FFD algorithm is a single objective algorithm that the inputs are resource requirements of VMs (with sorting order and one by one) and algorithms find the first available PMs for each VM. DVFS or Dynamic Voltage and Frequency introduce a trade-off between the energy consumed by the PM and computing performance. In Cloudsim, DVFS concept has been used as power aware algorithm but the VMs are processed as same order as they created in Cloudsim. Local regression (Gmach et al., 2008) is another VM allocation algorithm which is implemented in Cloudsim version 3. According to (Beloglazov and Buyya, 2012c), the Local Regression (Gmach et al., 2008) is used to fit models to build up a curve that helps to estimate the original data.

To evaluate our ACO multi-objective approach with another approach, we used GA in the Matlab Optimization toolbox with the same objective function. The numerical simulation results are shown in Table 4.3. In this Table, we have used a different number of Hosts (PMs), of VMs, and of Cloudlets (tasks). The different types of VM placement algorithms are compared with each other in terms of energy consumption in KW, resource wastage in percentage, number of active PMs that have been used, the amounts of energy gained (in comparison with FFD) and finally number of communication between source and destination that cause energy consumption and we assume as energy communication cost.

Table 4.3 Comparison of FFD, DVFS, LR and ACO algorithms

# of HOSTS	# of VMs	# of Cloudlets	Algorithm (Policy)	Energy, KW	Resource Wastage (%)	# of active PMs	Energy communication cost	Energy gain (%)
30	40	50	FFD	496.88	40.00	25	1458	
			DVFS	376.88	1.67	13	1232	24.15
			LR	312.64	3.33	13	1197	37.08
			MACO	294.38	0.00	12	1114	40.75
			MGA	300.47	21.66	14	1304	39.52
50	80	100	FFD	993.75	60.49	50	2783	
			DVFS	745.78	1.23	26	1490	24.95
			LR	683.81	1.23	26	1202	31.19
			MACO	594.38	1.23	24	1406	40.19
			MGA	796.88	30.86	38	1867	19.81
125	200	300	FFD	2484.38	59.05	125	3322	
			DVFS	1741	0.48	63	1896	29.92
			LR	1683.2	0.48	63	1226	32.25
			MACO	1455	0.00	58	1654	41.43
			MGA	2334.84	51.42	116	2462	6.01
250	400	500	FFD	4968.75	59.29	250	3875	
			DVFS	3480.94	0.48	126	2127	24.94
			LR	3365.57	0.48	126	1306	32.27
			MACO	3001	0.00	120	1334	39.62
			MGA	3897.19	29.76	188	1387	21.56
500	800	1000	FFD	9937.5	59.40	500	4159	
			DVFS	6905.16	0.12	251	2605	30.51
			LR	6725.03	0.12	251	2093	32.33
			MACO	6337.5	0.00	249	2214	36.23
			MGA	7868.91	29.76	375	2321	20.81
700	1000	1500	FFD	12421.88	44.60	625	5290	
			DVFS	8589.38	0.00	313	3035	30.85
			LR	8406.45	0.21	314	2260	32.33
			MACO	7833.75	0.07	307	2760	36.94
			MGA	9273.16	23.4	442	2915	25.34

In Figure 4.1, different algorithms are compared with each other in terms of energy consumption. Based on the results, FFD yields higher energy consumption due to the sorting mechanism of the VMs to the first available PMs without any attention to the resources available in other PMs. It means whenever FFD receives a new VM request (e.g. MIPS) as an input, it tries to find the first PM with enough resource for that particular PM. If it cannot find any active PM for that VM, it will activate new PM. However MACO uses the randomness of a meta-heuristic approach based on defined objective function and input parameters. It tries to find Pareto optimal solutions for current active PMs. If MACO is not able to find any PM for placing VM, it will activate new PM. According to (2.1) with the growth of number of hosts (PMs), energy consumption increases.

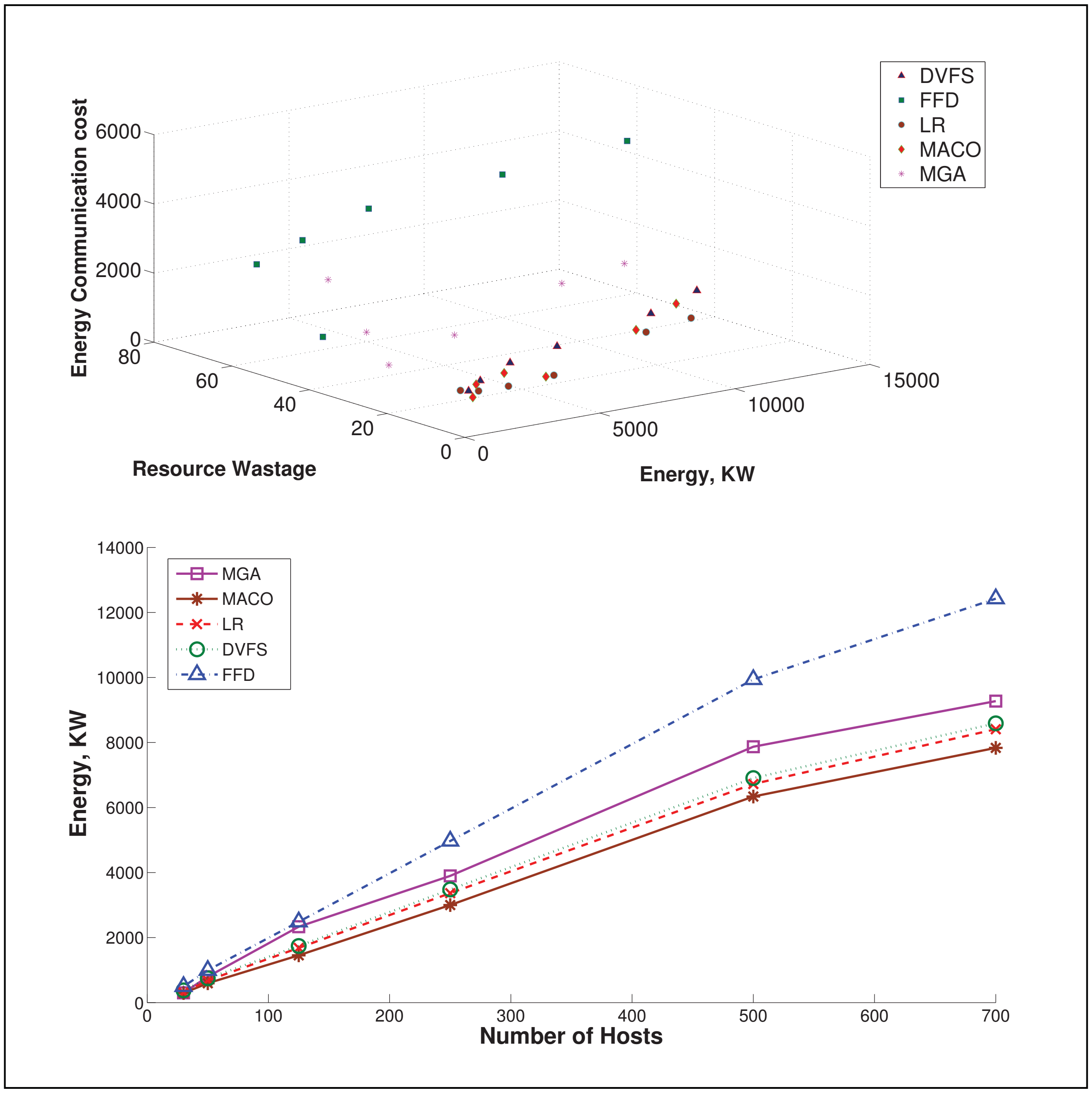


Figure 4.1 Energy consumption of placement algorithms

The results presented in Figure 4.2 shows the dependency between the numbers of active hosts and energy consumption of data center. This is due to the fact that each PM needs a stable power supply and CPU utilization and it consumes particular amounts of energy. When we set number of hosts, number of VMs and number of Cloudlets to 700, 1000 and 1500 respectively, FFD used 625 PMs with 12422 KW as energy consumption. However MACO used only 307 of PMs and 9273 KW as energy consumption. Therefore the less number of PMs are used in data center the less energy is consumed.

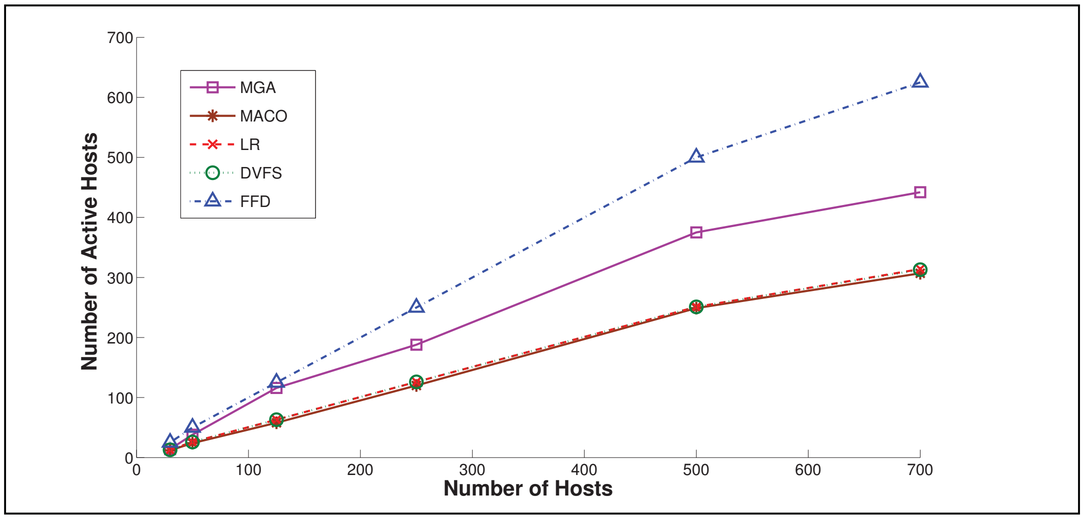


Figure 4.2 Number of active Hosts of placement algorithms

In Figure 4.3, the comparison of resource wastage for different algorithms has been presented. The MACO tries to use all available resources of PMs to place all the VMs. As an example for 30 PMs, 40 VMs and 50 Cloudlets we got 40%, 1.67%, 3.33%, 0% and 21.66% resource wastage for FFD, DVFS, LR , MACO and MGA. Among 25 servers that FFD used for its placement, 40% of their resources have not been used however MACO used all available resources of only 12 PMs.

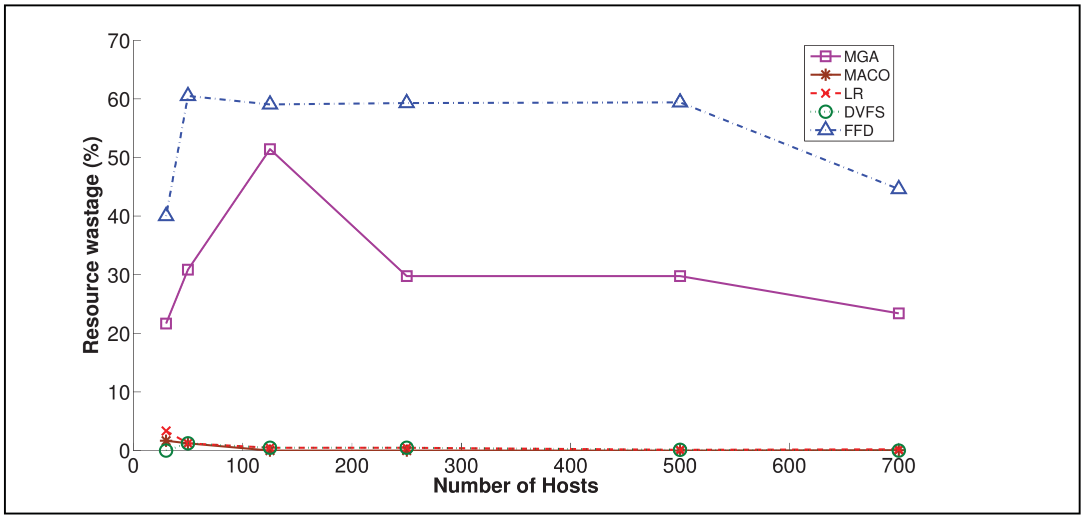


Figure 4.3 Resource wastage of placement algorithms

Table 4.3 and Figure 4.4 show that MACO enable better energy communication that MGA, DVFS and FFD, but less than LR. However, among the candidate MACO solutions with the same energy consumption and resource wastage, MACO is able to select a solution with the lowest energy communication cost. The main goal of single objective algorithms is to find global solution. But MACO and MGA gives a set of optimal solutions with respect to all objectives (minimize energy consumption, resource wastage and energy communication cost) unlike single objective algorithms. Among the algorithms that have been analyzed in the present report, MACO could save more energy than FFD, LR, DVFS and MGA. On average 39.19% of energy were preserved by MACO approach whereas MGA preserved energy by almost 22.175% (see Figure 4.5).

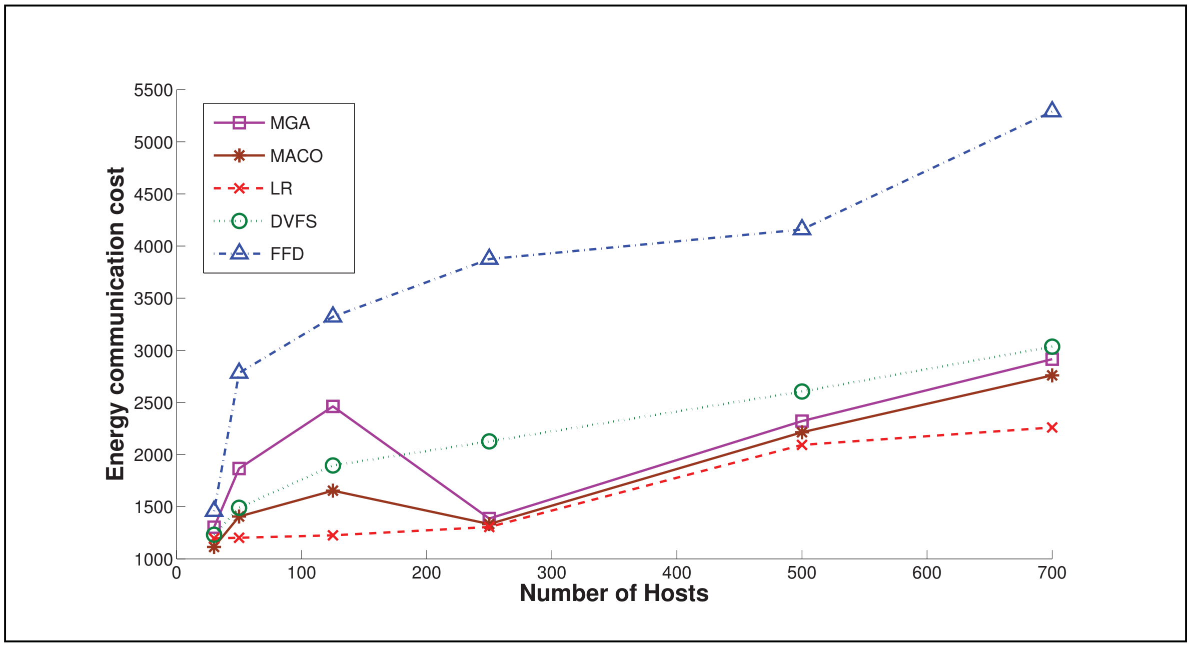


Figure 4.4 Comparison of Energy communication cost between placement algorithms

Figure 4.5 illustrates the amounts of energy that have been conserved by other algorithms in comparison with FFD algorithm. The results show that MACO could gain more energy than LR, DVFS and MGA. For instance, when the number of hosts is equal to 125, MGA could gain only 6% of energy whereas for other number of PMs it could conserve more energy. As shown in Figure 4.3, when the number of hosts is equal to 125, MGA wastes almost 51% of the resources. The results show that with the growth of the resource wastage, the amounts of energy that have been gained decrease. As a result the more number of PMs are used, the more energy consumed.

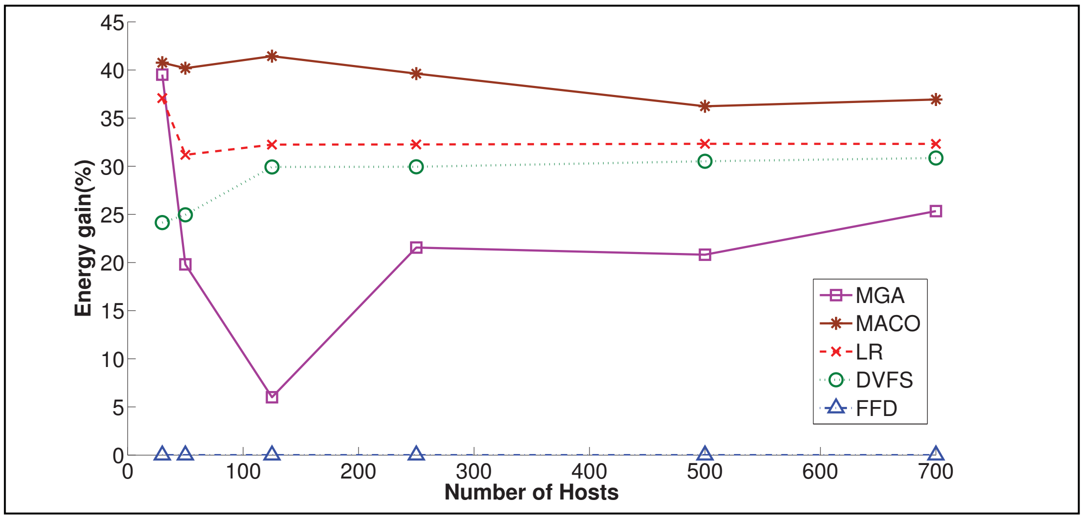


Figure 4.5 Energy gained of placement algorithms

4.4 Performance analysis of consolidation algorithms

4.4.1 Multi objective ACO and GA algorithms performance analysis - First approach

In the first approach, three objectives are defined: energy consumption, resource wastage and energy communication cost. We already explained FFD as a single objective algorithm in this chapter. Another single objective algorithm which we propose to analyze is Single Threshold (ST). In this algorithm the upper utilization threshold is set for active PMs and VMs will assign to PMs based on this threshold. In addition we have implemented ACO multi objective consolidation algorithm which have been proposed by Feller (Feller, 2013) in order to compare it with our algorithms results with this algorithm. Also to evaluate our ACO multi-objective approach with another approach, we used GA in the Matlab Optimization toolbox with the same objective functions defined for ACO algorithm.

Table 4.4 Comparison of VM consolidation algorithms

# of HOSTS	# of VMs	# of Cloudlets	Algorithm (Policy)	Energy, KW	# of Migrations	# of SLA Violations	SLA Violation Percentage (%)
30	40	50	FFD	109.17	37	203	79.30
			ST	95.29	38	207	80.54
			MACO (Feller)	93.53	2	5	2.36
			MACO	93.44	4	6	2.82
			MGA	95.51	6	8	3.74
50	80	100	FFD	189.32	76	427	82.75
			ST	192.22	78	430	83.01
			MACO (Feller)	158.34	3	10	2.36
			MACO	159.3	6	6	1.42
			MGA	160.8	7	8	1.89
125	200	300	FFD	543.07	199	1095	84.30
			ST	488.27	198	1087	83.74
			MACO (Feller)	449.05	3	14	1.33
			MACO	450.35	8	8	0.76
			MGA	450.5	8	10	0.95
250	400	500	FFD	988.53	397	2180	83.98
			ST	983.69	398	2181	83.95
			MACO (Feller)	811.58	6	17	0.81
			MACO	812.6	10	10	0.48
			MGA	814.53	12	14	0.66
500	800	1000	FFD	2046.69	799	4398	84.59
			ST	1968.52	798	4372	84.11
			MACO (Feller)	1691.31	4	18	0.43
			MACO	1691.59	8	10	0.24
			MGA	1693.55	12	15	0.36
700	1000	1500	FFD	2478.92	999	5488	84.46
			ST	2463.76	998	5466	84.12
			MACO (Feller)	2046.33	7	28	0.53
			MACO	2043.61	10	10	0.19
			MGA	2048.28	14	32	0.61

In Figure 4.6, the energy consumption of different consolidation algorithms is compared with each other. As it is shown in this figure, FFD and ST yield higher energy consumption in comparison with other three meta-heuristics approaches. We elaborate this fact that these algorithms (FFD and ST) are not developed to take into consideration current placement of VMs among PMs. That's why they find their solutions and move the whole VMs based on their method in each iteration. However other meta-heuristics approaches are able to take into consideration the current placement of VMs and PMs. According to Table 4.1, MACO (Feller) could consume less energy than MACO and MGA due to the objective function which has been used in Feller algorithm. In our algorithm we considered SLA violation metric and with the growth of this parameter the energy consumption decreases and vice versa. In Feller's MACO algorithm (Eugen, 2013), the objective functions tries to maximize the number of released PMs, maximize the variance of the scalar valued for capacity of PM and smaller migration plans.

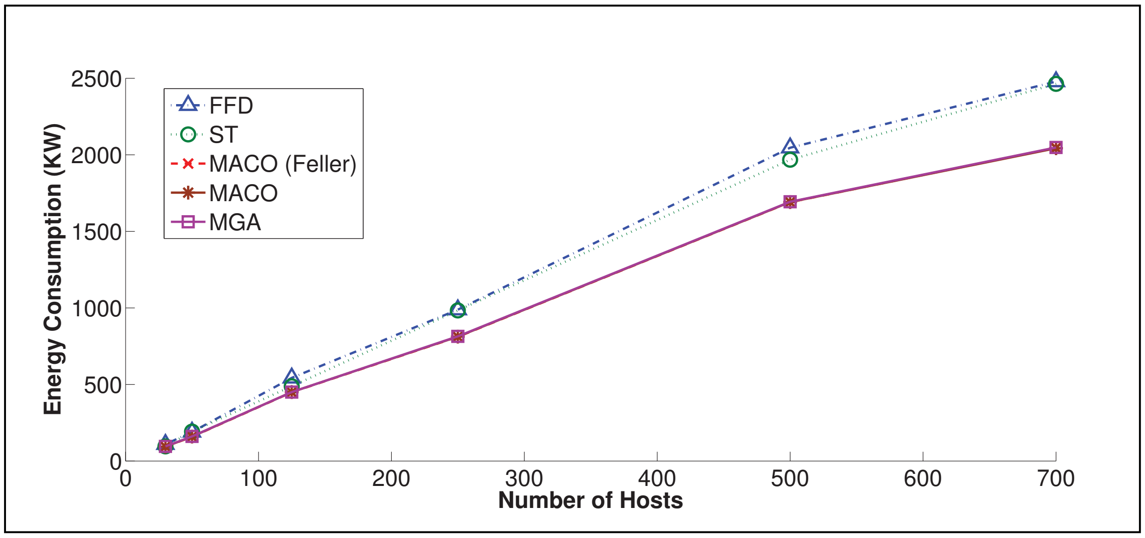


Figure 4.6 Comparison of Energy consumptions between consolidation algorithms

In Figure 4.7 we compared the results of different algorithms in terms of number of migration. The result shows that FFD and ST enable high number of migrations for VMs consolidation compared to the other meta-heuristic approaches. As an example whenever FFD receives a new VM request as an input, it tries to find the first PM with enough resource for that particular PM. If it cannot find any active PM for that VM, it will activate new PM disregarding to the current activated VMs and PMs.

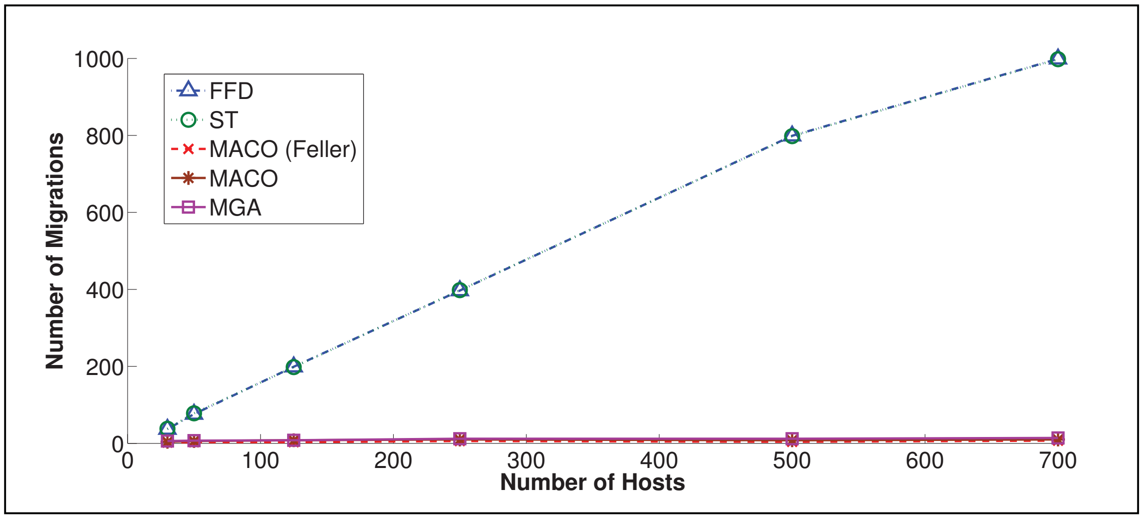


Figure 4.7 Comparison of number of migrations between consolidation algorithms

In Figure 4.8 we focus on three meta-heuristic results in terms of numbers of migrations. The result shows that among these algorithms, MACO (Feller) could find solutions with better numbers of migrations. In addition MACO could get better results than MGA.

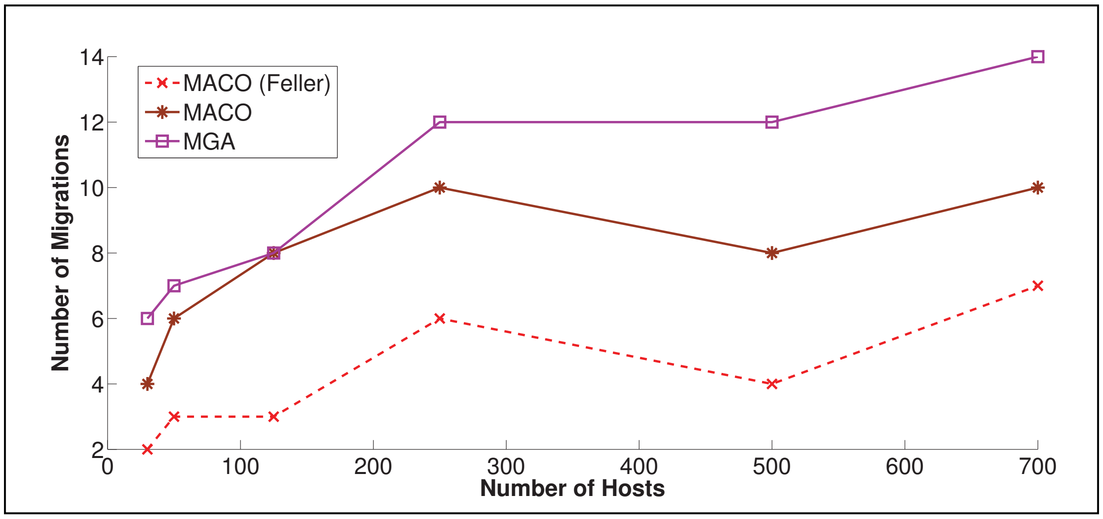


Figure 4.8 Number of migrations of MACO and GA consolidation algorithms

In Figure 4.9 we compare all algorithms in terms of numbers of SLA violations. This figure shows that FFD and ST algorithms increase the number of SLA violations. This is due to fact that they want to place the maximum number of VMs in each PM. As a result, the CPU usage of PMs might be of 100% and the number of overloaded PMs will increase. The overloaded

PMs cannot respond to user requests immediately and the cost of SLA violations will increase.

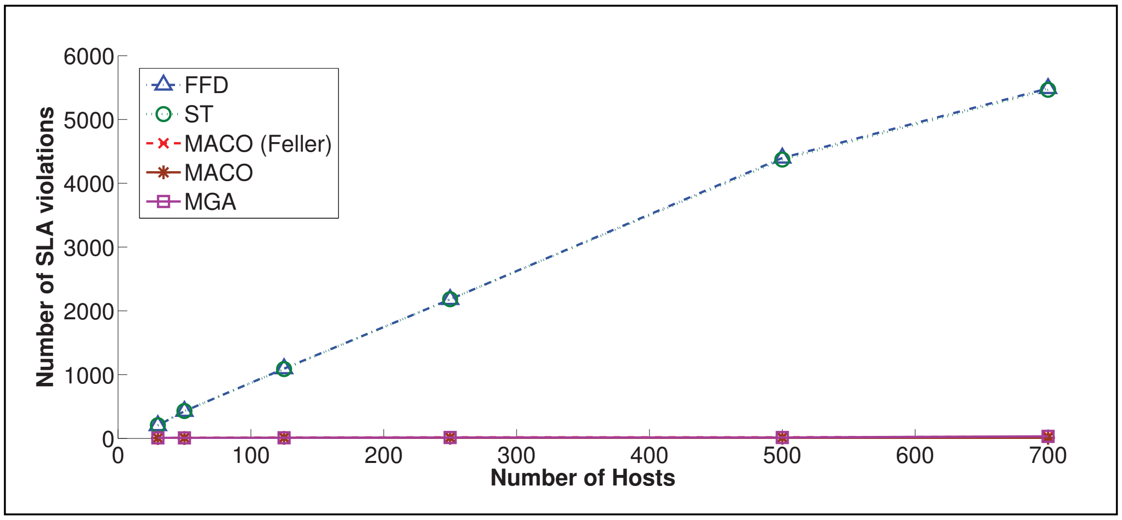


Figure 4.9 Number of SLA violations of consolidation algorithms

In Figure 4.10 we analyze the number of SLA violations for MACO and MGA approaches. The result shows that MACO and MGA enable less number of SLA violations than MACO (Feller). As we already explained we take into consideration the number of SLA violations in the proposed ACO and GA algorithms however this parameter has not been applied in Feller's approach.

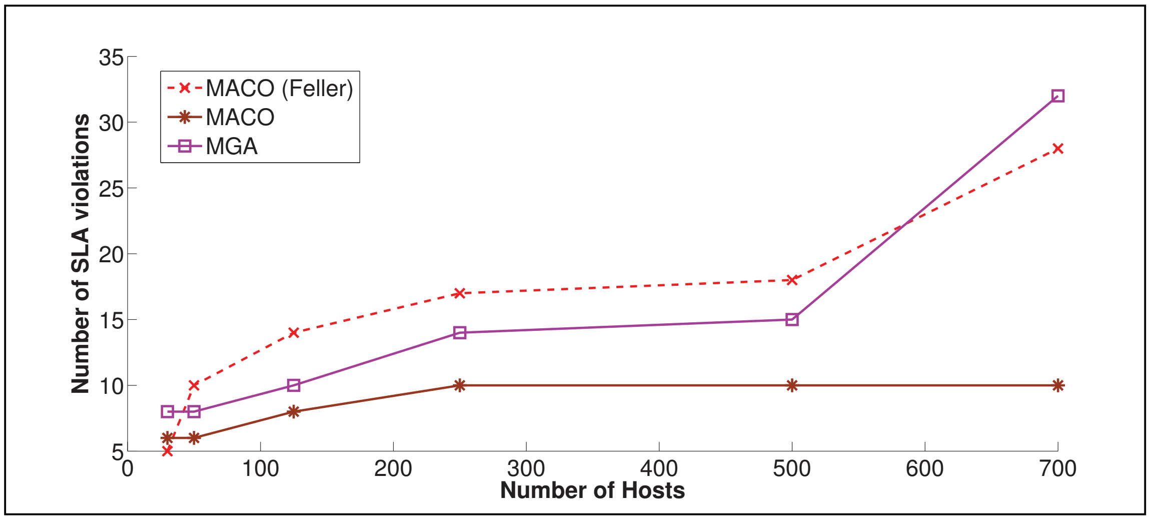


Figure 4.10 Number of SLA violations MACO and GA consolidation algorithms

In Figure 4.11 and Figure 4.12 illustrate SLA violation percentage for each algorithm. Figure 4.11 shows that MACO and MGA approaches decrease the risk of SLA violations in comparison with FFD and ST. In Figure 4.12 we presented that the number of SLA violations are increased with the number of hosts. But if we consider the percentage of these numbers in total number of PMs, the SLA violation percentage is decreased with increasing total number of hosts (active PMs), as shown in Figure 4.12.

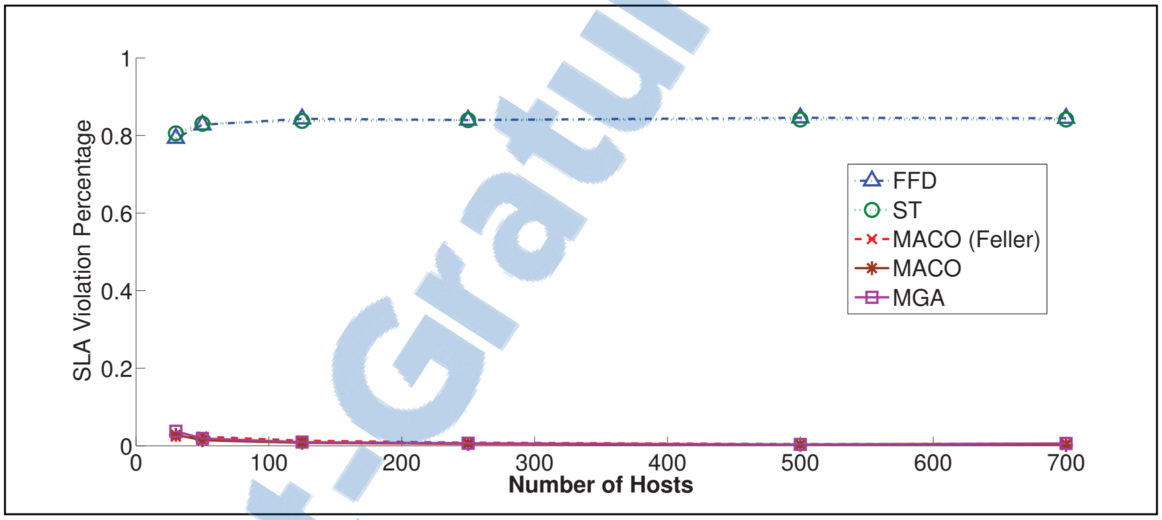


Figure 4.11 SLA violation percentage of consolidation algorithms

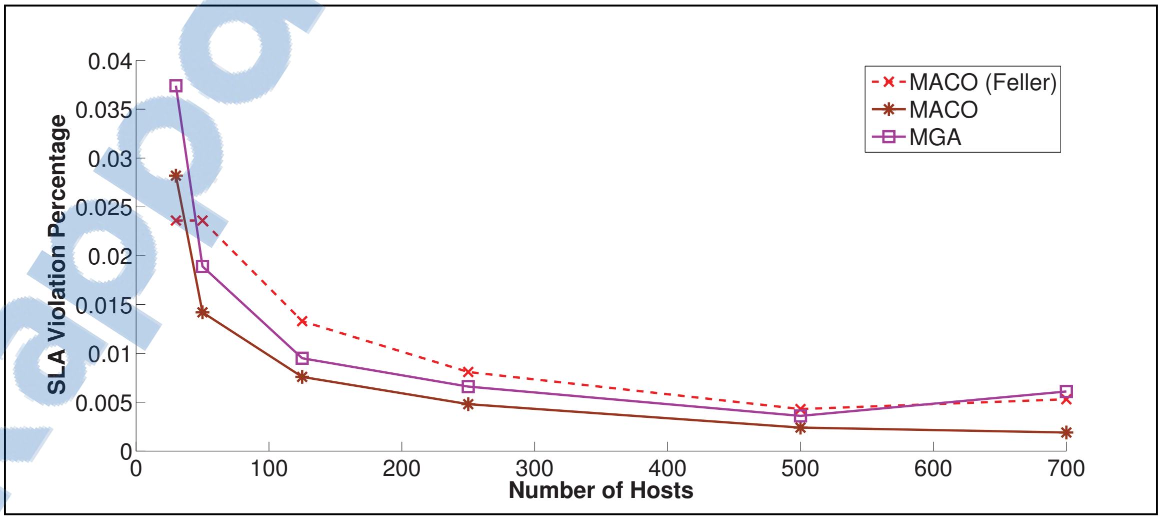


Figure 4.12 SLA violation percentage of MACO and MGA consolidation algorithms

4.4.2 Multi objective ACO and GA algorithms performance analysis – Second approach

As explained in section 3.3.4, we have added another objective (minimize number of active PMs) in order to evaluate the impact of this objective in our results and compare new results with consolidation algorithms of previous section. Also we have changed Cloudsim configurations to test the proposed algorithms in different situations with random number of VMs and PMs. Table 4.5 shows the main obtained results with four objectives.

Table 4.5 Comparison of VMs consolidation algorithms – second approach

# of HOSTS	# of VMs	# of Cloudlets	Algorithm (Policy)	Energy, KW	# of Migrations	# of SLA Violations	# of active PMs
11	22	22	FFD	58.91	10	216	4
			ST	89.34	20	223	7
			MACO (Feller)	60.18	2	34	5
			MACO (3 obj)	59.17	2	14	5
			MACO (4 obj)	59.17	2	14	5
			MGA	58.27	1	7	5
93	185	185	FFD	254.63	75	1,163	29
			ST	732.32	169	1,906	69
			MACO (Feller)	243.87	4	103	38
			MACO (3 obj)	246.17	6	77	40
			MACO (4 obj)	241.65	9	35	39
			MGA	247.74	4	78	41
180	320	320	FFD	409.38	118	1,734	50
			ST	1275.29	298	3,299	122
			MACO (Feller)	389.8	20	112	65
			MACO (3 obj)	393.9	6	93	89
			MACO (4 obj)	390.2	22	90	67
			MGA	393.87	6	112	92
340	710	710	FFD	885.15	259	3,711	111
			ST	2859.25	633	7,328	264
			MACO (Feller)	838.64	49	195	129
			MACO (3 obj)	843.12	32	135	134
			MACO (4 obj)	841.37	47	186	131
			MGA	848.4	52	176	136

580	1020	1020	FFD	1276.46	367	5,106	160
			ST	4051.22	953	10,599	389
			MACO (Feller)	1207.17	47	199	185
			MACO (3 obj)	1209.28	44	149	197
			MACO (4 obj)	1207.59	47	192	186
			MGA	1210.58	50	186	199

In Figure 4.13, the different consolidation algorithms are compared with each other in terms of energy consumption. Among these algorithms, Single Threshold (ST) consumes more energy because PMs and VMs are selected randomly. According to Table 4.5, when the number of hosts increases, the variation of energy consumption is increased.

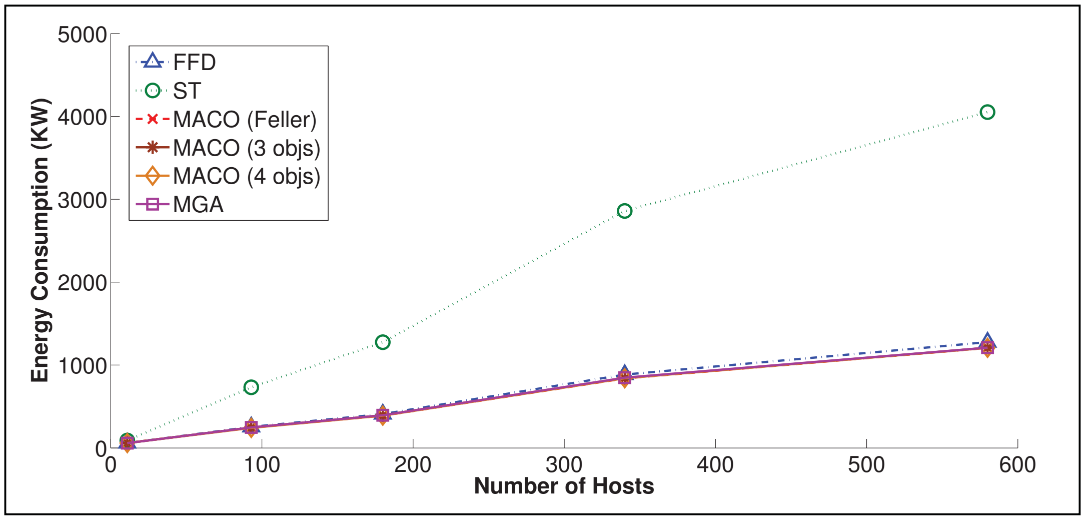


Figure 4.13 Energy consumptions of consolidation algorithms

We calculate number of migrations based on the new assignments of VMs and PMs in comparison with their previous assignment matrix. Figure 4.14 presents the comparison of algorithms in terms of number of migrations. It shows that ST and FFD need more number of migrations than MACO and MGA approaches. We already expected to get these results for FFD and ST. Because these algorithms are single objective and static. They do not take into consideration other objectives in their solution unlike MACO and MGA approaches. Now the question is which one of the Meta heuristic approaches need less number of migrations for resource consolidation.

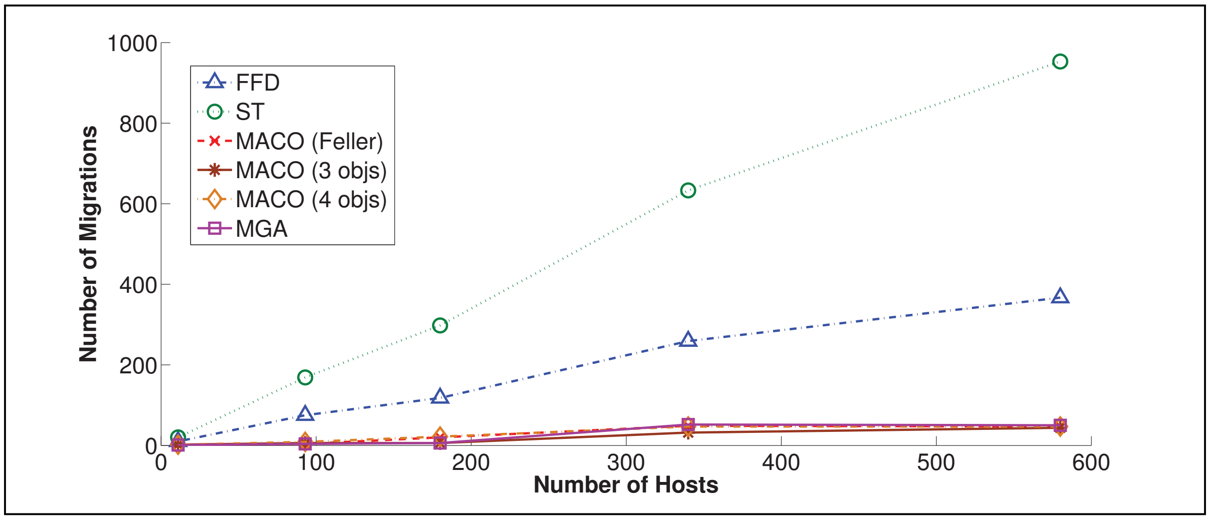


Figure 4.14 Number of Migrations of consolidation algorithms first and second approaches

In Figure 4.15, we focused on four Meta heuristic algorithms which have been tested. It presents MACO with three objectives. This algorithm is able to find solutions with less number of migrations compared to FFD and ST algorithms. MGA provides almost the same number of migrations as MACO (three objectives) with less number of migrations for small number of Hosts. But when we increased the number of Hosts, the result shows that MGA could not be able to find good results. On the other hand MACO (four objectives) and MACO (Feller) could find better results than MGA. When we use four objectives, MACO should trade-off between four objectives in order to find optimal solution. However with three objectives, MACO has less complexity than four objectives to find better solution. So with more numbers of objectives, we cannot necessarily find better solutions.

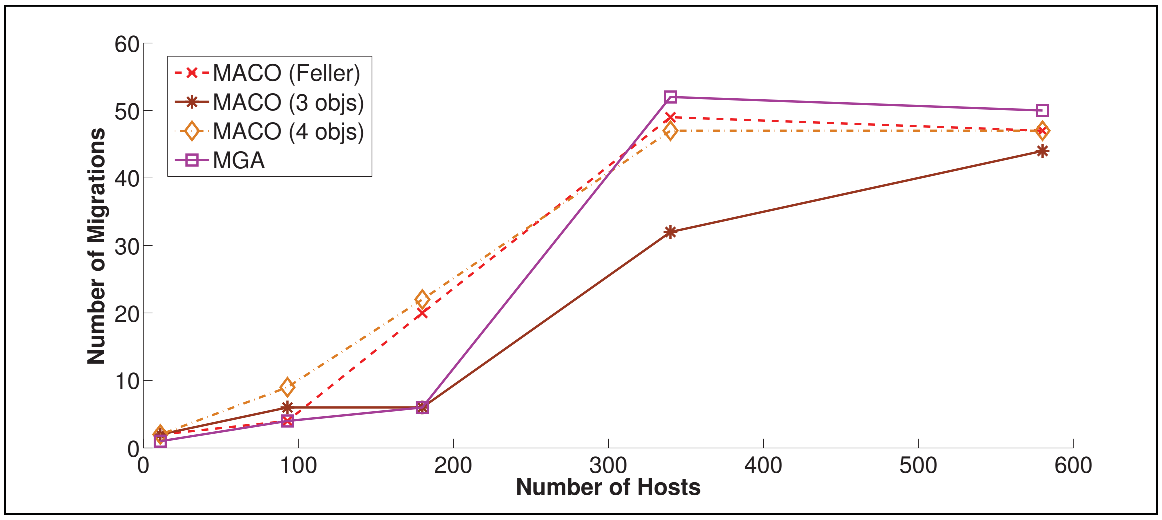


Figure 4.15 Number of Migrations of MACO and MGA consolidation algorithms

We also compared the consolidation algorithm s in terms of number of SLA violations. Figure 4.16 shows the number of SLA violations for each algorithm. In single objective algorithms like ST and FFD, they try to find solution based on their predefined objective. But SLA violations have not been defined as an objective for these algorithms. That’s why they have more numbers of SLA violations than the other algorithms,

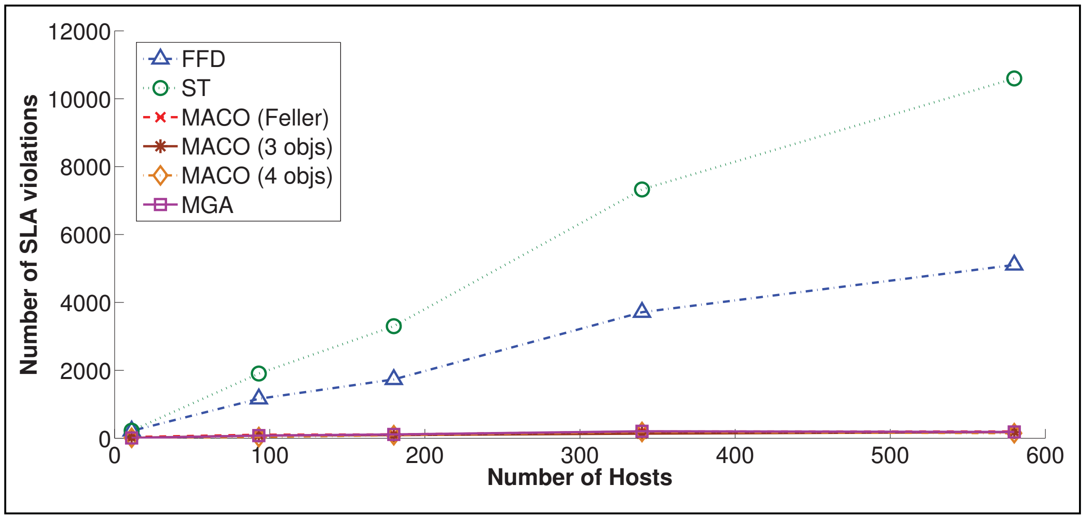


Figure 4.16 Number of SLA violations consolidation algorithms

According to Figure 4.17 when the number of hosts increases, MACO with three objectives is able to find solutions with minimize number of SLA violations. When number of hosts is equal to 580, MGA also finds better minimized number of SLA violations than MACO with four objectives. However, the solutions of MACO (Feller) yield to higher number of SLA violation than the other three Meta heuristic approaches. In MACO which presented by Feller, the number of SLA violation is not taken into consideration. The three objectives of MACO (Feller) are: 1) Maximize number of released hosts; 2) Minimize migrations; 3) Maximize variance of resource utilization (Feller, 2013).

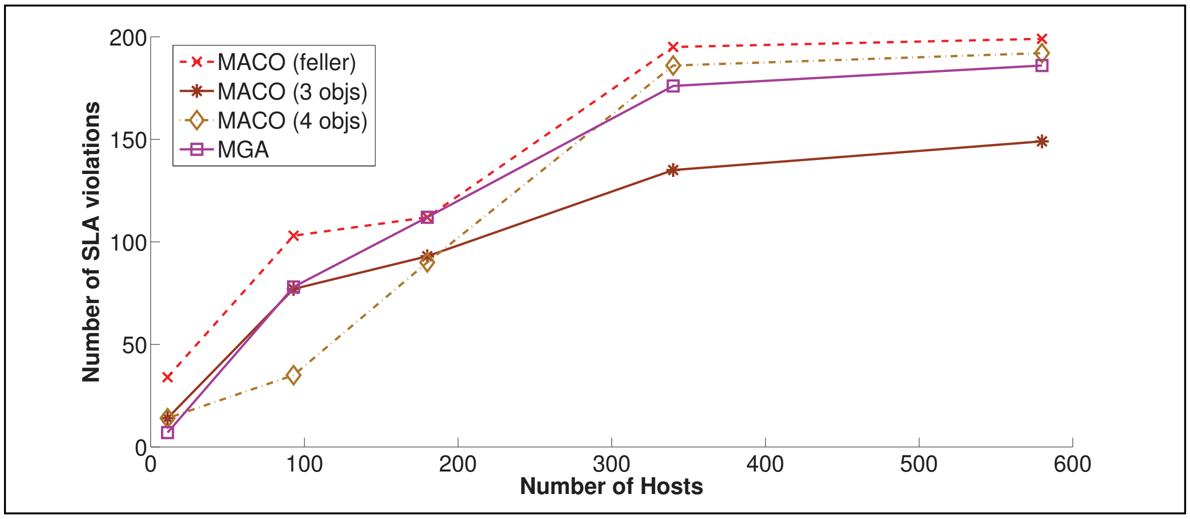


Figure 4.17 Number of SLA violations of MACO and MGA consolidation algorithms for the first and second approaches

Now it is the time to analyze the results that we got for number of active hosts' objective. In Figure 4.18 presents the number of active hosts of the four consolidation algorithms. As shown in Figure 4.18, The FFD finds the solution with the minimum number of active hosts than other algorithms.

The objective for maximizing released hosts and minimizing number of active hosts have been used in MACO (Feller) and MACO (four objectives) respectively. The result shows these two algorithms could find better solutions than MACO (three objectives) and MGA in terms of number of active hosts. The ST algorithm needs high number of active hosts for VM consolidation because of predefined utilization rate. With this static threshold, some resources are wasted in hosts and obviously more hosts should be activated to meet the requirements of VMs.

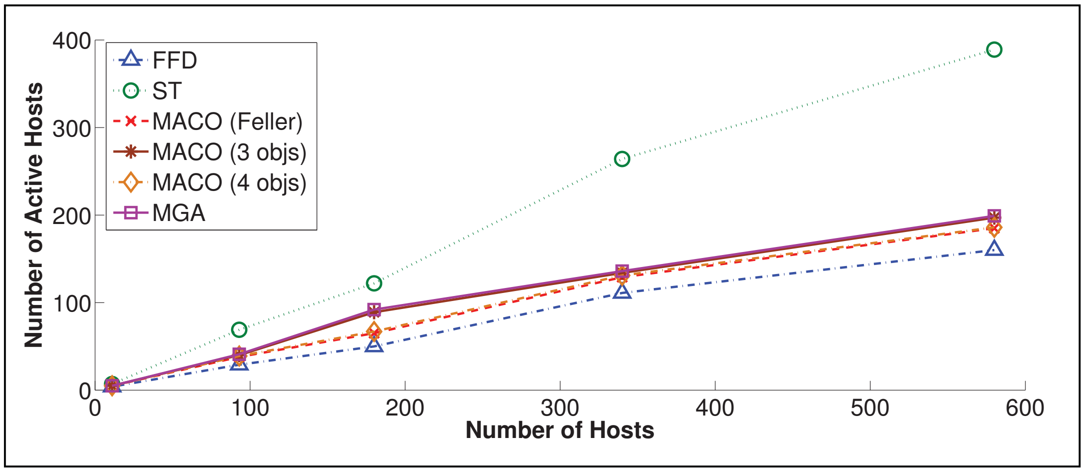


Figure 4.18 Number of Active Hosts of consolidation algorithms

In Figure 4.19, we compared the MACO and MGA algorithms in terms of number of active hosts. In order to calculate number of active hosts, we simply count the number of PMs (hosts) for global solution of each algorithm. As an example, if we have 93 PMs in a datacenter, each algorithm gives a solution at the end of each consolidation step. As shown in Table 4.5, MACO (Feller) could find a solution with 38 active hosts, 39 active hosts for MACO (4 objectives), 40 for MAC (3 objectives) and 41 for MGA.

In this figure we see that MACO (three objectives) and MGA did not take into consideration this objective as well as MACO (four objectives) and MACO (Feller).

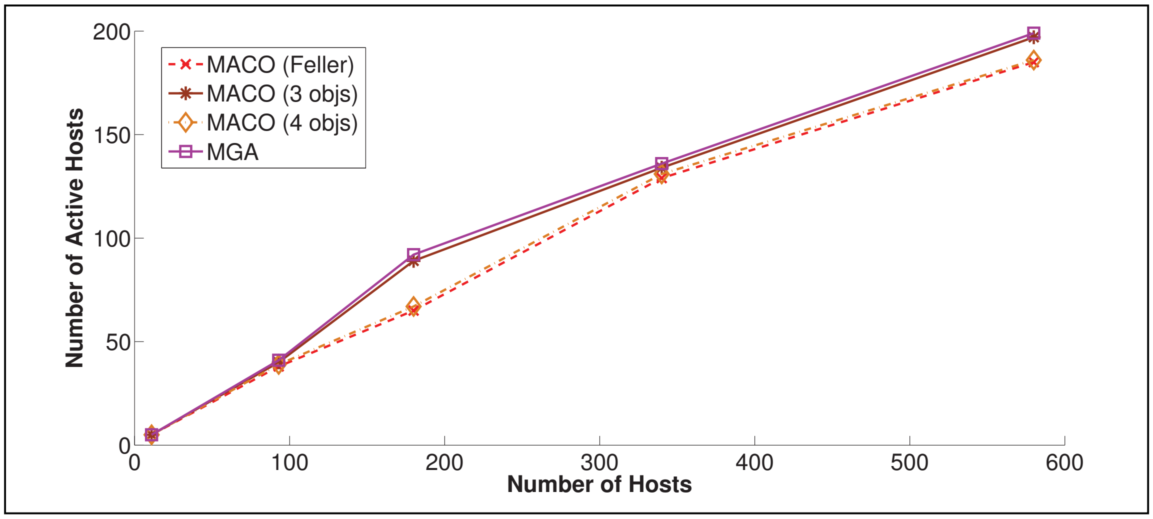


Figure 4.19 Number of Active Hosts of M ACO and MGA consolidation algorithms

CONCLUSION

Cloud computing technology is a new technology which allows users to use their desired services based on the pay-as-you-go model. This technology is becoming more popular and customers' demands are growing day by day. In order to be able to meet customers' requirements, data centers need a larger amount of computing resources with more efficiency.

This thesis has focused on energy-aware and QoS-aware VM placement and VM consolidation algorithms in a data center. The proposed approach has two parts. In the first phase (placement), we assume VMs have not been assigned to any PM. This approach allows minimizing energy consumption, resources' wastage and energy consumed by the communication network. The next phase (consolidation), we assume VMs have already been assigned to PMs but these mappings should be optimized. Consolidation approach enables to reduce energy consumption, to minimize SLA violation as well as the number of migrations. To achieve our goals, the related works have been analyzed for cloud computing, virtualization technology and resource management algorithms in cloud computing environments. Our main contributions are: 1) Propose a multi-objective placement optimization to minimize energy consumption, resource wastage and energy communication cost. 2) Propose a multi-objective consolidation algorithm to optimize the solution of placement algorithm based on new users' requests and to minimize the total energy consumption of a data center, minimize number of migrations, minimize SLA violations and minimize number of PMs.

A multi objective ACO placement algorithm have been proposed in order to minimize the energy consumption of PMs, minimize the resource wastage of PMs and minimize the energy communication cost between network elements of a data center. The Pareto approach has been used to find a set of non-dominated solutions. In order to simulate a data center and build our model, As testbed, we used Cloudsim tool to simulate a virtualized environment (Calheiros et al., 2011) and hierarchical topology (Kliazovich et al., 2010). The proposed

ACO algorithm (MACO) has been compared with single objective algorithms and a multi-objective GA algorithm of Matlab Optimization tools. The results demonstrate that MACO save more energy than other algorithms. On average, 39.19% of energy was conserved through the ACO by comparison with the FFD algorithm. However, FFD needs less execution time to compute the placement of VMs among other algorithms. The execution time of ACO and GA should be optimized in order to be more applicable to large-scale cloud computing environments. VM migration is one of the most useful techniques in cloud computing environments in order to move workloads from one PM to another PM and adapt users' requests with available recourses of data center dynamically (Jing and Fortes, 2010). By using consolidation algorithms and VM migration strategies, we can maximize the resource usages of PMs and can transit idle PMs into a low power state mode. Moreover, optimization of resource allocation aims at providing better energy consumption and take into consideration SLA violations in order to enable QoS and guarantee better quality of experience. In addition we assume one data center in our approach however in order to be able to apply the proposed algorithm in real environment, it should be extended to more different geographical data centers.

Moreover, we proposed a multi objective ACO consolidation algorithm aiming at minimizing the energy consumption of PMs, minimizing the number of migrations and minimize the number of SLA violations. The proposed ACO algorithm (MACO) has been compared with single objective algorithms (FFD and ST) and multi-objective ACO proposed by Feller and multi-objective GA algorithm of Matlab Optimization tools. The results demonstrate that MACO reduce the number of SLA violations in comparison with other consolidation algorithms. However, MACO (Feller) enables better energy consumption than other algorithms.

Future works

There are a number of open research challenges and future research directions. A number of improvements should be done in resource management of cloud computing environments. In this thesis we have focused on resource management mechanism within a single data center. However these mechanisms should be extended to more numbers of datacenters with different geographical locations in the world. In this case, we should take into consideration the location of each data center to assign VMs between different PMs while considering energy communication cost and satisfying QoS.

In addition, we should take into consideration the network characteristics in order to adapt resource management algorithms to different workloads. Also, we need to calculate accurate execution time of each algorithm for consolidating resources. Through these calculations we are able to predict possible performance degradation which we might be faced and reduce risk of SLA violation. In a cloud computing environment, the users' requests change rapidly. If algorithms need more execution time, the users' requests should wait in a queue. As a result, the performance will be degraded. Data mining algorithms can assist us to predict future behavior of workloads and to predict execution time of each algorithm in different environments.

On the other hand, in a real environment VMs might be dependent to each other. For instance, different VMs (web servers) might have a dependency on a VM (database). In this research, we assumed that VMs are independents. But, in real environment, some dependencies between VMs may exist and the resource optimization algorithm should take them into consideration in order to guarantee better QoS and better user experience.
Another future direction is to define optimization algorithms based on several resource metrics (ex. CPU, memory). In this thesis, we consider only one resource metric: the CPU usage. For some applications, other resources metrics such as memory and disc space may have an impact on the system performances.

Finally as discussed in Chapter 4, the results have been compared with small to mid-size data center. These algorithms should be compared with each other in a real environment with dynamic numbers of PMs and VMs with various datacenters' sized and locations.

APPENDIX I

CLOUDREPORTS SIMULATOR

CloudReports is considered as a cloud computing simulation tools based on CloudSim engine with graphical user interface and report generation feature. In this simulation tools the VM provisioning has been created at two different levels as follows (Cloud Report CloudSim Simulator, 2013):

- Host level which is responsible to assign each core to each VM based on calculating overall processing power of each core.
- VM level which is responsible for assigning a fixed amount of processing power to cloudlets (tasks).

Two policies have been defined for both levels:

In both levels, there are two default policies available (CloudSim FAQ, 2012):

- xSpaceShared: If the number of VMs/Cloudlets are more than available Processing Elements then the new VMs/Cloudlets have to wait in a queue.
- xTimeShared: The available Processing Elements are being shared among active VMs/Cloudlets.

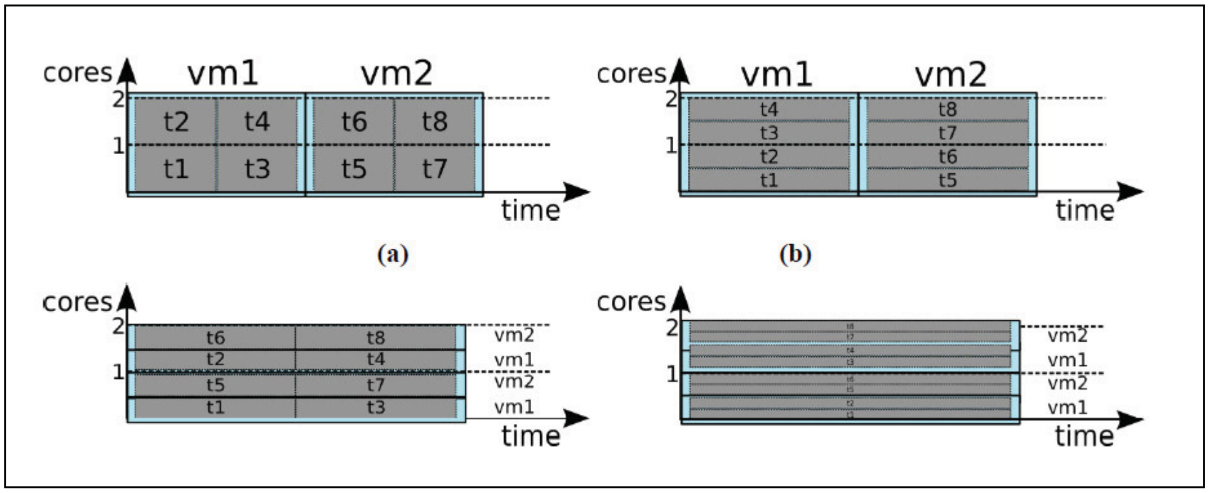
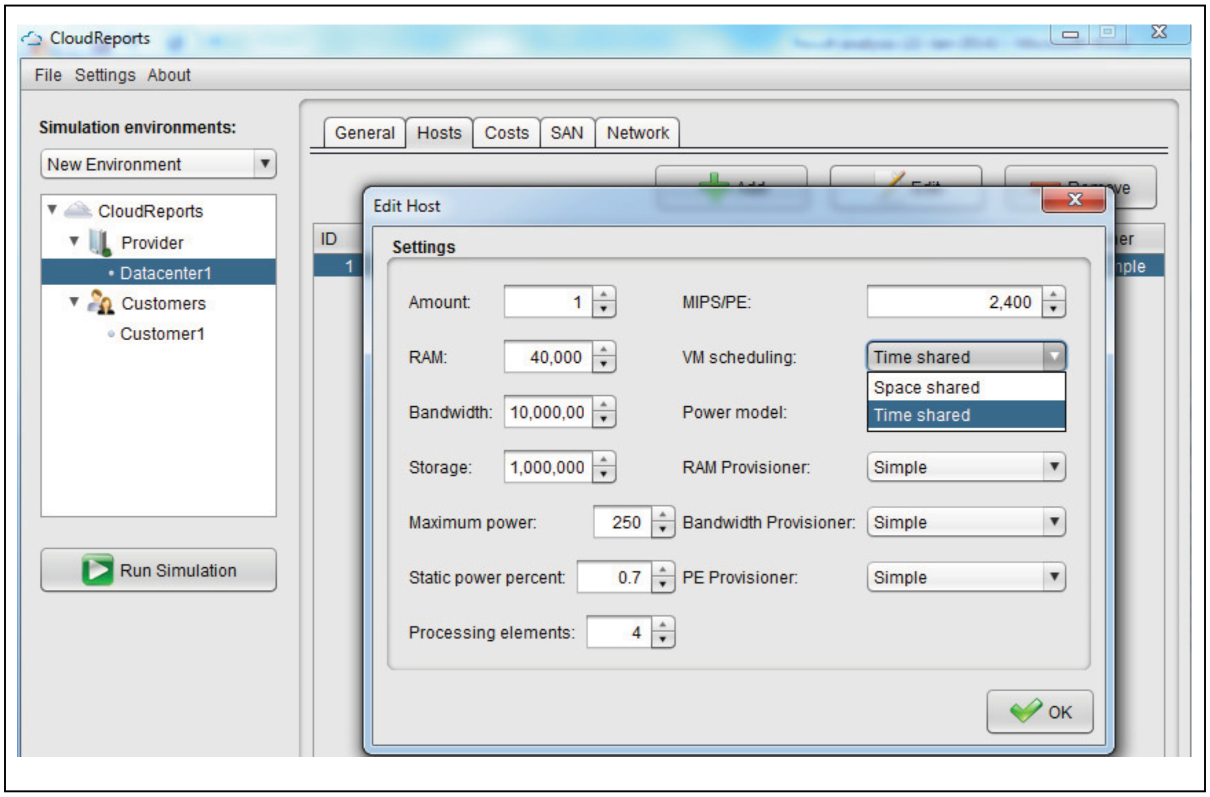
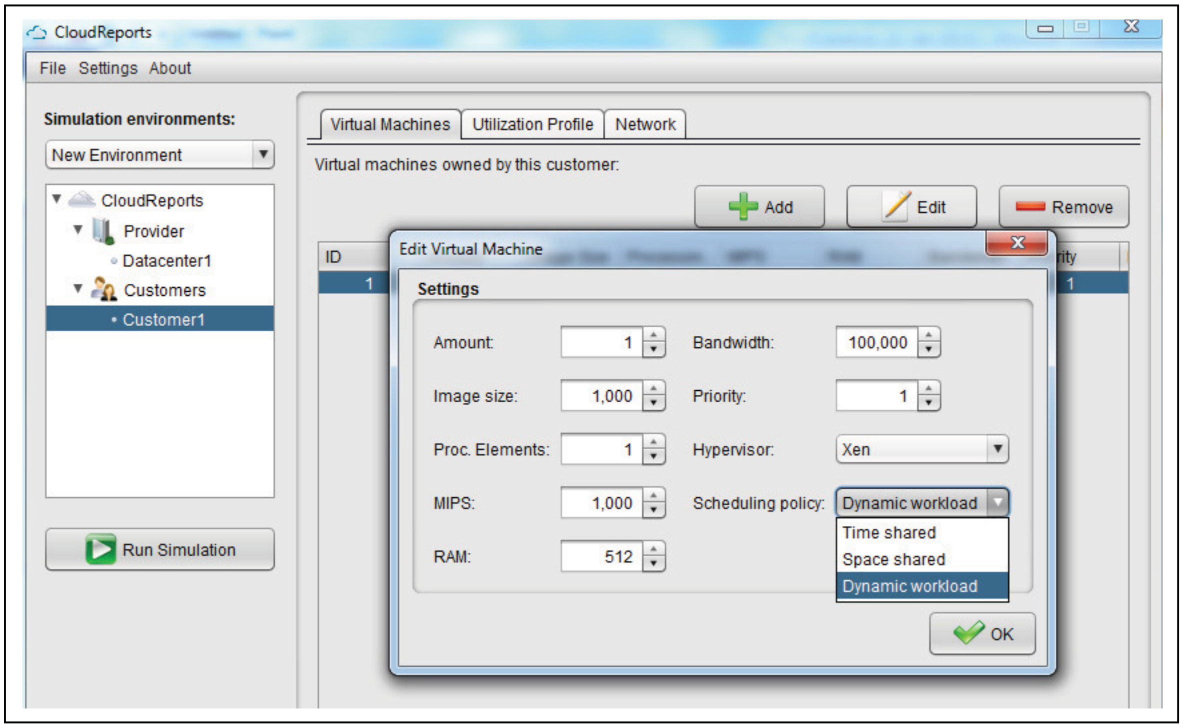


Figure A I. 1 Differences between TimeShared and SpaceShared
Taken From (Calheiros et al., 2011)



Amount: 1
RAM: 40,000
Bandwidth: 10,000,00
Storage: 1,000,000
Maximum power: 250
Static power percent: 0.7
Processing elements: 4
MIPS/PE: 2,400
VM scheduling: Time shared
Space shared
Time shared
Power model:
RAM Provisioner: Simple
Bandwidth Provisioner: Simple
PE Provisioner: Simple

Figure A I. 2 Different types of VM Scheduling in Cloud Reports



Virtual machines owned by this customer:

Edit Virtual Machine

Settings

Amount: 1

Image size: 1,000

Proc. Elements: 1

MIPS: 1,000

RAM: 512

Bandwidth: 100,000

Priority: 1

Hypervisor: Xen

Scheduling policy: Dynamic workload

Time shared

Space shared

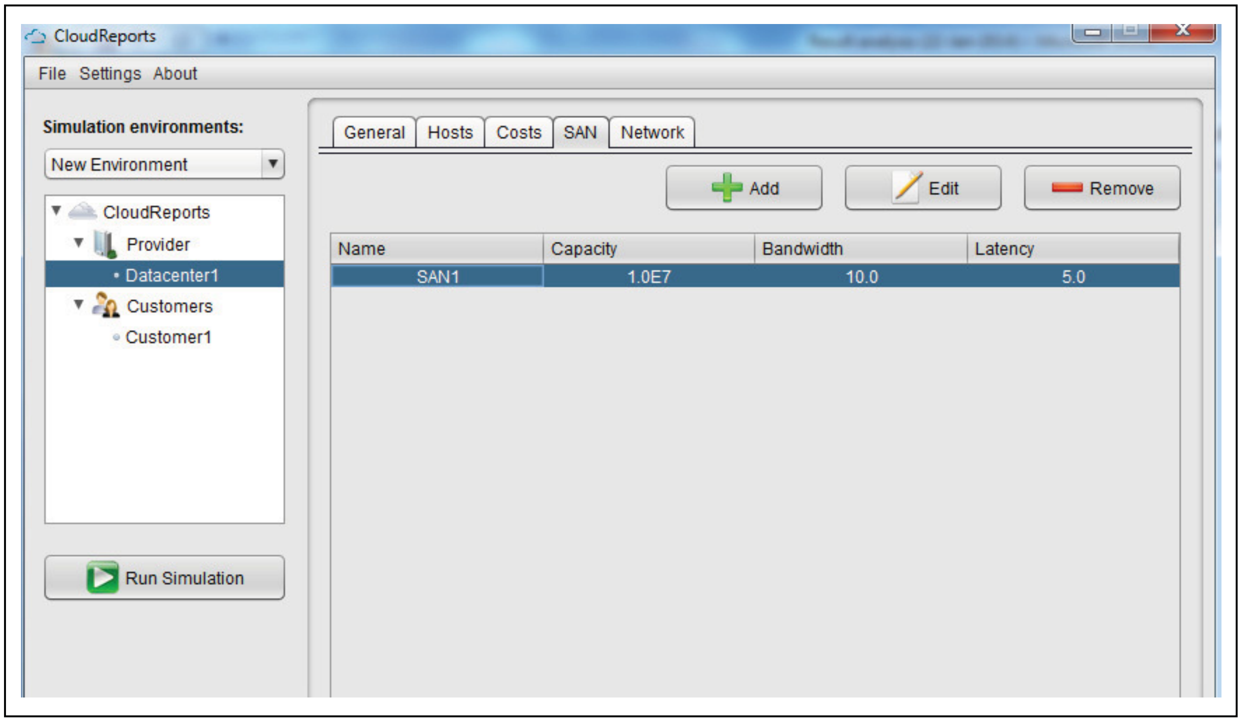
Dynamic workload

Figure A I. 3 Different types of Scheduling policy in Cloud Reports

CloudReport features

San Storage

This feature is used to store large amounts of data in data centers. If network bandwidth is available, San Storage can simulate storage and access files through SAN during execution time (Rodrigo N. Calheiros, 2010).



Name	Capacity	Bandwidth	Latency
SAN1	1.0E7	10.0	5.0

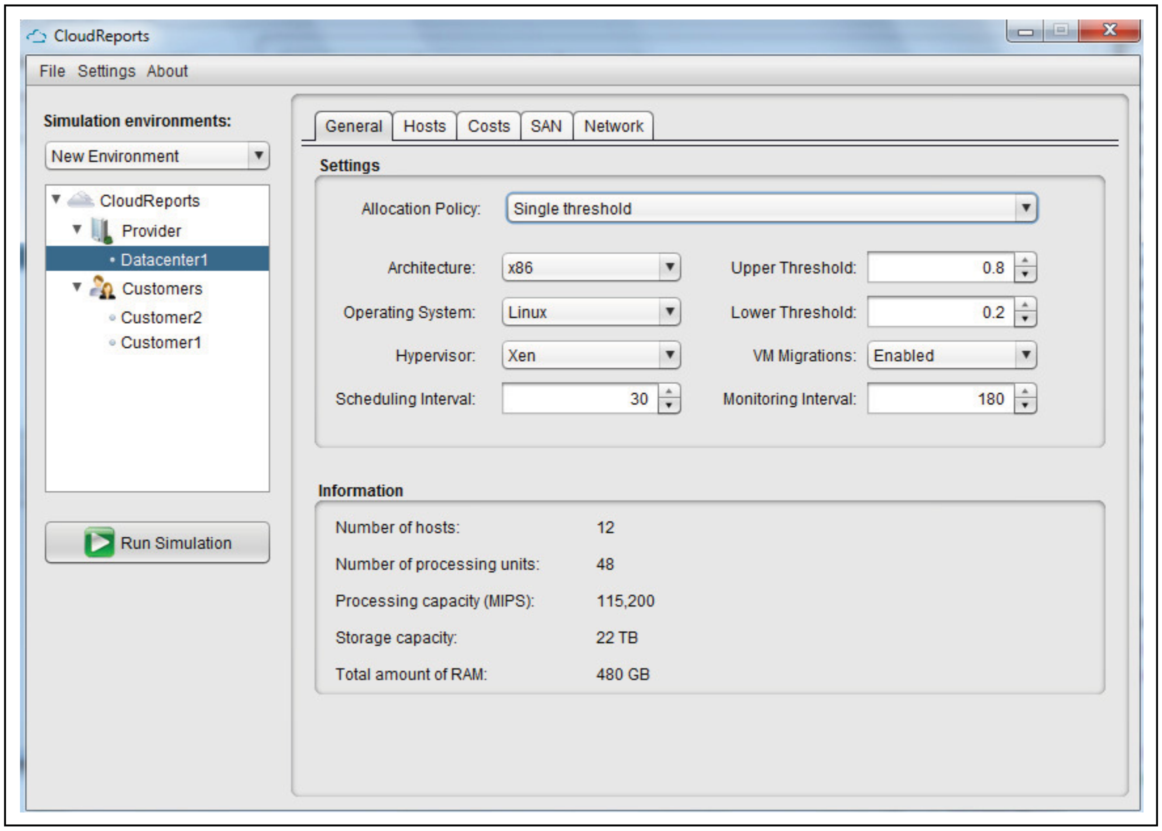
Figure A I. 4 SAN Storage in CloudReports

Single Threshold allocation policy

In Single Threshold allocation policy the network topology and the location of Cloud elements have not been considered whereas we should consider network topology based on our objectives.

The follow steps have been implemented in Single Threshold of CloudReports which it is little bit different from NetworkTopology steps:

1) Initialize the CloudSim package.
2) Create Datacenters. In order to execute CloudSim, Datacenters and their parameters (CPU, Memory, BW …) should be set.
3) Create Broker
4) Create one virtual machine and submit vm list to the broker
5) Create one cloudlet and submit cloudlet list to the broker



Settings

Allocation Policy: Single threshold

Architecture: x86

Operating System: Linux

Hypervisor: Xen

Scheduling Interval: 30

Upper Threshold: 0.8

Lower Threshold: 0.2

VM Migrations: Enabled

Monitoring Interval: 180

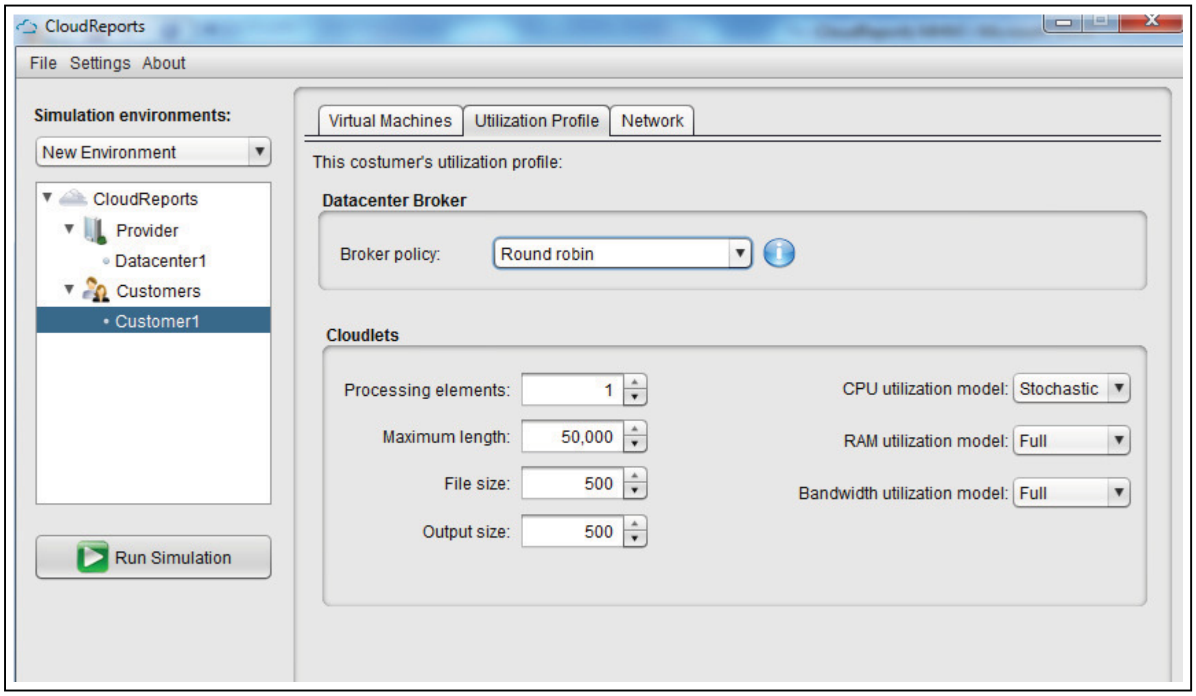
Information

Number of hosts:	12
Number of processing units:	48
Processing capacity (MIPS):	115,200
Storage capacity:	22 TB
Total amount of RAM:	480 GB

Figure A I. 5 Single threshold allocation policy in CloudReports

Broker policy

The broker policy defines how this customer will choose datacenters to allocate its virtual machines and run its cloudlets. Round-Robin load balancing policies have been used for selection of data centers existing in same region for distribution of load among them. This policy results into efficient resource utilization and better service quality from the Cloud Service Provider’s perspective.



Simulation environments: New Environment

- CloudReports
  - Provider
    - Datacenter1
  - Customers
    - Customer1

This costumer's utilization profile:

Datacenter Broker

Broker policy: Round robin

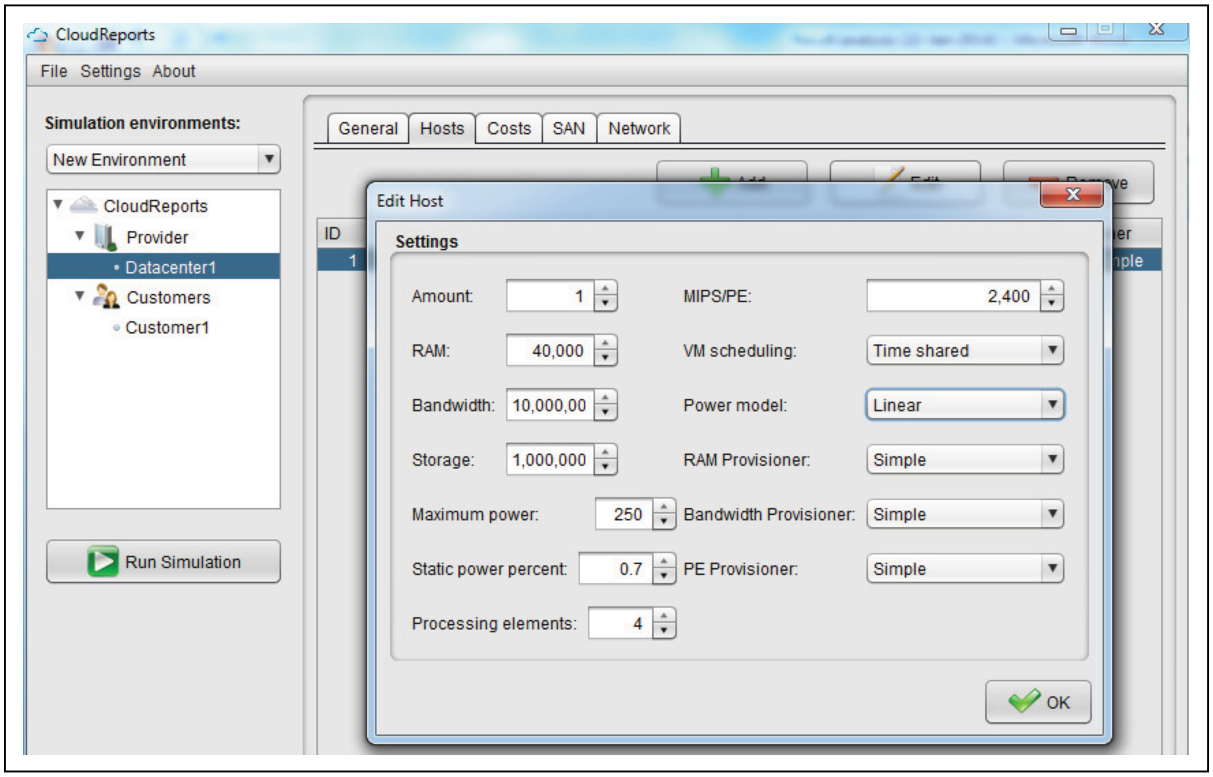
Cloudlets

Processing elements:	1	CPU utilization model:	Stochastic
Maximum length:	50,000	RAM utilization model:	Full
File size:	500	Bandwidth utilization model:	Full
Output size:	500		

Figure A I. 6 Brocker policy in CloudReports

Power model

The PowerModel has been designed to provide a model for power consumption of components within a system (Sankaranarayanan, Sharangi and Fedorova, 2011).

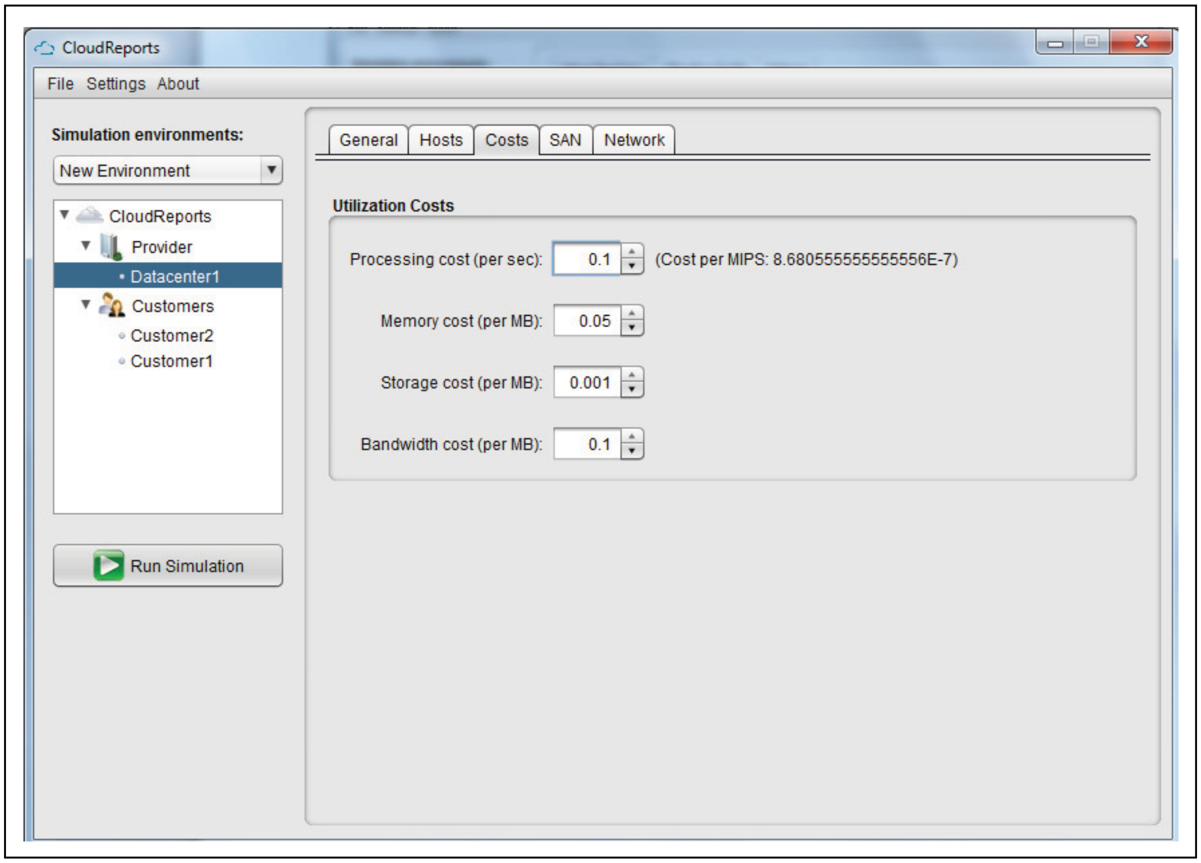


Amount: 1
RAM: 40,000
Bandwidth: 10,000,00
Storage: 1,000,000
Maximum power: 250
Static power percent: 0.7
Processing elements: 4
MIPS/PE: 2,400
VM scheduling: Time shared
Power model: Linear
RAM Provisioner: Simple
Bandwidth Provisioner: Simple
PE Provisioner: Simple

Figure A I. 7 Hosts Setting in CloudReports

Cloud Market

From cloud customers' point of view, the costs of CPU, memory, storage and bandwidth should be clarified. In the following figure we can see these parameters in CloudReports.



Utilization Costs

Processing cost (per sec): 0.1 (Cost per MIPS: 8.680555555555556E-7)

Memory cost (per MB): 0.05

Storage cost (per MB): 0.001

Bandwidth cost (per MB): 0.1

Figure A I. 8 Cloud Market in CloudReports

APPENDIX II

CLOUDSIM MAIN FEATURES FOR CUSTOMIZATION

The following classes are considered as main classes of CloudSim which can be customized depending on various requirements (CloudSim FAQ, 2012):

- DatacenterBroker
- VmAllocatonPolicy
- VmScheduler
- CloudletScheduler
- PowerVmAllocationPolicyMigrationAbstract

APPENDIX III

MODELING THE NETWORK IN SIMULATOR

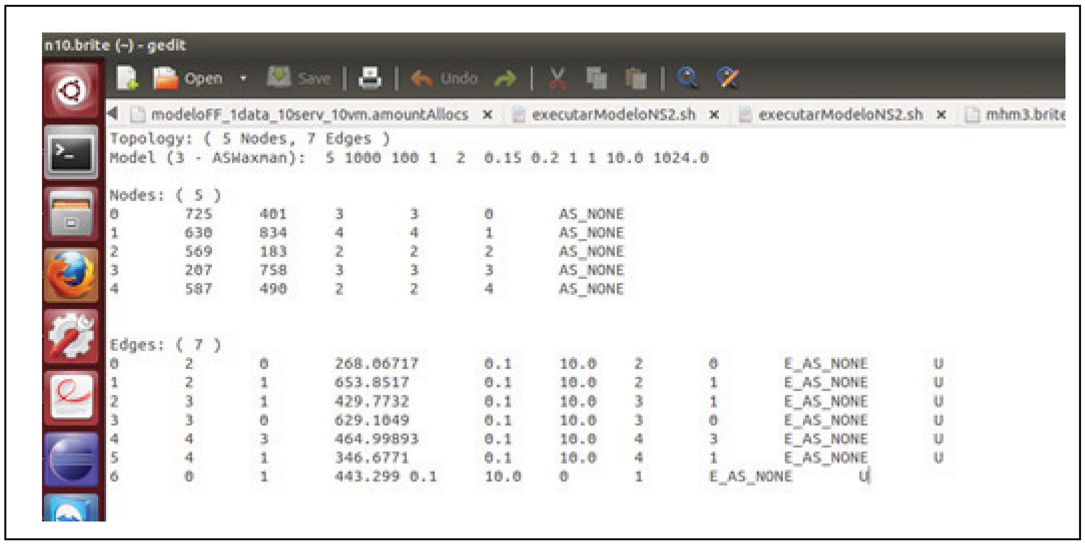
In order to model realistic environment in cloud computing, a networking concepts should be integrated in Cloudsim, as follows (Rodrigo N. Calheiros, 2010):

- Hosts, storage, end-users and Cloud brokers are considered as entities in Cloud computing
- Data centers, SaaS providers, hosts and end-users are intermediate entities between main entities.
- Routers or switches are not available in Cloudsim
- Network latency is simulated with the concept of latency matrix
- When a message is transferred from i to j, the delay of this tranformation is represented by e_{ij}.
- BRITE format includes network nodes which represents entities in Cloudsim and whenever Cloudsim is intilized, the BRITE will be updated
- The follow matrix is an example of Latency matrix which represents the logical connection between entities in the configuration file (BRITE file).

$$\begin{bmatrix} 0 & 40 & 120 & 80 & 200 \\ 40 & 0 & 60 & 100 & 100 \\ 120 & 60 & 0 & 90 & 40 \\ 80 & 100 & 90 & 0 & 70 \\ 200 & 100 & 40 & 70 & 0 \end{bmatrix}$$

Figure A III. 1 Latency Matrix
Taken From (Rodrigo N. Calheiros, 2010)

In Networktopology class generateMatrices() Generates the matrices used internally to set latency and bandwidth between elements.



```
Topology: ( 5 Nodes, 7 Edges )
Model (3 - ASWaxman):  5 1000 100 1  2  0.15 0.2 1 1 10.0 1024.0

Nodes: ( 5 )
0       725     401     3       3       0       AS_NONE
1       630     834     4       4       1       AS_NONE
2       569     183     2       2       2       AS_NONE
3       207     758     3       3       3       AS_NONE
4       587     490     2       2       4       AS_NONE


Edges: ( 7 )
0       2       0       268.06717       0.1     10.0    2       0       E_AS_NONE       U
1       2       1       653.8517        0.1     10.0    2       1       E_AS_NONE       U
2       3       1       429.7732        0.1     10.0    3       1       E_AS_NONE       U
3       3       0       629.1049        0.1     10.0    3       0       E_AS_NONE       U
4       4       3       464.99893       0.1     10.0    4       3       E_AS_NONE       U
5       4       1       346.6771        0.1     10.0    4       1       E_AS_NONE       U
6       0       1       443.299 0.1     10.0    0       1       E_AS_NONE       U
```

Figure A III. 2 BRITE Format with 5 nodes and 7 edges

Topological-node-information:

0 | x is: 725 y is: 401

1 | x is: 630 y is: 834

2 | x is: 569 y is: 183

3 | x is: 207 y is: 758

4 | x is: 587 y is: 490

Node-link-information:

from: 2 to: 0 delay: 0.1

from: 2 to: 1 delay: 0.1

from: 3 to: 1 delay: 0.1

```
from: 3 to: 0 delay: 0.1

from: 4 to: 3 delay: 0.1

from: 4 to: 1 delay: 0.1

from: 0 to: 1 delay: 0.1
```

Creates the delay matrix:

```
delayMatrix = new DelayMatrix_Float(graph, false);
```

delay-matrix is:

```
        0       1       2       3       4

0       0.0     0.1     0.1     0.1     0.2

1       0.1     0.0     0.1     0.1     0.1

2       0.1     0.1     0.0     0.2     0.2

3       0.1     0.1     0.2     0.0     0.1

4       0.2     0.1     0.2     0.1     0.0
```

Creates the bw matrix:

```
bwMatrix = createBwMatrix(graph,false);
```

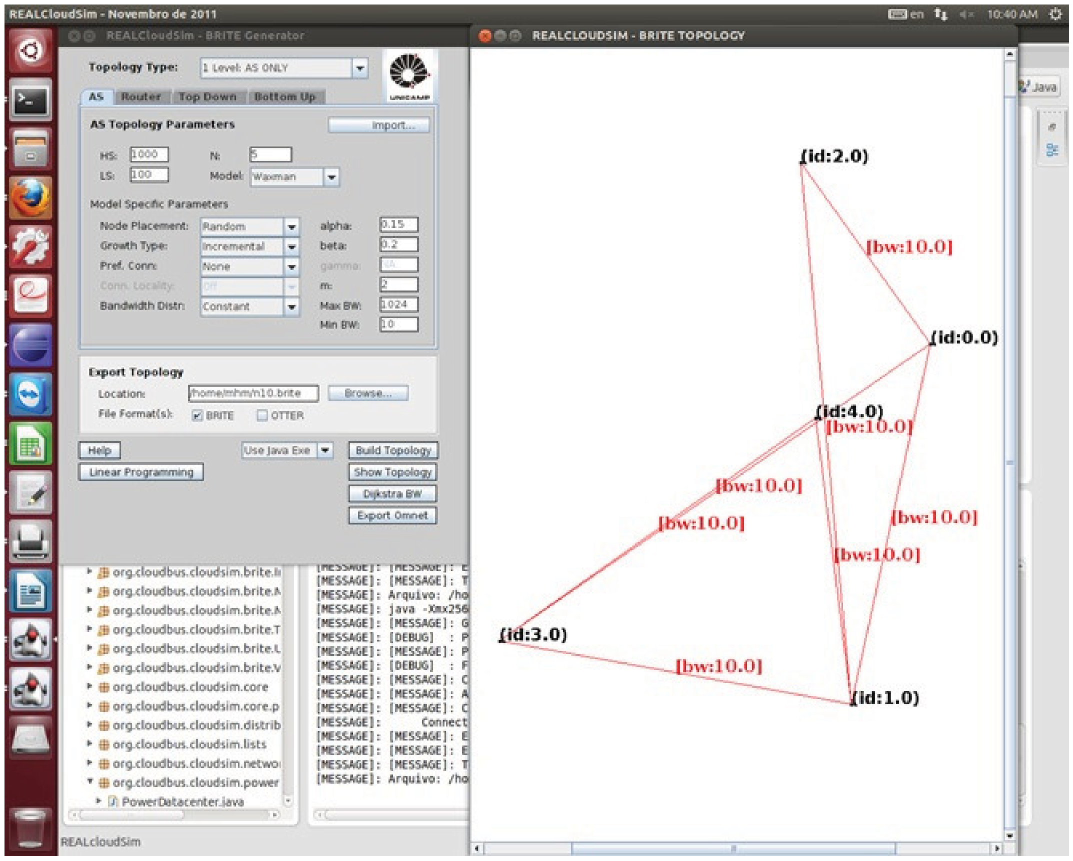
```
[[0.0, 10.0, 10.0, 10.0, 0.0], [10.0, 0.0, 10.0, 10.0, 10.0], [10.0, 10.0, 0.0, 0.0, 0.0], [10.0,
10.0, 0.0, 0.0, 10.0], [0.0, 10.0, 0.0, 10.0, 0.0]]
```

APPENDIX IV

REALCLOUDSIM

In order to show graphical interface to read BRITE network topology we have used RealCloudSim. This simulator allocates virtual mach ines based on CloudSim engine (Rocha, 2013).

The follow figures show the graphical view of the mentioned BRITE with 5 nodes:



Topology Type: 1 Level: AS ONLY

AS Topology Parameters

HS: 1000 N: 5
LS: 100 Model: Waxman

Model Specific Parameters

Node Placement:	Random	alpha:	0.15
Growth Type:	Incremental	beta:	0.2
Pref. Conn:	None	gamma:	NA
Conn. Locality:	Off	m:	2
Bandwidth Distr:	Constant	Max BW:	1024
		Min BW:	10

Export Topology

Location: /home/mhm/n10.brite

File Format(s): [✓ BRITE] OTTER

Use Java Exe

```
org.cloudbus.cloudsim.brite.I
org.cloudbus.cloudsim.brite.A
org.cloudbus.cloudsim.brite.A
org.cloudbus.cloudsim.brite.T
org.cloudbus.cloudsim.brite.L
org.cloudbus.cloudsim.brite.V
org.cloudbus.cloudsim.core
org.cloudbus.cloudsim.core.p
org.cloudbus.cloudsim.distrib
org.cloudbus.cloudsim.lists
org.cloudbus.cloudsim.netwo
org.cloudbus.cloudsim.power
  PowerDatacenter.java
```

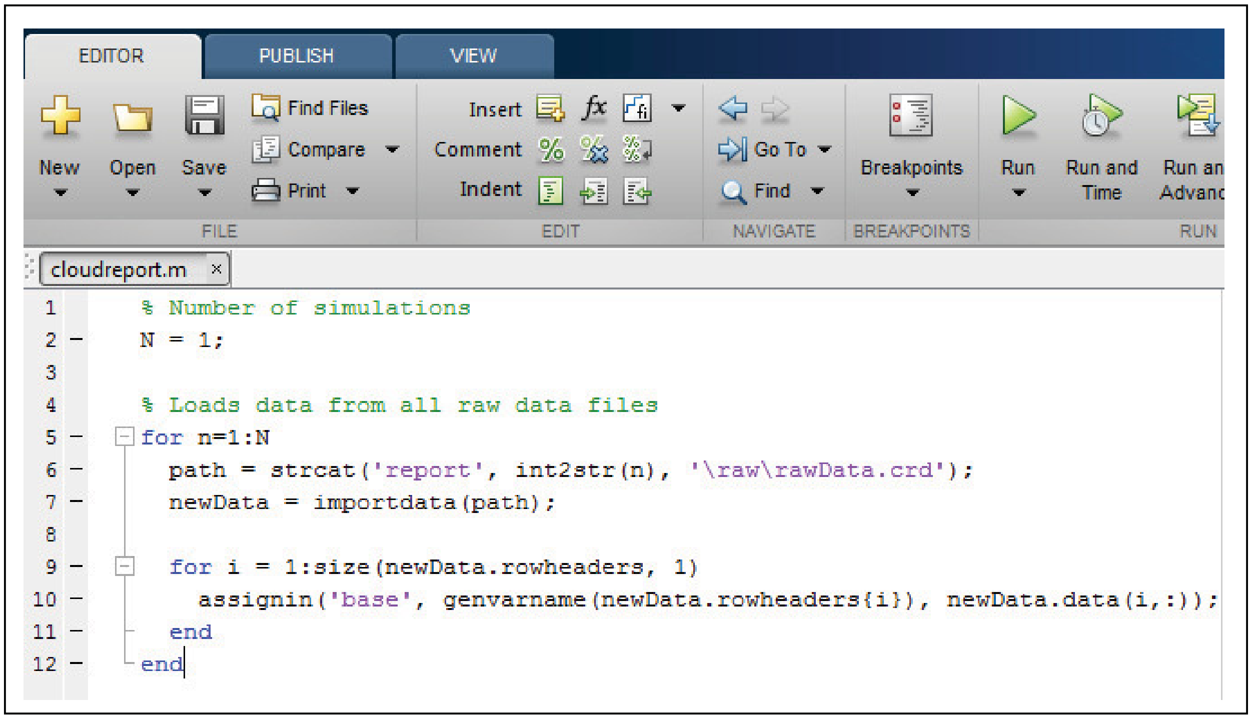
```
[MESSAGE]: [MESSAGE]: E
[MESSAGE]: [MESSAGE]: T
[MESSAGE]: Arquivo: /ho
[MESSAGE]: java -Xmx256
[MESSAGE]: [MESSAGE]: G
[MESSAGE]: [DEBUG]   : P
[MESSAGE]: [MESSAGE]: P
[MESSAGE]: [DEBUG]   : F
[MESSAGE]: [MESSAGE]: C
[MESSAGE]: [MESSAGE]: A
[MESSAGE]: [MESSAGE]: C
[MESSAGE]:      Connect
[MESSAGE]: [MESSAGE]: E
[MESSAGE]: [MESSAGE]: E
[MESSAGE]: [MESSAGE]: T
[MESSAGE]: Arquivo: /ho
```

Figure A IV. 1 BRITE graphical interface in RealCloudSim

APPENDIX V

IMPORTING DATA TO MATLAB

Loading data through GUI needs a lot of time. The code below can be used to load data from several reports at once.

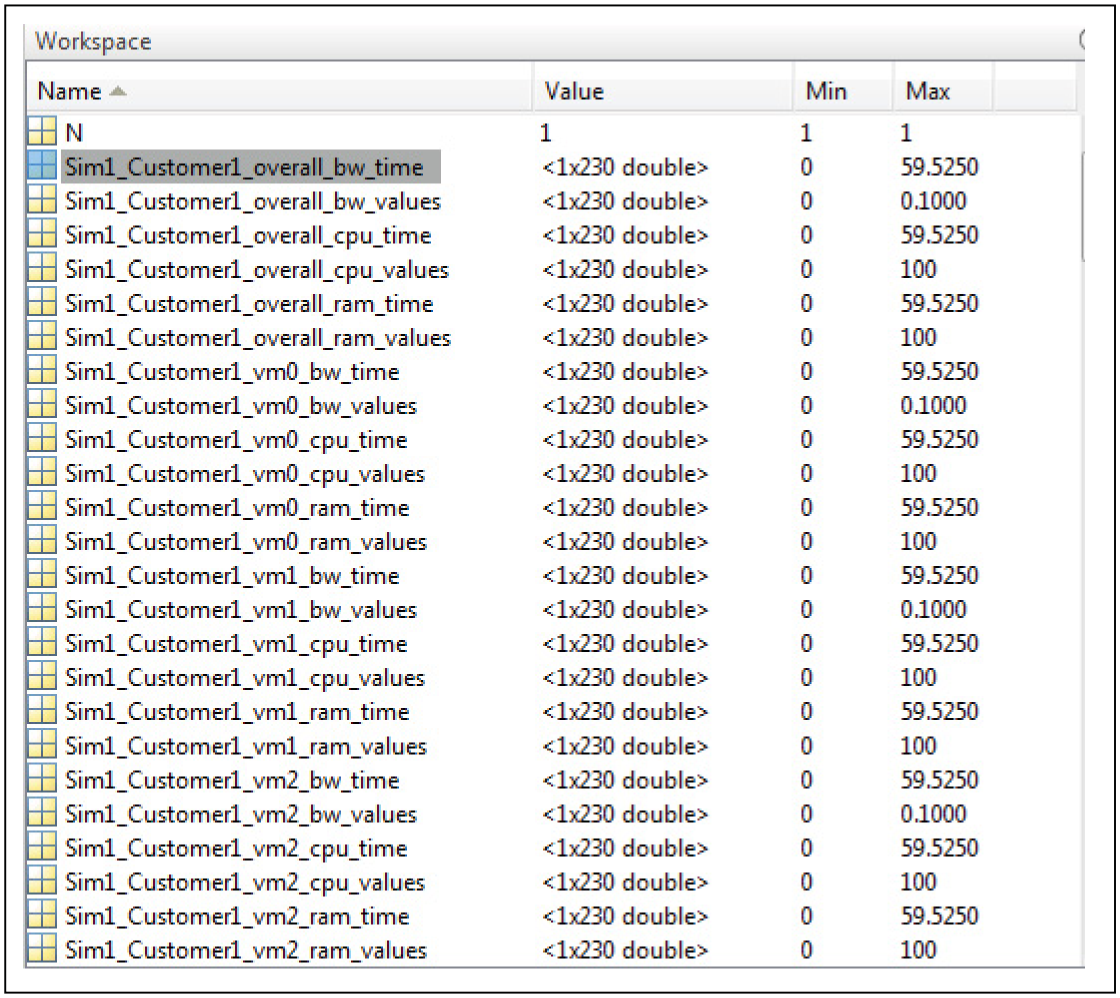


```
% Number of simulations
N = 1;

% Loads data from all raw data files
for n=1:N
  path = strcat('report', int2str(n), '\raw\rawData.crd');
  newData = importdata(path);

  for i = 1:size(newData.rowheaders, 1)
    assignin('base', genvarname(newData.rowheaders{i}), newData.data(i,:));
  end
end
```

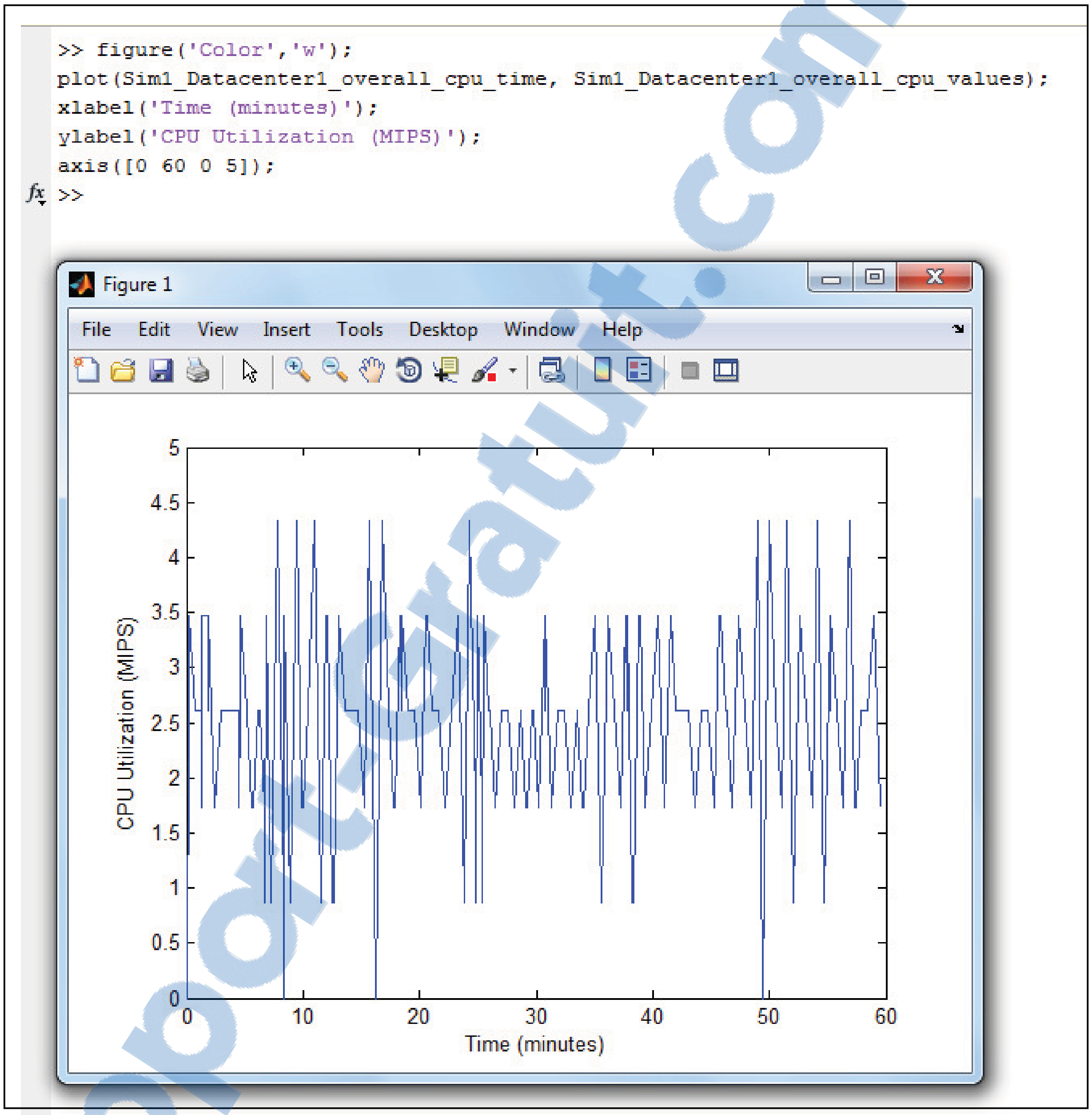
Figure A V. 1 Script of loading data into Matlab



Workspace

Name	Value	Min	Max
N	1	1	1
Sim1_Customer1_overall_bw_time	<1x230 double>	0	59.5250
Sim1_Customer1_overall_bw_values	<1x230 double>	0	0.1000
Sim1_Customer1_overall_cpu_time	<1x230 double>	0	59.5250
Sim1_Customer1_overall_cpu_values	<1x230 double>	0	100
Sim1_Customer1_overall_ram_time	<1x230 double>	0	59.5250
Sim1_Customer1_overall_ram_values	<1x230 double>	0	100
Sim1_Customer1_vm0_bw_time	<1x230 double>	0	59.5250
Sim1_Customer1_vm0_bw_values	<1x230 double>	0	0.1000
Sim1_Customer1_vm0_cpu_time	<1x230 double>	0	59.5250
Sim1_Customer1_vm0_cpu_values	<1x230 double>	0	100
Sim1_Customer1_vm0_ram_time	<1x230 double>	0	59.5250
Sim1_Customer1_vm0_ram_values	<1x230 double>	0	100
Sim1_Customer1_vm1_bw_time	<1x230 double>	0	59.5250
Sim1_Customer1_vm1_bw_values	<1x230 double>	0	0.1000
Sim1_Customer1_vm1_cpu_time	<1x230 double>	0	59.5250
Sim1_Customer1_vm1_cpu_values	<1x230 double>	0	100
Sim1_Customer1_vm1_ram_time	<1x230 double>	0	59.5250
Sim1_Customer1_vm1_ram_values	<1x230 double>	0	100
Sim1_Customer1_vm2_bw_time	<1x230 double>	0	59.5250
Sim1_Customer1_vm2_bw_values	<1x230 double>	0	0.1000
Sim1_Customer1_vm2_cpu_time	<1x230 double>	0	59.5250
Sim1_Customer1_vm2_cpu_values	<1x230 double>	0	100
Sim1_Customer1_vm2_ram_time	<1x230 double>	0	59.5250
Sim1_Customer1_vm2_ram_values	<1x230 double>	0	100

Figure A V. 2 The loaded data into Matlab by differen t categories



```
>> figure('Color','w');
plot(Sim1_Datacenter1_overall_cpu_time, Sim1_Datacenter1_overall_cpu_values);
xlabel('Time (minutes)');
ylabel('CPU Utilization (MIPS)');
axis([0 60 0 5]);
>>
```

Figure A V. 3 The CPU uti lization of PMs within a data center during one hour

LIST OF REFERENCES

Atashpaz-Gargari, E., and C. Lucas. 2007. « Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition ». In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*. (25-28 Sept. 2007), p. 4661-4667.

Beloglazov, A., and R. Buyya. 2012a. « Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers Under Quality of Service Constraints ». *Parallel and Distributed Systems, IEEE Transactions on,* vol. PP, n° 99, p. 1-1.

Beloglazov, A., and R. Buyya. 2013. « Managing Overloaded Hosts for Dynamic Consolidation of Virtual Machines in Cloud Data Centers under Quality of Service Constraints ». *Parallel and Distributed Systems, IEEE Transactions on,* vol. 24, n° 7, p. 1366-1379.

Beloglazov, Anton. 2013. « Energy-Efficient Management of Virtual Machines in Data Centers for Cloud Computing ». THE UNIVERSITY OF MELBOURNE.

Beloglazov, Anton, Jemal Abawajy and Rajkumar Buyya. 2012. « Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing ». *Future Generation Computer Systems,* vol. 28, n° 5, p. 755-768.

Beloglazov, Anton, and Rajkumar Buyya. 2012b. « Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers ». *Concurr. Comput. : Pract. Exper.,* vol. 24, n° 13, p. 1397-1420.

Beloglazov, Anton, and Rajkumar Buyya. 2012c. « Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in Cloud data centers ». *Concurrency and Computation: Practice and Experience,* vol. 24, n° 13, p. 1397-1420.

Bikeborg. 2012. « Cloud computing layers ». < https://commons.wikimedia.org/wiki/File:Cloud_computing_layers.png >.

Bo, Li, Li Jianxin, Huai Jinpeng, Wo Tianyu, Li Qin and Zhong Liang. 2009. « EnaCloud: An Energy-Saving Application Live Placement Approach for Cloud Computing Environments ». In *Cloud Computing, 2009. CLOUD '09. IEEE International Conference on*. (21-25 Sept. 2009), p. 17-24.

Buyya, Rajkumar, Chee Shin Yeo, Srikumar Venugopal, James Broberg and Ivona Brandic. 2009. « Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility ». *Future Gener. Comput. Syst.,* vol. 25, nº 6, p. 599-616.

Calheiros, Rodrigo N., Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose and Rajkumar Buyya. 2011. « 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, p. 23-50.

Cardosa, Michael, Madhukar R. Korupolu and Aameek Singh. 2009. « Shares and utilities based power consolidation in virtualized server environments ». In *Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*. (New York, NY, USA), p. 327-334. 1688986: IEEE Press.

Chase, Jeffrey S., Darrell C. Anderson, Prachi N. Thakar, Amin M. Vahdat and Ronald P. Doyle. 2001. « Managing energy and server resources in hosting centers ». In *Proceedings of the eighteenth ACM symposium on Operating systems principles*. (Banff, Alberta, Canada), p. 103-116. 502045: ACM.

Chowdhury, N. M. M. K., and R. Boutaba. 2009. « Network virtualization: state of the art and research challenges ». *Communications Magazine, IEEE,* vol. 47, nº 7, p. 20-26.

« Cloud Report CloudSim Simulator ». 2013. < http://cloudsim-setup.blogspot.ca/2013/01/cloud-report-cloudsim-seulator.html >.

« CloudSim FAQ ». 2012. < https://code.google.com/p/cloudsim/wiki/FAQ >.

Coffman Jr, EdwardG, János Csirik, Gábor Galambos, Silvano Martello and Daniele Vigo. 2013. « Bin Packing Approximation Algorithms: Survey and Classification ». In *Handbook of Combinatorial Optimization*, sous la dir. de Pardalos, Panos M., Ding-Zhu Du and Ronald L. Graham. p. 455-531. Springer New York. < http://dx.doi.org/10.1007/978-1-4419-7997-1_35 >.

Dorigo, Marco, Gianni Di Caro and Luca M. Gambardella. 1999. « Ant algorithms for discrete optimization ». *Artif. Life,* vol. 5, nº 2, p. 137-172.

Dutta, D., and R. C. Joshi. 2011. « A genetic: algorithm approach to cost-based multi-QoS job scheduling in cloud computing environment ». In *Proceedings of the International Conference &#38; Workshop on Emerging Trends in Technology*. (Mumbai, Maharashtra, India), p. 422-427. 1980111: ACM.

Eduardo Pinheiro , Ricardo Bianchini , Enrique V. Carrera, Taliver Heath. 2001. « Load balancing and unbalancing
for power and performancee in cluster-based systems, in: Proceedings of the
Workshop on Compilers and Operating Systems for Low Power ». p. 182–195.

Eugen, Feller. 2013. « Autonomic and Energy-Efficient Management Of Large-Scale Virtualized Data Centers ». UNIVERSITÉ DE RENNES.

Fan, Xiaobo, Wolf-Dietrich Weber and Luiz Andre Barroso. 2007. « Power provisioning for a warehouse-sized computer ». *SIGARCH Comput. Archit. News,* vol. 35, nº 2, p. 13-23.

Feller, E., L. Rilling and C. Morin. 2011. « Energy-Aware Ant Colony Based Workload Placement in Clouds ». In *Grid Computing (GRID), 2011 12th IEEE/ACM International Conference on*. (21-23 Sept. 2011), p. 26-33.

Feller, Eugen. 2013. « Autonomic and Energy-Efficient Management Of Large-Scale Virtualized Data Centers ». UNIVERSITÉ DE RENNES.

Friedman, Marian. 2011. *IBM Data Center Networking: Planning for Virtualization and Cloud Computing*. IBM Corporation.

Gao, Yongqiang, Haibing Guan, Zhengwei Qi, Yang Hou and Liang Liu. 2013. « A multi-objective ant colony system algorithm for virtual machine placement in cloud computing ». *Journal of Computer and System Sciences,* vol. 79, nº 8, p. 1230-1242.

Glauco Estácio Gonçalves, Patrícia Takako Endo, Thiago DamascenoCordeiro, André Vitor de Almeida Palhares, Djamel Sadok, Judith Kelner,Bob Melander, Jan-Erik Mångs. 2011. « Resource Allocation in Clouds: Concepts, Tools and Research Challenges ». *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos - SBRC 2011.*

Gmach, D., J. Rolia, L. Cherkasova, G. Belrose, T. Turicchi and A. Kemper. 2008. « An integrated approach to resource pool management: Policies, efficiency and quality metrics ». In *Dependable Systems and Networks With FTCS and DCC, 2008. DSN 2008. IEEE International Conference on*. (24-27 June 2008), p. 326-335.

Gmach, Daniel, Jerry Rolia, Ludmila Cherkasova and Alfons Kemper. 2009. « Resource pool management: Reactive versus proactive or let's be friends ». *Comput. Netw.,* vol. 53, nº 17, p. 2905-2922.

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., 372 p.

Guérout, Tom, Thierry Monteil, Georges Da Costa, Rodrigo Neves Calheiros, Rajkumar Buyya and Mihai Alexandru. 2013. « Energy-aware simulation with DVFS ». *Simulation Modelling Practice and Theory,* vol. 39, nº 0, p. 76-91.

Hai, Zhong, Tao Kun and Zhang Xuejie. 2010. « An Approach to Optimized Resource Scheduling Algorithm for Open-Source Cloud Systems ». In *ChinaGrid Conference (ChinaGrid), 2010 Fifth Annual*. (16-18 July 2010), p. 124-129.

James M. Kaplan, William Forrest, Noah Kindle. 2008. « Revolutionizing Data Center Energy Efficiency ».

Jing, Xu, and J. A. B. Fortes. 2010. « Multi-Objective Virtual Machine Placement in Virtualized Data Center Environments ». In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*. (18-20 Dec. 2010), p. 179-188.

Jinhua, Hu, Gu Jianhua, Sun Guofei and Zhao Tianhai. 2010. « A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment ». In *Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on*. (18-20 Dec. 2010), p. 89-96.

Kliazovich, D., P. Bouvry, Y. Audzevich and S. U. Khan. 2010. « GreenCloud: A Packet-Level Simulator of Energy-Aware Cloud Computing Data Centers ». In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*. (6-10 Dec. 2010), p. 1-5.

Koomey, J. G. 2008. « Worldwide electricity used in data centers ». vol. 3.

Koomey, Jonathan. 2011. « Growth in Data center electricity use 2005 to 2010 ». *Analytics Press*.

Kyong, I. Ku, Choi Won-Hyuk, Chung Moonyoung, Kim Kiheon, Kim Won-Young and S. J. Hur. 2010. « Method for distribution, execution and management of the customized application based on software virtualization ». In *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*. (7-10 Feb. 2010) Vol. 1, p. 493-496.

L. Minas , B. Ellison. 2009. « Energy Efficiency for Information Technology: How to Reduce Power Consumption in Servers and Data Centers ». *Intel Press*.

Leinberger, W., G. Karypis and V. Kumar. 1999. « Multi-capacity bin packing algorithms with applications to job scheduling under multiple constraints ». In *Parallel Processing, 1999. Proceedings. 1999 International Conference on*. (1999), p. 404-412.

Mohammadhossein Malekloo, Nadjia Kara. 2014. « Multi-objective ACO Virtual Machine Placement in Cloud Computing Environments ». In *Globecom 2014 workshop- Cloud computing systems, Networks and Applications*. (Austin, TX).

NIST. < http://www.nist.gov/itl/cloud/index.cfm >.

Obitko, Marek. 2011. « XI. Crossover and Mutation ». < http://www.obitko.com/tutorials/genetic-algorithms/crossover-mutation.php >.

P. Johnson, T. Marker. 2009. « Data centre energy efficiency metrics ». *Tech. Rep.*

Raghavendra, Ramya, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang and Xiaoyun Zhu. 2008. « No "power" struggles: coordinated multi-level power management for the data center ». *SIGARCH Comput. Archit. News,* vol. 36, n^o 1, p. 48-59.

Rajkumar Buyya, Anton Beloglazov. 2010. « Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges ». In *PDPTA 2010*. (July 12-15, 2010). Las Vegas, USA.

Rebecca Scudder. 2011. « Visualizing the Workings of Cloud Computing With Diagrams ». < http://www.brighthub.com/environment/green-computing/articles/127086.aspx >. Consulté le 11/29/2011.

Rocha, Lucio Agostinho. 2013. *REALcloudSim version 9.8 Documentation*.

Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, César A. F. De Rose, Rajkumar Buyya. 2010. « CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms ».

Rossi, Francesca, Peter van Beek and Toby Walsh. 2006. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc.

Sahoo, J., S. Mohapatra and R. Lath. 2010. « Virtualization: A Survey on Concepts, Taxonomy and Associated Security Issues ». In *Computer and Network Technology (ICCNT), 2010 Second International Conference on*. (23-25 April 2010), p. 222-226.

Sankaranarayanan, Ananth Narayan, Somsubhra Sharangi and Alexandra Fedorova. 2011. « Abstract only: global cost diversity aware dispatch algorithm for heterogeneous data centers ». *SIGSOFT Softw. Eng. Notes,* vol. 36, n^o 5, p. 41-41.

Schrijver, Alexander. 1986. *Theory of linear and integer programming*. John Wiley \&amp; Sons, Inc., 471 p.

Schwan, R. Nathuji and K. 2007. « VirtualPower: Coordinated Power Management in Virtualized Enterprise Systems ». *ACM SIGOPS Operationg Systems Review,* vol. 41, n^o 6, p. 265-278.

Sosinsky, Barrie. 2011. *Cloud Computing Bible*. John Wiley & Sons.

SOUALHIA, Mbarka. 2013. « RESOURCES MANAGEMENT ARCHITECTURE AND ALGORITHMS FOR VIRTUALIZED IVR APPLICATIONS IN CLOUD ENVIRONMENT ». ÉCOLE DE TECHNOLOGIE SUPÉRIEURE.

Stillwell, M., F. Vivien and H. Casanova. 2012. « Virtual Machine Resource Allocation for Service Hosting on Heterogeneous Distributed Platforms ». In *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*. (21-25 May 2012), p. 786-797.

Stillwell, Mark, David Schanzenbach, Fr, #233, #233, ric Vivien and Henri Casanova. 2010. « Resource allocation algorithms for virtualized service hosting platforms ». *J. Parallel Distrib. Comput.,* vol. 70, n° 9, p. 962-974.

Talia, D. 2012. « Clouds Meet Agents: Toward Intelligent Cloud Services ». *Internet Computing, IEEE,* vol. 16, n° 2, p. 78-81.

Teng, Fei. 2012. « MANAGEMENT DES DONN´EES ET ORDONNANCEMENT DES T^ACHES SUR ARCHITECTURES DISTRIBU´EES ». E´COLE CENTRALE PARIS.

The MathWorks, Inc. 2014. « Genetic Algorithm Options ». < http://www.mathworks.com/help/gads/genetic-algorithm-options.html >.

Vaquero, Luis M., Luis Rodero-Merino, Juan Caceres and Maik Lindner. 2008. « A break in the clouds: towards a cloud definition ». *SIGCOMM Comput. Commun. Rev.,* vol. 39, n° 1, p. 50-55.

Voorsluys, William, James Broberg and Rajkumar Buyya. 2011. « Introduction to Cloud Computing ». In *Cloud Computing*. p. 1-41. John Wiley & Sons, Inc. < http://dx.doi.org/10.1002/9780470940105.ch1 >.

W. F. James M. Kaplan, Noah Kindle. 2008. « Revolutionizing Data Center Energy Efficiency ».

Wikipedia. < http://en.wikipedia.org/wiki/Cloud_computing >.

Wu, Grant, Maolin Tang, Yu-Chu Tian and Wei Li. 2012. « Energy-Efficient virtual machine placement in data centers by genetic algorithm ». In *Proceedings of the 19th international conference on Neural Information Processing - Volume Part III*. (Doha, Qatar), p. 315-323. 2426976: Springer-Verlag.

Xindong, You, Xu Xianghua, Wan Jian and Yu Dongjin. 2009. « RAS-M: Resource Allocation Strategy Based on Market Mechanism in Cloud Computing ». In

ChinaGrid Annual Conference, 2009. ChinaGrid '09. Fourth. (21-22 Aug. 2009), p. 256-263.

Yue, Minyi. 1991. « A simple proof of the inequality FFD (L) $\leq$ 11/9 OPT (L) + 1, $\forall$L for the FFD bin-packing algorithm ». *Acta Mathematicae Applicatae Sinica,* vol. 7, n° 4, p. 321-331.

Zhu, Xiaoyun, Donald Young, Brian J. Watson, Zhikui Wang, Jerry Rolia, Sharad Singhal, Bret Mckee, Chris Hyser, Daniel Gmach, Robert Gardner, Tom Christian and Ludmila Cherkasova. 2009. « 1000 islands: an integrated approach to resource management for virtualized data centers ». *Cluster Computing,* vol. 12, n° 1, p. 45-57.