THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier)*

Présentée et soutenue le *03/07/2014* **par :**

CHEIKHOU THIAM

Anti Load-Balancing for Energy-Aware Distributed Scheduling of Virtual Machines.

JURY

FREDERIC MAGOULES	Professeur d'Université	Président du Jury
LAURENT LEFÈVRE	Chargé de recherche, HDR	Rapporteur
PHILIPPE ROOSE	Maître de Conférence, HDR	Rapporteur
JEAN-PAUL BAHSOUN	Professeur d'Université	Examinateur
JEAN-MARC PIERSON	Professeur d'Université	Directeur de thèse
GEORGES DA COSTA	Maître de Conférence	Co-directeur

École doctorale et spécialité :
MITT : Domaine STIC : Réseaux, Télécoms, Systèmes et Architecture

Unité de Recherche :
Institut de Recherche en Informatique de Toulouse (UMR 5505)

Directeur(s) de Thèse :
Jean-Marc Pierson et *Georges Da Costa*

Rapporteurs :
Laurent Lefèvre et *Philippe Roose*

Abstract

The multiplication of Cloud computing has resulted in the establishment of large-scale data centers around the world containing thousands of compute nodes. However, Cloud consume huge amounts of energy. Energy consumption of data centers worldwide is estimated at more than 1.5% of the global electricity use and is expected to grow further. A problem usually studied in distributed systems is to evenly distribute the load. But when the goal is to reduce energy consumption, this type of algorithms can lead to have machines largely under-loaded and therefore consuming energy unnecessarily. This thesis presents novel techniques, algorithms, and software for distributed dynamic consolidation of Virtual Machines (VMs) in Cloud. The main objective of this thesis is to provide energy-aware scheduling strategies in cloud computing for energy saving. To achieve this goal, we use centralized and decentralized approaches. Contributions in this method are presented these two axes. The objective of our approach is to reduce data center's total energy consumed by controlling cloud applications' overall energy consumption while ensuring cloud applications' service level agreement. Energy consumption is reduced by dynamically deactivating and reactivating physical nodes to meet the current resource demand. The key contributions are:

- First, we present an energy aware clouds scheduling using anti-load balancing algorithm : concentrate the load on a minimum number of severs. The goal is to turn off the machines released and therefore minimize the energy consumption of the system.

- The second axis proposed an algorithm which works by associating a credit value with each node. The credit of a node depends on its affinity to its jobs, its current workload and its communication behavior. Energy savings are achieved by continuous consolidation of VMs according to current utilization of resources, virtual network topologies established between VMs, and thermal state of computing nodes. The experiment results, obtained with a simulator which extends CloudSim (EnerSim), show that the cloud application energy consumption and energy efficiency are being improved.

- The third axis is dedicated to the consideration of a decentralized dynamic scheduling approach entitled Cooperative scheduling Anti-load balancing Algorithm for cloud. It is a decentralized approach that allows cooperation between different sites. To validate this algorithm, we have extended the simulator MaGateSim.

With an extensive experimental evaluation with a real workload dataset, we got the conclusion that both the approach using centralized and decentralized algorithms can reduce energy consumed by data centers.

Keywords: Energy, Heuristic, Virtual Machines, Cloud, Migration

Acknowledgments

Completion of my PhD required countless selfless acts of support, generosity, and time by people in my personal and academic life. Foremost, i can only attempt to humbly acknowledge and thank the people and institutions that have given so freely throughout my PhD career and made this dissertation possible. I am thankful to the Université de Thiès (UT) of Sénégal for providing uninterrupted funding support throughout my PhD career. I would like to express my greatest appreciation and gratitude to my supervisory committee chair (Jean-Marc Pierson and Georges Da Costa) for their guidance during my doctoral studies. I also express my gratitude to my supervisory committee members (Patricia Stolf, François Thiébolt, Amal Sayah) for their help. I would like to thank my advisor Georges Da Costa for all his enthusiasm and outstanding support without which this thesis would not have been possible. Despite having many responsibilities he has always found time to discuss about new ideas and provide detailed feedbacks and suggestions on my work. A very special thanks to Jean-Marc Pierson for his excellent advising during my stay and supporting my activities at the SEPIA team. Thank you Jean-Marc and Georges, for being excellent persons and researchers. It has been a great honour and pleasure for me to work with you during the last four and a half years and I hope to continue in the future.

I would like to gratitude the members of my jury. Thank you Philipe Roose and Laurent Lefevre for making me the honour to review my work. I also thank Frederic Magoules and Jean-Paul Bahsoun for taking your time to evaluate my work.

I want to thank the team of Grid'5000 for making available workloads used for the validation of our algorithms and some information about topology.

The SEPIA team for all our nice discussions, coffee breaks, and lunches. Chantal Morand which contributed to the warm atmosphere which i have experienced during my stay in France. This thesis is devoted to my parents, my mother Khady, my brothers Bounama and Abdou Rakhmane. I can't thank you enough for everything you did for me. Huge thanks to my wife Aminata Thiam for always being open minded and ready to discuss about my work. Your love, patience, and support has

always been a great motivation for me to ever move forward. The contributions presented in this thesis were funded by the UT. Lastly, I can only wish if my father, El Hadji Lamine Thiam, was still alive to embrace me on achieving this milestone. He will always remain my constant.

*To my **children** who have endured during my long absences,*
*to my father **El Hadji Lamine** , who always dared to dream,*
*and to my mother **Khady** and my wife **Aminata** (late) for making it a reality.*

Contents

Part I

Part I Complete dissertation: English version

Chapter 1

Introduction

Contents

This chapter introduces the context of the research explored in this thesis. It starts with the fundamental motivations behind decentralized and coordinated organization of Grid/Cloud systems; including resource allocation systems. The chapter thereafter provides discussion on the problem and the issues and scope of the work and the outline of the thesis.

1.1 Motivation

Cloud computing has emerged as a new business model of computation and storage resources based on the pay-as-you-go model access to potentially significant amounts of remote datacenter capabilities. Customer billing depends on the services they have consumed so far. One particularly cloud Services offered by cloud providers which has gained a lot of attraction over the past years is Infrastructure As-a-Service (IaaS). In IaaS clouds, resources such as compute and storage are provisioned on-demand by the cloud providers. Many cloud providers such as Amazon,Google, Rackspace, SalesForce have appeared and are now offering a huge amount of services such as compute capacity and data storage on demand. In order to support the

customers increasingly service demands, cloud providers have deployed an increasing number of large-scale data centers. The management of such data centers, requires the cloud providers to solve a number of challenges. Particularly, cloud providers now design and implement large-scale IaaS cloud computing systems. As Cloud computing infrastructure consumes enormous amounts of electrical power leading to operational costs that exceed the cost of the infrastructure in few years, many researches have been improving energy efficiency.

There are three energy-saving approaches : "do less work", "slow down" and "turn off idle elements". In the "do less work" strategy, the minimum load shall be executed allowing optimization of process leads to a low energy consumption. With "slow down" strategy, the faster a process runs, the more resource intensive it becomes.There are two ways of slowing down processes. They can be run with adaptive speeds, by selecting the minimal required speed to complete the process in time. Alternatively, buffering can be introduced so that instead of running a process immediately upon arrival, one can collect new tasks until the buffer is full and then execute them in bulk. This allows for components to be temporarily switched off resulting in lower power consumption. This allows for components to be temporarily switched off resulting in lower power consumption. The "turn off idle elements" strategy refers to the possibilities offered by exploiting a low-consumption state (sleep mode). Basically, the sleep mode aims at switching into an idle mode the devices during periods of inactivity. Unloaded servers can be dynamically put into sleep mode during low-load periods, contributing to great power savings. For data centers and grid/cloud infrastructures, if properly employed, the sleep mode may represent a very useful mean for limiting power consumption of lightly loaded sites.

In cloud computing, existing techniques for energy savings can be divided into two categories : (1) First, Dynamic Voltage/Frequency Scaling (DVFS). Power savings are obtained by adjusting the operating clock to scale down the supply voltages for the circuits. Although this approach may achieve a significant reduction of the energy consumption, it depends on the hardware components' settings to perform scaling tasks; (2) Then one of the technologies to reduce energy consumed by a

data center is to use consolidate technique. It is the merger of several things into one. Applied to the computer world, it is to assemble more resources into one. Virtualization was mainly used in server consolidation projects. Server consolidation aims at minimizing the number of physical servers required to host a group of virtual machines. In this thesis we will use the technique of consolidation. Given the importance of energy savings, energy-efficient IaaS cloud management systems must be designed. Several attempts have been made over the past years to design and implement IaaS cloud management systems to facilitate the creation of private IaaS clouds.

Given the increasing data center scales, such systems are faced with challenges in terms of scalability, autonomy, and energy-efficiency. However, many of the existing attempts to design and implement IaaS cloud systems for private clouds are still based on centralized architectures, have limited autonomy, and lack of energy saving mechanisms. Consequently, they are subject to points of failure like limited scalability and low energy efficiency.

1.2 Research Problems and Objectives

The goal of this thesis is to design, implement, and evaluate energy aware scheduling algorithms in distributed systems. To achieve its main goal this thesis investigates the following research problems :

- **How to define a scheduling algorithm**. Job scheduling strategies have been studied within a variety of scenarios related to the complexity of business and scientific computing processes. The goal of job scheduling is to meet the performance requirements, and get the lowest total power consumption of all tasks. Energy saving management of distributed systems scheduling strategy must take the following aspects into account: cyclical, continuity, communication overhead between node server, data correlation, node heterogeneity and time limit. In the case of energy consumed in data centers, consolidation is one of the multiple solutions used. With distributed scheduling algorithms, it is recommended to chose cooperative techniques;

- **Network structure .** The type of algorithm defined in the previous section depends on how is structured the network : centralized, distributed or hybrid;

- **Migration :**

  - When to migrate VMs. Dynamic VM consolidation comprises two basic processes: (1) migrating VMs from overloaded servers to avoid performance degradation; and (2) migrating VMs from underloaded servers to improve the utilization of resources and minimize energy consumption. An important decision that must be made in both situations is determining when migrating VMs to minimize energy consumption, while satisfying the defined QoS constraints;

  - Which VMs to migrate. Once a decision to migrate VMs is made, it is necessary to choose one or more VMs from the set of VMs allocated to the server, which must be reallocated to other servers. The problem consists in determining the best subset of VMs to migrate to servers which are not in the same situation, that will provide the most beneficial system reconfiguration.

  - Where to migrate the VMs selected for migration. This is to determine the best placement of new VMs or the VMs selected on server for migration to other servers is another essential aspect that affects the quality of VM consolidation and energy consumption by the system.

- **Consolidation :**

  - When and which physical nodes to switch on/off. To optimize energy consumption by the system and avoid violations of the QoS requirements, it is necessary to determine of effective manner, when and which physical nodes should be disabled to save energy, or reactivated to handle increases in the demand for resources;

  - How to design distributed dynamic VM consolidation algorithms. To provide scalability and eliminate single points of failure, it is necessary

to use a decentralized approach for dynamic VM consolidation algorithms. The problem is that traditionally global resources management algorithms are centralized. Therefore, a good approach is to propose a distributed dynamic VM consolidation system .

To deal with the challenges associated with the above research problems, the following objectives which are to improve cloud's total energy efficiency by controlling cloud applications' overall energy consumption while ensuring cloud applications' service level agreement, have been delineated :

- Explore, analyze, and classify the research in the area of energy-efficient computing in order to gain a systematic understanding of the existing techniques and approaches;
- Develop algorithms for energy-efficient distributed dynamic VM consolidation for cloud environments satisfying QoS constraints;
- Design and implement a distributed dynamic VM consolidation simulator that can be used to evaluate the proposed algorithms;
- Ease of task Management : we design a system which is flexible enough to allow for dynamic addition and removal of servers. As system components can fail at any time, it is desirable for a system to heal in the event of failures or reallocate tasks to other servers without human intervention.
  - Conservation of the execution context : It must be possible to stop the execution process of the task and restart it where it has stopped. Execution time can be reduced when the task migrates to a more powerful node;
  - Slowdown prevention : task slowdown increases the execution time and therefore increases the energy consumed by hosts and impact users.
- Energy saving : One of our goals is to propose task placements management algorithms which are capable of creating idle times, transitioning idle servers

in a power saving state and waking them up once required (e.g. when load increases).

1.3 Methodology

We study different techniques used in the operation of the Cloud and data centers. Some research on distributed systems and cloud and energy savings were studied in chapter 2. To test the proposed algorithms , simulators which manage energy-efficiency for cloud environment have been developed. These simulators operate in centralized and decentralized level. We used technical tools such as cooperation, collaboration and virtualization. To meet the full meaning of sustainability that can be built on energy-efficient scheduling algorithms proposed , we analyze existing solutions and compare them to choose one that we are improving. And we show how the consolidation is able to decrease energy consumption compared to the reduction of energy costs taking into account the QoS constraints .

1.4 Contributions

This thesis makes the following contributions:

- **Scheduling Anti load-balancing Algorithm.** Our first contribution is a Scheduling Anti load-balancing Algorithm (ALBA) technique. This technique works in conjunction with scheduling algorithms presented in our centralized and decentralized approaches. It is a technique for real-time tasks which decides where exactly a task shall execute in a distributed system. In the latter (distributed systems), a problem usually studied is how to evenly distribute workload. But when the goal is to reduce energy consumption, this type of algorithms can lead to have computers largely under-loaded and therefore consuming energy unnecessarily. Our fields of study will be the management of virtual machines in the cluster or grid-type systems. Here we will therefore look at the opposite problem : concentrate the load on a minimum number of machines. The goal is to turn off the released servers and therefore minimize

the energy consumption of the system. We will use an algorithm of global decision: a server is selected, it distribute its load to other servers and then it switched off. To study this problem we have proceeded in three phases: (i) Model the problem (ii) Simulation (iii) Experimentation based on virtual machines.

- **Energy aware clouds scheduling using anti load-balancing algorithm.** Cloud computing is a highly scalable and cost-effective infrastructure for running HPC, enterprise and Web applications. However rapid growth of the demand for computational power by scientific, business and web-applications has led to the creation of large-scale data centers consuming enormous amounts of electrical power. Hence, energy-efficient solutions are required to minimize this energy consumed. The objective of our approach is to improve data center's total energy efficiency by controlling cloud applications' overall energy consumption while ensuring cloud applications' service level agreement. This contibution presents an Energy aware clouds scheduling using anti load-balancing algorithm (EACAB). The proposed algorithm works by associating a credit value with each node. The credit of a node depends on its affinity to its jobs, its current workload and its communication behavior. Energy savings are achieved by continuous consolidation of VMs according to current utilization of resources, virtual network topologies established between VMs and thermal state of computing nodes.

- **Cooperative scheduling Anti load-balancing Algorithm for cloud.** Due to the characteristics of clouds, meta-scheduling turns out to be an important scheduling pattern because it is responsible for orchestrating resources managed by independent local schedulers and bridges the gap between participating nodes. Likewise, to overcome issues such as bottleneck, overloading, under loading and impractical unique administrative management, which are normally led by conventional centralized or hierarchical schemes, the distributed scheduling scheme is emerging as a promising approach because of its capability with regards to scalability and flexibility.

In this contribution, we introduce a decentralized dynamic scheduling approach entitled Cooperative scheduling Anti load-balancing Algorithm for cloud (CSAAC) which is based on the idea of CASA [Huang 2013b]. The goal of CSAAC is to optimize performance and to achieve energy gain over the scope of overall cloud, instead of individual participating nodes.

- **Simulators.** We develop a simulator Enersim 1 which exends ALEA and CloudSim. It is a tool that will be used to simulate a centralized approach. Then we develop a simulator Enersim 2 which extend MagateSim and Enersim 1. Enersim 2 simulates decentralized algorithms. MaGateSim is extended by adding properties that allow it to take into account energy and migration.

1.5 Outline of the Thesis

This thesis is organized as follows:

- Chapter 2 covers the state of the art. Particularly, it first presents the context of this dissertation by giving a brief introduction to server virtualization, autonomic computing, and cloud computing. Then, existing energy management approaches in computing clusters are reviewed. Understanding the energy saving approaches is mandatory to position the energy management contributions of this work.

- Chapter 3 presents our contribution. First a scheduling algorithm named Anti load-balancing Algorithm is presented. Then an energy aware clouds scheduling using anti load-balancing algorithm is presented. Finally, in order to improve its scalability, a fully decentralized VM consolidation system based on an unstructured P2P network of physical machines (PMs) is proposed.

- Chapter 4 presents results of the centralized VM consolidation algorithm and the fully decentralized VM consolidation algorithm.

- Chapter 5 concludes this manuscript by summarizing our contributions and presenting future research directions.

Chapter 2

State of the art

Contents

This PhD thesis proposes an energy-efficient IaaS cloud management system for large-scale virtualized data centers. To provide the necessary background for our work, in this chapter we present the state of the art in related fields which include server virtualization, cloud computing, energy efficient management of cloud and task placement. First, the concept of server virtualization is detailed. Server virtualization is a fundamental technology which can be used to enable efficient data center resources utilization. Then, cloud computing is presented. Finally, related works on schedulling and energy management in cloud are presented.

2.1 Virtualization

This section gives a brief definition of virtualization. Compared to our work, we focus on CPU virtualization only. Obviously, other hardware subsystems (e.g. memory and I/O devices) need to be virtualized as well to enable complete server virtualization. System virtualization is becoming ubiquitous in contemporary datacenter. Consolidating physical servers by building virtual machines (VM) clusters is universally adopted to maximize the utilization of hardware resources for services. Two fundamental but challenging requirements are to minimize virtualization overhead [Mergen 2006] and to guarantee the reliability of the built virtualized infrastructure. Therefore, low level design of VM architecture is of great significance.

It is common for a company to have a couple of servers running at 15% capacity, the latter being there to deal with any time to punctual peak loads. A server loaded at 15% do not consume much less power than a server that is loaded at 90%, and to consolidate workloads of several servers on only one can be profitable if their peak

loads do not always coincide. It would even be true if virtualization had a load of 30 % on the computer. Currently with the progress of the technique this value is greatly exceeded. Virtualization can improve the effectiveness and availability of IT resources and applications. Server virtualization is supposed to help businesses save on IT costs. It reduces costs by reducing data center physical infrastructure and energy consumption. Our study is based on saving energy therefore this virtualization technique will be a promising mechanism. Indeed virtualization is fundamental when it is necessary to use consolidation to reduce the energy consumed in data centers. With the cloud paradigm being applied broadly and the increased use of Green IT, virtualization technologies are gaining increasing importance. They promise energy and cost savings by sharing physical resources, thus making resource usage more efficient.

After the introduction, the state of the art of virtualized techniques is presented. Finally, VM live migration is introduced as a basic mechanism allowing to move VMs between physical machines (PM) with ideally no service downtime.

2.1.1 What is Virtualization?

Virtualization is a technology which provides a virtual version of a device or resource, such as a server, storage device, network or even an operating system where the resource is divided into multiple execution environments. In addition, different instances of the operating system run simultaneously on a single computer.

A virtual machine must satisfy the following conditions: (i) insulation: two machines can share physical resources of a single computer, they remain completely isolated from each other as if they were separate physical machines, (ii) compatibility: a virtual machine can be used to run any software running on an real computer, (iv) encapsulation: *encapsulates* a complete set of virtual hardware resources and software within a software package, (v) independence material: totally independent of their underlying hardware.

Multiple isolated virtual environments are created by a software installed on the server. The virtual environments are sometimes called virtual private servers, but they are also known as guests, instances, containers or emulations. When

one physical server is partitioned into several virtual machines, multiple operating system instances can be deployed, operated and managed at once on that single physical server (see Figure 2.2). This reduces the number of physical servers used by a company and therefore increases the gain of energy, the objective of this thesis. Tasks that would normally run on multiple servers will be from now on a single server. [Huai 2007] propose to improve resource utilization. Authors provides a

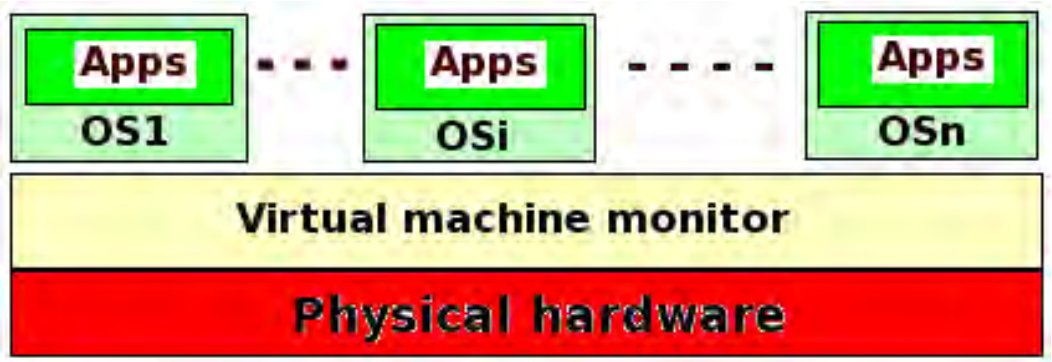

Figure 2.1: Virtual Machine Monitor example

unified integrated operating platform. They summarize up the four characteristics of virtual machine: transparent, isolation, packaging and heterogeneity.

Virtualization can also help a company cut down on energy consumption, since there are fewer physical servers consuming power. That's especially important, given the trend toward green IT planning and implementation.

2.1.2 Virtualization Techniques

There are many virtualization techniques. Insulation is a technique to trap the execution of applications in contexts. Another technique is full virtualisation. The guest OS is not aware of being virtualized and has no way of knowing that sharing the server with other OS. In contrast to virtualization, with paravirtualization the OS should be modified to run on a paravirtualization hypervisor. Paravirtualization is capable to deliver performance by allowing the hypervisor to be aware of the idioms in the operating system. The performance evaluation of these virtualization

techniques are studied by [Padala 2007], [Quétier 2007] and [Soltesz 2007].

2.1.3 Benefits of a virtualization

Server virtualization is a technique that can be used in a situation where several underutilized servers take up more space and consume more resources. This allows for more efficient use of server resources.

Technology today can combine these requirements with the need for power servers that are increasingly sought by setting mobility applications and dissemination of data. The efficiency gains achieved through virtualization provides small businesses with economies which only large companies had previously low. Servers operate at a very low percentage of their capacity. It is possible through virtualization to increase this rate to 80% usage, allowing to save money without sacrificing essential performance of applications.

Virtualization consolidates applications onto fewer physical host, which can be easily managed as a shared resource. Nothing changes for applications. Performance levels remain high, even rise to dramatic benefits. Virtualization consolidates apps on many fewer physical machines, with high levels performance and benefits such as : (1) 80% greater utilization of every server; (2) Server count reduced by a ratio of 10:1 or better; (3) And up to 50% savings in annual capital and operating costs. [virtualizationVmware 2013]. In case of failure of an application or operating system, the problem is completely isolated, without impact on other workloads. And when a server fails, the affected applications may be restarted automatically on another host. Advances of virtualization techniques are affecting all aspects of data center operation and hardware. Virtualization has allowed many companies to reduce expenditures in computer equipment.

2.1.4 VM Live Migration

Live migration is the ability to move a running VM dynamically from one server to another, without stopping it. When a server is overloaded, some VMs can be relocated dynamically to another server. When a server is underloaded its VMs

can be reallocated to another server. Figure 2.2 shows three physical machines

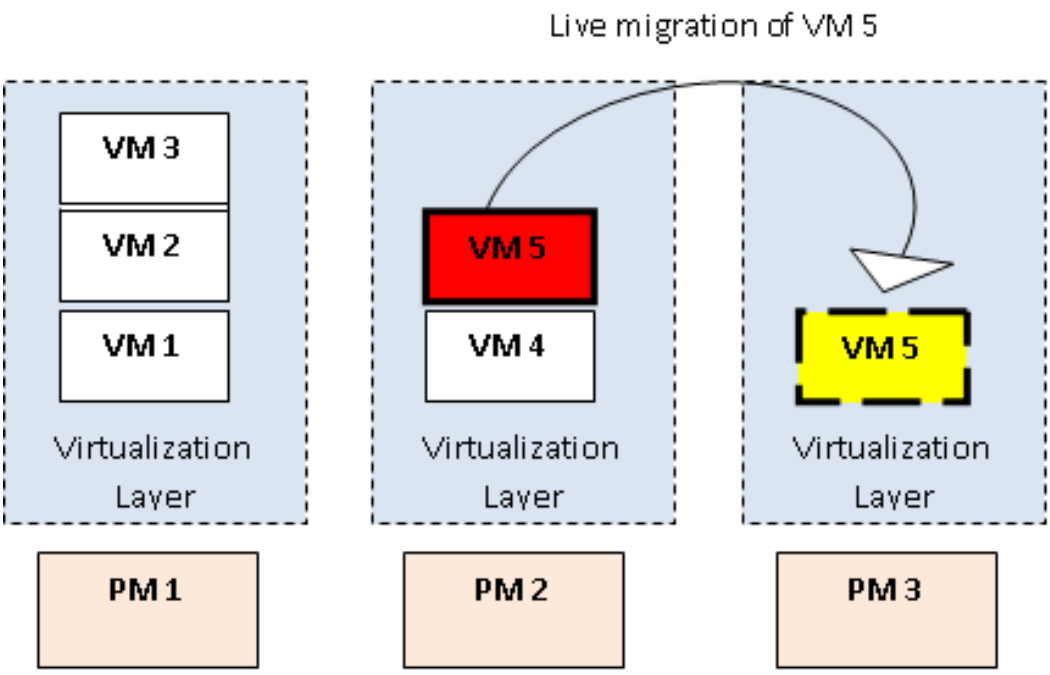

Figure 2.2: Virtualized servers with live migration capability.

with virtualization layer host five instances of operating systems (with one being migrated between PM 2 and PM 3).

Xen [Barham 2003] and the Internet Suspend-Resume Project [Kozuch 2002] developed technology to perform migration at the server level. Some commercial products such as VMotion from VMware [Ward 2002] are well known on the market. In this thesis we leverage the VM migration mechanism which plays a key role in consolidation.

2.1.5 Summary

Today, platforms such as Google's App Engine [GoogleAppEngine 2013], Amazon's EC2 [Amazon 2010], AT&T's Synaptic Hosting [Hosting 2014], and Salesforce's Force.com [Salesforce.com 2014] use virtualization techniques to host a variety of distributed applications.

Virtualization transforms radically computing for the better utilization of resources available in the data center reducing overall costs and increasing agility. It reduces operational complexity, maintains flexibility in selecting software and hardware platforms and product vendors. It also increases agility in managing heterogeneous virtual environments.

Virtualisation solution appear as alternative approaches for companies to consolidate their operation services on a physical infrastructure, whil preserving specific functionalities inside the Cloud perimeter (e.g., security, fault tolerance, relliability). Consolidation approaches are explored to propose some energy reduction while switching OFF unused computing nodes. Virtualization is a key feature of the Clouds, since it allows high performance, improved manageability, and fault tolerance. It is promoted by several researches to decrease the energy consumption of large-scal distributed system.

One of the benefits of virtualization is a server and application consolidation. The benefits include savings on hardware and software, environmental costs, management, and administration of the server infrastructure. In this thesis we will use virtual machines which can be used to consolidate the workloads of under-utilized servers on to fewer machines, perhaps a single machine.

2.2 Cooperation and coordination in Distributed Systems

In recent years there has been increased interest in decentralized approaches to solve complex real-world problems such as problems of scalability, coordination of data center with different local constraints. Many such approaches fall into the area of distributed systems, where a number of entities work together with a common interest of solving problems.

Coordination mechanisms are introduced by Christodoulou, Koutsoupias, and Nanavati [Christodoulou 2004]. A coordination mechanism is a local policy that assigns a cost to each strategy s, where the cost of s is a function of the users who have chosen s. Consider, for example, a selfish scheduling algorithm in which

there are n jobs owned by independent users, m servers, and a processing time t_{ij} for job i on servers j. Each user selects a server on which to schedule its job with the objective of minimizing its own completion time. The social objective is to minimize the maximum completion time. A coordination mechanism for this algorithm is a local policy used by a server leader which determines how to schedule jobs assigned to this server.The role of the server leader makes members group work together for a goal or effect to fulfill desired, here minimizing time. It is important to emphasize that a server's policy is a function only of the jobs assigned to that server. This allows the policy to be implemented in a completely distributed fashion. Coordination mechanisms are closely related to local search algorithms. A local search algorithm iteratively selects a solution "close" to the current solution which improves the global objective. It selects the new solution from among those within some search neighborhood of the current solution. Centralized algorithms have been designed for this problem in [Bejerano 2004]. We hope to use ideas from centralized approach to design decentralized algorithms for this problem. It would be interesting to study coordination mechanisms in this context given that users may exhibit selfish behavior.

This section briefly introduces cooperation theory, which is a paradigm that utilizes distributed systems. Then, cooperative and coordination mechanisms are defined. Finally, related works on cooperative mechanism for pure P2P are presented.

2.2.1 Cooperation theory

The conditions under which cooperation can emerge in systems with self-interested participants, when there is no centralized control, and no trusted third-parties, have been studied in cooperation theory (c.f. [Axelrod 1984]).

The repeated prisoner's dilemma (PD) provides a useful context as a pivot case to understand how cooperation can emerge. In the PD, players can adopt either defect or fully cooperate : defect (D) or cooperate (C). Cooperation results in a benefit b to the opposing player, but incurs a cost c to the cooperator (where $b > c > 0$); defection has no costs or benefits. If the opponent plays C, a player

gets the reward $R = b - c$ if it also plays C, but it can do even better and get a temptation $T = b$ if it plays D. On the other hand, if the opponent plays D, a player gets the lowest payoff $S = -c$ if it plays C, and it gets a punishment $P = 0$ if it also defects. In either case, i.e. independent of whether the opponent plays C or D, it is, therefore, better to play D. In evolutionary settings, payoffs determine reproductive fitness, and it follows that D is the evolutionarily stable strategy [Smith 1993]. Another formulation [Taylor 1978] [Hofbauer 1998], admits pure defection as the only stable equilibrium. The Iterated Prisoner's Dilemma consists of two or more agents supposedly accused of a robbery; agents have to choose between confessing to the crime or refuse participation in it. The settings are such that it is rational for individual agents to deny, but it is in their collective interest for all to confess. Given these payoffs, users are tempted to defect because the payoff for defection is larger then the payoff for cooperation, whether or not the other user cooperates or defects. The dilemma arise because if both users defect they each get the lowest payoff, when they could each earn a better payoff if they had cooperated. Many systems are based on the PD. This is a good example cited in the problems of cooperation. The prisoner's dilemma provides a general context for thinking about situations where two or more nodes have an interest in cooperating for the migration of their loads but even stronger interest not to do so if the other does, and no way to coerce the other.

2.2.2 Cooperative Mechanism for Multi-agent system

In multi-agent problem solving, motivation for benevolence among agents is having a common goal. Various approaches have been developed for effective resource allocation to multiple agents which need to interact, and they need to behave cooperatively rather than greedily to accomplish a common objective.

The main question is, in general, how to better establish cooperation. It is important to know if agents should be implemented as a diverse set of autonomous actors and in this case what kind of communication is necessary for agents to cooperate effectively in the task. An interesting question is whether or not agents should be coordinated by a central controller.

In [Yong 2001], authors explore these questions in the context of machine learning, where a team of neural networks is evolved using genetic algorithms to solve a cooperative task. The Enforced Subpopulations method of neuroevolution (ESP) [Gomez 1997][Gomez 1999]), which has proven highly efficient in single-agent reinforcement learning tasks, is first extended to multi-agent evolution. The method is then evaluated in a pursuit-and-evasion task where a team of several predators must cooperate to capture a fast-moving prey. Their contribution was to show how the different coordinations of a team of agents affect performance. Results show that a set of autonomous neural networks, but driven by other behavior, each evolving cooperatively to control a single predator performs well.

2.2.3 Cooperative Mechanism for Pure P2P

Cooperative distributed systems, commonly called "systems peer-to-peer" are systems where constituent entities pool local resources to build a global service. These systems are often in opposition with the client-server systems, however, the distinction between the roles of client and server is blurred, and participants are generally two consummers service (eg, clients) and producers (eg server). The emergence of this class of systems is often associated with the decreasing cost of computing and communications resources during the 1990s.

Cooperative distributed systems offer many services. The most commonly used are content distribution, file sharing, distributed data processing, the execution of tasks etc...

These systems enable massive resource and information pooling at low cost per participant and at scales that are difficult to achieve with traditional client-server systems, while local autonomy and network effects provide resilience against faillures and atttacks.

The main distinction between these systems and the traditional client-server model is widespread cooperation between participants in terms of sharing the resources of a inforlation. The main advantage of this system is the cooperation that makes them very vulnerable to non-cooperative behavior on a large scale. It is therefore necessary for the system to be designed such that the participants generally

cooperate.

Peer-to-peer systems provide a powerful infrastructure for distributed computing applications on a large scale due to the pooling of cooperative resources of participants. Cooperation and enough resources are key to enabling a variety of new applications such as file sharing, content distribution on a large scale and distributed data processing. Performance in these systems are related to the level of cooperation between participants in the system.

2.2.3.1 Peer-to-Peer Architectures

The peer-to-peer architectures can be classified according to their different characteristics. We argue that peer-to-peer architectures can be categorize in terms of their structure: Unstructured and structured. Their architecture is based on the extent of (de)centralization. Based on this we distinguish the following combinations: centralized unstructured, pure unstructured, hybrid unstructured and pure structured systems.

- Degree of Decentralization

  - **Centralized peer-to-peer architectures**, This architecture contains a central server running the vital functions of the system. The central server is most often used as a directory server or as a central directory server that stores an overview of the nodes and the available network resources. It allows peers or nodes to find, locate and share resources with other peers. A major drawback of these systems is the risk of bottlenecks. The advantage of using central directory servers is the availability of data retrieved;

  - **Pure decentralized architectures** Nodes perform functions without the intervention of centralized components. These types of architectures are theoretically scalable and a high level of fault tolerance. Examples of pure decentralized peer-to-peer networks are Gnutella 0.4 [Ripeanu 2001], Freenet [Clarke 2002] and Chord [Stoica 2001].

  - **Hybrid systems** are often hierarchical networks that have characteristics both centralized and decentralized pure architectures . Therefore they combine the advantages (eg, the location of effective resources, scalability) while avoiding the disadvantages (eg, bottlenecks, limited QoS) of these systems. KaZaA [Shin 2006] and Gnutella 0.6 [Ripeanu 2001] are some examples of such architectures. Further, we argue that BitTorrent [Izal 2004] can also be regarded as a hybrid system.

- **Degree of Structure.** Whether a system is structured depends on how nodes and data are positioned in the network.

  - Unstructured. A system is unstructured when the layout of nodes and data require no rules and nodes are of an ad hoc manner in the network. A main feature is a high consumption of bandwidth in the matter of traffic of messages.
  - Structured. In this type of networks, nodes and data are being placed in a structured way in the network as to be able to efficiently locate data which increases the possible scalability. Distributed routing tables make possible to efficiently the connection of nodes, data or other resources to specific location. The structured systems include: Chord [Stoica 2001], CAN [Ratnasamy 2001] and Tapestry [Rhea 2001] Freenet [Clarke 2002].

2.2.3.2 Cooperative Mechanism

The mechanism for pure P2P networks of file-sharing applications gives peers which interact and cooperate with each other in an efficient and effective way. There are several publications on cooperative scheduling algorithms to enhance collective performance and efficiently utilize network resources [Stoica 2002]. In a P2P network, a peer can play along with the role of client and server. At its creation, a member of P2P network has their own behavior and acts freely as an individual is in a group or society. Thus, a peer sends requests in the network to find the desired file. A response message is sent by other peers in the network to respond to the request. Because the query can reach distant peers, relay message to the request

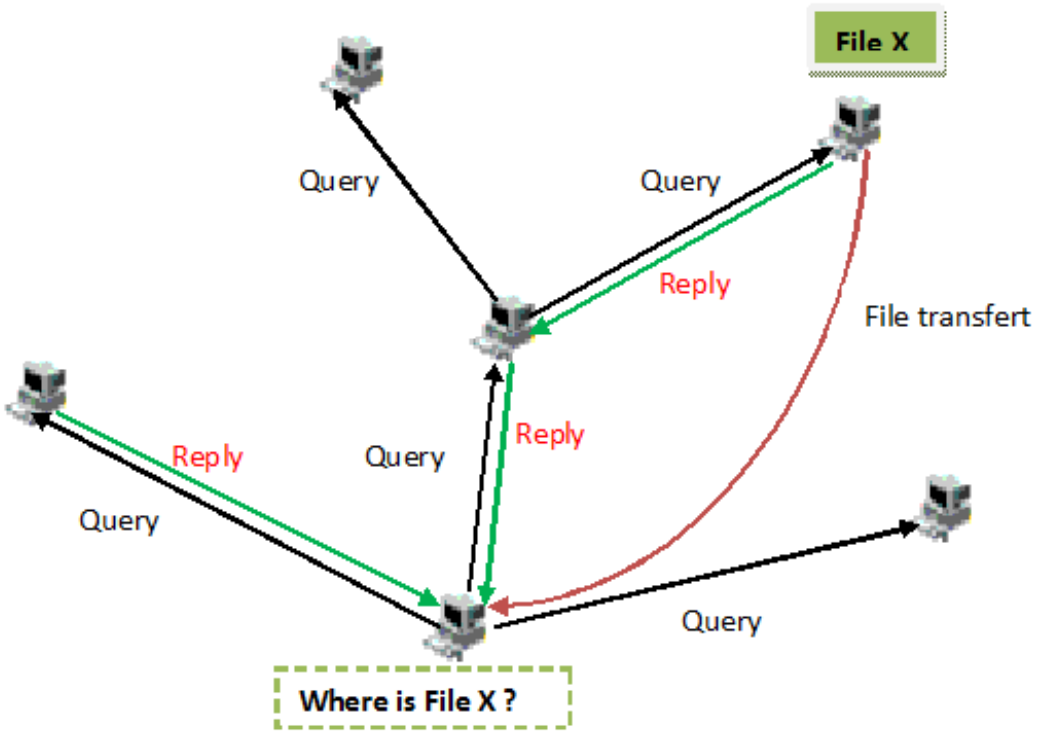

Figure 2.3: Flooding in a pure P2P file-sharing network

can be sent to their neighbors peers (fig. 2.3). Therefore it is a system that utilizes the flooding techniques. A pair relays a request message to all neighbor peers. It is a powerful system to find a file in a P2P network However, it has been pointed out that flooding techniques lack scalability because the number of query messages that traverses a network significantly increase with the growth in the number of peers. In pure P2P networks logical links can be established between two or more cooperative peers each time in cooperative system. With this cooperation, requests are distributed more efficiently and even finds a file more efficiently. Cooperation may prevent some even having selfish behavior in trying to strengthen and improve their QoS.

2.2.3.3 An example of cooperative pure P2P file-sharing networks [Konishi 2006]

Cooperation pure P2P networks in the case of file sharing starts by establishing logical peer cooperation links. These peers are selected among peers candidates within each P2P network. These peer candidates are those who are willing to play the role of cooperation to strengthen and improve their QoS. Consequently reply messages are transmitted via the logical link between peer cooperation (Fig.2.4).

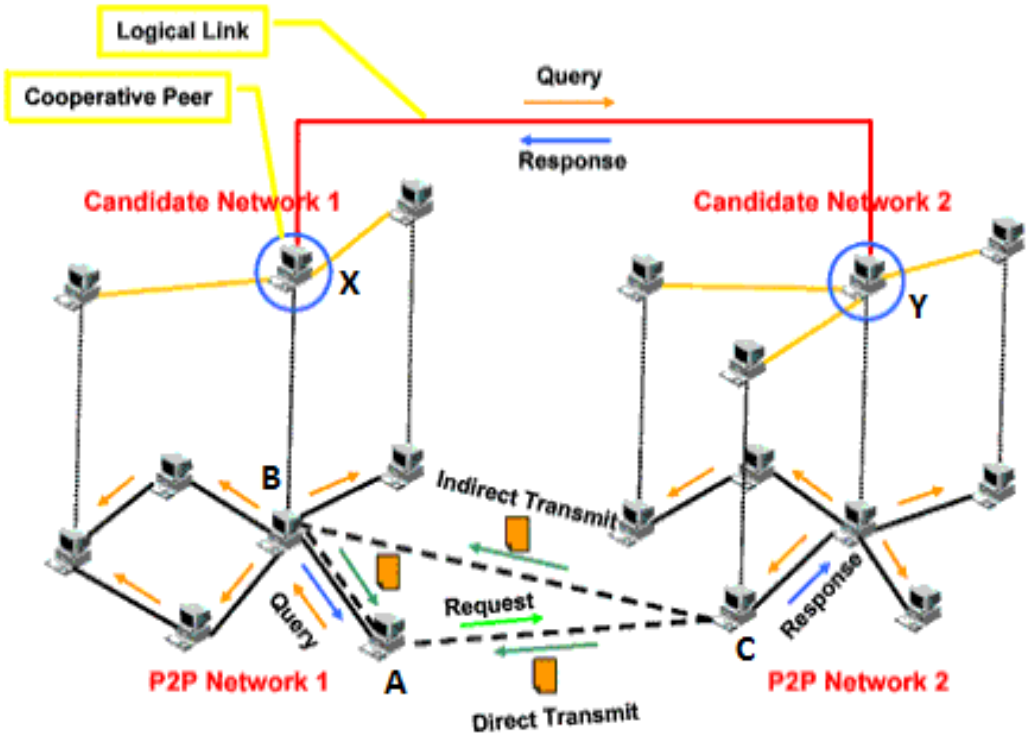

Figure 2.4: Cooperation of pure P2P file-sharing networks [Konishi 2006]

Several steps are needed to set up cooperation mechanism . The selection of peer cooperation , the discovery of other P2P networks, the decision to start cooperation , the relay of messages and file transfer , and the decision to terminate cooperation in detail are diferent steps of the process. Among these steps, we retain three :

- **Establishing a Candidate Network.** A pair is a candidate for the execution of a program of cooperation. In the case where a peer has intending to enhance and improve its application-level QoS, it runs the cooperation

program independently of others. The peer does not care whether the other peers in the same P2P network will benefit from the cooperation or not. Then, it becomes a candidate peer, i.e., a candidate for cooperative peers. A new candidate peer finds another candidate peer in the same P2P network by flooding a special message over the P2P network. On the other hand, candidate peers in a candidate network send packet messages containing a service identifier and its address to the network periodically. A new candidate peer receives one of their packet messages and establishes a logical link to the candidate peer. An example of cooperation is illustrated in Fig.2.4.

In our decentralized approch we hope that a node (request) can send a flooding message. Some nodes (response) send a reponse message. The request node choose a node (candidate) that meet requirements. The example that has just been explained may be appropriate for our decentralized approach that will be presented in Chapter 3.

- **Selecting Cooperative Peers.** Only peers that receive requests for cooperation can become cooperative peers. Peer candidates are promoted cooperatives pairs and can then launch another cooperative application .

  To effectively disseminate messages, cooperative peers must be well chosen. Logical links are established between peers candidates and building a network of candidates to exchange information for the selection of peer cooperation. A peer is selected among the candidates peers , then confirms it is appropriate as a peer or not to cooperate. When a cooperative peer receives a message that does not contain the identifier of the network, it sends a request to the candidat peers that are in a different network. And the selection is initiated in this new network.

- **Finding other P2P networks** A new cooperative peer X first finds another candidate peer Y by flooding a special message over the P2P network. When the new cooperative peer X in another P2P networks receives this packet message that contains a different network address of hers, it sends a cooperation request to the candidate peer Y, i.e., the sender of the packet message, in

another P2P network. Next, the selection of a cooperative peer is initiated by the candidate peer in a newly found P2P network. Then, the cooperation request is forwarded from the candidate peer to a new cooperative peer. Finally, a logical link is established between those cooperative peers.

- **Decision of Starting Cooperation.** Finally , after confirmation, a pair provisional cooperative is selected . The start of the cooperation is related to the compatibility of the cooperative peer and their ability to find other peers in a network. A message is initiated to the network in all cooperative peers. A cooperative peer must get all the information about the type and characteristics of the latter. After it sets priorities to each of them. When the two cooperative peers both fulfill requirements then, collaboration is started.

2.2.4 Entities cooperation and coordination in the Grid/Cloud

A distributed sytem is composed of autonomous entities able to perform functionalities but also share their capabilities. Their main challenge is managing the interdependencies between entities that are no centralized systems. The efficiency of this management depends on the coordination and cooperation between the different entities of the system. Lack of coordination or cooperation results in communication overhead and eventually reduces performance of the system. The coordination process (cooperation respectively) with respect to task scheduling and resource management in distributed system requires the dynamic exchange of information between the different entities of the system.

For Techopedia [Janssen 2013] a cooperative storage cloud is based on peer-to-peer architecture . The role of cloud storage service providers is to make available this service. It builds specialized P2P software to manage the storage and retrieval of data. In the Symform Cooperative Storage Cloud [symform 2013]each local computer contributes to cheap storage and exchange of valuable cloud-based storage.

Problems related to interdependencies can be solved by coordination mechanism. A market based mechanism considers the cloud as a virtual market in which economic entities interact with each other through the purchase and sale of computing,

storage resources and services. Such a coordination mechanism is used to facilitate efficient resource allocation. In such mechanism, the resource provider works as a manager that exports its local resources to contractors, and resource brokers are responsible for decision regarding admission control based on negotiated Service Level Agreements (SLA).

Solving problems related to the interdependence uses the model of decision making which is the coordination structure. The interaction among entities is coordinated by the utilization of a simple broadcast communication but it is using a lot of messages so is very expensive : One-to-many communication. Anther type, One-to-one communication can drastically reduce overhead by adopting among the resource providers and consumers through establishment of a Service Level Agreement.

2.3 Cloud Computing System

The visions of both the Cloud and the Grid are the same. It is mostly environment created to reduce the cost of computing and especially increase storage capacity and computing [Foster 2008]. The term Cloud can be seen as another marketing term hype of the Grid computing due to the fact they have similar goals. A similar view is given by many experts defined in [Geelan 2009]. This section briefly introduces cloud computing, which is a computational paradigm that utilizes networked computing systems in which applications plug into a "power Grid" of computation for execution. First, the basic principles behind cloud computing are defined. Afterwards, the cloud characteristics, service models, and deployment models are presented. Finally, existing attempts to design and implement IaaS cloud management systems are reviewed.

2.3.1 What is Cloud Computing?

Cloud Computing is a new pattern of information service provision which is raised by Google and IBM, and advocated by other IT giants. The concept means providing users who submit requests with software and information services on demand

from computing resources that are heterogeneous and autonomous, which are regarded as a whole from the external. The existence of cloud computing is based on the development and use of Internet computer technology. It is an abstraction of the various infrastructure that contains the internet. Cloud computing is very essential, which allows on demand access to remote resources. Cloud computing delivers infrastructure, platform, and software (applications) as services that are made available to consumers in a pay-as-you-go model.

In industry these services are referred to as Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) respectively. Figure 2.5 shows the architecture of the cloud computing. Cloud computing system scales applications by effective use of resources and by optimizing concurrency. The cloud computing enabled IT companies to reduce their spending on equipment and software. This especially thanks to innovative ideas for new application services. Cloud computing has allowed IT company to release the maintenance and purchase of equipment and software. Thus they can focus more on management and creation of business values for their application services [Goldberg 1989]. Some of Cloud based application services include storage, web hosting, content delivery, and real time instrumented data processing. Each of these application types has different characteristic, mainly configuration, and deployment requirements. Quantifying the performance of provisioning (scheduling and allocation) policies in a real Cloud computing environment (Amazon EC2 [Armbrust 2010], Microsoft Azure [MicrosoftAzure 2013], Google App Engine [GoogleAppEngine 2013]) for different application models.

The description of the applications in the cloud computing enables those are extended to be accessible through the Internet.These cloud applications are running or are stored in large data centers and powerful servers that host Web applications and Web services.A cloud application can be accessed via internet with a suitable Internet connection and a standard browser. Three elements compose the traditional computational model : computational power (processors and memory), storage, and software (services). The overall goal of Cloud computing is to allow applications to utilize these elements. The cloud can be classified based on their

use. The elements of a cloud are distributed among different nodes. We can classify the Cloud computing systems as:

- Computation: dicates a system that has a large amount of distributed processors.
- Data: therefore it may be data warehouses
- Service: refers to systems that provide services which can be an aggregate of multiple services

The primary goal of cloud computing is to reduce the costs of infrastructures management [Zhang 2010]. Many cloud computing definition have been proposed over the past years [Vaquero 2008]. However, as of today still no standard definition exists. In this work we rely on the definition presented in [Mell 2011], where Peter Mell and Tim Grance define cloud computing as: a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources that can be configurable and rapidly provisioned. This model must be released easily without the intervention of a provideur with little effort.

A comprehensive definition is provided by the National Institute of Standards and Technology (NIST) [Mell 2011], which we will follow throughout this thesis. Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable com-puting resources (e.g., networks, servers, stor-age, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider inter-action.

2.3.2 Characteristics

Cloud Computing model is distinguished by the following five essential characteristics.

- On-demand self-service. A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.

- Broad network access. Resources are available at any time and anywhere on the network via standard mechanisms with light and heterogeneous platforms (eg, mobile phones, laptops and PDAs).

- Resource pooling. Regrouping of IT resources of providers to serve multiple clients. Customers may purchase multiple IT resources that can be physical or virtual;

- Rapid elasticity. Capabilities can be rapidly and elastically provisioned, in some cases automatically;

- Measured Service. resource use is automatic and mostly it is optimized. Measurement or monitoring of the provision of services is done by the cloud provider.

2.3.3 Cloud Computing Services

Cloud computing models can be mapped with layers of business value pyramid as shown in below diagram:

- **Cloud Software as a Service (SaaS)**. In SaaS, an application is hosted by a service provider and then accessed via the World Wide Web by a client. An easy way to think of SaaS is the web-based email service offered by such companies as Microsoft (Hotmail), Google (Gmail), and Yahoo!(Yahoo Mail).Each messaging service has basic criteria: the provider (Microsoft, Yahoo, etc.) hosts all the programs and data in a central location.

  - Line of business services. These business solutions available for businesses. To purchase must go through a subscription service.
  - Customer-oriented services In category we can have webmail services, online gaming and consumer banking, among others.

  To put them in a SaaS system, the technique used is virtualization. With the the growth of virtualization it is easier for independent software vendors (ISVs) to adopt SaaS. The growing popularity of some SaaS vendors using Amazon's

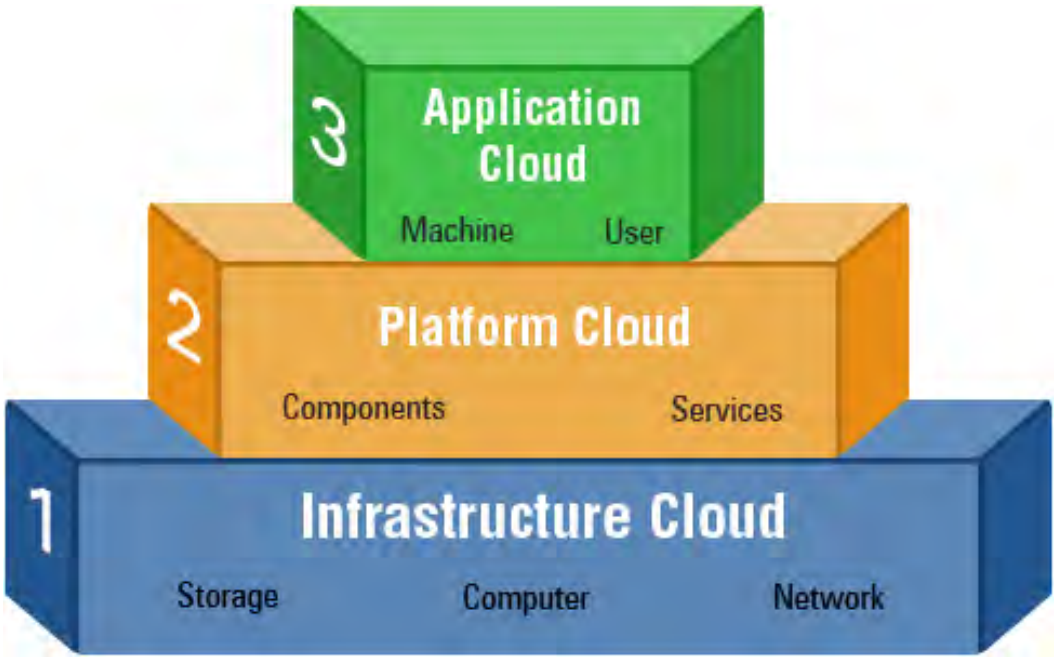

Figure 2.5: Cloud computing : layers of business value pyramid

EC2 cloud platform and the overall popularity of virtualized platforms help with the development of SaaS.

- **Cloud Platform as a Service (PaaS)**. The consumer has the ability to deploy applications created , its use programming languages. It does not need to manage or control the cloud infrastructure but has control over the deployed applications and possibly configure the application hosting environment.

- **Cloud Infrastructure as a Service (IaaS)**. Also known ass Everything as a Service. In this scenario, the client is using the cloud provider's machines, but mostly a virtualized server executes its software. In this domain the most widespread is Amazon Elastic Compute Cloud (EC2).

- **Software plus Services**. This is the strengthening of the typical SaaS. This service is used by Microsoft. The on-premise software requires you to reach out to the cloud for additional services.

2.3.4 Deployment Models

- **Private cloud**. In this case the cloud infrastructure is dedicated to an organization who can run or by a third party and may exist on premise or off premise.
- **Community cloud**. The cloud infrastructure is shared by several organizations and supports a specific community that has shared concerns
- **Public cloud**. The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.
- **Hybrid cloud**. The cloud infrastructure is a composition of two or more clouds (private, community, or public)

2.4 Scheduling in the Grid/Cloud

Scheduling is a decision-making process that is used on regular basis in many manufacturing and services industries. It deals with the allocation of resources to tasks over given time periods and its goal is to optimize one more objectives. The resources may be processing units in a computing environment. The tasks may be executions of computer programs. Each task may have a certain priority level, an earliest possible starting time and due date. The goal of a scheduler is to find ways to appropriately assign tasks to limited resources that optimize one or more objectives. The objectives can also take many different forms. One objective may be the minimization of the execution time of tasks and another may be minnimizaation fo the energy consumption.

These are solutions to planning center given that the batch starts depending on available resources. A set of rules is defined with route optimization to reduce travel time and mileage between jobs.

Start times of the day offers many advantages , and a process must take into account all the dynamics of the business day to day . In environments with modern facilities , it is important to consider the performance parameters such as the

number of calls engineers engineers can émetttre day. As customers become increasingly demanding in imposing stricter sanctions cumulative ALS organizations successful services closely scrutinize their productivity and operational costs. They use advanced analytics and business models of historical data and can use a more strategic approach to their activities and ensure they can meet the demand for efficient service , the allocation of revenue targets and profitability to drive the process.

Application submitted in the cloud are divided into several tasks. When the parallel traitement is adopted in the execution of these tasks, we must consider the following questions: 1) how to allocate resources to tasks, 2) in what order the cloud should perform the tasks, and 3) how to plan overhead when virtual machines prepare, terminate or change jobs. Resource allocation and task scheduling can solve these three problems. Resource allocation and task scheduling [Cantú-Paz 1998] [Jiang 2007][Sharma 2010] [Shenai 2012] [Huang 2013a]. Huang et al. in [Huang 2013a], proposes a aurvey on resource allocation policy and job scheduling algorithms of cloud computing. However, the autonomic feature within clouds [Rahman 2011][Guerout 2013][Serban 2013] and the execution of VM require different algorithms for resource allocations and task scheduling in the IaaS cloud computing. A major objective of this thesis is the implementation of scheduling algorithms for reducing energy consumed in data centers.

This section introduces scheduling in the Grid/Cloud. First, the basic principles behind scheduling are defined and give the objective. Afterwards, scheduling phases market mechanism properties and scheduling problem are presented. Finally, scheduling approaches in the Grid/Cloud are reviewed.

2.4.1 The Grid/Cloud Scheduling Phases

In this subsection, we introduce the Grid/Cloud scheduling logical architecture as shown in Figure 2.6.

Planning grid has three main phases : (i)resource discovery, which generates a set of potential resources collecting information on resources, (ii) the selection of a better set, and (iii) perform the work, which includes file used and cleaning.

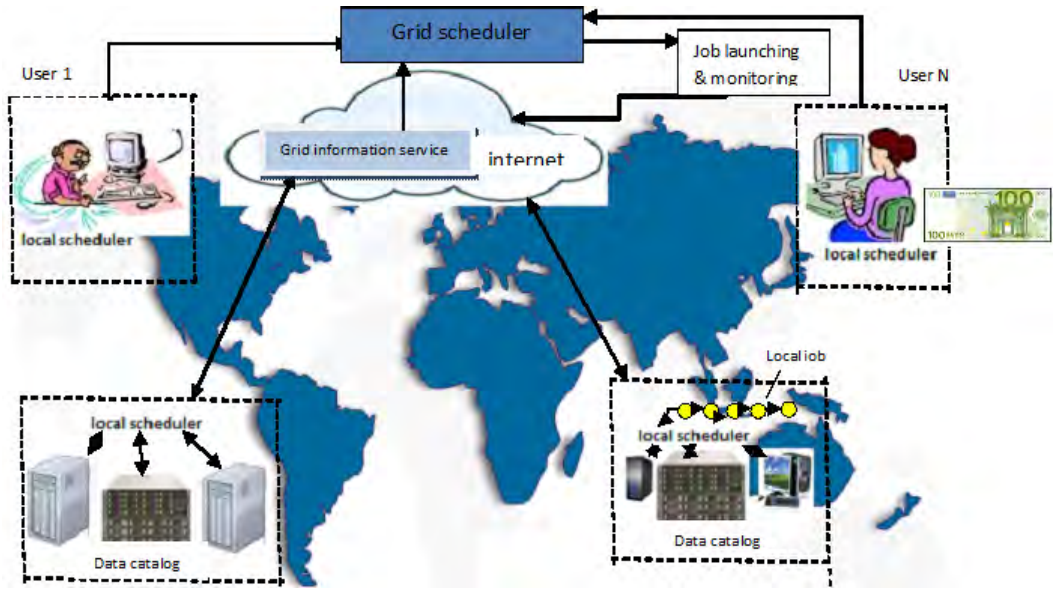

Figure 2.6: Grid/Cloud scheduling logical architecture

These phases, and the steps that make them up, are shown in Figure 2.7. We note that there is no planner of the current grid that implements all the stages of this architecture.

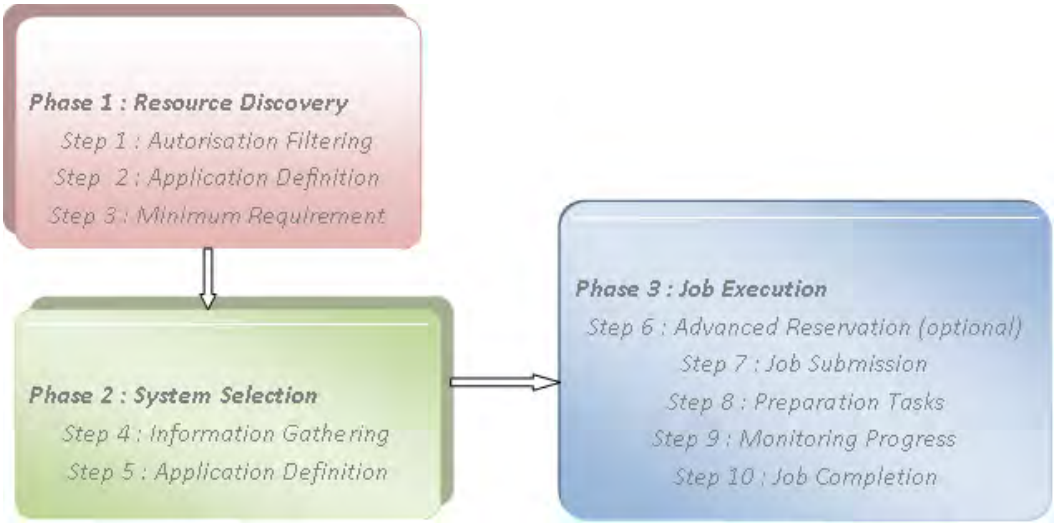

Figure 2.7: Three-Phase Architecture for Grid Scheduling

2.4.1.1 Phase 1 : Resource Discovery

To a given user, the first stage of any scheduling interaction is the discovery of the available resources. During this phase a list of resources is selected . The Phase 1 begins by an empty list of resources; at the end of this phase, the potential resources selected is the list that has the minimum feasibility requirements. This phase is composed of three steps: authorization filtering, job requirement definition, and filtering to meet the minimal task requirements.

- Step 1: Authorization Filtering. To properly select all possible adequate resources , the user must be able to specify the requirements of the smallest task. The set of possible task requirements can include static details such as the operating system or hardware, or the specific architecture as well as dynamic details such as a minimum RAM requirement, connectivity, or space. The first step of resource discovery in job scheduling is to determine the set of resources that exist. At the end of this step the user will have a list of machines or resources to which he or she has access.

- Step 2: Application Requirement Definition. To make a good selection of a set of feasible resources, the user is to be able to specify the minimum task requirements. The set of possible job requirements can be be very wide and varies considerably between task. It may include static details (the operating system or hardware for which a binary of the code is available, or the specific architecture for which the code is best suited) as well as dynamic details (for example, a minimum RAM requirement, connectivity needed, storage space needed). .

- Step 3: Minimal Requirement Filtering. Given a set of resources to which a user has access and at least a small set of job requirements, the third step in the resource discovery phase is to filter out the resources that do not meet the minimum task requirements. At the end of this step, the user acting as a Grid/Cloud scheduler will have a reduced set of resources to explore.

  Current Grid Information Services are set up to contain both static and dy-

namic data. Many of them cached data with a lifetime associated that allows faster response time of long-term data , including information on basic resources such as operating system software available and hardware configuration. Since resources are distributed and getting data to make scheduling decisions can be slow, this step uses basic, mostly static, data to evaluate whether a resource meets some basic requirements. This is similar to the discovery stage in a monitoring and discovery service.

2.4.1.2 Phase 2 : System Selection

All possible resources meeting the minimum requirements of the task, those on which to schedule the task were selected. This selection is usually done in two stages: the acquisition of knowledge and making a decision. these two steps will be studied separately, but they are intrinsically linked, the decision process depending on the information available.

- Step 4: Dynamic Information Gathering. In order to make the best task scheduling in Grid/Cloud computing environment, detailed dynamic information about the available resources is needed. Since this information may vary with respect to the application being scheduled and the resources being examined, no single solution will work in all, or even most, settings. The dynamic information gathering step has two components: what information is available and how the user can get access to it.

- Step 5: System Selection. Detailed information gathered in step 4 to enable this step to determine which resource (or a set of resources) to use. Maui [Scheduler 2013] is an open-source job scheduler for use on clusters and supercomputers. It submits a full job description to each of the local schedulers, each of which individually returns feasible time ranges, including estimated execution times, cost, and resources used. It improves the manageability and efficiency of machines ranging from clusters of a few processors. Adaptive's Moab HPC Suite [Moab 2012] is a descendant of the Maui scheduler. The Moab is an intelligent cloud management end-to-end platform.

2.4.1.3 Phase 3: Job Execution

The third phase of Grid scheduling concerns running a job. Thus a number of measures must be taken, some of which have been defined uniformly between resources.

- Step 6: Advance Reservation (optional). To best use a given system, some or all of the resources can be reserved in advance . Advance booking can be easy or hard to do and can be done with mechanisms or human resources. Moreover , reservations may or may not expire with or without costs . A question having advance bookings are most common is the need for lower-level resource to support services based on indigenous resources .

- Step 7: Job Submission. Once resources are chosen, the application can be submitted to the resources.

- Step 8 : Preparation Tasks. With this preparation stage which may contain setup, claiming a reservation, or other actions it is necessary to prepare the selected resource to run the application.

- Step 9: Monitoring Progress. Depending on the service and its running time, users may monitor the progress of their services.

- Step 10: Job Completion. The user must be notified at the end of the task. In many cases, the submission scripts for parallel machines include a parameter notification e-mail. Due to the fault tolerance, however, this notification can be surprisingly difficult. Furthermore, with so many interacting systems, one can easily imagine situations in which a state of completion can not be reached. Much more work is needed.

- Step 11: Cleanup Tasks. The user may need to retrieve the results after a job is done from this resource. They can then make the data analysis results, delete the temporary settings, and so on. One of the current systems that staging (step 8) also manages the cleaning. Condor also supports a variation of this stage, in which it runs an extra process before and after the application is run to do staging and clean-up.

2.4.2 Integration of the Grid Computing environment and a job based scheduling scheme

A job scheduling scheme which is mostly utilized in a Grid Computing environment is summarized in figure 2.8. This scheme integrates all of the three phases of Grid scheduling we have just seen. Requests for resource allocations are made between the Application Layer and the Collective Layer. The Application layer contain normally a task manager which is given the responsibility of finding the available resources. Requests are sent by the task manager to manager calendar which is present in the collective layer just to process these applications . The program manager then looks in a central directory to check the available resources. Which corresponds to the phase 1 view previously (Resource Discovery). Between collective layer and the resource manager program executes various scheduling algorithms based and identifies best available resources that yield the most effective results. This part of the scheme may match the phase 2 (System Selection). The rest of the scheme corresponds to the phase 3. Between the resource layer and the connecting layer there is often a prediction mechanism which measures the performance of the system, and based on the current output of the collected results . These results are very important. They are often used to estimate the rate of tasks completion. Various sorts of data managers and tools are used to measure the performance of connectivity between the layer and the fabric layer . These tools measuring system and application information use sensors of various kind. Thus, the presence of various tools or managers present at different levels of treatment (so that once the problem is entered into the system , the problem would logically pass through a series of transactions) imply queues to reach a final result. Without a queue based system, there is always a possibility that the available resources are not used efficiently.

2.4.3 Scheduling Approaches in the Grid/Cloud

Task scheduling techniques in data centers are studied in great detail with the aim of making use of these systems efficiently and especially in order to reduce their

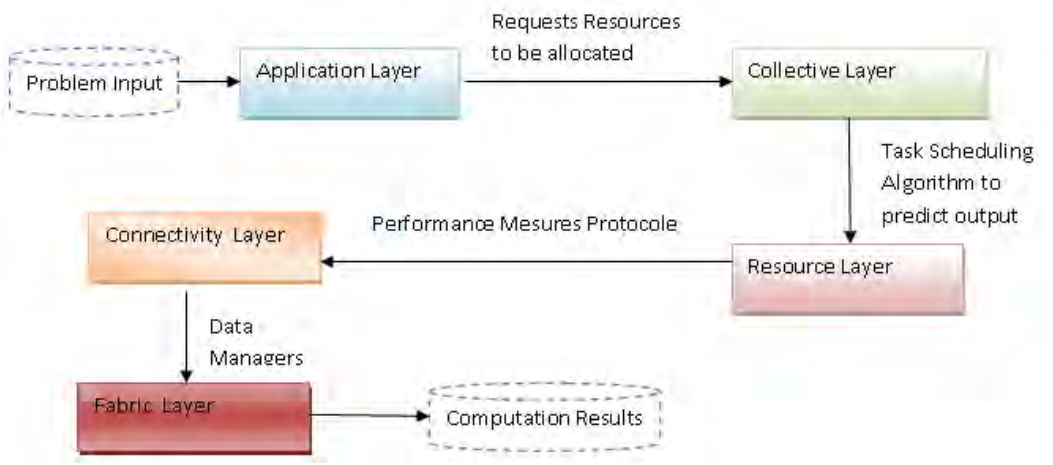

Figure 2.8: Integration Of The Grid Computing Environment And A Job Based Scheduling Scheme

energy consumption. Task scheduling algorithms are typically classified into two subcategories: static scheduling algorithms and dynamic scheduling algorithms. In static task scheduling algorithms, the task assignment to resources is determined before applications are executed. Information about task execution cost and communication time is supposed to be known at compilation time. In [Wang 2010] the task graph clustering technique is defined as an effective static scheduling heuristic used for scheduling parallel tasks. In a task graph, the process of mapping task graph nodes onto marked clusters is called clustering. One processor executes all tasks of the same cluster. In traditional task scheduling heuristics, the process of clustering tasks is an optimization of reducing the makespan of the scheduled graph.

A task is always running on the resource to which it is assigned. Dynamic task scheduling algorithms normally schedule tasks to resources in the runtime to achieving load balance among processor elements (PEs).They are based on the redistribution. List scheduling algorithm is the most popular algorithm in the static scheduling [Mtibaa 2007]. List based scheduling algorithms assign priorities to tasks and sort tasks into a list ordered in decreasing priority. Then tasks are scheduled based on the priorities.

There is important to note that scheduling strategies deployed in clusters have

a large impact on overall system performance. This is due to the fact that many parallel applications run in clusters require intensive data processing and data communication. Basically, parallel scheduling strategies can be classified to three primary categories, called priority-based scheduling, cluster-based scheduling, and duplication-based scheduling, respectively. Priority-based scheduling involves the assignment of priorities to tasks and then maps those tasks to processors based upon assigned priorities [Sih 1993].

Duplication scheduling is more efficient than other scheduling algorithms in most cases, especially when communication time is greater than the execution time of parallel applications. With the duplication of many tasks and the execution of more than once by multiple processors, the performance is improve but energy consumption increases. Lai et al. [Lai 2008] proposed a duplication-based task scheduling algorithm for distributed heterogeneous computing (DHC) systems, which improve system utilization and avoid redundant resource consumption, resulting in better schedules. Authors proposed a heuristic strategy called the Dominant Predecessor Duplication (DPD) scheduling algorithm, which allows for system heterogeneities and communication bandwidth to exploit the potential of parallel processing. Zong et al. [Zong 2011] proposed two energy-efficient duplication-based scheduling algorithms Energy-Aware Duplication (EAD) scheduling and Performance-Energy Balanced Duplication (PEBD) scheduling. These algorithms strive to balance schedule lengths and energy savings by judiciously replicating predecessors of a task if the duplication can aid in performance without degrading energy efficiency.

In real-time events, dynamic scheduling is of great importance for the successful implementation of real-world scheduling systems. In dynamic scheduling job release times are not fixed at a single point in time, i.e. jobs arrive to the system at different times. In dynamic scheduling problems, one considers scheduling one customer arrival only, assuming that there are a number of scheduled customers already. Real world scheduling problems are usually of dynamic nature. From the point of view of combinatorial optimisation the question of how to sequence and schedule jobs in a dynamic environment looks rather complex and is known to be NP-hard to almost every state. Chryssolouris et al. [Chryssolouris 2001] proposed a

dynamic scheduling of manufacturing job shops using genetic algorithms. Dynamic scheduling algorithm have higher performance than static algorithm but has a lot of overhead compare to it.

2.4.3.1 Allocation process of a scheduler

The allocation process of a scheduler consists of two parts, the selection of the machine and the scheduling over time.

- Selection-Strategies.

  There are various scheduling techniques. The Greedy heuristic is literally a combination of the Min-min and Max-min heuristics. Results show that it outperforms both and uses the better solution [Armstrong 1998] [Freund 1998]. Min-Min and Max-Min are discussed first.

  - **Greedy Algorithm.** Greedy Algorithm [Hazewinkel 2001] works by selecting at every step the best choice available at any moment; it never reconsiders this decision, whatever situation may arise later. Greedy algorithms are used to solve optimization problems. Many heuristics are greedy algorithms and practical experience shows that they usually perform quite well [Perea 2011].
  - **Min-Min Algorithm.** Min-Min begins with a set of unassigned tasks. First, it computes minimum completion time for all tasks on all resources. Then among these minimum times the minimum value is selected which is the minimum time among all the tasks on any resources. Then that task is scheduled on the resource on which it takes the minimum time and the available time of that resource is updated for all the other tasks. It is updated in this manner; suppose a task T is assigned to a machine and it takes 20 seconds on the assigned machine (AM), then the execution times of all the other tasks on AM will be increased by 20 seconds. After this the assigned task (AT) is not considered and the same process is repeated until all the tasks are assigned resources. This approach was proposed by Maheswaran et al. [Maheswaran 1999].

- **Max-Min Algorithm.** In Max-Min, after having discovered the completion time, the minimum execution times are discovered for each and every task. Then among the set of minimum times the maximum value is selected which is the maximum time among all the tasks on any resources. Then that task is scheduled on the resource on which it takes the minimum time and the available time of that resource is updated for all the other tasks. The updating is done in the same manner as for the Min-Min. All the tasks are assigned resources by this procedure. The approach is proposed in [Maheswaran 1999] and [Mandal 2005].

- Scheduling Algorithms.

  Most common algorithms in scheduling are based on list-scheduling. In the following some variants are presented :

  - First-Come-First-Serve (FCFS)[Denning 1965] [Schwiegelshohn 1998] : The scheduler starts the jobs in the order of their submission. If not enough resources are currently available, the scheduler waits until the job can be started. The other jobs in the submission queue are stalled. This strategy is known to be inefficient for many workloads as wide jobs waiting for execution can result in unnecessary idle time of some resources;

  - Random [Seiden 2000] : The next job to be scheduled is randomly selected among all jobs that are submitted therefore not yet started then are in wait state, therefore the schedule is nondeterministic. No job is preferred, but jobs submitted earlier have a higher probability to be started before a given time instant;

  - Backfill[Lifka 1995] : This is a version of FCFS scheduling whose its rule is to try to prevent the unnecessary idle time caused by wide jobs. Two common variants are EASY- and conservative-backfilling [Feitelson 1995]. In case that a wide job is waiting for execution other jobs can be started under the premise that the wide job is not delayed. Note, that the

performance of this algorithm relies on a sufficient backlog. Most commonly used is first come first serve (FCFS) combined with backfilling [Keller 2000][Skovira 1996][Mu'alem 2001], as on average a good performance for the utilization and response time is achieved. However, with certain job characteristics other scheduling policies might be superior to FCFS. For example, for mostly long running jobs, longest job first (LJF) is beneficial, whilst shortest job first (SJF) is used with mostly short jobs [Feitelson 1994].

– round robin scheduling [Demers 1989]. In the round robin scheduling, processes are dispatched in a FIFO manner but are given a limited amount of CPU time called a time-slice or a quantum. If a process does not complete before its CPU-time expires, the CPU is preempted and given to the next process waiting in a queue. The preempted process is then placed at the back of the ready list.

2.4.3.2 Local scheduling vs. Global scheduling

We can note a distinction between the two algorithms at the highest level. The local scheduling subjet determines how the processes resident on a single CPU is reallocated and executed; a global scheduling policy uses information about the system to allocate processes but also to multiple processors to optimize a system-wide performance objective. Obviously, Grid scheduling falls into the global scheduling branch which shows its popularity.

2.4.3.3 Static Scheduling Algorithmn vs Dynamic scheduling algorithm

- Static scheduling Algorithms. In the case of static scheduling, information regarding all resources in the Grid as well as all the tasks in an application is assumed to be available by the time the application is scheduled. Every task comprising the job is assigned once to a resource. Thus, the placement of an application is static. The following benefit were noted : estimate of the cost of the computation can be made in advance of the actual execu-

tion, information regarding all resources in the Grid as well as all the tasks in an application is assumed to be available by the time the application is scheduled, the assignment of tasks is fixed a priori, and estimating the cost of jobs is also simplified, the static model allows a global view of tasks and costs. A major benefit of the static model is that it is easier to program from a scheduler's point of view. The following disadvantages were noted. Cost estimate based on static information is not adaptive to situations such as one of the nodes selected to perform computation fails, becomes isolated from the system due to network failure. Heavily loaded with jobs that its response time becomes longer than expected. Unfortunately, these situations are quite possible and beyond the capability of a traditional scheduler running static scheduling policies. To address this problem, some important mechanisms such as rescheduling mechanism are introduced at the cost of overhead for task migration to reduce energy consumption. Another side-effect of introducing these measures is that the gap between static scheduling and dynamic scheduling becomes less important. In the decentralized approach presented in chapter 3 we introduced a stage of rescheduling.

- Dynamic Scheduling Algorithm (On line Scheduling Algorithms.) In the case of dynamic scheduling, the main idea is to perform task allocation dynamically. This is useful when execution time can not be found, as well as direction of branches and number of iterations in a loop as in the case where jobs arrive in a real-time mode [Yu 2008].

2.4.3.4 Approximate vs. Heuristic

Another class of algorithms is approximate algorithm . These algorithms use the technique used is model algebra. However, instead of searching the entire solution space for an optimal solution, it stops when it finds a solution that is qualified sufficiently "Good". In the case where a metric is available for evaluating a solution, this technique can be used to decrease the time taken to find an acceptable schedule.The factors that determine whether this approach can be continued include: the

availability of a function that can evaluate a solution, the time required to evaluate a solution, the ability to judge the value of an optimal solution according to some metric and the availability of a mechanism for intelligent pruning of the solution space.

In distributed system, traditional metrics are usually used for task scheduling. These metrics are especially makespan, but also the dynamic nature of distributed system will violate the above conditions. Approximate algorithms in Grid scheduling are based on a proposed in [Fujimoto 2003] objective function named Total Processor Cycle Consumption. An important branch in the suboptimal category currently widely used is called heuristic. It represents the class of algorithms which make the most realistic assumptions about a priori knowledge concerning process and system loading characteristics. It also represents the solutions to the scheduling problem where it is difficult to give an optimal answers but only requires the most reasonable amount of cost. Most of the algorithms to be discussed in the following are heuristics.

2.4.3.5 Distributed vs. Centralized

In dynamic scheduling scenarios, the responsibility for making global scheduling decisions may lie with one centralized scheduler, or be shared by multiple distributed schedulers. In a computational Grid and in Cloud, there might be many applications submitted or required to be rescheduled simultaneously. The centralized strategy has the advantage of ease of implementation, but suffers from the lack of scalability, fault tolerance and the possibility of becoming a performance bottleneck. Sabin et al [Sabin 2003] propose a centralized metasheduler which uses backfill to schedule parallel jobs in multiple heterogeneous sites. Similarly, Arora et al [Arora 2002] present a decentralized, dynamic and sender-initiated scheduling. Authors use the technique of load balancing algorithm for the Grid environment. A feature of this algorithm is that it uses a smart search strategy to find partner nodes to which tasks can migrate. It also overlaps this decision making process with the actual execution of ready jobs, thereby saving precious processor cycles.

2.4.3.6 Cooperative vs. Non-cooperative

Given the increasing data center scales, such systems are faced with challenges in terms of scalability, autonomy, and energy-efficiency. In large-scale Cloud, the centralized approach is clearly unfeasible. Firstly, centralized scheduling [Yu 2009] requires accurate, centralized information about the state of the whole system. Secondly, sites forming the grid maintain some level of autonomy, yet classic algorithms implicitly assume a complete control over individual resources. Thus, many of the existing attempts to design and implement cloud systems are still based on centralized architectures, have limited autonomy, and lack of energy saving mechanisms.

In contrast, decentralized scheduler [Ranjan 2008] negates the limitations of centralized structures with respect to fault-tolerance, scalability, autonomy, and most importantly the adequacy for the Cloud computing environment. Jobs are submitted locally, but they can be migrated to another cluster, if the local cluster is overloaded. The possibilities of migration are, however, limited, so that migrated jobs do not overload the host system. A decentralized scheduling approach assumes that each entity is autonomous and has its own control that derives its scheduling decision based on its policies. However, if the decisions are taken by several independent units, it might be the case that these units aim at optimizing their own objectives rather than the performance of the system as a whole. Such situations call for models and techniques that take the strategic behavior of individual units into account, and simultaneously keep an eye on the global performance of the system. In most cases, self-interested entities have to cooperate to achieve their respective objectives, but any cooperation must be self-enforcing and not enforced by binding agreements through third parties.

In the case of the adoption of a distributed scheduling algorithm, another problem that should be considered is if the nodes involved in job scheduling are working cooperatively or not. In the non-cooperative case, individual selfish schedulers act alone as autonomous entities and arrive at decisions regarding their own optimum objects independent of the effects of the decision on the rest of system. Good examples of such schedulers in the Grid and Cloud are application-level schedulers

which are closely associated with particular applications and optimize their private individual objectives. In the cooperative case, each scheduler has the responsibility to carry out its own portion of the scheduling task, but all schedulers are working toward a common system-wide goal. Each scheduler's local policy is concerned with making decisions in concert with the other schedulers in order to achieve some global goal, instead of making decisions which will only affect local performance or the performance of a particular job. As already define in section 2.2 a cooperative scheduler can be seen as a system including a set of consumers that be applications or tasks, and a set of resources as energy and processing power. To make the most of all available resources, a proper distribution of tasks among the devices such as to optimize the energy consumption, represents a key issue to be addressed.

The importance but also the effectiveness of this type of algorithm is more obvious, especially since they allow avoided some selfish behavior in distributed systems. Those behaviors that do not contribute to energy efficiency but good system performance. Our decentralized approach defined in chapter 3 adopts cooperation.

2.4.3.7 Task Dependency of an Application

In data centers the task execution information is mainly composed by a task execution cost and communication cost. It can be obtained by some profiling tools and compiler aid. The task graph clustering technique is considered as an effective static scheduling heuristic for scheduling parallel tasks. Given a task graph, clustering is the process of mapping task graph nodes onto labeled clusters. All tasks of the same cluster are executed in the same processor. In traditional task scheduling heuristics, the process of clustering tasks is an optimization of reducing the makespan of the scheduled graph. When the relations among tasks in a distributed system application are considered, a common dichotomy used is dependency vs. independency. In the case of dependency there are precedence orders existing in tasks. In this situation a task cannot start until all its parent are done. Dependency has consequences to the design of scheduling algorithms;

- **Independent Task Scheduling.** As a set of independent tasks arrive, a common strategy should be assigned for them according to the resources load in order to achieve high system throughput from a system point of view. Furthermore from the point of view of applications, some static heuristic algorithms based on execution cost estimate can be applied.

  Min-min and Max-min algorithms are simple and can be easily amended to adapt to different scenarios. Wu, Shu and Zhang [Wu 2000] gave a Segmented Min-min algorithm, in which tasks are first ordered by the expected completion time, then the ordered sequence is segmented, and finally Min-min is applied to all these segments. The segment improves the performance of typical Min-min when the lengths of the tasks are dramatically different by giving a chance to longer tasks to be executed earlier than in the case where the typical Min-min is adopted.

- **Dependent Task Scheduling.** Directed acyclic graph (DAG) often applies when tasks composing a job have precedence orders. This model is popular. In this case a node represents a task and a directed edge denotes the precedence orders between its two vertices. In some cases, weights can be added to nodes and edges to express computational costs and communicating costs respectively. As distributed system infrastructures grows and become more and more mature and powerful, support for complicated workflow applications, which can be usually modeled by DAGs, are provided. We can find such tools like Condor DAGMan [Condor2013 2013] and CoG [Von Laszewski 2000].

The study in this thesis focuses on the scheduling of independent tasks

2.4.3.8 Non-traditional Approaches for Grid Task Scheduling

The distributed system as Grid is a system composed of a large number of autonomous resource providers and consumers, which are running concurrently, changing dynamically, interacting with each other but self-ruling. In nature and human society, there are some systems having the similar characteristics. Therefore many

approaches, such as the Grid Economy [Buyya 2005] and other heuristics inspired by natural phenomena, were proposed to address the challenges of Grid computing and Cloud. Ideas behind these approaches are not originally applied to the scheduling problem and mapping from the problem spaces in which they are initially used to Grid scheduling problems is usually required.

2.4.3.9 Grid Economic Model

The use of economic methods in Grid scheduling involves interacting processes between resource providers and users, analogous to various market behaviors, such as bargain, bid, auction and so on. Buyya et al [Buyya 2002a] discuss some economic models that can be applied to the Grid world, including the Commodity Market Model, Tender/contract-net Model, and Auction Model. As economic models are introduced into Grid computing, new research opportunities arise. Because the economic cost and profit are considered by Grid users and resource providers respectively, new objective functions and scheduling algorithms optimizing them are proposed. Economic methods for scheduling problems are very interesting because of their successes in our daily lives. Some models can only support relatively simple Grid scheduling problems such as independent tasks. For more complex applications, such as those consisting of dependent tasks and requiring cross-site cooperation, more sophisticated economic models might be needed. The idea of reward and pay will be included in one of our contribution presented in the following.

2.4.3.10 Scheduling Methods Inspired by Nature's Laws

As scheduling is usually a process to find optimal solutions. That's why several analogies from natural phenomena have been introduced to form good heuristics, which have proven to be highly successful. Some of the common characteristics of Nature's heuristics are the close resemblance to a phenomenon existing in nature, namely, non-determinism, the implicit presence of a parallel structure and adaptability.

- Genetic Algorithm (GA) GA is an evolutionary technique used in many re-

searches. The general procedure of GA search is defined in [Braun 2001]. For its simplicity, GA is popular Nature's heuristic used in algorithms for optimization problems;

- Simulated Annealing (SA) SA is a technique used in physical research on the physical process of annealing. The simulation of annealing process which is the thermal process, can produce low-energy crystalline states of a solid. It is simulated. A SA algorithm is implemented in [Braun 2001];
- Tabu Search (TS) TS is a technique used in heuristics to overcome local optimality. It has become an established optimization approach that is used in many searches in many fields. In [Braun 2001], a TS is implemented beginning with a random mapping as the initial solution, generated from a uniform distribution;
- Combined Heuristics GA can be combined with SA and TS to create combinational heuristics. These Nature's heuristics were only relatively introduced into the scheduling area and more work needs to be done to fit them in a Grid context.

2.4.3.11 Scheduling with virtual machines

Virtualization of data centers resources is a successful direction which aims to solve many current problems related not only to Grid scheduling. Today, various users' requirements can be more or less fulfilled by introducing different queues, job priorities, advanced reservations and so on. However, without the ability to preempt running jobs a scheduling system can hardly guarantee immediate execution of interactive or high priority jobs. While certain scheduling systems may provide support for job preemption, they are limited to the provided support of underlying operating systems. Job preemption is usually achieved by completely suspending the job or reducing its priority. Even in such situation still the high priority job may be slowed down by a preempted job. For doing so swapping is one of solutions. Condor support job preemption due to recompilation of applications, which substitutes all I/O system calls with variants that allows to performing the preemption

safely. Complete suspend might also have serious impacts on parallel jobs or processes communicating over the network. With virtual machine technology, which provides strong isolation of processes running within different virtual machines, better preemption without unpredictable performance losses can be achieved. Many scheduling algorithms of the system mainly viewpoint, are classified in a hierarchical taxonomy, such as dynamic or static, distributed or centralized. Currently there are many other very important aspects forming a scheduling algorithm which are not yet covered by this method.

Through our survey on current scheduling algorithms working in the distributed system scenario, we can find that heterogeneity, dynamism, computation and data separation but mainly energy consumption in data centers are the primary challenges concerned by current research on this topic.

2.4.4 Cloud computing Scheduling

The previous discussion highlights the need to develop a comprehensive approach for job schedulings in cloud. Since cost of each task in cloud resources is different with one another, scheduling of user tasks in cloud is not the same as in traditional scheduling methods.

One of the challenging scheduling problems in Cloud data centers which we must pay attention, is to consider the allocation and migration of (VMs) with full life cycle constraints, which is often neglected [Kim 2011]. Beloglazov et al. [Beloglazov 2012] considered off-line allocation of VMs by modified best-fit bin packing heuristics. Kim et al. [Kim 2011] modeled a real-time service as a real-time VM request, and used dynamic voltage frequency scaling schemes. In [Tian 2013] autors consider online energy-efficient scheduling of real-time virtual machines (VMs) for Cloud data centers. They associate to each request a starttime, a end-time, a processing time and demand for a Physical Machine (PM) capacity. Their goal was to schedule all of the requests non-preemptively in their start-timeend-time windows, subject to PM capacity constraints, such that total busy time of all used PMs is minimized (called MinTBT-ON for abbreviation). In [Emeneker 2007], the authors proposed an image caching mechanism to reduce the overhead of loading disk image in vir-

tual machines. The authors of [Fallenbeck 2006] presented a dynamic approach to create virtual clusters to deal with the conflict between parallel and serial jobs. In this approach, it has an automatic adjustment of job load without running time predictions. A system which can automatically scale its utilization of infrastructure resources is designed in [Ruth 2006]. Another resource sharing system which can trade machines in different domains without infringing autonomy of them is developed in [Ruth 2005]. Studies described above, however, do not consider the issue of preemptable task scheduling. In [Sotomayor 2009b], a suspend/resume mechanism is used to improve utilization of physical resource. The overhead of suspending/resume is modeled and scheduled explicitly. The VMs model considered in [Sotomayor 2009b] is homogeneous, so the scheduling algorithm is not applicable in heterogeneous VMs models. The study presented in [Shivle 2004] focuses on scheduling in heterogeneous mobile ad hoc grid environments. However the scheduler algorithms can not be used in cloud computing. Parallel processing in the cloud system can shorten the execution of jobs. Parallel processing requires a mechanism to scheduling the executions order as well as resource allocation. To improve the utilization of resources in clouds Jiayin Li et al. [Li 2010] present a preemptable job scheduling mechanism in cloud system. [Sadhasivam 2009] presents the implementation of an efficient Quality of Service (QoS) based meta-scheduler and Backfill strategy based light weight Virtual Machine Scheduler for dispatching jobs. [Kailasam 2010] proposes three flavors of autonomic cloud-bursting schedulers: 1) greedy scheduler, 2) order preserving scheduler and 3) size-interval based bandwidth splitting for order preserving scheduler. For the scheduling of data access, [Bein 2010] studies the problem of actually allocating the memory of servers in a data center based on online requests for storage. It presents two algorithms that basic principle is bin packing problem. And these algorithms are used in serving online sequence of requests. [Garg 2011] proposes near-optimal scheduling policies that exploit heterogeneity across multiple data centers for a Cloud provider. It considers a number of energy efficiency factors (such as energy cost, carbon emission rate, workload, and CPU power efficiency) which change across different data centers depending on their location, architectural design, and management system. In

[Sun 2011], a novel virtual machine scheduling algorithm by a clustering mechanism for maximizing cloud computing system utility is put forward. Theoretical as well as experimental results conclusively demonstrate that the scheduling algorithm has high potential as it takes both preference and fairness into account, and maximizes cloud computing system utility by the clustering mechanism in cloud computing environments.

In [Huang 2013b], authors introduce a decentralized dynamic scheduling approach called community aware scheduling algorithm (CASA). The CASA functions as a two phase scheduling decision and contains a collection of sub-algorithms to facilitate job scheduling across decentralized distributed nodes. The first one, job submission phase, finds the proper node from the scope of the overall grid and the second one, the dynamic scheduling phase, aims to iteratively improving scheduling decisions. CASA great difference when comparing with the aforementioned approaches is that it aims to an overall performance improvement, rather than individual hosts performance boosting. The algorithm utilize nodes which cooperate in the whole system. The authors, by conducting a series of experiments have shown significant results. First of all, by applying the CASA in a decentralized scheduling setting could lead to the same amount of executed jobs comparing with the centralized solution. Autors improve job slowdown and waiting times. In addition, the authors claim that improvements were also noticed on the scheduling performance including response and waiting time and the messages overhead. The CASA, in contrast with aforementioned algorithms, is based on contacted nodes' real time responses. Our decentralized approch is based on this algorithm.

2.5 Energy Management in Cloud

Cloud computing has now become a new business model of computation and storage resources based on on-demand access to potentially significant amounts of remote datacenter capabilities. As the field matures together with the nonstop growth of the Internet and the world's businesses, it is expected that more Cloud providers will appear and provide a more diverse selection of different resources and services.

However, the deployment of datacenters in Clouds uses more and more computers which need new intallation electric, each year, increasing energy consumption and negative pressure on the environment. Research conducted by [Bianchini 2004] shows that running a server which has a power of 300-watt during a year can cost about $338, and more importantly, can emit as much as 1,300 kg CO2, without mentioning the cooling equipment. This section introduces Energy Management in Cloud. First, understanding Power Consumption is studied. Afterwards an overview about energy is presented. Finally, researchs in energy management in cloud are given.

2.5.1 Computer Power

In 2009 the US Environmental Protection Agency (EPA) reports that a PC left on overnight unnecessarily consumes between $25 and $75 of electricity a year. This gives a huge amount wasted if you multiply that number by the hundreds and thousands of machines worldwide.

To design efficient strategies for energy savings it is essential to understand the relationship between power consumption, CPU utilization and the transition delay between different server's states . This relationship by measuring power consumption of typical machines in different states is examined.

There is generally three options to conserve power when using a computer which are shutdown, hibernate or sleep state mode.

- **Sleep .** The sleep state of the machine is its situational break. When the machine restarts from this state, the machine finds the same situatution it had with pausing (with the same applications running)

  Sleep mode had different names: (1) First "Suspend" for Window 95 and Linux; (2) then "Sleep" from Windows Vista, Mac OS and Linux for now. When a machine is running, it is put on Random-Access Memory (RAM). When the machine enters in Sleep state, the RAM is placed in state of minimum energy allowing it to retain its data (eg application running). Due to currently recommended energy saving, laptops (when not in use), automat-

ically switch to this state when operating on battery power and if the lid is closed;

- **Hibernation.** When a computer enters in hibernation state mode all operational data are saved on the hard disk before turning computer off completely. When computer returns in the previous state it is restored with all programs and files open, and unsaved data intact. In standby mode, computer's state is saved in RAM; in hibernation mode, computer's state is saved on the hard disk. Hibernation can save electrical power. In hibernate state mode, the hardware of a machine is almost completly powered down in constrat of the shutdown which is completly. A small amount of power is used to power the CMOS powered, and prevents the user from entering the BIOS setup upon resuming the system. A machine which is in hibernated state mode uses less electrical power than one which is in sleep or suspend mode;

- **Shutdown.** Stopping off a machine cuts the power of the main components (such as processors, RAM modules and hard drives) of a computer. However, some internal components, such as an internal clock, may retain power.

One study performed by [Duy 2010] give results represented in figures 2.9 and 2.10.

In their experimental environment they used :

- Linux machine with AMD Phenom 9500 Quad-Core Processor 2.2GHz,
- A Windows machine with AMD Athlon 64 X2 Dual-Core Processor 5000+ 2.6GHz.
- Machines were connected to a System Artware SHW3A watt-hour meter at the power plug to record power consumption of the whole machines.

In Linux machine, from their results, they assert that the suspend-to-RAM seems to be the best state, in terms of both power consumption and transition delay, as it needs only 10 seconds to come to this state from the idle state, and 20 seconds for the opposite direction. Similarly, the standby state consumes less power than the idle state in Windows machines. It consumes only 3.7W, and takes as little as 5 and 10 seconds for transition delays.

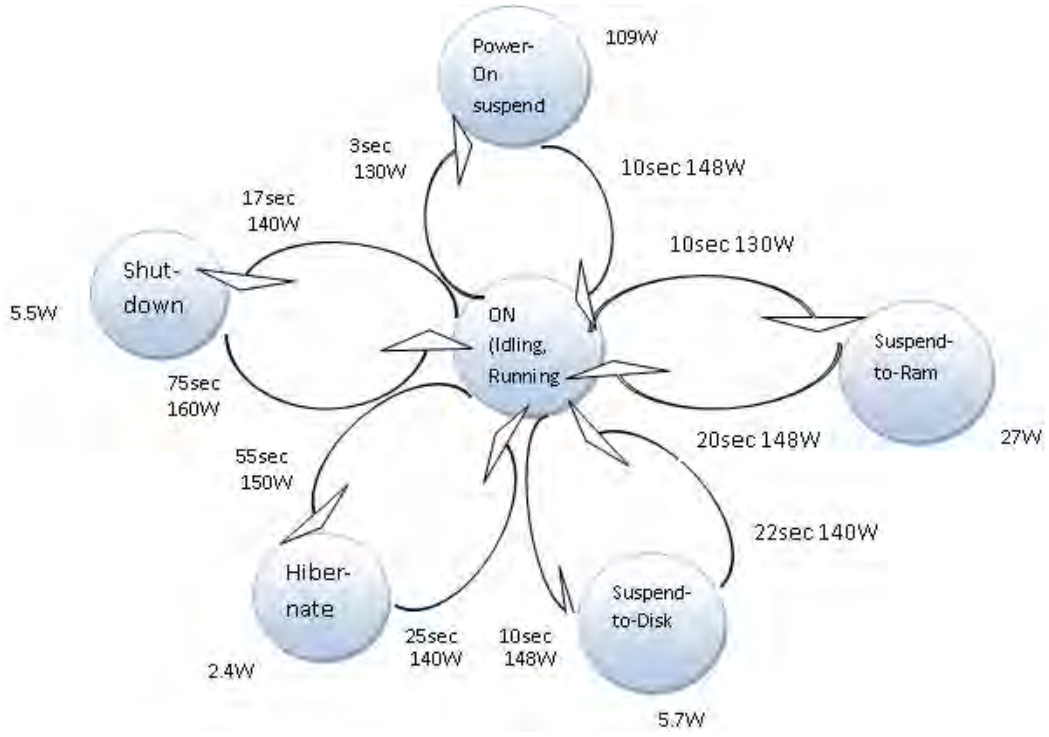

Figure 2.9: State transition of the Linux machine.

2.5.2 Data Center Characteristics

In [Brown 2008] authors argue that data centers are not designed for people but for computers. consequently, data centers generally have no windows and minimal circulation of fresh air. They are often housed in new construction dedicated for that or in existing locals that have been renovated.

The size of the areas containing the data center varies with that of the latter. Large data centers housed in large buildings are becoming more and more, with advances in technology, common as smaller data centers consolidate. Data center rooms contains rows of IT equipment racks, network equipment and many others. To protect all this equipenments, uninterruptible power supply (UPS) unit is used. Electricity is first supplied to UPS unit before reaching the IT equipment rack.

The server power supply unit (PSU) convert current (AC) to direct current (DC) the electicity coming out of UPS. The low-voltage DC power supplied from the

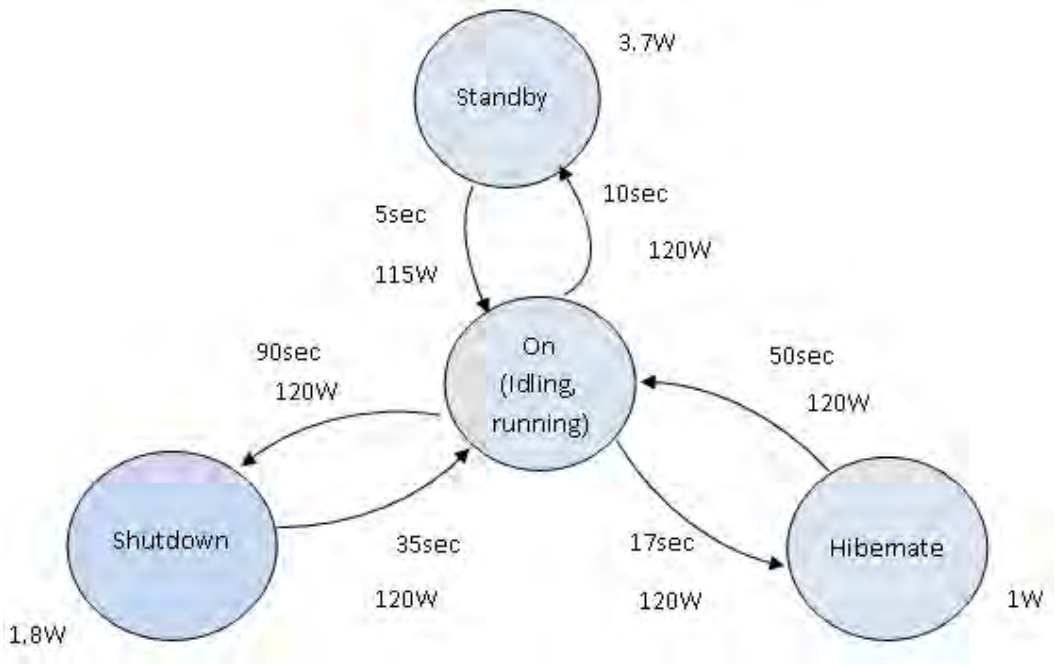

Figure 2.10: State transition of the Windows machine.

UPS is used by the server's internal components of, such as the central processing unit (CPU), memory, disk drives, chipset, and fans. The DC voltage serving the CPU is adjusted by load specific voltage regulators (VRs) before reaching the CPU. Electricity is also routed to storage devices and network equipment, which facilitate the storage and transmission of data.

It is very important for the proper function of the data center to reduce considerably the huge amount of heat delivered by the IT equipment and power which continuously operating. Because of the configuration of buildings which have no window , computer room air conditioning (CRAC) are often used for cooling. In most configurations the entire air handling unit (AHU) is situated on the data center floor and contains fans, filters, and cooling coils and is responsible for conditioning and distributing air throughout the data center.

Data centers use a huge amount of energy to supply three key components: IT equipment, cooling, and power delivery. These energy needs can be better understood by examining the electric power needed for typical data center equipment in

and the energy required for cooling. A lot of research work exist on how to reduce the energy consumed in data centers. This is one major objective of this thesis.

2.5.3 About energy consumption

Introduced by the Green Grid [Azevedo 2012], Power Usage Effectiveness (PUE) is a measure of efficiency used usually to better evaluate energy in data centers. It is defined by this formula : $PUE = \frac{Totalfacilityenergy}{ITequipmentenergy}$. The total facility energy concern the energy used solely by the data center mainly everything that supports the IT equipment including power, cooling, lighting, etc.; while the IT equipment energy corresponds to the energy consumed by equipment that is used to manage, process, store, or route data within the compute space. See figure 2.11.

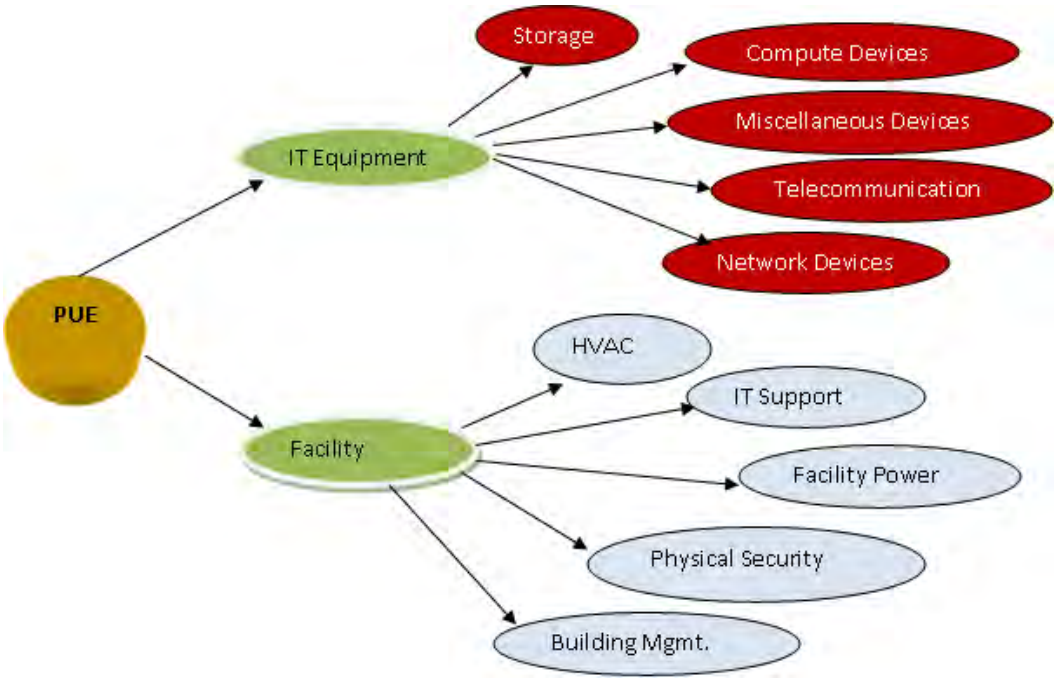

Figure 2.11: Overview of the subcomponents within a typical data center's facility and IT equipment

One of the best ways to reduce energy savings in computing is to reduce the amount of IT equipment. Virtualization is the key to the consolidation which eliminate a large number of underused physical servers while providing the same

level of service with a small number of highly used servers.

As was already said in the previous section, there is opportunity in data centers to reduce heating, ventilation, and air conditioning (HVAC) energy by employing row containment strategies to reduce the mixing of cool supply air with hot evacuation air. In-rack cooling solutions offer even greater gains by reducing the volume of conditioned space from the entire data center to the inside of the racks.

The energy consumption of data centers has been estimated at 61 billion kilowatt-hours in 2006, about 1.5% of total US consumption. The report estimates this consumption to 5.8 million average U.S. homes. In the report to US Congress on server and data center energy efficiency (August 2007), authors think that the real problem is that data center power consumption is rising faster than overall demand. In the report to US Congress on server and data center energy efficiency (August 2007) says that in 2006, data servers account for nearly 1.5% of total electricity consumption in the U.S. at a cost of approximately $4.5 billion per year. According to a study by Pike Research [pikeresearch 2013] "the adoption of cloud computing will lead to a 38% reduction in worldwide data center energy expenditures by 2020, compared to a business as usual (BAU) scenario for data center capacity growth, according to Cloud Computing Energy Efficiency".

A study conducted by Lawrence Berkeley National Laboratory (Berkeley Labi [Berkeley 2013] with funding from Google indicates that moving common data used by 86 million U.S. workers to the cloud could can save enough electricity annually to power Los Angeles for a year. The report worked at three common business applications that are email, customer relationship management software, and bundled productivity software. The transfer of these applications in a local server to a centralized cloud services could reduce information technology energy consumption by up to 87% - about 23 billion kilowatt-hours. This is roughly the amount of electricity used each year by all the homes, businesses and industry in Los Angeles

A report published this year by DCD Intelligence [dcd 2013][dcd 2014] states that North American data center energy consumption is currently 11.55GW, an increase of 6.8% compared to 2013. This helped, thanks to a great attention paid to the management and increased monitoring and the effectiveness of practices, down-

grade forecast energy consumption energy by 2016 compared to 2013 projections 8% per annum growth from 2014 to 6% a saving of 1 GW by 2016.

2.5.4 Researches on power consumption

In the past few years, people started to realize that the energy consumption is a critical issue since energy demands of clusters grow, with an increasing number of data centers. Several strategies for energy saving in heterogeneous clusters have been proposed and studied. The existing techniques for energy savings in the area of enterprise power management at a server farm can roughly be divided into two categories: dynamic voltage/frequency management inside a server and shutting down servers when not in use. However, several other techniques exist (e.g New hardware design and Os management). In the former, power savings are gained by adjusting the operating clock to scale down the supply voltages for the circuits. Although this approach can provide a significant reduction in power consumption, it depends on the hardware components' settings to perform scaling tasks.

2.5.4.1 Dynamic voltage-frequency scaling (DVFS)

Dynamic voltage and frequency scaling (DVFS) had been widely used for power and energy optimization in embedded system design. There is a large field of researches on using DVFS in single and multiple processor systems to minimize energy consumption. Currently, temperature related problems in some chips had typically been addressed by techniques that lower the average temperature or keep the temperature under a given threshold in order to reduce energy consumed. Dynamic voltage-frequency scaling (DVFS) is an example of such techniques. Dynamic voltage-frequency scaling is a technology now present in high-performance microprocessors that its function is to reduce the supply voltage to the CPU consumes less power. Energy-aware scheduling in multiprocessor systems and grid systems always concerned researchers as the following overview shows. In [Xian 2007] authors present an energy-aware algorithm which schedules real-time tasks in multiprocessor systems that support a technique used in many system : dynamic voltage

scaling (DVS). In [AlEnawy 2005] the DVFS capability treats to energy minimization for periodic preemptive hard real-time tasks that are scheduled on an identical multiprocessor platform. AlEnawy and Aydin [AlEnawy 2005] suggest partitioned scheduling in their work and assume that the tasks are assigned rate-monotonic priorities. To find a solution to this problem, they proposed an integrated approach that include rate monotonic scheduling, an admission control test, a partitioning heuristic and a speed assignment algorithm. These works propose to control the energy consumption of hardware by adjusting voltage levels. A significant bottleneck of this method is the performance impact associated with stalling or slowing down the processor. Srinivasan et al. in [Srinivasan 2004] argue that when the workload that is going to run on the system is known, voltage/frequency levels, architecture configuration or job allocation can be adjusted at the design stage to avoid dynamic thermal management as much as possible. With this technique , several temperature-aware job allocation and task migration techniques have been proposed (e.g. [Bartolini 2013], [Zhao 2013]) to reduce thermal hot spots and temperature variations dynamically at low cost.

2.5.4.2 Consolidation

The utilization ratio of data center resource is only 30% [Kliazovich 2012]. So, to concentrate the workload in a minimum set of the computing resources is very important for saving energy. In addition, energy waste in VM migration can be a significant percentage of total energy consumption in cloud computing [Nagothu 2010]. A number of studies demonstrate that VM migration can cost a big percentage of energy use, as well as cause extensive runtime [Baliga 2011] and VMs migrations are considered as zero energy cost. Live migration of VMs allows transferring a VM between physical nodes without suspension and with a short downtime. However, live migration has a negative impact on the performance of applications running in a VM during a migration. Voorsluys et al. have performed an experimental study to investigate the value of this impact and find a way to model it [Voorsluys 2009]. They found that bad performance and downtime depend on the application behavior, i.e.,the number of memory pages the application updates during its exe-

cution. However, for the class of applications with dynamic workloads, such as web-applications, the average performance degradation including the downtime can be estimated as approximately 10% of the CPU utilization. Most of the existing algorithms are focusing on the resource utilization.

Consolidation is a solution which increases energy efficiency in data centres. As already mentioned above the key to the consolidation is virtualization which partitions computational resources and enables the sharing of physical server. Many services often need only a small fraction of the available computational resources. Many services often need few computational resources [View 2007] of a data centre server. However, even when run at a low utilization, servers typically need up to 70% of their maximum power consumption [Hintemann 2010].

2.5.4.3 Other techniques

Other techniques such as energy-efficient Cloud infrastructure, new hardware design and Os management can be used. In [ORGERIE 2010], Orgerie et al study the impact of virtual machines aggregation in terms of energy consumption. As part of the migration of virtual machines within the infrastructure, the authors presented several load balancing strategies. Their research deals with the support of energy-efficient frameworks dedicated to Cloud architecture. They present the design of a new original energy-efficient Cloud infrastructure called Green Open Cloud (GOC). The GOC architecture supports swiching OFF unsused computing, and predicting computing resources usage in order to switch ON the nodes which are required iin a near future. They proposed software frameworks able to reduce the energy usage of Cloud infrastructure.

In [Chase 2001], Chase et al. illustrated a method of determining the aggregate system load and the minimal set of servers that can process the load. A similar idea in cluster load balancing determines when to turn machines on or off to handle a given load [Pinheiro 2001], [Pinheiro 2003]. A critical problem for these ideas is that in order to turn lightly loaded machines off or to assign workload to newly turned-on machines, the task need to be transferred from one machine to another. But almost all the operating systems used in the real clusters, e.g. Windows, Unix and Linux,

cannot support such kind of operations. So in their research specific OS features have to be developed and applied, which in turn limits the practicability of their approaches. In [Pan 2005], Feng et al. proposed a method of finding the best match of the number of cluster nodes and their uniform frequency (called energy gear in their research). But they did not consider much about the effect of scheduling algorithms. There are indeed a few high-performance computers designed with energy-saving in mind, such as BlueGene/L [Adiga 2002], which used a "system on chip" to reduce energy consumption, and Green Destiny [Warren 2002], which used low-power Transmeta nodes. But their concern on energy saving is only confined to the design of hardware, with nothing to do with the strategies for power control at run-time, which also plays an important role.

There is also a large effort in saving energy for desktop and mobile systems. In fact, most of the early researches in energy-aware computing were on these systems. At the system level, there has been work in trying to make the OS energy-aware by making energy the first class resource [Ellis 1999]. On device-specific energy saving, studies have been conducted on saving the energy consumed by CPU [Flautner 2001], by disk [Helmbold 1996], and by memory and network [Krashinsky 2005]. A number of good methods and ideas in these studies could be introduced to the energy saving schemes in cluster systems.

In [Aupy 2012], Aupy et al. aim at minimizing the energy consumption while enforcing two constraints: a prescribed bound on the execution time (or makespan), and a reliability threshold. Because DVFS lowers the reliability of a schedule while reducing the energy consumption, the heuristics use re-execution of some tasks to improve it. They assess the complexity of the tri-criteria scheduling problem (makespan, reliability, energy) of deciding which task to reexecute, and at which speed each execution of a task should be done, with two different speed models : either the CONTINUOUS or the VDD-HOPPING model where processor can run at a finite number of different speeds and change its speed during a computation. They propose several novel tri-criteria scheduling heuristics under the continuous speed model. Heuristics turn out to be very efficient and complementary.

Energy management in computing clusters can be achieved either by means of

static or dynamic power management (SPM resp. DPM) [Chedid 2002]. SPM, sometimes also referred to as low-power computing is applied at design time of a system. For instance, by improving the CPU microarchitecture and/or using low-power CPUs. Most recent examples of systems following this approach are the BlueGene/Q [Chen 2011] supercomputers which are among the most energy efficient computing systems available today [Green500List2013 2013].

2.5.5 Meta-scheduling solutions

Generally meta-scheduling solutions are classified into three categories, namely the centralized, hierarchy, and decentralized schemes. In a centralized scheduling architecture [Yu 2009], scheduling decisions are made by a central controller for all VMs. The scheduler maintains all information about the VM and keeps track of all available resources in the system. Centralized scheduling organization is simple to implement and easy to deploy. A. Beloglazov and Rajkumar Buyya [Hamscher 2000] have proposed and evaluated heuristics for dynamic reallocation of VMs. The goal was to minimize energy consumption, while providing reliable QoS. Their results show that the technique of dynamic reallocation of VMs and switching off the idle servers brings substantial energy savings and is applicable to real-world Cloud data centers. This work has not investigated setting the utilization thresholds dynamically according to a current set of VMs allocated to a host, leveraging multi-core CPU architectures, and decentralization of the optimization algorithms to improve scalability and fault tolerance. In order to avoid scheduling self competition, some Clouds only allows one scheduler to manage each virtual organization. However, Centralized scheduling organization is not adequate for the Cloud because of the nature of the Cloud computing environment. Again, these centralized services limit their scalability.

In distributed scheduling, there is a central manager and multiple lower-level entities. This central manager is responsible for managing the complete execution of a VM and assigning the individual VM to the low-level providers. Each lower-level entity scheduler is responsible for mapping the individual tasks into Cloud resources. R. Ranjan et al [Ranjan 2006] proposed a meta-scheduling framework.

Each resource consumer may value various resources differently depending on its QoS based utility functions and may want to negotiate a particular price for using a resource based on demand, availability and its budget. An SLA is the agreement negotiated between a meta-scheduler, entitled the Grid Federation Agent (GFA), and the Local Resource Management System (LRMS) of the local sites in terms of acceptable job QoS constraints, such as job response time and budget spent. It highlights a bid-based SLA contract negotiation model. Furthermore, the contract net protocol [Smith 1980] based SLA bids are restricted with a certain expiration time, and different economic parameters such as setting price, user budget and deadline. Authors propose a greedy backfilling heuristic for application on the participating LRMSs during their cooperation with the meta-schedulers. For managing peering arrangements between grids, authors proposed InterGrid. [Dias de Assunção 2008] Although the structure of the overall Inter- Grid ecosystem is hierarchical, the InterGrid Gateways employed upon the top of each participating grid are distributed in a decentralized manner. We can say that the grid has a hybrid structure. Each IGG is aware of the agreements with other IGGs, and is capable of enabling resource allocation across multiple grids with pluggable policies. The InterGrid/IGG relies on external decentralized approaches.

Such approaches are not adequate since it requires entities to deploy different scheduling policies to the central manager [Yu 2009]. The failure of the central manager results in entire system failure.

In decentralized meta-scheduling, each job has its local schedule which receive job submissions from local users. It assigns jobs to the local resource management system. Local schedulers of different nodes are capable of exchanging information and sharing jobs between each other in order to balance the resource load amongst participating nodes. Besides the issue of efficiency and overhead, the decentralized scheme brings better scalability, compared to other scheduling schemes. C. Comito et al [Comito 2011] proposed a task allocation scheme for mobile networks focusing on energy efficiency. To conservatively consume energy and maximize network lifetime they have introduced a heuristic algorithm that balances the energy load among all the devices in the network. Authors have implemented a prototype of

the system and evaluated the scheduling strategy through simulation experiments. Results show that the proposed scheduler greatly enhances the performance of the system compared to time-based traditional schedulers like the round-robin. They achieved improvements in terms of network lifetime, number of active devices and number of completed tasks. Authors refer to a cooperative Energy-Aware Scheduling strategy that assigns computational tasks over a network of mobile devices optimizing the energy usage.

MaGateSim [Huang 2009], a Simulation Environment for a Decentralized Grid Scheduler, is designed to be a decentralized grid scheduler that focusing on grid scheduler interoperation, and is complemented by a dynamic resource discovery approach on decentralized network. In order to share the jobs submitted from a local MaGate to other MaGates within the same grid community, a set of community scheduling important parameters are evaluated and discussed to address various job delegation scenarios between different MaGates. The same authors propose a decentralized dynamic scheduling approach named the community-aware scheduling algorithm (CASA) [Huang 2013b]. However, the problem has not been explored in the context of the optimization of energy consumption and in cloud computing.

2.5.6 Summary

The figure 4.1 below gives a comparative table of some algorithms. This chapter has introduced the state of the art of this thesis. It has started with a brief introduction into virtualization which is a fundamental technique enabling server consolidation in many cloud data centers such VM live migration techniques. Then, cooperation and coordination in distributed Systems and cloud computing were introduced. Finally, related work on scheduling in cloud and energy management in cloud are presented. In computing paradigm server virtualization is typically used to ease compute infrastructure management perform server consolidation for energy savings. First, cooperation and coordination in distributed Systems were introduced by presenting its mechanisms and a few selected cooperation and coordination in cloud. Afterwards, cloud computing was presented as a promising computing paradigm whose goal is to offer resources (e.g. compute, storage) on-demand based on the pay-as-

Algorthm	Dec	Cooperative	Energy	Virt
[Arora 2002][Lai 2008]	+	-	-	-
[Yu 2009] [Aupy 2012] [ORGERIE 2010]	-	-	+	+
[Buyya 2002a]	+	-	-	+
[Tian 2013] [Beloglazov 2012] [Kim 2011] [Zong 2011]	-	-	+	+
[Sotomayor 2009b] [Fallenbeck 2006] [Emeneker 2007]	-	-	-	+
[Bein 2010]	-	-	+	-
[Sun 2011]	-	+	+	-
[Xian 2007]	-	-	-	-
[AlEnawy 2005]	-	-	+	-
[Nagothu 2010]	-	-	-	-
[Ranjan 2008]	+	+	-	-
[Pinheiro 2001]	-	+	-	-
[Huang 2013b]	+	+	-	-
[Comito 2011]	+	-	+	-
[Hamscher 2000]	-	-	-	+
[Ranjan 2006]	+	-	-	+
Our decentralized approach (chapter 3)	+	+	+	+

Table 2.1: Characteristics of some existing algorithms. Dec=Decentralized - Virt=vituluzation

you-go model. To be precise, we have first provided a cloud computing definition and introduced the cloud characteristics, service, and deployment models. Then, deployment models. Finally, we have conducted a review of scheduling and energy management approaches in cloud computing. Particularly, presenting the scheduling phases, we have also reviewed current researches. In addition, after defining the terminology and presenting traditional power measurement techniques, we have discussed techniques used for energy saving. Our study has shown that despite the fact that a lot of efforts have been made over the past years to propose scheduling algorithms, to design and implement IaaS cloud management systems, as well energy management techniques in computing clusters, much work is still left to be done. The three key observations from this chapter are:

- Despite the vision of consolidation, cooperation and coordination existing

IaaS cloud management systems still must be optimized to reduce energy consumption in data center. Moreover, most of the IaaS cloud management systems are based on centralized architectures thus limiting their scalability.

- There is a clear lack of an experimentally validated holistic energy-efficient IaaS cloud management system which federates the introduced VM management algorithms (i.e. VM placement, underload and overload management, VM consolidation, and power management).

- A huge amount of attention has been given to the design of decentralized VM management algorithms. A considerable low amount of attention has been given to the application of cooperative meta-scheduling algorithms.

Chapter 3

An Energy-Aware Scheduling Strategy for Allocating Computational Tasks

---

Contents

In the previous chapter we have presented the context of this work and reviewed the state of the art on the design and implementation of energy-efficient cloud management systems. Our analysis has shown that existing cloud management systems are mostly based on centralized architectures and energy management mechanisms are suffering several limitations. To address these limitations, our contribution is to design, implement, and evaluate a novel cloud management system which provides a holistic energy-efficient VM management solution by integrating advanced VM management mechanisms such as underload mitigation, VM consolidation, and power management. We propose two novels energy-efficient cloud management heuristics.

This chapter presents the design of these algorithms. It is structured as follows. Section 3.1 introduces the design principles. Section 3.2 describes the model and objectives. Section 1 introduces the algorithm Anti Load-Balancing. Section 3.4 presents the centralized approach. Section 3.5 discusses the decentralized approach. Finally, Section 3.6 summarizes the contributions.

3.1 Design principles

The main goal of this thesis is to design and implement a scalable, and energy-efficient IaaS cloud management system. Thereby, several properties have to be fulfilled by a cloud management system in order to achieve these goals. First, the cloud management system architecture has to scale across many thousands of nodes. Second, nodes and thus framework management components can fail at any time. Therefore, the system needs to self-heal and continue its operation despite of component failures. Finally, the cloud management system has to be easily configurable. While achieving this goal, our system must also improve energy efficiency as discussed in Chapter 2.

Like managing load distribution, energy consumption is a fundamental problem

for large scale systems. Choice of location of running jobs has a large impact on the behavior of such systems. A minimal number of computers can be selected in order to run requested job at a particular time, even taking into account user requirements. Users can even add more criteria to obtain an optimal power consumption. Difficulties arise as dynamism is introduced and as jobs arrive, leave and change over time. Several solutions exist to reduce power consumption due to computational nodes and data centers. Also, mixing energy consumption and performance (or QoS) objectives rise the difficulty level needed to find a good placement for tasks. Furthermore, to achieve energy savings while maintaining system performance, we need: i) a good description of the characteristics of applications and user feedback, ii) a complete and concise description of the system. A problem usually studied in distributed systems is how to evenly distribute workload. But when the goal is to reduce energy consumption, this type of algorithms can lead to have computers largely under-loaded and therefore consuming energy unnecessarily. Our study will be the management of virtual machines in the data center. Here we will therefore look at the opposite problem : concentrate the load on a minimum number of machines. The goal is to turn off the released computers and therefore minimize the energy consumption of the system. To achieve this goal a node is selected, its load is distributed to other nodes and then switched off. To study this problem we have proceeded in three phases: (i) Model the problem, (ii) Design an algorithm and develop a simulator, (iii)Simulate.

To save energy, as seen in Chapter 2, two main techniques exist: dynamic voltage frequency/scaling (DVFS) and consolidation. Our work will be based on the latter. This will involve to move VMs in order to switch off underloaded nodes for better energy efficiency. To study this problem we developed an heuristic "anti load-balancing" to provide a power gain in managing data centers.

In this chapter we will optimize the jobs's migration in order to increase the gain of energy with anti load-balancing in distributed systems. We developed an heuristic named *Energy aware clouds scheduling using anti load-balancing algorithm* (EACAB) which works by associating a credit value with each node. The credit of a node depends on its affinity to its jobs, its current workload and its commu-

nication behavior. We propose an approximation approach that places tasks based on the load on resources. This algorithm is based on the use of an indicator called credit for each job that quantifies its affinity for each machine and determine the best candidates for migration. Apart from the algorithm itself, this thesis presents an implementation and evaluation of a simulator Enersim 1 which derive of the simulators Cloudsim [Calheiros 2011] and Alea [Klusáček 2010]. Another version of Ensersim 1 named Enersim 2 is used for the evaluation of our decentralized cooperative algorithm. The two simulators are presented in chapter 4. We present our algorithms and will show experimental results in the next chapter to indicate how our approximations approach can provide solutions closer to the optimal in terms of energy efficiency.

3.2 Model and objectives

The overall objective of the energy management policy is to reduce energy consumption, while satisfying the users' performance demand within Cloud. In this section, we discuss the model, hypotheses, constraints and objectives.

This section presents the system model of energy used for an energy-efficient IaaS cloud management for private clouds. First, the distributed system model is introduced. Then, objectives are presented.

3.2.1 Model

In the following we will use interchangeability the terms job, task and VM. The nodes are the machines that host virtual machines ie the host nodes of virtualization. The table 4.1 gives a list of notations and terminology used.

We consider a cloud data center environment consisting of $H = \sum_{i=1}^{N} H_i$ heterogeneous physical nodes. Each site i has H_i nodes. There are N sites. Each node is characterized by the CPU performance defined in Millions Instructions Per Second (MIPS). We consider T_i tasks associated to VM_i VMs, that run on the site i.

- R_i : load of the site i. This load depends on the number of VM (VM_i) executed by the site and their load ($l_{i,j,k}$ is the requested load of VM k in site

Table 3.1: List of notations and terminology used.

H and N	set of nodes and number of sites
H_i	set of nodes in site i
$H_{i,j}$	node j in site i
$P_{i,j}^{min}$ and $P_{i,j}^{max}$	minimun power (pmin) and maximum power (pmax) of node j in site i
V	set of VMs
VM_i	set of VM in site i
$VM_{i,j}$	set of VMs in node j in site i
$VM_{i,j}^{w}ait$	set of VMs waiting in node j in site i
$VM_{i,j,k}$	VM k in node j in site i
$VM_{i,j,k}^{min}$	smallest VM k in node j in site i
$l_{i,j,k}$	load of VM k in node j in site i
R_i	requested load of site i
$r_{i,j}$	the aggredated load of all VM on node j in site i
$VM_{i,j,k}^{pe}$	number of processing elements (PEs) requested by VM k in node j in site i
$VM_{i,j,k}^{mips}$	speed $VM_{i,j,k}$ in node j of in site i
$VM_{i,j,k}^{RAM}$	Ram of $VM_{i,j,k}$ in node j of in site i
$D_{i,j,k}$	the estimated execution duration of VM k in node j in site i
$D_{i,j,k}^{w}$	the waited duration of VM k in node j in site i
$w_{i,j,k}$	represent the weight of VM k in node j in site i
nH	number of nodes
nH_i	number of nodes in site i
nVM	number of VMs
nVM_i	number of VMs in site i
$nVM_{i,j}$	number of VMs in node j
ε and γ	over-loaded threshold and under-loaded threshold
$H_{i,j}^{under}$	under-loaded node j in site i
$H_{i,j}^{over}$	overloaded node j in site i
$M_{i,j,k}^{req}$	request message sent by node j in site i for VM k
$H_{i,j}^{req}$	requester node j in site i
$H_{i',j'}^{resp}$	responder node j' in site j' for execution VM k
$M_{i',j',k}^{ack}$	an accept message sent by node j' in ste i'
$M_{i',j',k}^{ass}$	an assign message sent from a node j in site i to another node j' in site i' for exceution of a job k.
$M_{i",j",k"}^{info}$	inform message sent from a node j" in site i" to inform other nodes that a $VM_{i",j",k"}$ is being rescheduled

i on node j). $r_{i,j}$ is the aggregated requested load of all VM on node j in site i. Note that if VM k is not running on node j in site i, then $l_{i,j,k}$=0.

$R_i = \sum_{j=1}^{H_i}(r_{i,j})$

$r_{i,j} = \sum_{k=1}^{T_i}(l_{i,j,k})$

- Load C_i and speed V_i of site i

  $C_i = \sum_{j=1}^{H_i}(c_{i,j})$

  $V_i = \sum_{j=1}^{H_i}(v_{i,j})$

  $c_{i,j}$ Actual load of node j in site i

  $v_{i,j}$ Maximum speed of node j in site i in Mips

- Job satisfaction S_i of site i (same for $s_{i,j}$, job satisfaction of node j in site i)

  $S_i = \frac{C_i}{R_i}$

  $s_{i,j} = \frac{c_{i,j}}{r_{i,j}}$

- $VM^{pe}_{i,j,k}$: number of processing elements (PEs) requested by a VM k in node j (CPUs) on site i. "Note: It will be assumed in experiment part that this value is 1".

- $VM^{RAM}_{i,j,k}$ is constant accross the node and only depends on k. j and i are kept for coherence of notations.

- $D_{i,j,k}$: the estimated execution duration of VM $VM_{i,j,k}$

- $w_{i,j,k} = VM^{pe}_{i,j,k}.D_{i,j,k}$: represent the weight of VM $VM_{i,j,k}$

- The load of $VM_{i,j,k}$ depends on the speed estimation $VM^{mip}_{i,j,k}$ of the $VM_{i,j,k}$ in node j of site i, its estimated execution duration : $l_{i,j,k} = w_{i,j,k}.VM^{mips}_{i,j,k}$.

3.2.2 Hypothesis

- Communications are modeled by a linear model of latency and bandwidth.

- For each scenario there is at least one VM.

- Migration cost. To migrate a VM, only RAM has to be copied to another node. The migration time depends on the size of RAM of $VM_{i,j,k}$ in node j of site i and the available network bandwidth.

  VM migration delay = $VM^{RAM}_{i,j,k}$ / bandwidth + C (C is a constant).

  Bandwidth is considered as constant.

- Nodes have two different power states : Switched on and switched off. While switched on, power consumption is linear in function of load between $P_{i,j}^{min}$ and $P_{i,j}^{max}$.

We will use the classical linear model of power consumption in function of load : $\forall i, j \qquad P_{i,j} = P_{i,j}^{min} + c_{i,j}(P_{i,j}^{max} - P_{i,j}^{min})$ if node j is switched on, $P_{i,j} = 0$ otherwise. Therefore the total power consumption of the system is:

$$P = \sum_{i=1}^{N} \sum_{j=1}^{H_i} P_{i,j}$$

To obtain energy consumed during a time slice, instantaneous power is integrated over time $\int_{t1}^{t2} \mathrm{P(t)}\,\mathrm{d}t$. Total energy is then obtained by summing all the energy of those time slices.

3.2.3 Objectives

The main objective of our approach is to improve cloud's total energy efficiency by controlling cloud applications' overall energy consumption while ensuring cloud applications' service level agreement. Therefore our work aims to satisfy several objectives :

- Ease of task management : we design a system which is flexible enough to allow for dynamic addition and removal of nodes. As system components can fail at any time, it is desirable for a system to heal in the event of failures without human intervention. Consequently, we aim at designing a system using self-healing mechanisms to enable high availability.

- Energy Efficiency: One of our goals is to propose task placements management algorithms which are capable of creating idle times on nodes, transitioning idle nodes in a power saving state and waking them up once required (e.g. when load increases).

3.3 Anti load-balancing

In many fields of research, performance of distributed systems, the load-balancing technique is often used in order to guaranty good performance. This technique leads

to spread load on all available nodes, which is efficient from the performance point of view, but not from the energy point of view. For energy savings, it is the opposite problem of consolidation which is studied. This load concentration or unbalancing operation saves the power consumed by the powered-down nodes, but can degrade the performance of the remaining nodes and potentially increase their power consumption. Thus, load concentration involves an interesting performance vs. power tradeoff. The goal is to turn off the unused nodes and therefore minimize the whole energy consumption of the system. It is critical to reduce energy consumption in information systems, especially in large scale distributed systems. There are many algorithms on how to reduce the total power consumption in such systems, one of the most efficient being to turn off a maximum number of nodes without impacting the jobs running on the system. In these algorithms heavily loaded nodes turn into heating points. It leads to an increase in the energy consumption on the cooling infrastructure. Most researches on energy efficiency try to reduce energy consumption of nodes, but they usually do not take into account the cost of cooling systems and related infrastructure. Thus it is important to set a minimum threshold to consolidate jobs, but also to avoid to load nodes heavily by concentrating too many jobs on the same node.

Dynamic consolidation of virtual machines (VMs) and switching idle nodes off allow Cloud providers to optimize resource usage and reduce energy consumption. However, the obligation of providing high quality of service to customers leads to the necessity of dealing with the energy-performance trade-off. Consolidation, concentrating the workload onto fewer physical nodes, can save energy. Current virtualization techniques have a low overhead, allowing to consolidate dozen of virtual machines on one node with a low impact.

The main contribution of this section is to propose an efficient algorithm that migrate VMs to reduce energy-consumption while preserving performance and preventing hot spots. We propose an energy-efficient heuristic Anti Load-Balancing (ALBA), for virtual machine placement in cloud computing virtualized data centers, in oder to reduce a low energy consumption. The anti load-balancing technique is the basis of our two centralized and decentralized approaches that we will present in

the remainder of this chapter. This run-time algorithm will migrate tasks while they are running, using on-the-fly migration technology. *Anti load-balancing* algorithm based on migration techniques use mainly two threshold based rules. Depending on a node load compared to a threshold, VMs are migrated in order to switch off nodes. Also if over a threshold, load is balanced on other nodes or a new node is switched on in order to keep a good quality of service.

The proposed algorithm works by associating two threshold values with each node. When a node is under-loaded (load $< \gamma$), all its tasks are migrated to a comparatively more loaded node. We also use the over-loaded threshold ε, which we call saturation, to measure the saturation of a node.

In this section we present our algorithm ALBA. First, probem description, node underload and overload detection and the VMs placement are detailed. Then, algorithm statements and scenarios are presented. Finally, the algorithm is presented.

3.3.1 Problem description and VMs placement model

3.3.1.1 Assumption

For simplicity, we assume that each physical node consists of one processor. The nodes are heterogeneous. One or more VMs can run in a node. The tasks or subtasks submitted by end users are encapsulated in VMs. We assume that each new VM has a predefined resource requirement. If a node runs VMs, we call it an active node. Accordingly, idle nodes are called inactive nodes.

3.3.1.2 Problem description

In cloud computing environments, it is not sure when the task will be submitted and how many resources it will need. So the VMs will be sent to a node dynamically. When the VM finishes or the size of VM changes, it can be interesting to rebuild the correspondence relationships between VMs and nodes. To put the resource into effective use and minimize the number of active nodes, VM migrations are inevitable. But VM migration belongs to coarse-grain relocation, and the energy cost caused by it can not be looked down. Therefore, we will focus on how to make

a placement scheme to increase resource utilization rate and reduce energy waste.

3.3.2 Node Underload Detection

Scheduling a workflow is a process of finding the mapping of tasks to the suitable resources so that the execution can be completed with the satisfaction of objective functions, such as execution time minimization. Existing workflow scheduling approaches are non-coordinated, where workflow schedulers perform scheduling related activities independent of the other schedulers in the system. They directly submit their tasks to the underlying Cloud resources without taking into account the current load, priorities, and utilization. This leads to over-utilization or a bottleneck on some valuable resources, while leaving others largely under-utilized.

Although complex underload detection strategies can be applied, for the purpose of demonstration in this chapter a simple approach is used. First, all the underload nodes are found using the selected underload detection algorithm, and the VMs selected for migration are allocated to the destination nodes. Then, the system finds a node with the minimal utilization compared with the underloaded threshold, and attempts to place all the VMs from this node on other nodes, while keeping them not overloaded. If such a placement is feasible, the VMs are set for migration to the determined target nodes. Once the migrations are completed, the source node is switched to sleep mode to save energy. If all the VMs from the source node cannot be placed on other nodes, the node is kept active.

3.3.3 Node Overload Detection

Each compute node periodically executes an overload detection algorithm to deconsolidate VMs when needed in order to avoid performance degradation and SLA violation. A static CPU utilization threshold is used in this case. One of the simplest overload detection phase is based on an idea of setting a CPU utilization threshold distinguishing the non-overload and overload states of the node. When the algorithm is invoked, it compares the current CPU utilization of the node with the defined overloaded threshold. If the threshold is exceeded, the algorithm detects

a node overload.

This process is iteratively repeated for all non-overloaded nodes

3.3.4 VMs placement model

Energy cost is caused by VMs running, VMs migrations and the fixed consumption of starting nodes. The energy use for VMs running is determined by the tasks encapsulated in them, and we can not cut this part through optimizing. The resource size and migration number determine the wastage of energy caused by migration operations. So, the smaller resource size and the smaller migration number, the less energy will be wasted. The energy consumption for running a node is fixed and unmodifiable by algorithm. As a result, we will minimize the number of nodes and the migration operations.

3.3.5 Algorithm statements

Consider a system with an algorithm of centralized decision : it is usually a strategy of client-server / master-slave. A node is responsible for allocation. A node $H_{i,j}$ decides which node $H_{i',j'}(j' \neq j)$ move away its load and is switched off. For the centralized algorithm (client-server) there are several way to manage the list of VMs : (i) a central list, (ii) each node has a list of VMs. In the second approach, each node initializes its list with its list of VMs. In the case when a node $H_{i,j}$ is lightly loaded, its load is migrated to another node $H_{i',j'}$ and it goes in sleep mode. This centralized approaches has several advantages such as a simple implementation and policy update information, and good efficiency (low overhead since few communication occur)

3.3.6 Scenario

Our approach is to develop systems that turn nodes off to save power under lighter load. The key component of our systems is an algorithm that makes load unbalancing decisions by considering both the power and performance implications of turning nodes off. In more detail, the technique periodically considers whether nodes should

be added to or removed from the site (or VMs added or removed from nodes), based on the expected performance and power consumption that would result, and decides how the existing load should be re-distributed in case of a configuration change.

Using migration of VMs as a solution for improving energy efficiency can lead to dynamically consolidate VM on a minimal number of physical nodes according to their current resource requirements. The second phase enables distributing or redistributing the load of some nodes after an addition or removal decision. For a node load ($c_{i,j}$) two thresholds are defined : under-loaded threshold (γ) and over-loaded threshold (ε). The goal is that $c_{i,j}$ verifies the formula $\gamma <= c_{i,j} <= \varepsilon$. The system periodically checks whether there are underloaded nodes ($H_{i,j}^{under}$) or overloaded ($H_{i,j}^{over}$). For each $H_{i,j}^{under}$ the system seeks, first if it exists nodes that can receive load of $H_{i,j}^{under}$. If yes, load is moved to the node and the $H_{i,j}^{under}$ is switched off; if there is no node the $H_{i,j}^{under}$ remains active. Then another check is made on $H_{i,j}^{over}$. The system always tries to reduce the load of these nodes whose load is greater than ε. The proposed approach to dynamic consolidate VM or Distributed Dynamic VM Consolidation consists in splitting the problem into 4 steps :

- **Step 1 :** determining when a node is considered as being overloaded; requiring migration of one or more VMs from this node. If the decision is to add one or more nodes, the algorithm must determine what part of the current load should be sent to the added nodes. Obviously, the load to be migrated should come from nodes undergoing excessive demand for resources.

- **Step 2 :** determining when a node is being underloaded in order to migrate all VMs from this node and switch the node to the sleep mode.

- **Step 3 :** selection of VMs that should be migrated from an overloaded node.

- **Step 4 :** finding a new place for the VMs selected for migration from the overloaded and underloaded nodes. Placement of new requested VM or selected VMs for live migration is a critical issues in virtualized cloud computing and should be performed in a way that ensures QoS, SLA and also prevent to increase total energy consumption by physical nodes in a data cen-

ter, since one of the important requirement for cloud computing environment is providing QoS or SLA for consumers.

This approach has two major advantages compared with traditional VM consolidation algorithms : (1) splitting the problem simplifies the analytic treatment of the steps; and(2)the approach can be implemented in a distributed manner by executing the underload / overload detection and VM selection algorithms on nodes, and the VM placement algorithm on replicated controller nodes.

3.3.7 Algorithm

Our systems use load concentration as a first-class technique, rather than as a remedial technique like in systems that harvest idle workstations or as a management technique for manually excluding a node. The key component of our system is an algorithm that makes load unbalancing with concentration decisions by considering both the total load imposed on the node and the power and performance of different node configurations. In more detail, the algorithm periodically considers whether VMs should be added to or removed from the node, based on the expected performance and power consumption that would result, and decides how the existing load should be re-distributed in case of a configuration change. In the folowing algorithm 1 :

- Line 12 to 14 execution of the step 1,
- Line 8 to 10 execution of the step 2,
- Line 14 to 15 execution of the step 3,
- Line 7 to 16 execution of step 4,

To maximize their Return On Investment (ROI), Cloud providers need to apply energy efficient resource management strategies, such as dynamic VM consolidation with switching idle nodes to power-saving modes. However, such consolidation is not trivial, as it may result in violations of the SLAs negotiated with customers. This chapter proposed novel heuristics for distributed dynamic VM.

Algorithm 1 Pseudo-code for node configuration and anti load-balancing algorithm

```
1: N=number of sites
2: H_{i,j}= node j in site j
3: c_{i,j}= node H_{i,j} load
4:
5: Periodically do
6: for (H_{i,j} in H_i) do
7:   if node support being switched off then
8:     if (c_{i,j} < γ) then
9:       determine nodes to receive c_{i,j} and ask H_{i,j} to migrate c_{i,j} out and ask
         H_{i,j} to turn itself off
10:  else
11:    if (c_{i,j} > ε) then
12:      if (exist switched on node that can receive c_{i,j}) then
13:        move load to this node
14:      else
15:        turn on new nodes
16:        determine the charge to be sent to those added nodes and ask H_{i,j} to
           share c_{i,j} between added nodes
```

3.4 Centralized approach

This section presents an Energy aware clouds scheduling using anti load-balancing algorithm (EACAB). The proposed algorithm works by associating a credit value with each node. The credit of a node depends on its affinity to its jobs, its current workload and its communication behavior. Energy savings are achieved by continuous consolidation of VMs according to current utilization of resources, virtual network topologies established between VMs and thermal state of computing nodes.

The rest of the section is organized as follows. Subsection 3.4.1 discusses related Credit based Anti load-balancing model, followed by the algorithme description presented in Subsection 3.4.2.

3.4.1 Credit based Anti load-balancing model

The algorithm proposed aims at maximizing *Credit* which is a value used when calculating the energy-efficiency of the system behavior.

This Credit algorithm is an adaptation of the *Comet Algorithm* [Chow 2002], a load balancing algorithm. Comet calculates credit for mobile agent. Each software

agent is trying to maximize its own credit by moving between nodes. An agent a_i uses the following formula:

$$C_i = -x_1 w_i + x_2 h_i - x_3 g_i$$

Where w_i : computation load of the node running agent a_i, h_i and g_i : communication load inside and outside agent a_i, and where x_1, x_2 and x_3 are positive float coefficients which constitute dependence assigned to each agent from its creation to estimate its affinity relative to its node. Thus an agent will move to a new node if it result in a lower node load, or if it reduces external communication or if it increases internal communication. This algorithm does not take int account migration cost.

In the same way, the proposed algorithm in this section works by associating a credit value with each node. The credit of a node depends on the node, its current workload, its communications behavior and history of task execution. When a node is under-loaded ($c_{i,j} < \gamma$), all its VMs are migrated to a comparatively more loaded node.

In dynamic load unbalancing schemes, the two most important policies are *selection policy and location policy*. Selection policy concerns the choice of the node to unload. Location policy chooses the destination node of the moved VMs. An important characteristic of selection policy is to prevent the destination node to become overloaded. Also, migration costs must be compensated by the performance improvement.

Each node has its own *Credit*, which is a float value. The higher a node Credits, the higher the chance its VMs to stay at the same node. It is equivalent to say that chances of its VMs to be migrated is lower. The credit of a node increases if:

- Its workload or the number of VMs in the node increases
- Communication between its VMs and other nodes increases
- Its load increases while staying between the under-loaded threshold γ and the over-loaded threshold ε

On the contrary, the credit of a node decreases in the cases below:

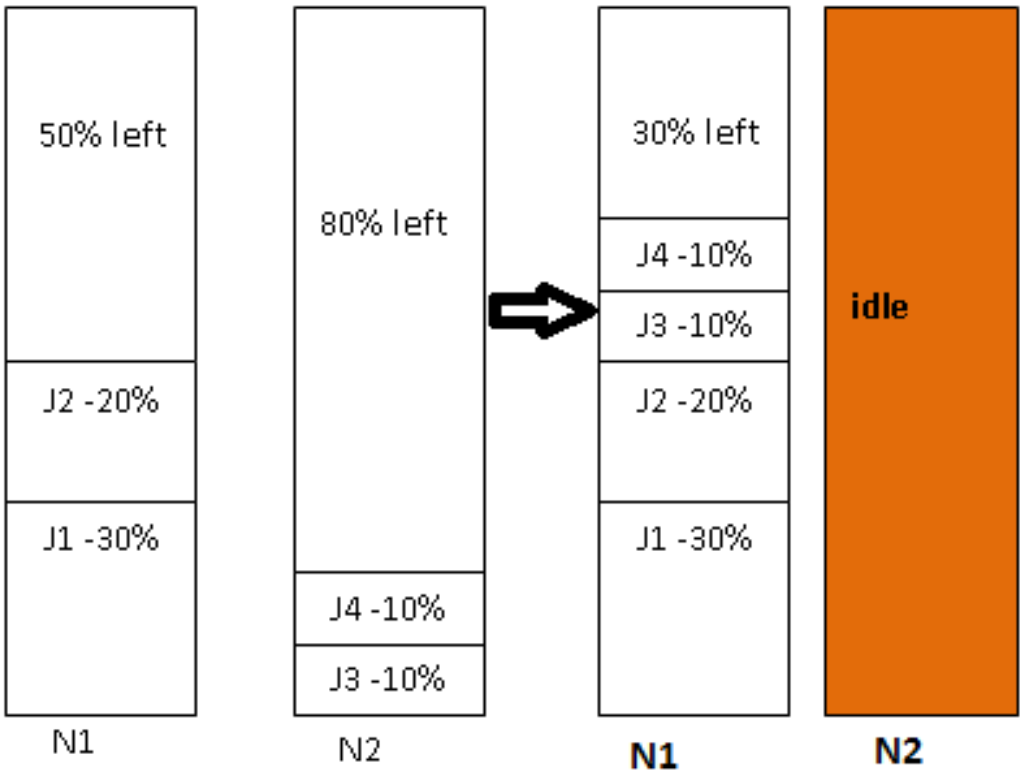

Figure 3.1: Tasks migration.The percentage represents here the processor usage of each VM on the node.

- Its workload or its number of VMs decrease
- It has just sent or received a message from the scheduler which indicates that the node will probably become empty in a short while.

The Credit of a node will be used in the selection policy: the node which credit is the lower is selected for VMs migration(c.f Fig. 3.1). The location policy identifies the remote node with the highest credit which is able to receive the VMs selected by the selection policy without being over-loaded.

However, as shown in figure 3.2, such consolidation is not trivial, as it can result in overloading a node. Also keeping one particular node heavily loaded for a large period of time can lead to a heating point. It has a negative effect on the cooling system compared to several colder points.

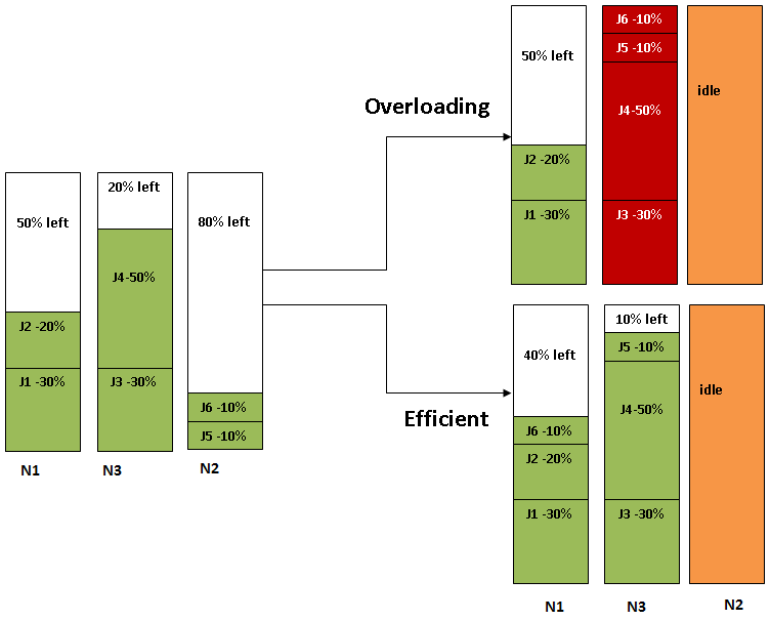

Figure 3.2: Different contexts for a migration. Both cases are possible but the first one creates a hot spot on N_3

In Figure 3.2 we have two nodes N_1 and N_2. Node N_2 is under-loaded because its load is below the underload threshold ($\gamma = 50\%$). In this situation the load N_2 migrates to the node N_1. Then N_2 is switched to sleep mode.

3.4.2 Algorithm

In Comet mobile agents move between nodes according to their affinities (credit) to achieve load balancing. Here we work with VMs which migrate depending on the load of the node. We apply the Credit concept to the migration of VMs. EACAB algorithm is based on the technique of calculating credit ($\sigma_{i,j}$) of each node j (in site i) by the same method of Comet[Jeon 2010]. In EACAB, the formula is then:

$$\sigma_{i,j} = c_{i,j} - t_{i,j}s_{i,j} + \varepsilon - \gamma$$

$$t_{i,j} = (100 * (\varepsilon - c_{i,j}) * (c_{i,j} - \gamma) * (\lambda * c_{i,j} + \delta))/s_{i,j}$$

λ and δ are two constants respectively 10 and 20. The values 10 and 20 were

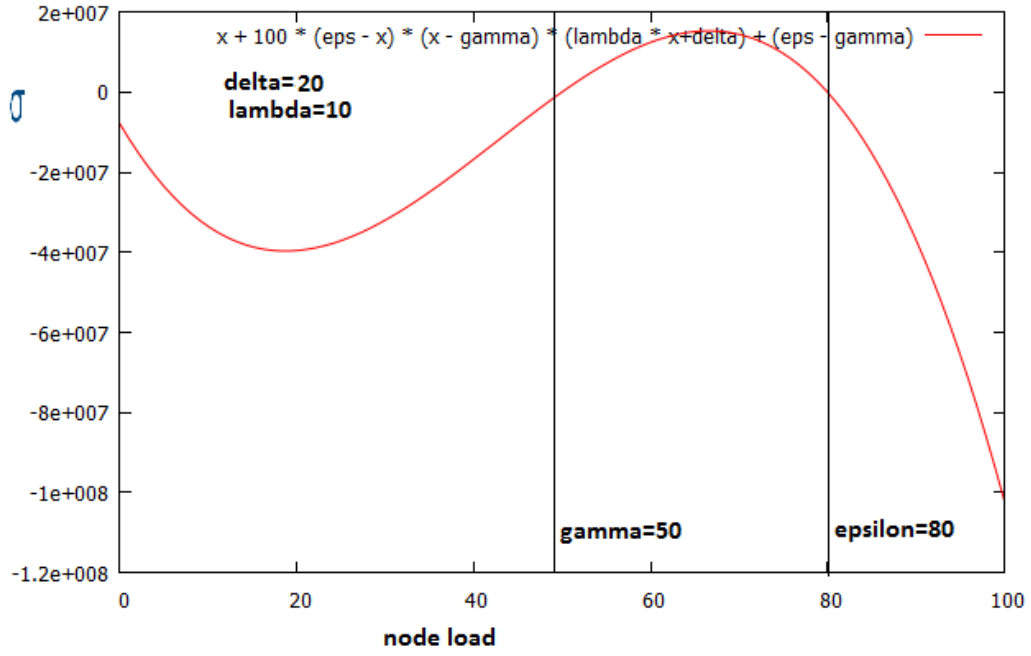

Figure 3.3: Credit of node. $\gamma = 50\%$ and $\varepsilon = 80\%$

chosen in order to favor the avoidance of hotspots (overload) compared to avoiding underload nodes. Indeed, the $\sigma_{i,j}$ value is used for sorting the nodes to consider in the following algorithm 2. A smaller value of $\sigma_{i,j}$ leads a node j in site i o be considered first, and actions are firstly held on this one. On Figure 3.3, one can see that overloads leads to smaller values than underloads. Other values than 10 and 20 would lead to different behaviour of the algorithm: It could favor underload versus overload nodes first, for instance. In the experiment part, we only considered the previous scenario. Were $c_{i,j}$ is the actual load of node j in site i, $r_{i,j}$ is its requested computation load, $s_{i,j}$ is its VM satisfaction, and γ and ε are respectively under-load and over-load threshold. $t_{i,j}$ is the influence of the communications on the behavior of the node.

EACAB provides task scheduling strategy, which dynamically migrate tasks among computing nodes, transferring tasks from underloaded nodes to loaded but not overloaded nodes. It balances load of computing nodes as far as possible in order to reduce program running time. The decision making algorithm behaves

globally as follows:

- If $\sigma_{i,j} < 0$, the node j of site i is over-loaded or under-loaded.
- If $c_{i,j} > \varepsilon$, the node j of site i is over-loaded
- If $c_{i,j} < \gamma$, the node j of site i is under-loaded

This algorithm is described in Algorithm 2. For the sake of simplicity, corner cases such as all nodes over-loaded are not included. Selection policies take into account credits and migration cost. The selected node (node j' in site i') is the one with the minimun $\sigma_{i',j'}$ weighed by the migration cost between the current position of the VM and the potential node. If $\tau_{i,j,i',j'}$ is the migration cost between the node j in site i and the node j' in site i', the selected node is the one that minimize :

$$\sigma_{i',j'} . \frac{\tau_{i,j,i',j'}}{Max_{i",j"}(\tau_{i,j,i",j"})} \tag{3.1}$$

Algorithm 2 Energy aware clouds scheduling using anti load-balancing algorithm for each node $H_{i,j}$(EACAB)

1: Calculate $c_{i,j}$, $\sigma_{i,j}$ *// Load, credit of node j in site i*
2: Sort in ascending order other nodes $(\forall (i,j) \neq (i',j'))$ according to the value of their credit (all, but $H_{i,j}$)
3: **if** $(c_{i,j} < \gamma)$ **then**
4: **for** (all nodes j' in all sites i' sorted by their credits) **do**
5: **if** $(H_{i,j} \neq H_{i',j'})$ and $(c_{i',j'} \geq \gamma)$ and $(c_{i,j} + c_{i',j'} < \varepsilon)$ **then**
6: Add $H_{i',j'}$ to potential destination set *Potential*
7: Migrate all $VM_{i,j}$ from $H_{i,j}$ to the element with lower credits weighted by migration cost in *Potential* (see equation 3.1)
8: **else**
9: **while** $(c_{i,j} > \varepsilon)$ **do**
10: *// $c_{i,j}$ and $\sigma_{i,j}$ have changed after migration of $VM_{i,j,k}$*
11: Calculate $c_{i,j}$
12: Calculate $\sigma_{i,j}$
13: Calculate $l_{i,j}^{min}$ *// load of the lightest task*
14: Let $VM_{i,j,k}$ the task with $l_{i,j,k}^{min} = l_{i,j,k}$
15: **for** (all nodes j' in all sites i' sorted by their credits) **do**
16: **if** (($H_{i,j} \neq H_{i',j'}$) and $(c_{i',j'} + l_{i,j}^{min} < \varepsilon)$) **then**
17: Migrate $VM_{i,j,k}$ from $H_{i,j}$ to $H_{i',j'}$.

This migration algorithm's is composed of two parts. In the first part (line 6 to 13), it checks for each node j if the load is below the threshold. If this is the case, it locates the node j' that will receive all tasks of node j. The second part (line 15 to 25) manages hotspots. To reduce the load of an overloaded node, it begins to migrate the lightest task. Selection policy will choose the task that will stay the longest on the node. Policy of localization will then identify the node that will receive the task without exceeding its capacities (ie. its load after migration will still be under ε). So this node will be the new destination of the task.

3.5 Decentralized approach

The dynamic scheduling of a large number of VMs as part of a large distributed infrastructure is subject to important and hard scalability problems that become even worse when VM image transfers have to be managed. Consequently, most current algorithm schedule VMs statically using a centralized control strategy. An analysis of virtual infrastructure managers (VIMs) reveals that most of them still schedule VMs in a static manner [Hoffa 2008] [Nurmi 2009]. Indeed, the advanced solutions that promote the implementation of dynamic VM scheduling [Hermenier 2009] [Sotomayor 2009a] are subject to strong limitations regarding scalability, reactivity, and fault-tolerance aspects. Although scheduling is a non deterministic polynomial time-hard problem, most of these proposals rely on a master/worker design using a single daemon that is in charge of performing the different phases: monitoring all resources, computing the schedule, and applying the reconfiguration. The choice of a single master node leads to several problems. First, during the computation and the application of a schedule, this manager does not enforce QoS properties anymore, and thus cannot react quickly to QoS violations. Second, because the manipulation of VMs is costly, the time needed to apply a new schedule is particularly important: the longer the reconfiguration process, the higher the risk that the schedule may be outdated, due to the workload fluctuations, when it is eventually applied. Finally, a single master node can lead to well-known fault-tolerance issues or a node can be overloaded; a subgroup of VMs may be temporarily isolated from

the master node in case of a network disconnection; QoS properties may not be ensured any more if the master node crashes. Some nodes could be overloaded which increases energy consumed. Even if a better design of the master can help to separate each phase in order to resolve some of these issues (a multi-threaded model, for instance, can help to track workload changes so that a scheduling process can be stopped if need be), a centralized approach will always be subject to scalability, reactivity, and fault-tolerance issues.

In this thesis, we investigate whether a more decentralized approach can tackle the aforementioned limitations. Indeed, scheduling takes less time if the work is distributed among several nodes, and the failure of a node does not impede scheduling strongly any more. Several proposals have been made precisely to distribute dynamic VM management [Feller 2010] [Yazir 2010]. However, the resulting prototypes are still partially centralized. Firstly, at least one node has access to a global view of the system. Secondly, several virtual infrastructure managers (VIMs) consider all nodes for scheduling, which limits scalability. Thirdly, several VIMs still rely on service nodes, potential single points of failure. [Quesnel 2012] introduce distributed VM scheduler (DVMS), a VIM that schedules and manages VMs cooperatively and dynamically in distributed systems. Author designed it to be non-predictive and event-driven, to work with partial views of the system, without any potential single points of failure. Our VIM thus has the same characteristics and is more reactive, more scalable, and more tolerant to nodes crashes or network disconnections. In this section, we present distributed VM scheduler, an algorithm that enables VMs to be scheduled cooperatively and dynamically in large-scale distributed systems. We describe, in particular, how several VM reconfigurations can be dynamically calculated in parallel and applied simultaneously. Reconfigurations are enabled by partitioning the system (i.e., nodes and VMs) on the fly. Partitions are created with a minimum of resources necessary to find a solution to the reconfiguration problem.

Distributed VM consolidation algorithms enable the natural scaling of the system when new nodes are added, which is essential for large-scale Cloud providers. An illustration of the importance of scalability is the fact that Rackspace, a well

known IaaS provider, has increased the total node count in the second quarter of 2012 to 84,978 up from 82,438 nodes at the end of the first quarter leading to 8% per year. Another benefit of making VM consolidation algorithms distributed is the improved fault tolerance by eliminating single points of failure: even if a compute or controller node fails, it would not render the whole system inoperable. The proposed heuristics efficiently implement dynamic VM consolidation in a distributed manner according to the current utilization of resources applying live migration, switching idle nodes to the sleep mode, and thus, minimizing energy consumption. The proposed approach can effectively adhere to strict QoS requirements, as well as handle multi-core CPU architectures, heterogeneous infrastructure and heterogeneous VMs.

The key contributions of this section are the following.

- The introduction of a distributed approach to energy and performance efficient dynamic VM consolidation.

- Novel heuristics for the problem of energy and performance efficient dynamic VM consolidation following the introduced distributed approach.

The remainder of this section is organized as follows. First we will present the decentralized dynamic scheduling anti load-balancing Algorithm for Clouds (CSAAC). Then we will present the algorithm description and finally conclude the section.

3.5.1 The Cooperative scheduling Anti load-balancing Algorithm for cloud

The approach use a multi-phase decentralized scheduling solution. It is comprised of migration phase, the job submission phase responsible for job dissemination, as well as the dynamic scheduling phase responsible for iterative scheduling improvement. Furthermore, the decentralized approach is a collection of implementation-independent interfaces and heuristics, which are used to facilitate job scheduling across decentralized distributed nodes.

3.5.1.1 Algorithm statements

There are in total N sites and in each site a set H_i of nodes distributed in a cloud data center system with the same start time U, where in some of the nodes have local job submission. Each time when one node (initiator) attempts to (re)assign a task to another node (or the same node) for execution, the initiator is called the requester node, and the node receiving such a request is called the responder node. Each task $VM_{i,j,k}$ $i \in [1, 2, ..., N]$ and $j \in [1, 2, ..., nH_i]$ is sent from node $H_{i,j}$. The decentralized approach is comprised of two phases, namely the job submission phase and the dynamic scheduling phase, which work together to ensure both a rapid job distribution and an optimized rescheduling effect. Figure 3.4 shows an example of execution of a job in a decentralized and cooperative environment. After job submission a request for execution is sent to all nodes (using flooding technique). Nodes that fulfill the job requirements send responses to become a candidate. The process follows at least one of the phases of the algorithm.

- **Job submission phase.** This phase is the first phase of the algorithm. Each time when a node j, receives a $VM_{i,j,k}$ submitted by its local user, node j behaves as a requester node $H^{req}_{i,j}$ and generates a request message $M^{req}_{i,j,k}$ for $VM_{i,j,k}$. VM characteristic information including estimated execution time $D_{i,j,k}$ and requested amount of Processor Elements (PEs) $VM^{pe}_{i,j,k}$ will be appended to the generated request message. Afterwards, request message $M^{req}_{i,j,k}$ is replicated and disseminated to each of the discovered remote nodes asking for the job delegation possibilities. All nodes receiving the job delegation request message $M^{req}_{i,j,k}$, including the requester node j itself, are considered as responder nodes. Each responder node $H^{resp}_{i',j'}$ needs to send an accept message $M^{ack}_{i',j',k}$ to decide whether the node j' is able and willing to execute the received $VM_{i,j,k}$. A $M^{req}_{i,j,k}$ takes various factors, such the current state of the node which allows it to execute the VM using the most amount of energy relative to other nodes. If yes, each candidate, considered as responder node, computes an estimated completion time according to its current scheduling, mainly the energy consumed and resource status and delivers the information

by means of an accept message $M^{ack}_{i',j',k}$. In addition, the estimated response time and the estimated energy consumed if $VM_{i,j,k}$ is executed by node j' in site i' are also appended to the generated accept message, which can be utilized by the requester node for responder node evaluation and selection. Each time a request message $M^{req}_{i,j,k}$ is generated by the requester node and disseminated to contactable remote nodes, the requester node waits and collects all received $M^{ack}_{i',j',k}$, generates an assign message $M^{ass}_{i',j',k}$ and send it to a proper remote node $H^{ass}_{i',j'}$(assignee node) to which to delegate the $VM_{i,j,k}$.

An arbitrary node i in site i, due to the effect of the job submission phase, has received a set of jobs from either local users' submissions or remote nodes' delegations. A rescheduling process helps this approach to adapt to the changes of both underlying resources and arriving jobs.

- **Dynamic scheduling phase.** As a federated resource infrastructure, a Cloud data center is a naturally dynamic environment where resources contributed by different sites can join and leave through time; furthermore, issues like unexpected network delay, resource overhead, and VM status modification make the status of a cloud even more unpredictable. This phase solves some problems related to the uninterrupted changing data center infrastructure during VM submission phase. It allows for example a redistribution decision for a VM that is in a long tail and thus a node can not be executed instantly. Suppose the arbitrary node $H_{i",j"}$ finds that many customers applications (VMs) wait to be served such selected customers will be sent to others nodes (rescheduling) to improve their own VM makespans and the resource utilization, therefore the energy consumed of the overall cloud data center. The number of to-reschedule VMs is decided by considering the status of node $H_{i",j"}$ itself, such as length of local waiting customers and current node overhead. Afterwards, the algorithm finds a set of contactable remote nodes for VM rescheduling and re-allocation. $H_{i",j"}$ sends an inform message $M^{info}_{i",j",k"}$ for each to-schedule $VM_{i",j",k"}$, and disseminates those inform messages to all remote nodes discovered for negotiating VM rescheduling possibilities.

Each generated inform message $M^{info}_{i",j",k"}$, contains firstly the same information as the aforementioned request message, including estimated execution time $D_{i",j",k"}$, requested amount of PEs $VM^{pe}_{i",j",k"}$, and job characteristic profile. Furthermore, each inform message also includes the already made schedule for $VM_{i",j",k"}$, on the current node $H_{i",j"}$, i.e., the estimated VM finish time and the estimated energy consumed, which will be used for offer comparison by the contacted remote nodes later. It is noteworthy that the algorithm can be triggered by customized events as well depending on each node's local setting.

The selection of the node that will receive the task is the same as the submission phase except that the initiator is no longer a candidate node.

3.5.1.2 Scenario

Assuming that a cloud is comprised of interconnected nodes $H_{i,j}$, $i \in [1, 2, ..., N]$ and $j \in [1, 2, ..., nH_i]$, $VM_{i,j,k}$ is submitted to node $H_{i,j}$. The CSAAC then leads to the following phases and steps for job assignment and dynamic scheduling as shown in figure 3.4.

When a new VM arrives there is a VM submission phase.

- VM **Submission Phase.**
  - **Step 1 :** $VM_{i,jk}$ is submitted to node $H^{req}_{i,j}$ together with its characteristic data.
  - **Step 2 :** The initiator node $H^{req}_{i,j}$ generates a request message according to the retrieved execution requirement from $VM_{i,j,k}$, and broadcasts such a message to the cloud by the employed lightweight decentralized information system.
  - **Step 3 :** The initiator node $H^{req}_{i,j}$ then waits for some time to receive returned accept messages from other nodes of the data center.
  - **Step 4 :** Nodes receiving the broadcasted request message check if the required VM profile can be matched by the local resources.

- **Step 5 :** If yes, each candidate node computes an estimated VM response time according to its current scheduling, their loads (load increases while staying between the under-loaded threshold γ and the over-loaded threshold ε) and resource status, and sends the information back to the initiator node $H_{i,j}^{req}$ by means of an accept message.
- **Step 6 :** The initiator node $H_{i,j}^{req}$ evaluates received accept messages and selects the candidate node which fulfilled requirements and energy minimized. The selected node, referred to the assignee node, is assigned the VM by means of an assign message. A node $H_{i',j'}^{ass}$ is selected as the candidate node for $VM_{i,j,k}$'s delegation.

Regularly the system tries to schedule the waiting VM.

- **Dynamic Scheduling Phase.**
  - **Step 7 :** The assignee node $H_{i',j'}^{ass}$ takes the assigned $VM_{i,j,k}$ and puts it into its local list of VMs.
  - **Step 8 :** The assignee node $H_{i',j'}^{ass}$ periodically picks jobs from its local list of VMs, which have a long enough waiting time and have not been selected recently. Afterward, for each selected VM, the assignee node $H_{i',j'}^{ass}$ generates an inform message, which contains both VM estimated completion time and energy estimated for VM execution. In our case, $VM_{i,j,k}$ is selected with an assumed long waiting time.
  - **Step 9 :** The inform message of $VM_{i,j,k}$ is sent over the network using the employed low-overhead walking protocol (The walk starts at an initial node and at each step selects randomly a minimum number of its neighbors).
  - **Step 10 :** Nodes receiving the aforementioned inform message check if the local resource and scheduling status could match the requirement of $VM_{i,j,k}$; furthermore, they also need to check whether the estimated VM response time is short enough when compared to $VM_{i,j,k}$'s current response time upon the assignee node $H_{i',j'}^{ass}$.

- **Step 11 :** If the evaluation result from above step is positive, an accept message will be generated and delivered to the assignee node $H^{ass}_{i',j'}$.
- **Step 12 :** The assignee node $H^{ass}_{i',j'}$ evaluates the received accept messages according to the promised VM response time. As a result, node $H^{re-ass}_{i',j',k}$ is selected, and $VM_{i,j,k}$ is re-assigned by means of an assign message.
- **Step 13 :** To enable tracking of VMs for the purpose of node crash tolerance, each VM re-assignment is logged and notified to the initiator node $H^{req}_{i,j}$.
- **Step 14 :** To enable node weighting for future scheduling, VM completion status is sent back to the original assignee node $H^{ass}_{i',j'}$ and initiator node $H^{req}_{i,j}$.
- **Step 15 :** The final VM execution result is sent back to the initiator node $H^{req}_{i,j}$.

Regularly the scheduling restarts to optimize the metrics.

- **Migration Phase.** This phase comprises two steps :
  - **Step 1 :** For a node $H_{i,j}$ if $c_{i,j} > \varepsilon$ the node is considered as being overloaded; requiring migration of one or more VMs from this node. The algorithm must determine what part of the current load should be sent to other nodes. The algorithm seeks, for all nodes in all sites, one or more nodes, whose charge $\gamma < c_{i',j'} < \varepsilon$, which can receive the VM to migrate. If the node is found, the execution of the VM is stopped at the source node $H_{i,j}$ and continue to the destinations nodes. If the decision is to add one or more nodes, the algorithm must determine what part of the current load should be sent to the added nodes. Obviously, the load to be migrated should come from nodes undergoing excessive demand for resources.
  - **Step 2 :** if $c_{i,j} < \gamma$ the node is considered as being underloaded; requiring migration of all VMs from this node. The algorithm seeks, for

all nodes in all sites, one or more nodes whose charge $\gamma < c_{i',j'} < \varepsilon$ which can receive the load to migrate. If the node is found, the execution of all VMs is stopped at the source node $H_{i,j}$ and continue to the destinations nodes.

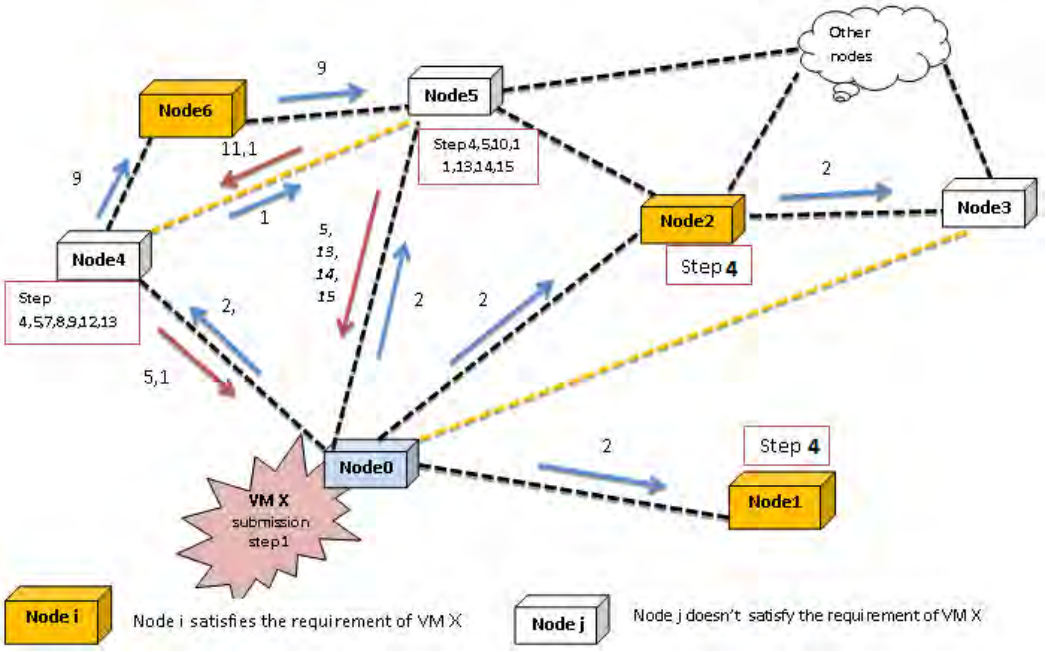

Figure 3.4: Scenario of decentralized approach.

3.5.2 Algorithms

As a decentralized scheduling solution, the Energy Efficient Decentralized Scheduling for Cloud proposed algorithm adopts the promised VM response time and the node load as main criterions to evaluate the nodes' capabilities. Participating nodes need to calculate their actual load and estimated response time for a concerned VM and bid for the VM delegation using the calculated and promised VM response time.

Each responder node computes an estimated completion time according to its current scheduling and resource status, calculates the necessary energy, and delivers the information by means of an accept message. In addition, the node load is also

added to the generated message, which can be utilized by the requester node for responder node evaluation and selection.

The selected node is the best candidate node based on several parameters, such as the promised time to complete, energy consumed, the node load between under-load threshold and over-load threshold, node weight due to historical interaction records, etc. Furthermore, during the execution of the VMs, all the time the system verifies if there are underloaded or overloaded nodes.

A self-healing is designed similar to the intelligent feedback loop supports not only scheduling but also re-scheduling activities. It enables CSAAC with self-healing capabilities to allow VMs that wait a long time in a node to be re-scheduled. The dynamic scheduling phase, but also the posibility to switch off or switch on nodes, and especially the cooperation between nodes help to maintain system efficiency. These properties of CSAAC algorithm constitue its self-healing system.

CSAAC provides task scheduling strategy, which dynamically migrate VMs among computing nodes, transferring VMs from underloaded nodes to loaded but not overloaded nodes. It balances load of computing nodes as far as possible in order to reduce program running time.

The decision making algorithm behaves globally as follows:

- If total Vm load on the node j of site $i > \varepsilon$,node is over-loaded
- If total Vm load on the node j of site $i < \gamma$, node is under-loaded

This algorithm is described in Algorithm 6. For the sake of simplicity, corner cases such as all nodes over-loaded are not included. Selection policies take into account migration cost. The selected node (node j' in site i') is the one with the minimun energy consumed with best execution time, weighed by the migration cost between the current position of the VM and the potential node.

To reduce the load of an overloaded node, it begins to migrate the slowest task. Selection policy will choose the task that will stay the longest on the node.

This migration algorithm's goal is to minimize the energy. During the execution of the task it may happen that a node is overloaded. We decided in this case to migrate VMs whose remaining execution time is greater. The algorithm 4 can

Algorithm 3 Find VM with longest remaining waiting time : FindVmWait

Input : $VM_{i,j}$ // *set of VMs*
Output : $VM_{i,j,k}$ // *$VM_{i,j,k}$ in node j of site i*
$D = 0$
for $VM_{i,j,k}$ in $VM_{i,j}$ **do**
  if $(D^{W}_{i,j,k} > D)$ **then**
    // *$D^{W}_{i,j,k}$ is the waiting time of VM k in node j of site i*
    $D = D^{W}_{i,j,k}$
    $k' = k$
RETURN $VM_{i,j,k'}$

Algorithm 4 Find VM with longest remaining execution time : FindVmExec

Input : $VM_{i,j}$ // *set of VMs*
Output : $VM_{i,j,k}$ // *Vm k in node j of site i*
$D = 0$
for $VM_{i,j,k}$ in $VM_{i,j}$ **do**
  // *$D^{Exec}_{i,j,k}$ is the execution time of VM k in node j of site i*
  if $D^{Exec}_{i,j,k} > D$ **then**
    $D = D^{Exec}_{i,j,k}$
    $k' = k$
RETURN $VM_{i,j,k'}$

Algorithm 5 Migration of VM based on sorted nodes : VerifyLoad

Calculate $c_{i,j}$ // *Load of node j in node i*
if $(c_{i,j} < \gamma)$ **then**
  // *In case $H_{i,j}$ is under-loaded*
  for ($H_{i',j'}$ in all nodes j' in all sites i' sorted by their P^{max}) **do**
    if $H_{i,j} \neq H_{i',j'}$ and $(c_{i',j'} \geq \gamma)$ and $(c_{i,j} + c_{i',j'} < \varepsilon)$ **then**
      Add $H_{i',j'}$ to potential destination set *Potential*
    Migrate all VM from $H_{i,j}$ to the element in Potential set with minimum migration cost
else
  // *$H_{i,j}$ can be over-loaded*
  while $(c_{i,j} > \varepsilon)$ **do**
    Calculate $c_{i,j}$
    // *In case $H_{i,j}$ is over-loaded*
    $VM_{i,j,k}$=$FindVMExec(VM_{i,j})$
    for ($H_{i',j'}$ in all nodes j' in all sites i' sorted by the value of their maximum power) **do**
      if $((H_{i,j} \neq H_{i',j'})$ and $(c_{i',j'} + l_{i,j,k} < \varepsilon)$ **then**
        Migrate $VM_{i,j,k}$ from $H_{i,j}$ to $H_{i',j'}$.

Algorithm 6 Cooperative scheduling Anti load-balancing Algorithm for cloud (CSAAC)

1: *Regularly*
2: γ, under-load threshold
3: ε, over-load threshold
4: **for** $(i = 1; i <= N)$ **do**
5: **for** $(j = 1; j <= nH_i)$ **do**
6: Calculate $c_{i,j}$
7: VerifyLoad()
8: **if** $(notemptyVM_{i,j}^{wait})$ **then**
9: $VM_{i,j,k"}$=$FindVMWait(VM_{i,j}^{wait})$
10: **for** ($H_{i',j'}$ in all nodes j' in all sites i' sorted by the value of their P^{max}) **do**
11: **if** $((H_{i,j} \neq H_{i',j'})$ and $(c_{i',j'} + l_{i,j,k"} < \varepsilon)$ **then**
12: Migrate $VM_{i,j,k"}$ from $H_{i,j}$ to $H_{i',j'}$.

find this VM. The algorithm 5 is composed of two parts. The first part (line 6 to 13), checks for underloaded node (load is less than γ) and migrates their load. The second part (line 14 to 25) manages hotspots (ie overloaded nodes). The algorithm 6 uses underloaded and overloaded detections and avoids underloaded and overloaded nodes. The last part (line 4 to) corresponds the dynamic scheduling phase. This migration algorithm's goal is to minimize the energy. During the execution of the task it may happen that a node is overloaded. We decided in this case to migrate VMs whose execution time remaining is greater.

Policy of localization will then identify the node that will receive the VM without exceeding its capacities (ie. its load after migration will still be under ε). So this node will be the new destination of the VM.

The algorithm 3 is used for the dynamic scheduling phase. It allows to find the VMs that has been waiting a long time on the list of VMs of a node.

Selection policies take into account migration cost. The selected node (node j' in site i') is the one with the minimun energy consumed with best execution time, weighed by the migration cost between the current position of the VM and the potential node. To reduce the load of an overloaded node, it begins to migrate the VM with the longest remaining execution time. Selection policy will choose the VM that will stay the longest on the node. Suppose that at time t1 a candidate

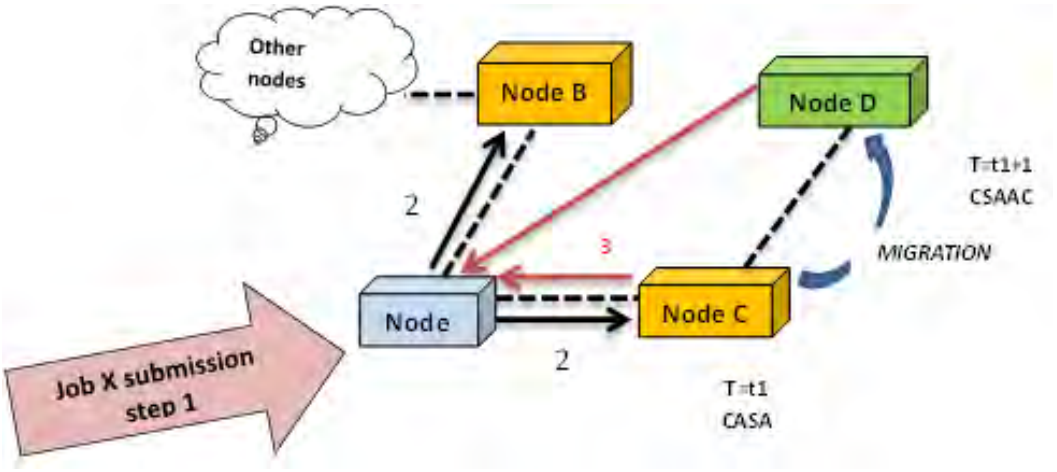

Figure 3.5: VM allocation with CSAAC compared to CASA. CSAAC can decide to migrate from C to D if D is more efficient or if can lead to switch off C for example

node C to the execution of a VM is selected (see Figures 3.5). With CASA the execution of the job is performed by the node C. With CSAAC, at time t1 +1, the VM migrates to the node D that meets the requirements and whose load is greater than that of node C. If node C had no load it is more benefic in terms of energy saving to not activate the node C if another node can execute loaded. Figures 3.5 and 3.6 show that CSAAC can produce better execution time due to migration.

3.5.3 Analysis of the algorithm

This section presents a novel decentralized dynamic scheduling approach named the Cooperative scheduling Anti load-balancing Algorithm for cloud (CSAAC), which is inspired by the motivation of enabling cloud scheduling for the scope of the overall cloud, instead of each single node. In contrast to conventional cloud scheduling solutions, this algorithm is designed to deliver relevant scheduling events, such as VM submission and scheduled information, to as many neighboring remote nodes as possible. Moreover, the algorithm enables the possibility of dynamic rescheduling in order to adapt to cloud data center characteristics such as instantaneity and volatility. The Cooperative scheduling Anti load-balancing Algorithm also uses

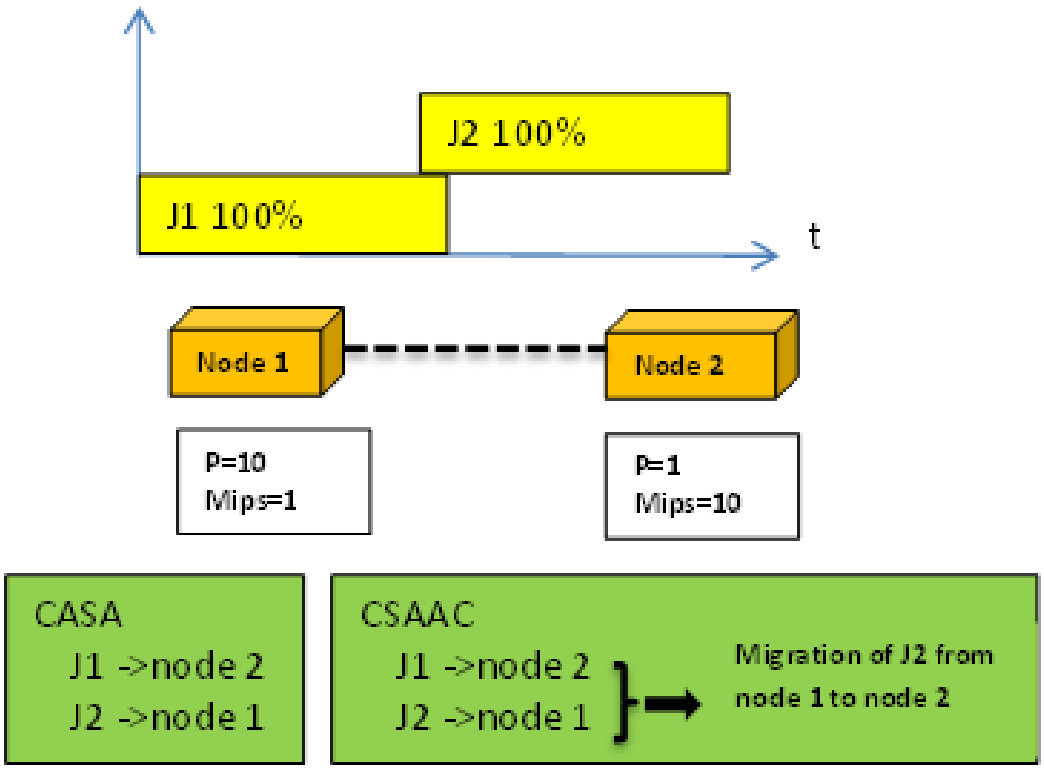

Figure 3.6: How migrations can reduce makespan. CSAAC compared to CASA. CSAAC can migrate J_2 when J_1 finishes to reduce makespan and energy consumed.

consolidation in order to minimize energy consumed. The Cooperative scheduling Anti load-balancing Algorithm is comprised of three phases, namely the VM submission phase, dynamic rescheduling phase and migration phase. Furthermore, the three phases of the Cooperative scheduling Anti load-balancing Algorithm are composed of sub-scheduling algorithms, in which the VM response time and the energy are used as critical criteria to evaluate the nergy consumed, the performance of scheduling and rescheduling decisions. With regard to no detailed information from participating nodes, such as length of local list of VMs and number of processing elements, is required during the interactions between the metaschedulers and local schedulers, the CSAAC is able to work together with different kinds of local scheduling algorithms.

3.6 Summary

It has been shown that good management of computing resources can lead to a significant reduction of energy consumption by a system, while still meeting performance requirements. One of the significant advancements that have facilitated the progress in managing nodes is the implementation consilidations techniques with switching idle nodes to power-saving modes. These technologies have enabled the reduction of energy consumption by reducing the number of switched on nodes. In this chapter, an algorithm, ALBA have been proposed for the problem of node underload and overload detection as a part of dynamic VM consolidation. We have presented first an algorithm ALBA that use consilidation techniques. Then, two centralized and decentralized approaches with two algorithms based on ALBA : EACAB and CSAAC. The two algorithms are based on the principle of *Anti load-balancing*. EACAB is a credit-based algorithm which is centralized. Each node can exchange VM, based on their respective credits. The algorithm CSAAC is a decentralized cooperative algorithm. It takes into account energy and aim to provide energy-efficiency improvement.

Experiments and results of these two algorithms are presented in the next chapter.

Chapter 4

Experiment results

Contents

4.1 Introduction

The aim of experiments is to verify that algorithms EACAB and CSAAC based on ALBA are capable of obtaining energy-efficient schedules. The proposed System Setup is presented in section 4.3, followed by an evaluation and analysis of the obtained experimental results in Sections 4.4, 4.5 and 4.6. The chapter is concluded with Section 4.7 providing a summary of results and contributions. This chapter starts with a description of the two versions of the simulator developped Enersim.

4.2 Simulators

There have been many studies using simulation techniques to investigate behaviour of large scale distributed systems such as the GridSim and CloudSim projects at the

University of Melbourne and Alea developped by Dalibor et al [Klusáček 2010]. This section investigates into extending these techniques to the behaviour of scheduling algorithms in a cloud environment and proposes a new simulation tool that can be used for simulating decentralized environment.

There are several desirable features of a tool similar to the one described such as :

- Ease of setting up and executing a simulation experiment and of using user interface which is intuitive yet comprehensive but also graphical;
- Ability to establish a simulation with high flexibility and configurability. During simulation, it is important to be able to enter and change parameters quickly and easily and repeat simulations;
- Repeating experiments is a very important requirement of a simulator as the same experiment with the same parameters should produce same results each time the simulation is executed;
- Easily exploitable outputs are highly desirable for producing gaphical output quickly;
- Ease of extension.

To evaluate our proposed approaches we need a simulator which has the following capabilities : *(1)it has to to take into account two metrics which are energy and makespan; (2)basics technics required are virtualisation, migration and the possibility of switching off/on physical nodes; (3)users must be able to enter and change algorithms quickly and easily to perform simulations and (4)it must simulate a decentralized cooperative cloud scheduler taking into account cloud scheduler interoperation, that implements a dynamic resource discovery approach on decentralized network.*

Table 4.1 summarizes the literature about various existing simulators. A remark that might raise here is the lack of a simulator that can be considered as *standard*. The simulator adapted to our work does not exist. In effect, for those who take into

account the energy, most use a centralized appproach. Most of evaluated solutions in cloud are usually based on centralized systems which is a limit when addressing large scale distributed systems. Despite efforts in this direction and compared to low level network simulation, cloud simulation does not appear to have reached the stage aging in order to qualify for having a referent simulator recognized by all. We will use our cloud-based simulator. It takes into account energy, manage virtual machines for the decentralized approach, the simulator will be able to simulate cooperative scheduling algorithms.

Taking into account some deficiencies noted in these solutions, we will try in the following section to propose a solution of anti load-balancing leading to energy savings in distributed systems.

Simulator	**Advantage**	**Disavantage**
PeerSim [Montresor 2009]	Scaling: 10^6 nodes in a cyclic mode	No model of physical network, Light documentation
GridSim [Buyya 2002b]	Network, grid, jobs, resources modeled, good Documentation	Energy management not taken into account
CloudSim [Calheiros 2011]	Energy management taken into account, Inherits GridSim, manages VM	Manages only the energy as a money cost
PlanetSim [García 2005]	Scaling: 10^5 nodes	No statistics and no model of physical network
SimGrid [Casanova 2008]	Scalability, network model, statistics supported	Energy management not taken into account, light Documentation
Dcworm [Kurowski 2013]	Scalability, network model, statistics supported	Energy management taken into account, light Documentation, centralized scheduler
MagateSimulator [Huang 2009]	Scalability, network model, statistics supported	Energy management not taken into account, light Documentation, decentralized scheduler
Alea 3 [Klusáček 2010]	Scalability, network model, statistics supported	Energy management not taken into account, light Documentation, centralized scheduler

Table 4.1: Characteristics of a selection of existing simulators.

- **Enersim 1.** In order to evaluate algorithms for centralized approach, we implemented the centralized simulator Enersim 1. This simulator extends ALEA simulator [Klusáček 2008] and CloudSim[Calheiros 2011].

  Alea is a centralized scheduler allowing to apply and compare various scheduling algorithms for cloud systems. The solution consists of the scheduler entity and other supporting classes which extend the original basic functionality of GridSim. Additional data structures are used to maintain information about the resource status, the objective functions and for collection and visualization of the simulation results. Several new scheduling algorithms and objective functions were included as well as the support of additional job and machine characteristics such as the machine usage, the average slowdown or the average response time are included.

  CloudSim is used to model, simulate and make experiments on designing Cloud computing infrastructures. It can simulate the operation of a data center where several activities such as simulating hardware and VM description but also its creation and its destruction, the management of VM including the allocation of physical hardware resources to each VM, take place. A significant activity is simulated by CloudSim is the execution of user programs or requests on the VMs. CloudSim provides a virtualization engine with extensive features for modelling the creation and life cycle management of virtual engines in a data center. Its framework is built on top of GridSim framework also developed by the GRIDS laboratory. Most of these features are directly used in Enersim 1. The main limits of CloudSim are a lack of cooperation between entities and nodes which does not take into account energy.

  The underlying first layer is the discrete-event simulation tool SimJava [Howell 1998]. SimJava provides a set of primitive APIs for developing and analyzing discrete-event simulators. GridSim is built on the top of SimJava. GridSim provides the modeling of different kinds of essential grid components, such as grid jobs with various parameters,heterogeneous grid resources, and grid users. On the top of GridSim we use Alea and CloudSim.

This work concentrates on the design of a system intended for study of advanced scheduling techniques for planning VMs in data center environment. The solution is able to deal with common problems of VM scheduling in data center like heterogeneous resources and OS, and dynamic runtime changes such as switching off of nodes and creation of new VMs. Enersim 1 simulator is based on the latest Alea and CloudSim simulators which we extended to provide a simulation environment to evaluate our research. We implemented an experimental centralised scheduler which uses advanced scheduling techniques for schedule generation and two thresholds (underload and overload thresholds). By now dynamic first and energy-aware clouds scheduling using anti load-balancing algorithms were tested.

The scheduler is capable to handle dynamic situation when VM appear in the system during simulation. In this case generated schedule is changing through time as some VMs are already switched off and new VMs are added.

EnerSim 1 brings significantly improvement of Alea and CloudSim to take into account energy with following features:

- To support energy in CloudSim and Alea, a new type of resource entity named Enerhost inherited from Host (in CloudSim) and GridResource (in Alea) is added. The Enerhost.java class contains newly designed methods to approximate resource energy consumed. Each node j in site i has two characteristics : minimum power ($p_{i,j}^{min}$) and maximum power ($p_{i,j}^{max}$).
- Similarly, a new abstract scheduler class called ReplayPolicy inherited from DatacenterBroker in Alea and AllocPolicy in CloudSim is also added. This class allows to randomly select a set of nodes with a given configuration, and to execute VMs repeatedly. This class use also the class OverReservation which allows to send a VM to a given node even if the distribution node is already full.
- The class Vm is inherited from the class Vm in CloudSim. Virtual machines are introduced in order to simulate cloud environments. We added

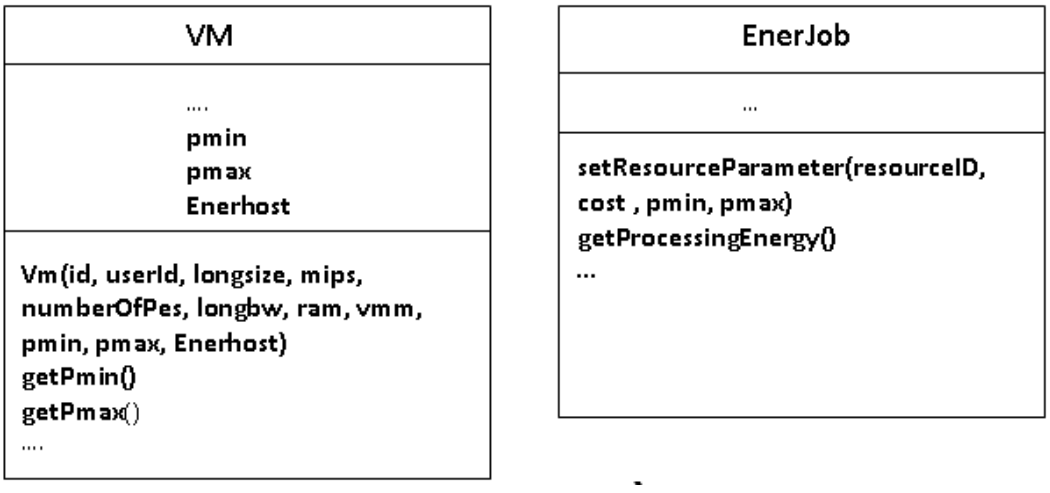

Figure 4.1: A simplified representation of Vm and EnerJob class with only the parameters and methods added.

parameters pmin and pmax to VMs. The class VM in EnerSim 1 takes into account energy. Figure 4.1 shows a representation of this class diagram with the parameters and methods added. Vm processes EnerJob.

– The class EnerJob inherited from class Gridlet (in Alea) allows to create a job which runs in a VM. The class job takes into account energy and limited resources allocation. We added a method setResourceParameter which sets the resource parameters for which this job is going to be executed. Parameters are the node ID, the pmin and the pmax the node and the cost running this GridResource per second. Another method is getProcessingEnergy which gets the total energy of processing or executing this job. Figure 4.1 shows the class diagram.

– Allocpolicy, inherited from VmAllocationPolicy class in CloudSim, is responsible of allocation of VM to nodes. This policy takes into account several hardware characteristics, such as number of CPU cores, CPU share, and $p_{i,j}^{min}$ and $p_{i,j}^{max}$ of physical node j in site i, that are allocated to a given VM instance. The proposed algorithm EACAB in Chapter 3 is implemented in this class.

– The classes Migration and OverReservation are added. Migration migrates tasks. The class OverReservation allows to specify which node will execute a given task. These classes are used by Allocpolicy to implement algorithms presented in sections 3.3 (ALBA) and 3.4 (EACAB).

– Enersim 1 also allows users to export results into cvs. or xls. format and also generating of graphs.

– Enersim 1 introduces a comprehensive GUI which can be used to configure the simulation.

Figure 4.2 shows the architecture of the simulator Enersim 1.

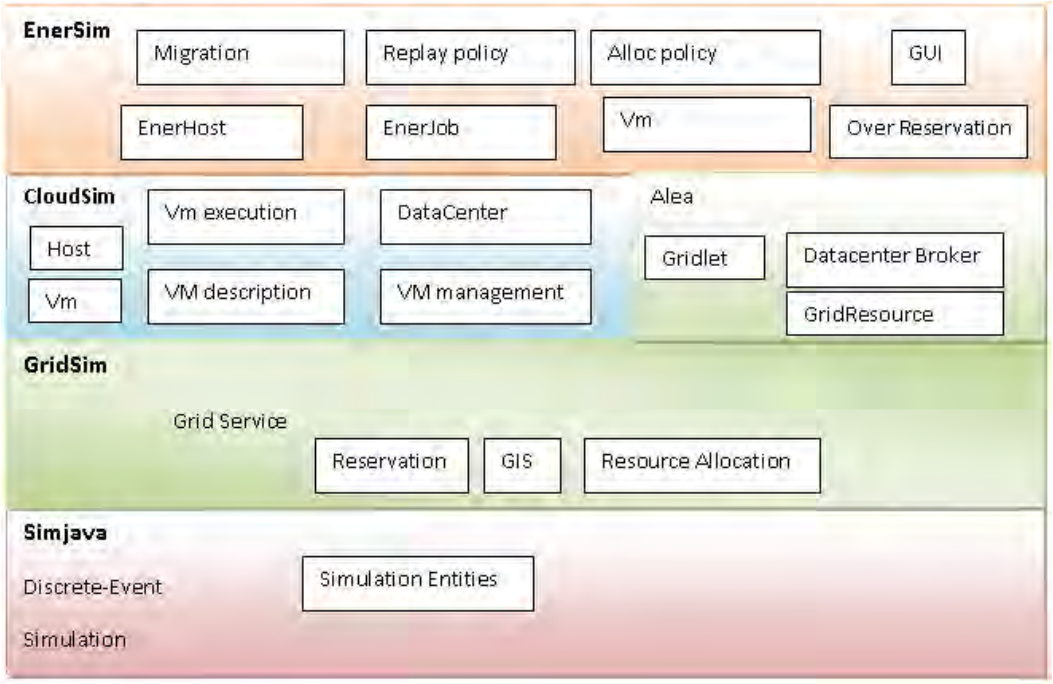

Figure 4.2: Layered architecture of Enersim 1

The class diagram on figure 4.3 describes the main set of classes of Enersim 1. These classes are responsible for the modelling and execution of the simulations. The GUI is designed loosely coupled from this main simulation framework and hence shown as a package in the main diagram. The class DatacentarBroker of CloudSim performs a dual role in VM management and

routing traffic to data centers. In EnerSim 1, the two roles are performed by VM management (CloudSim) and DatacenterBroker (Alea).

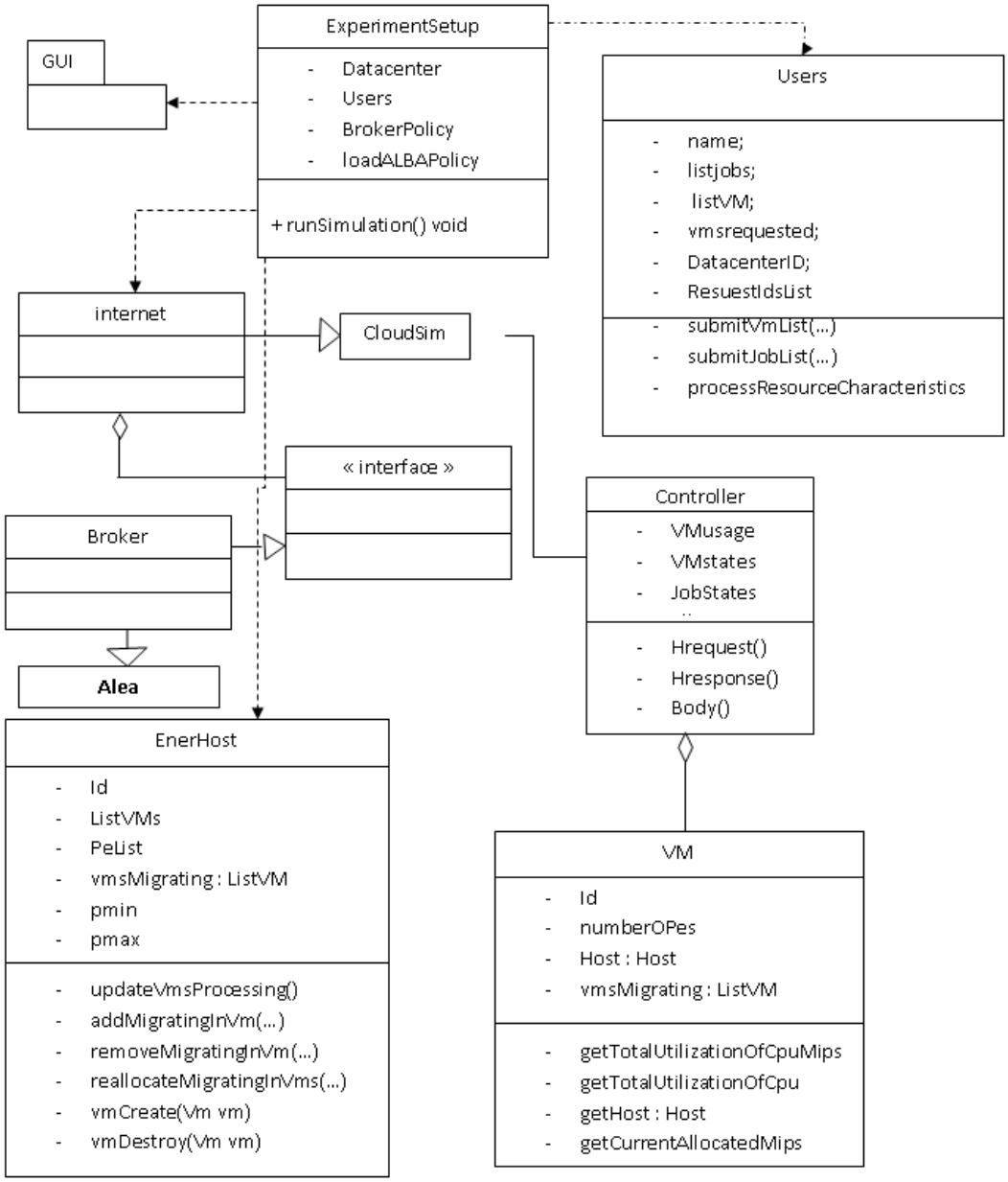

Figure 4.3: Class diagram showing the main classes of Enersim 1

Figure 4.4 present main screen of Enersim 1.

An advantage of Enersim 1 is that future simulators which are inherited do

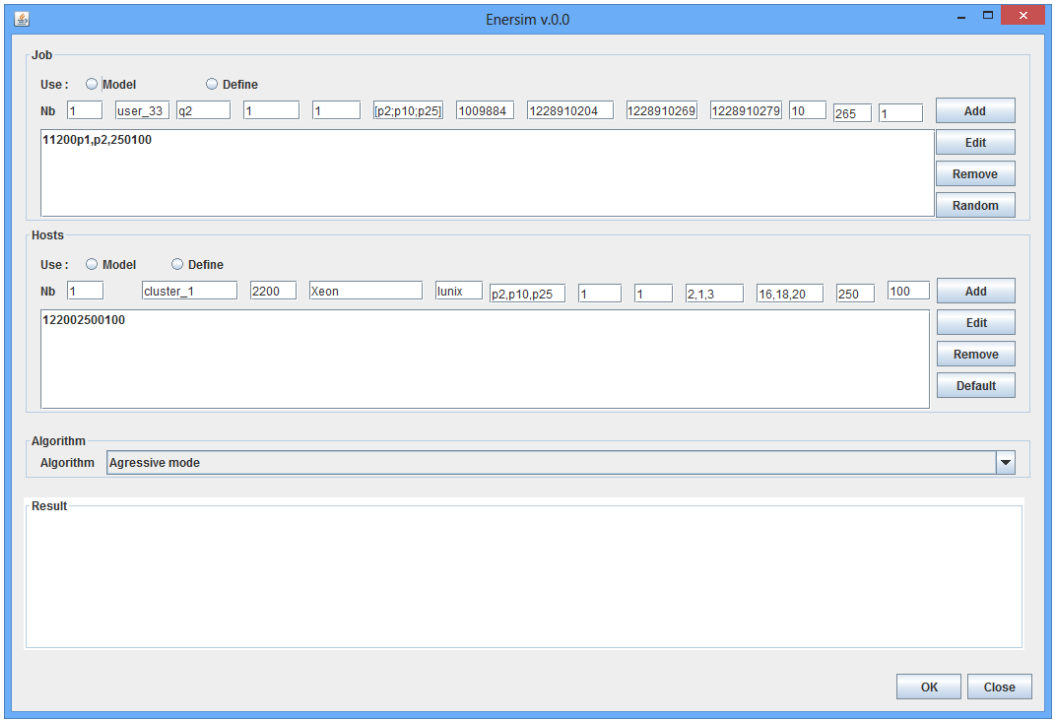

Figure 4.4: EnerSim 1 EnerSim and 2 main screen

not require modification of existing resource. But the limit is that it can simulate only centralized system.

- **EnerSim 2.** The Enersim 2 simulator solves the lack of Enersim 1 to simulate decentralized algorithms. EnerSim 2 extends MaGateSim and Enersim 1 and is selected as the simulator for experimental evaluation for the decentralized approach. EnerSim 1 has been extended with the ability to handle: (1)simulation of decentralized algorithms and (2)cooperative scheduling algorithms. MagateSim brings decentralized and cooperatives properties. MagateSim is a set of classes extending GridSim. Among the classes inherited from MagateSim we used mainly classes in modules Kernel Module (KM), Community Module (CM), Local Resource Management systems (LRM) and External Module (EM). KM is responsible for Magate self-management. It is related to three other modules. In these modules, the class ModuleController is in charge communication during the scheduler lifecycle, including :

job submission/scheduled/completion events, community knowledge updates, system self-inspection requests. An other class in the KM is MatchMaker. This class evaluates the adopted policy with knowledge of local resource capabilities, and decides whether the job could be executed locally or an appropriate remote nodes from the local cached direct neighbors list, or sends the propagated queries to the EM, in order to discover potential suitable remote nodes from the grid community directly. EM contains other features such as the management of neighboring nodes and internet services. CM allows for interoperability between other scheduleurs and facilitates the work (job) exchange among the interconnected grid community. LRM performs tasks allocations and tasks management on the local nodes.

EnerSim 2 extends MaGateSim by adding the power consumption and virtual machine (mainly their migration), the latter inherited from Enersim 1. The simulator offers computing resource which takes into account energy. Our simulator is designed to be a decentralized cloud scheduler that emphasizes on cloud scheduler interoperation, and complemented by a dynamic resource discovery approach on decentralized network. In Enersim 2 we changed classes that manipulate resources. Here we use the resources defined in Enersim 1, which take into account energy.

- Multiple inheritance in java is prohibit. Instead multiple inheritance of interfaces is proposed. In EnerSim 2 we added the interface IJOB which is implemented by the class Job of MagateSim. Thus, we create the new class EnerJobII which inherit from the class EnerJob of Enersim 1 and implement the interfase IJOB (see class in figure 4.5).
- We added the class simResourceInfo that gives nodes' informations (see fig. 4.6). This class stores dynamic information about each resource. e.g. prepared schedule for this resource, list of descriptions of jobs in execution. It also provides methods to calculate various parameters based on the knowledge of the schedule and resource status e.g. expected makespan, energy.

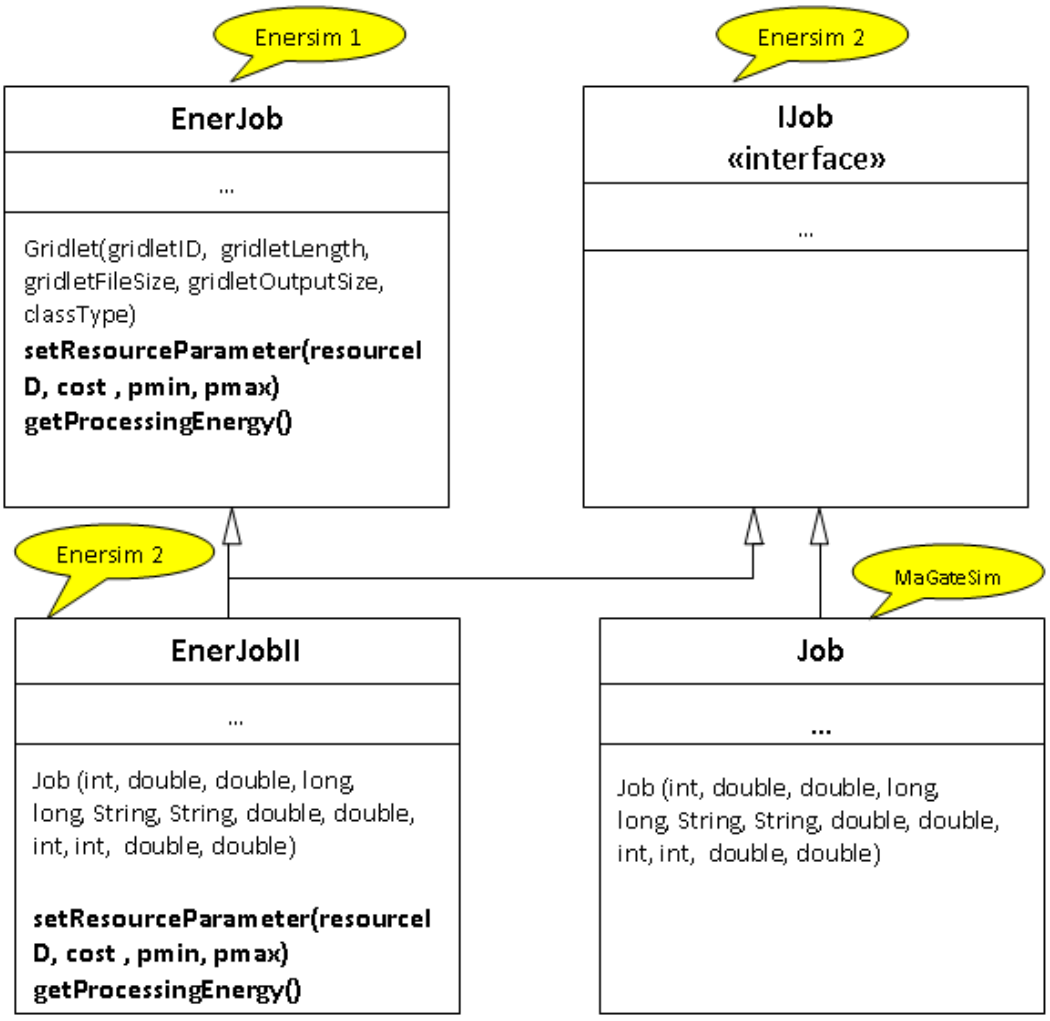

Figure 4.5: Class diagram for EnerJob in Enersim 2

- The class AdvancedPolicy extends the class AllocPolicy of Enersim 1. This class implement algorithms presented in our decentralized approach (CSAAC, FindVmWait, FindVmExec and VerifyLoad).

- GUI is the same in both simulators.

The conceptual architecture of the Enersim 2 is shown below. Enersim 2 is built on the top of Enersim 2 and MagateSim. Underlying layers, which are not shown in the figure, are the same as those presented in Enersim 1.

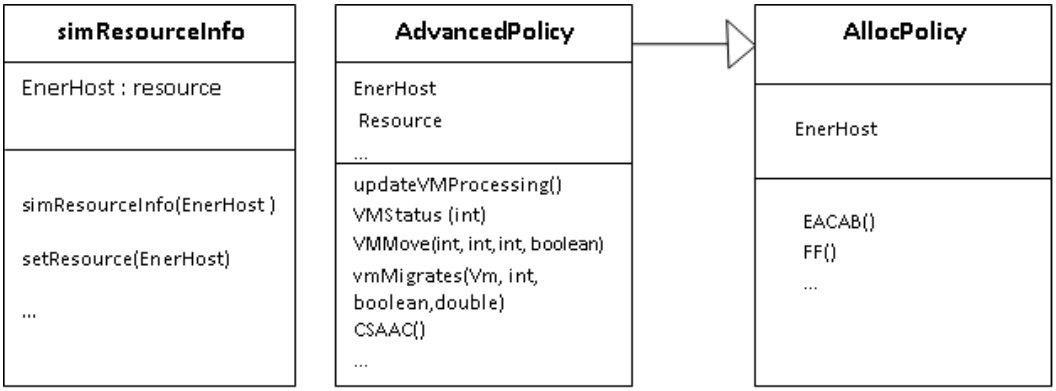

Figure 4.6: Class diagram for simResourceInfo and AdvancedPolicy in Enersim 2

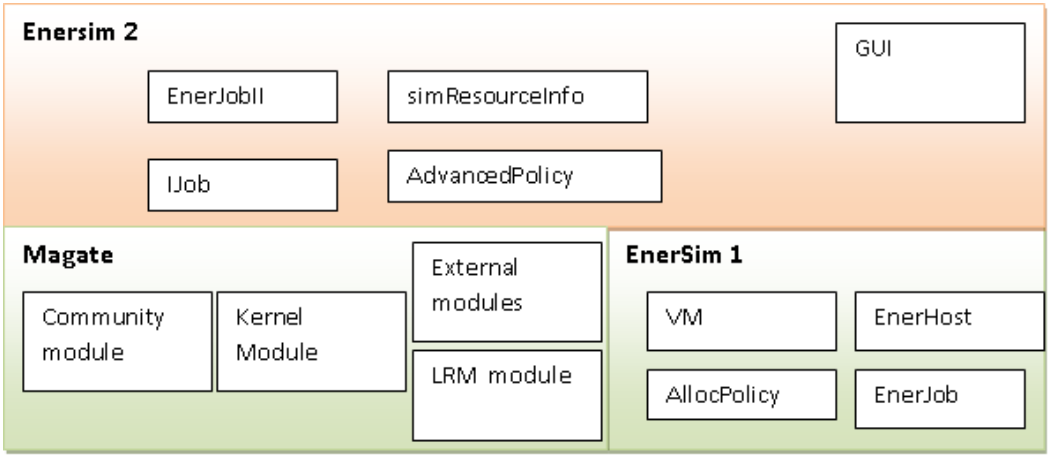

Figure 4.7: Layered architecture of Enersim 2 simulator.

4.3 System Setup

To verify our algorithms we initially used Enersim 1 to test the EACAB. Then, to compare the two algorithms, EACAB and CSAAC, we used the simulator Enersim 2.

4.3.1 Simulation environment with Enersim 1

- Data center : 100 clusters of 100 nodes each. Each node speed is randomly chosen between 1GHz and 3.06GHz

- Tasks : 1000 randomly generated tasks
  - Duration between 10 and 40s
  - Requested load between 10% and 100%
- Node shutdown and wakeup energy are assumed to be zero. In section 3.2.2 we defined the classical linear model of power consumption in function of load. When nodes are being switched off or wakeup, they have no load so their energy is zero.
- A node j in site i has two different power states for each core: Switched on and switched off. While switched on, power consumption is linear in function of load between $P_{i,j}^{min}$ and $P_{i,j}^{max}$. Those values are different for each node and are respectively between 75 and 150W, and 200 and 250W.

The First Fit algorithm for the first allocation problem is the well-known mapping algorithm Borgetto et al. [Borgetto 2012] modified by adding a power-aware component to the algorithm. In the following we compare EACAB with Dynamic First Fit (FF) which is a dynamic *First Fit* where nodes are sorted according to their maximum power consumption. FF allocates each VM to the first node on which it fits. Borgetto et al. added a power-aware component to the algorithm FF, and they allocate first on the PM that will have the smallest maximum power consumption. FF with migration use a re-allocation phase : choose underloaded PM, distribute its load in First Fit fashion. With migration all VMs are re-allocated, while without migration only the arriving VM is allocated.

4.3.2 System Setup for Enersim 2 simulator

We use Grid5000 workload (Date: 1 Apr. 2010, full load submitted tasks: 1020195) [Grid50002010 2010], resource topology for Grid5000 , 26 sites, 3194 nodes (see table 4.2). Nodes are equipped with Intel Xeon and AMD Opteron processors. Muti-core is not considered. Service nodes and storage nodes are not considered.

- The workload trace archive and resource deployment topology of the Grid5000

Table 4.2: Resource topology for Grid5000. A=AMD and I=Intel.

sites	AMD	INTEL	TOTAL	NODE	NumOfSite
Orsay	684	x	684	60(A), 372(A), 252(A)	3
Grenoble	x	68	68	68(I)	1
Lyon	252	x	252	112(A), 140(A)	2
Rennes	x	376	376	50(I), 128(I), 66(I),132(I)	4
Sophia	310	90	400	98(A), 112(A), 100(A),90(I)	4
Bordeaux	322	102	424	96(A), 102(I), 186(A),40(A)	4
Lille	198	92	290	100(A), 40(A), 52(A), 92(I)	4
Nancy	x	424	424	184(I), 240(I)	2
Toulouse	276	x	276	116(A), 160(A)	2
	2042	1152	3194		26

[Grid50002010 2010] is selected to organize the experiment of the distributed algorithm and the comparision of the EACAB and CSAAC.

- We assume that nodes have two different power states for each core : Switched on and switched off. While switched on, the power consumption depends on load, between $P_{i,j}^{min}$ and $P_{i,j}^{max}$. Those values are different for each node and are respectively between 75 and 150W, and 200 and 560W as measured on Grid5000.

4.4 Centralized approach : Experimental results

A part of this work concentrates on the design of a system intended for study of advanced scheduling techniques for planning various types of tasks in Cloud environment. The solution is able to deal with common problems of task scheduling in cloud like heterogeneity of tasks and resources, and dynamic runtime changes such as arrival of new tasks.

In order to evaluate the gains of the EACAB compared to classical algorithm Dynamic First Fit, we implemented our algorithm in Enersim 1.

In this subsection, we describe the simulation study performed to evaluate the performance of our algorithms in terms of energy minimization as well as the execution time and the number of migrations.

The first observation is that for three algorithms EACAB, dynamic First Fit

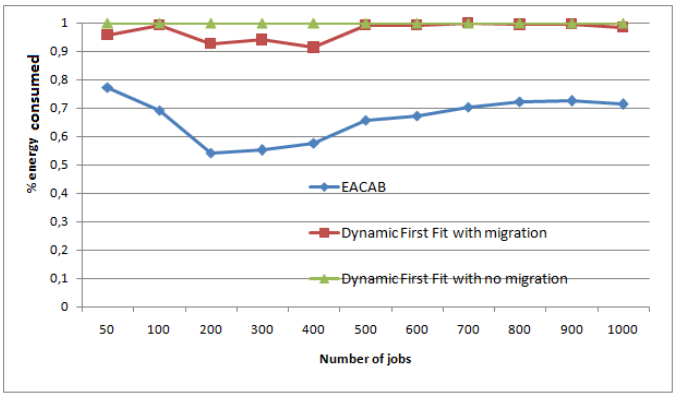

Figure 4.8: Simulation with Enersim 1 : Energy of EACAB compared to dynamic first fit with sorted nodes. Lower is better

with no migration and dynamic First Fit with migration. The algorithm EACAB consumes the least energy while Dynamic First Fit algorithms consume the most energy (see figure 4.8). The second observation is that this algorithm is able to reduce the energy consumption by 15 percent to 35 percent when task increases from 350 to 1000. EACAB checked regularly if not exite overloaded or underloaded node and migrates VMs to avoid the VMs slowdown or to have underloaded PM. FF does not verify the overload nodes.

Figures 4.9 and 4.10 show respectively the maximum and median number of switched on nodes as a function of task number. When number of tasks increase then the number of nodes switched on also increase, leading to a higher power consumption. This is particularly true if there is no migration after the initial placement of tasks. Hence, the gain of our algorithm increases power-wise with the number of tasks because migration is activated. The number of migrations is low in both EACAB and dynamic First Fit at the start of the experiment. In the case of the consolidation, less nodes are switched-on because we can adapt to the workload dynamism. The median has the same behavior but the maximum number of nodes is 50%. The observed gain increases with the number of tasks and becomes constant when nodes are saturated.

The good results of the EACAB comes from the fact that with the increase of the number of tasks, it has more possibilities to migrate tasks. It can then better

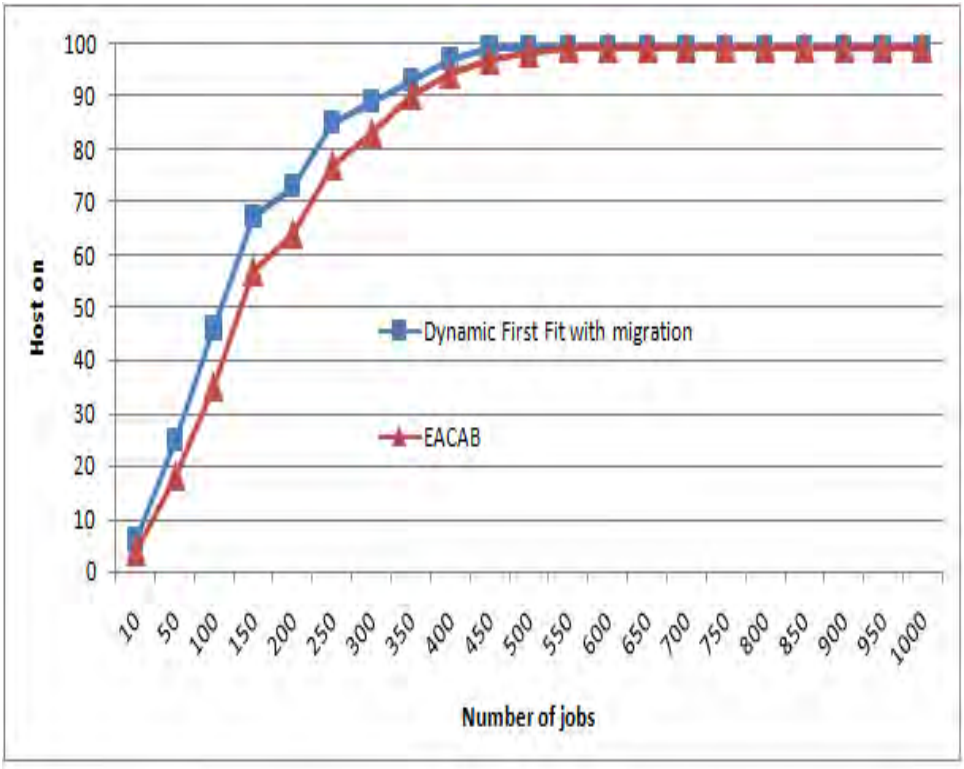

Figure 4.9: Simulation with Enersim 1 : Maximum nodes switched on with EACAB

allocate tasks on nodes, reducing the number of switched-on nodes.

Due to the thresholds of the EACAB, it would be possible to reduce further the number of switched on nodes but it would overload remaining nodes. Those nodes would become hot points and would have a negative impact on cooling. In order to prevent overloading, the EACAB adjusts load as shown in Figure 4.11. If the number of tasks increases, it will reduce the mean actual load they will obtain.

Figure 4.12 shows that our algorithm EACAB is better than classical *Dynamic First Fit* regardless of the threshold when the number of tasks is large.

The choice of γ is still important. There is an energy consumption difference of 10% between the best and the worst value. The worst value (100 jobs) is 10% more efficient than dynamic first fit, the best one (300 jobs) is 50% more efficient.

For small number of tasks, energy consumption increases because of the many migrations.

The EACAB has the shortest execution time when the number of jobs increases.

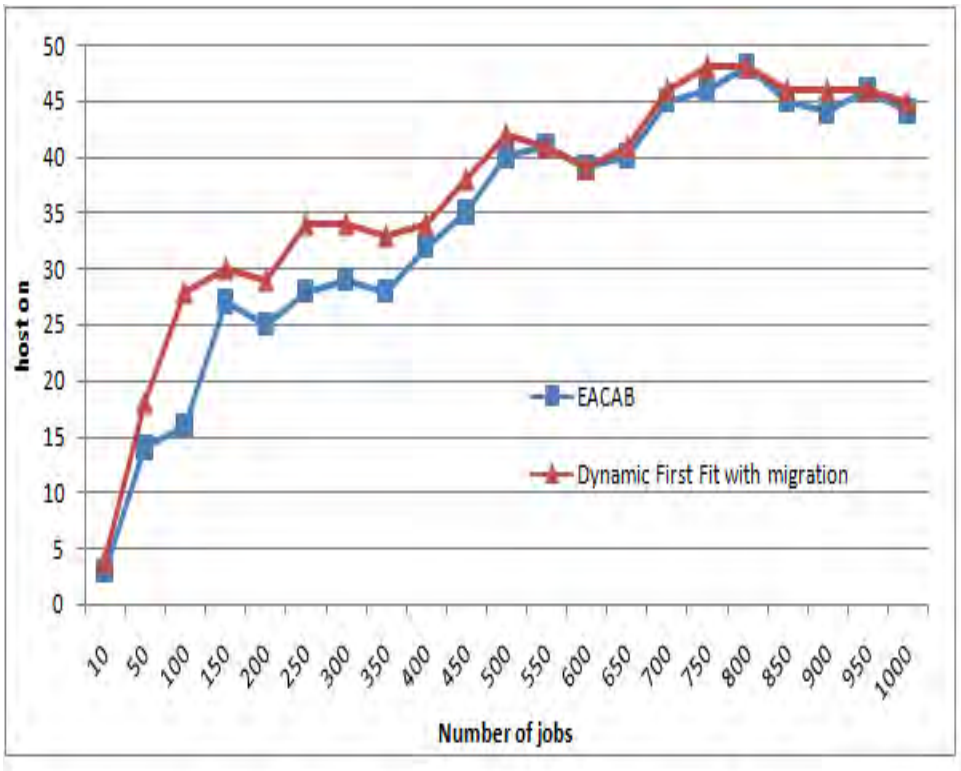

Figure 4.10: Simulation with Enersim 1 : Median nodes switched on with EACAB

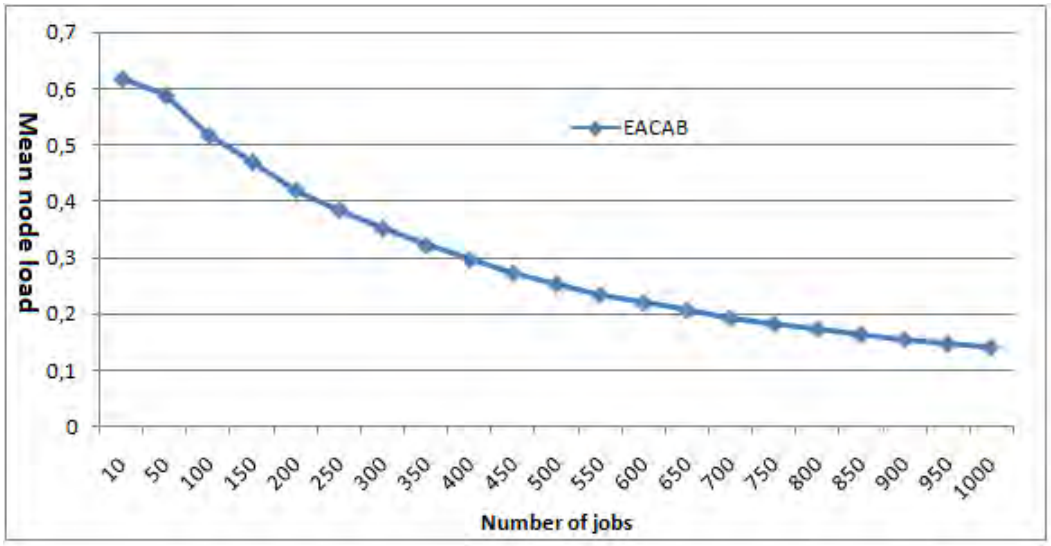

Figure 4.11: Simulation with Enersim 1 : Node mean load with EACAB

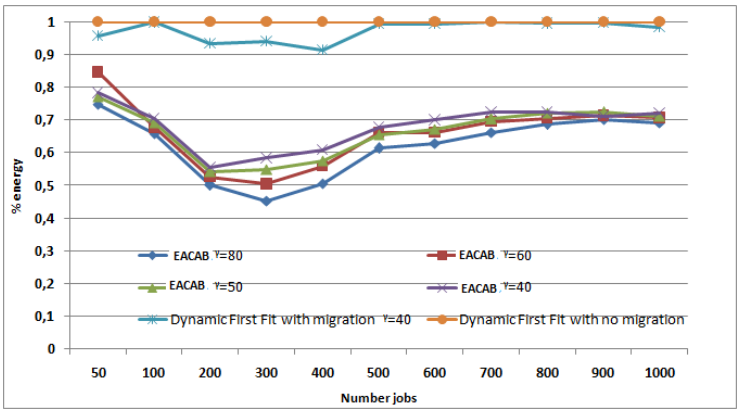

Figure 4.12: Simulation with Enersim 1 : Impact of threshold in energy gain with the EACAB

Table 4.3: Makespan of algorithms compared to First Fit with sorted nodes and migration

	Number of Jobs									
Algorithm	100	200	300	400	500	600	700	800	900	1000
Dynamic first Fit	265.53	374.4	580.37	631.22	615.04	870.55	813.83	1006.31	853.68	1270.73
EACAB	230	225.81	321.95	399.45	495.09	589.84	565.48	872.3	738.43	815.65

The result implicates that the scheduling EACAB algorithm can leverage migrations to achieve high performance and energy efficiency. Table 4.3 an Figure 4.13 show that our algorithm produces faster scheduling regardless of the number of jobs. Whatever the number of jobs EACAB makespan is lower than Dynamic First Fit. This can be justified by the fact that EACAB uses two thresholds for nodes load and thus avoids the jobs slowdown. The job slowdown may be due to overloading node.

From Figure 4.10 and 4.13, we understand the difference in Energy Consumption shown in Figure 4.8 between EACAB and Dynamic First Fit. This comes from two factors : less nodes are used, and for a smaller time.

4.5 Decentralized approach : Experiments and Results

In this subsection, we describe the simulation study performed to evaluate the performance of our algorithms in terms of energy minimization as well as the execution

Figure 4.13: Simulation with Enersim 1 : Makespan with EACAB compared to dynamic first fit (FF)

time and the number of migrations for the CSAAC. The proposed algorithms are evaluated by extensive simulations using the EnerSim 2 simulator.

In order to share the tasks submitted from a local scheduler to other scheduler within the same data center community, a set of community scheduling relevant parameters are evaluated and discussed to address various task delegation scenarios between different scheduler. The decentralized approach schedulers are driven to cooperate with each other, to provide intelligent scheduling for the scope of serving the data center community as a whole, not just for a single node individually.

In the following we compare our algorithms with the CASA with energy that produces energy-efficient schedules. CASA with energy is the version of CASA which takes into account the energy (see section 2.4.4 for details).

The first observation is that for two algorithms, the CSAAC consumes the least energy while the CASA algorithm consumes the most energy (see figures 4.14 and

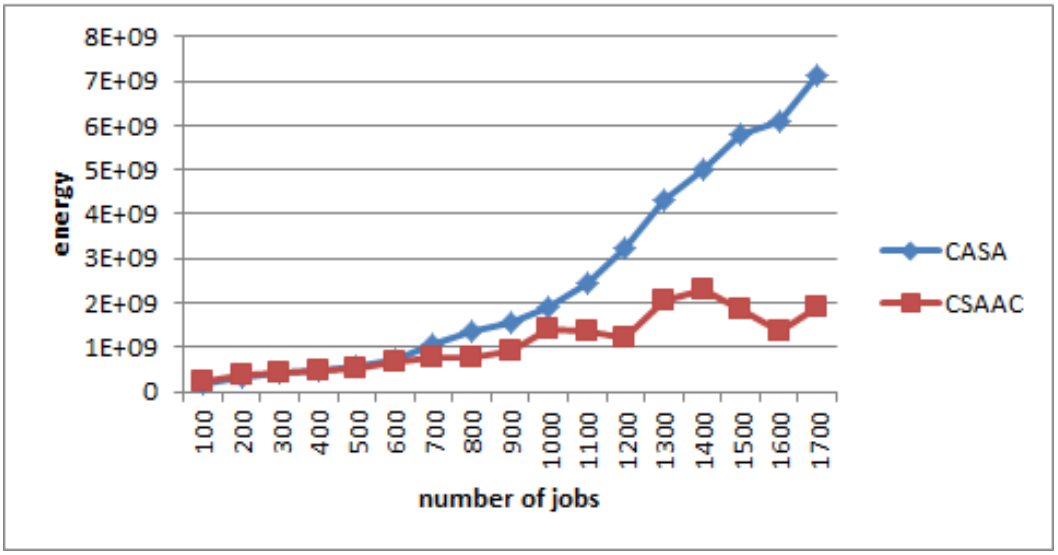

Figure 4.14: Energy of CASA with energy compared to the CSAAC with sorted nodes by pmax. Lower is better

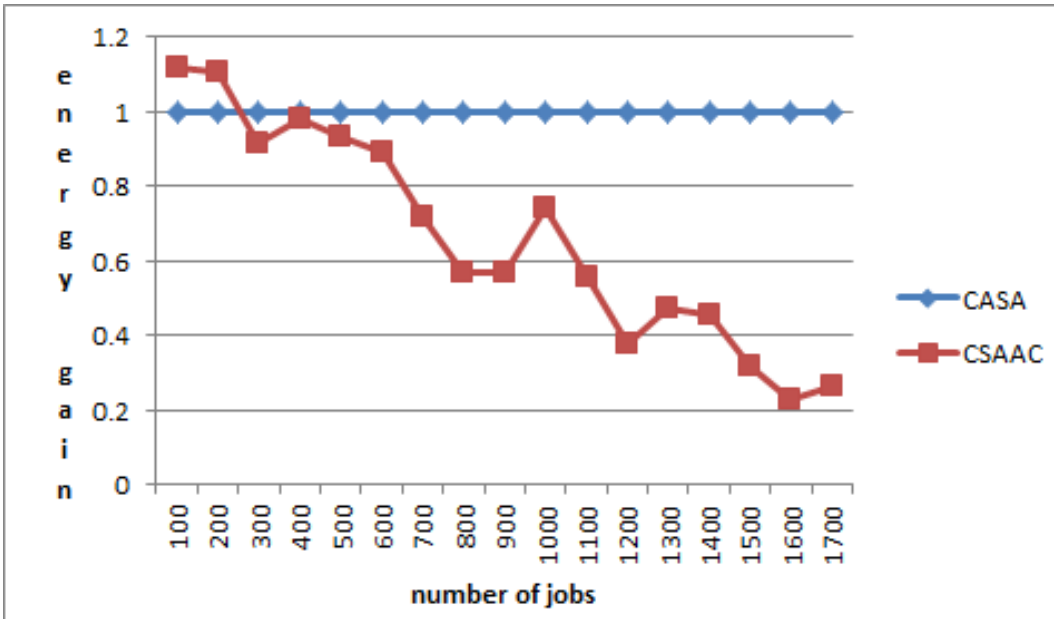

Figure 4.15: Energy gain of CSAAC compared to CASA with sorted nodes by pmax. Lower is better

4.15), when the number of jobs $T > 300$. For a small number of tasks our algorithm CSAAC leads to a significant energy consumption increase. The second observation is that the CSAAC is able to reduce the energy consumption by 5 percent to 80 percent when job increases from 300 to 1700. Figure 4.16 demonstrates the number of migrations incurred by 1700 jobs. An obvious observation is that migrations are beneficial to save energy. The second phase of our algorithm calls into question the choices and therefore modifies the node loads over time. If at any time the tasks are finished and that there are several machines under loaded, they can be consolidated. This is not the case for algorithms which never calls into question the allocation once the jobs are running. The impact of migration, however, may not be large enough to dominate the total energy gain when the number of jobs is less than 300.

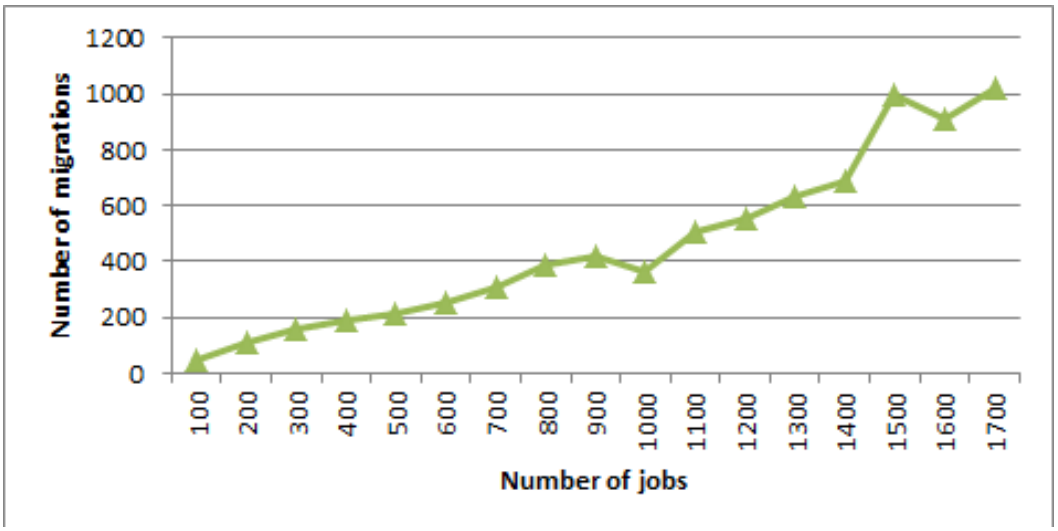

Figure 4.16: CSAAC : Migration

Figure 4.17 shows maximum number of switched on nodes. In CSAAC, when the number of jobs increases, the number of nodes switched on does not increase automaticaly. Sometimes a very powerful server is turned on causing the migration of VMS from several other servers to it. As the cloud is not saturated redistribution of tasks is often done. Whith CASA, when the number of jobs increases then the number of nodes switched on also increases, leading to a higher power consumption.

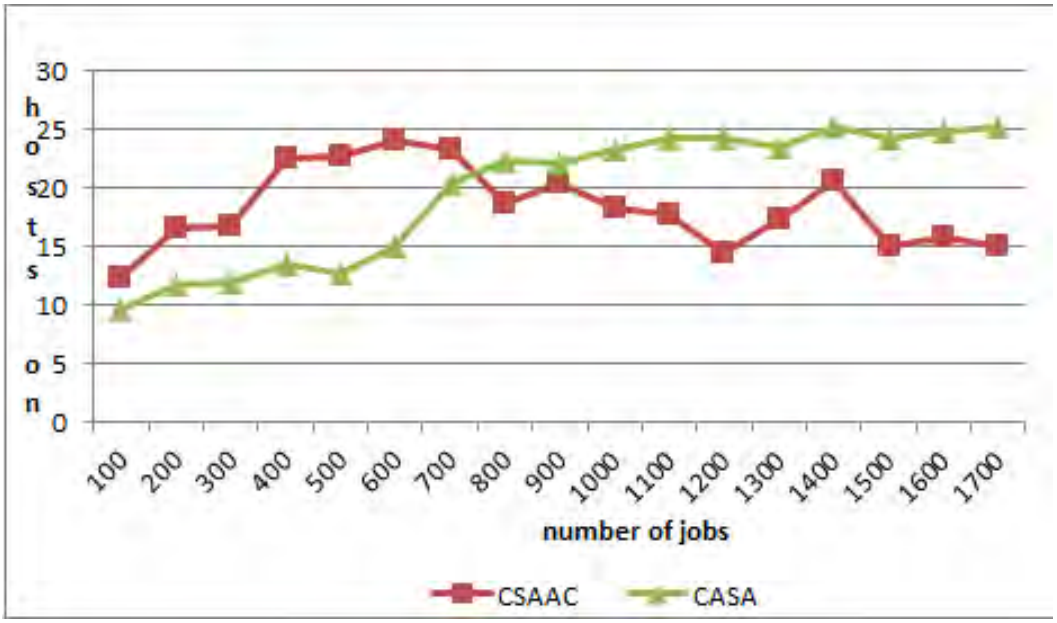

Figure 4.17: Maximum node switched on with the CSAAC compared to CASA

This is particularly true if there is no migration after the initial placement of tasks. Hence, the gain of our algorithm CSAAC increases power-wise with the number of tasks because migration is activated. Computing resources are fully used in both cases at the start of the experiment. In the case of the consolidation, less nodes are switched-on because we can adapt to the workload dynamism.

The CSAAC on the other hand, has the shortest execution time when the number of jobs increases. The result implicates that the scheduling algorithm such as CSAAC can leverage interconnects with migrations to achieve high performance and energy efficiency. Figures 4.18 shows that our algorithm produces faster scheduling regardless of the number of jobs. The good results of the CSAAC comes from the fact that with the increase of the number of tasks, it has more possibilities to migrate tasks. It can then better allocate tasks on computing resources, reducing the makespan and increasing energy gain. Therefore, our algorithm CSAAC is able to obtain better performance.

Due to the thresholds of the CSAAC , it would be possible to reduce further the number of switched on nodes but it would overload remaining nodes. Those nodes

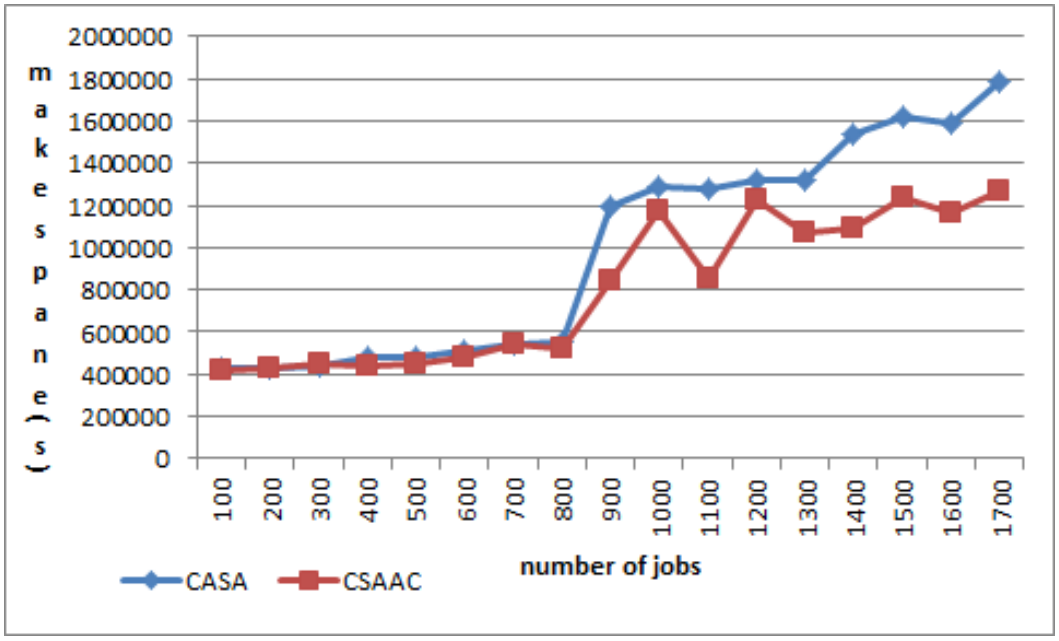

Figure 4.18: Makespan of CSAAC compared to CASA with sorted nodes by pmax. Lower is better

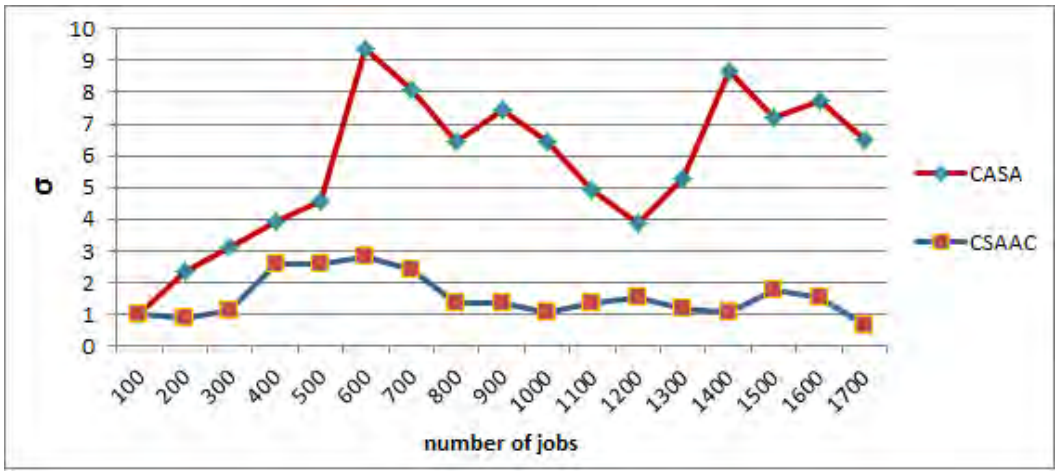

Figure 4.19: comparison between the CSAAC and CASA. Standard deviation σ of node load

would become hot points and would have a negative impact on cooling. In order to prevent overloading, the CSAAC adjusts load. Also the figure 4.19 shows how widely node loads are dispersed from the average value (the mean). In previous

results (figures 4.15 and 4.16), the CSAAC is in the lead in terms of energy gain and execution time, since in this algorithm there is cooperation between schedulers which allows an efficient consolidation in cloud. Figure 4.19, shows the standard deviation σ of the nodes load which confirms the good distribution of the load after consolidation. Thus figure 4.19 shows that the CSAAC gives the best standard deviation compared to CASA, which is an indication of the good predictability of the performance of the CSAAC.

4.6 Decentralized approach vs Centralized approach

Decentralized algorithms solve the main shortcomings of centralized algorithms algorithms such as scalability, fault tolerence and bottlenecks which can significantly degrade performance, the adequacy of the cloud computing environment, autonomy. However, we note two main disadvantages are the difficulty of implementation and selfish comprtement can have some nodes .

In this section, we compare two classes of proposed scheduling algorithms, centralized and decentralized. In centralized scheduling, one cloud scheduler maintains a complete control over the clusters. All the jobs are submitted through the cloud scheduler. In contrast, in decentralized scheduling, organizations maintain (limited) control over their schedules. Jobs are submitted locally, but they can be migrated to another cluster, if the local cluster is overloaded. The possibilities of migration are, however, limited, so that migrated jobs do not overload the node system. The aim of this section is to compare energy consumed by EACAB and CSAAC. Using EnerSim 2 simulation environment, we simulate both algorithms.

We run the simulation using a Grid5000 workload and measure the amount of energy consumption of the placement and the average execution times for both algorithms (i.e. centralized and decentralized). To derive the actual energy savings, the amount of energy spent for VM placement was estimated by the formula defined in section 3.2.

The final simulation results are depicted in Table 4.4. The gain in energy and makespan does not depend on the numbers of jobs but mostly of the distribution

of jobs between nodes.

Table 4.4: Simulation results. (Stdev=standard deviation)

Jobs	Approach	makespan	Makespan gain (%)	Makespan Stdev	energy	Energy gain (%)	Energy Stdev
500	cent	48770		6491	16358.5		10363419
	Dec	48984	−0.44	5574	16252.9	0.65	10475850
1000	cent	102089		40225	34473.4		56793603
	dec	100859	1.20	40676	31628.8	8.25	51366829
1500	cent	117143		24348	45480.4		104549039
	dec	130734	−11.6	61778	47263.9	−3.9	118552002
2000	cent	153386		51215	58819.5		154963019
	Dec	143047	6.74	39176	63349.97	−7.7	180703187
2500	cent	143977		37093	56716.3		129407887
	Dec	143541	0.3	36334	56108.85	1.07	116130251
3000	cent	250457		237465	61635.4		167952610
	Dec	211032	15.72	191168	56993.7	7.5	133578369
3500	cent	235614		215901	63370.7		164512761
	Dec	202696	13.97	162140	55093.6	13.06	130973378
4000	cent	221526		192048	54071.2		116554623
	Dec	233452	−5.38	223754	60270.2	−11.5	157261329
4500	cent	241048		222470	57983.4		131320566
	Dec	207189	14.05	173065	59478.3	−2.57	175986327

Figure 4.20: EACAB vs CSAAC : Energy

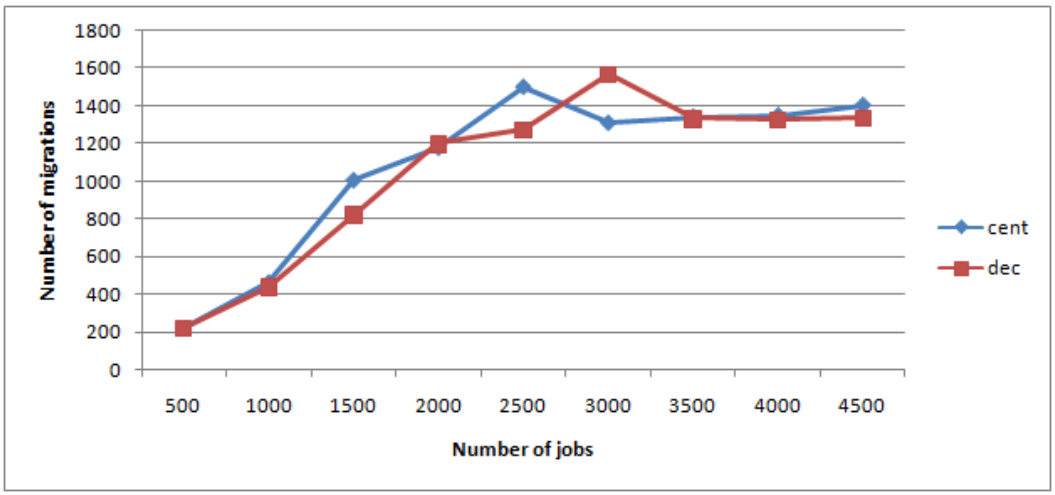

Figure 4.21: EACAB vs CSAAC : Migration

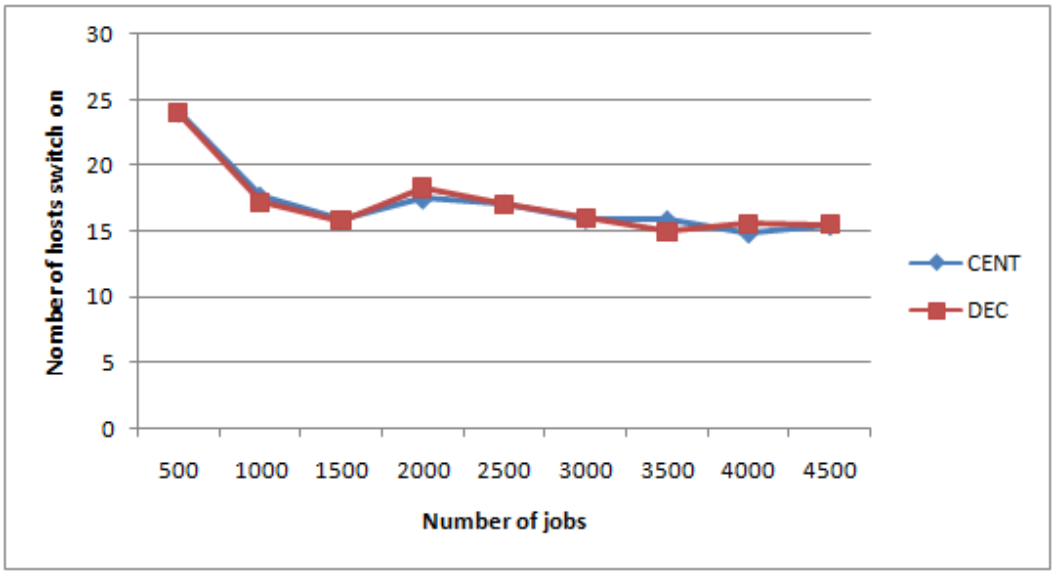

Figure 4.22: EACAB vs CSAAC : Maximum nodes switched on

We performed series of tests, comparing EACAB and CSAAC. Generally, the CSAAC schedules slightly worse in terms of makespan than the EACAB. Typical results of experiments are presented in Figures 4.20,4.21,4.22 et 4.23. In Figure 4.20 it is easy to notice that energy consumed by the distributed algorithm is comparable to centralized strategy for low number of jobs. These poor results

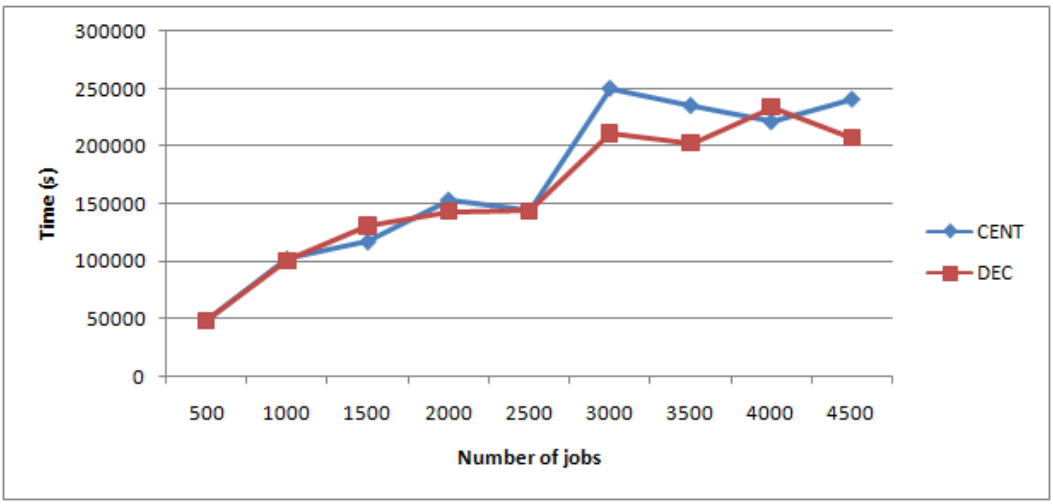

Figure 4.23: EACAB vs CSAAC : Makespan

are caused by low number of migrations since majority of jobs can be executed without exceeding their due dates. This situation changes for higher loads when number of migrations is increased and the distributed algorithm outperforms the centralized one some time. We achieved similar results for centralized algorithm which use migration and anti load-balancing techniques. When a cluster load is below the under-loaded threshold, centralized and decentralized algorithm are able to migrate jobs to more-loaded clusters and switch off under-loaded cluster. In this case, performance measures and energy depend strongly on the collaboration of less-loaded clusters. When their cooperation is too low the system as a whole starts to be inefficient, although the performance of the less-loaded clusters is not affected. Consequently, we consider that there must be some minimal cooperation that results from a cloud agreement. As in real systems the job stream changes, this minimal cooperation can be also interpreted as an "insurance" to imbalance the load.

From the experiments above, we can get the obvious conclusion that both the EACAB and CSAAC can reduce energy consumed of data centers. Figure 4.23 above shows the execution time for all tasks and both schedulers. We can see that the two algorithms have the same behavior.

4.7 Conclusion

In this chapter we have presented the two simulators (EnerSin 1 and EnerSim 2) used. We also evaluate the EACAB and CSAAC which are energy-efficient IaaS cloud management systems for large-scale virtualized data centers.

Overall, the EACAB can compute allocations effectively with an important energy gain. Experiments showed that with this algorithm energy gain can achieve 20% over standard algorithms.

The CSAAC has been extensively evaluated with the Grid'5000 workload and shown to be scalable and energy-efficient. Particularly, our experimental results have shown that: (1) the CSAAC is capable of obtaining energy-efficient schedules using less optimization time;(2)application performance is not impacted by performing migration using under load and under load thresholds; (3)the system scales well with increasing number of resources thus making it suitable for managing large-scale virtualized data centers; (4) The anti load-balancing technique used by the two approaches achieve substantial energy savings. (5) when $jobs > 300$, the CSAAC is able to reduce the average energy consumption by about 10 percent to 80%. (5)the execution time of jobs is also reduced by 5 percent to 25 percent when number of $jobs > 300$, when compared to the CASA algorithm

Finally we have compared the two algorithms and can get the obvious conclusion that both EACAB and CSAAC can reduce energy consumed of data centers.

Chapter 5

Conclusion

Contents

This thesis investigates VMs scheduling in cloud in developing solution approaches to the Cloud energy consumption problem. Our objective is to design energy aware clouds scheduling using anti load-balancing algorithm capable of coordinating the scheduling behaviors of independent entities in the Cloud. The developed solution mainly targets to reduce energy consumed based on centralized and decentralized scheduling that is adequate for the Cloud. This chapter summarizes the main contributions of this work, highlights our conclusions, and presents some future research directions.

5.1 Contributions

Cloud Computing has become another buzzword after Web 2.0. However, there are dozens of different definitions for Cloud Computing and there seems to be no consensus on what a Cloud is. On the other hand, Cloud Computing is not a completely new concept; it has an intricate connection to the relatively new but thirteen-year established Grid Computing paradigm, and other relevant technologies such as utility computing, cluster computing, and distributed systems in general. Cloud computing has recently emerged as a new computing paradigm which allows customers to lease services based on the pay-as-you-go model. Customers are charged for only what they use. To support the customers growing service demands cloud

providers are now building an increasing number of large-scale data centers. Managing such data centers is a challenging task as it involves the design of novel cloud management frameworks and algorithms which are not only able to operate at scale but also lower the data center energy consumption during periods of low resource utilization. This thesis has focused on the IaaS cloud service model whose goal is to offer compute infrastructure by provisioning VMs on-demand. Particularly, in this thesis we have investigated the challenge of designing, implementing, and evaluating an energy-efficient task placement system for private clouds. In order to achieve this goal, Chapter 2, has first introduced the context of this work, namely server virtualization, scheduling, cloud computing and energy management in data centers. Then, it has reviewed the related work on the design and implementation of scalable, and energy-efficient IaaS cloud management systems and highlighted their limitations. Based on the lessons learned this thesis has proposed the following four novel contributions which have been analyzed and designed in Chapter 3 :

- **Anti load-Balancing algorithm**. In this thesis, we analyzed and developed a scheduling model which is the inverse of the load balancing operation. This model concentrates work in fewer nodes, idling other nodes that can be turned off. This load concentration or anti load-balancing saves the power consumed by the powered-down nodes, but can degrade the performance of the remaining nodes and potentially increase their power consumption.

- **Energy-aware cloud scheduling using Anti load-Balancing algorithm.** In Chapter 3, a centralized dynamic VM consolidation algorithm has been analysed to obtain significantly energy gain and insights into designing online algorithms for dynamic VM consolidation. The centralized approach consolidates VMs on a subset of nodes judiciously chosen depending on the characteristics and state of nodes. This algorithm can be employed in practical settings. Overall, the proposed algorithm can compute allocations effectively with an important energy gain. We argue in this work that energy savings are achieved by continuous consolidation of VMs according to current utilization

of resources and by avoiding hot spots. The chapter has concluded with a discussion of taking into account performance in the algorithm. The utilisation of two thresholds of the CPU utilization defines the underload and overload states. For instance, a server can be considered to be overloaded when its utilization exceeds 80%. Such tuning is useful in cases when the application performance is known to degrade after a certain level of CPU utilization.

- **Cooperative scheduling Anti load-Balancing Algorithm in Cloud**. Chapter 3 has proposed a distributed dynamic VM consolidation consisting in splitting the problem into 4 sub-problems: (1) host underload detection; (2) host overload detection; (3) VM selection; and (4) VM placement. Splitting the problem allows executing algorithms for the first 3 sub-problems in a distributed manner independently on each compute node. Local managers eliminate the single point of failure and making the system completely distributed and decentralized. The model was analyzed and derived based on the structure and the characteristics of the Cloud. Comparing with other research work on the Cloud scheduling problem modeling, our model leads to a family of heuristics that perform well in terms of energy savings while still leading to good task performance.

  Then we distributed this algorithm, so that each cluster can exchange tasks, based on their respective loads. We presented and evaluated this decentralized approach for cloud. This algorithm is based on the principle of *CASA*. It takes into account energy and provides energy-efficiency improvement compared to classical load unbalancing algorithms. The decentralized approach schedulers are driven to cooperate with each other, to provide intelligent scheduling for the scope of serving the grid community as a whole, not just for a single grid node individually. Our main problem was to optimize energy consumption given task performance constraints. Energy consumption is to be taken in a broad way as we try to prevent hot spots to reduce impact on cooling. Overall, the proposed approach can compute allocations effectively with an important energy gain.

We have compared the CSAAC to CASA over a range of realistic problem instances using simulation in chapter 4. The decentralized approach parameters lead to a family of heuristics that perform well in terms of energy savings while still leading to good task performance. The proposed approach can compute allocations effectively with an important energy gain.

The simulation results in chapter 4 showed that our algorithm is capable of obtaining energy-efficient schedules using less optimization time. In particular, we showed that for the case when $jobs > 300$, our algorithm is able to reduce the average energy consumption by about 10 percent to 80 percent. At the same time, the execution time of jobs is also reduced by 5 percent to 25 percent when number of $jobs > 300$, when compared to the CASA algorithm.

The experimental results have exhibited significant improvement in terms of scheduling for energy savings.

The conclusions drawn from these results indicate that the proposed approaches of scheduling requires more work, but are otherwise promising with regards to energy efficiency.

- **Simulators.** We developed for the centralized approach the EnerSim 1 similator which is described in chapter 4. EnerSim exends ALEA and CloudSim. Secondly in order to simulate the cooperative scheduling, we extended MaGateSim by adding properties that allow it to take into account energy and migration. We have also performed an implementation of CASA within the context of current developments in cloud computing. The Enersim 2 extends MagateSim and Enersim 1.

  The implementation of Enersim 2 allowed us to compare the centralized approach and decentralized approach. Thus, Chapter 4 has proposed experiments consisting in comparing the centralize approach and the distributed dynamic VM consolidation. Experiments conducted on a Grid5000 topology have shown that dynamic VM consolidation is able to reduce energy consumption by the compute nodes by up to 30

5.2 Perspectives

As we just discussed in the section 5.1, the obtained results gave us enough motivation to consider following as future research topics to be investigated as a continuation of this Ph.D. thesis.

We plan to extend the heterogeneity of our performance model to other parts of IaaS cloud computing centers. We would like to support full heterogeneity in VMs, PMs and user tasks; more specifically, VMs are required to be varied in terms of memory, disk, CPU core and CPU; PMs may differ in computation capacity, networking, memory, disk, Graphics Processing Unit (GPU) and hypervisor (VMM) capabilities; user tasks may request various number of resources for different amount of time (i.e., different probability distributions for task service time).

Despite substantial contributions of the current thesis in energy-efficient distributed dynamic VM consolidation, there are a number of open research challenges that need to be addressed in order to further advance the area.

VM Network Topologies

In virtualized data centers VMs often communicate with each other, establishing virtual network topologies. However, due to VM migrations or non-optimized allocation, the communicating VMs may end up hosted on logically distant physical nodes leading to costly data transfers between them. If the communicating VMs are allocated to hosts in different racks or enclosures, the network communication may involve network switches that consume significant amounts of energy. To eliminate this data transfer overhead and minimize energy consumption, it is necessary to observe the communication between VMs and place the communicating VMs on the same or closely located nodes. In particular, to provide efficient reallocations, it is required to develop power consumption models of the network devices and estimate the cost of data transfer depending on the traffic volume. Moreover, VM migrations consume additional energy and have a negative impact on the performance. The VM placement algorithm has to ensure that the cost of migration does not exceed the benefits.

Dynamic threshold

Fixed values of the utilization threshold are unsuitable for an environment with dynamic and unpredictable workloads, in which different types of applications can share a physical node. The system should be able to automatically adjust its behavior depending on the workload patterns exhibited by the applications. It will be interresting to present a heuristic algorithm for auto-adjustment of the utilization threshold based on statistical analysis of historical data collected during the lifetime of VMs.

Dynamic and Heterogeneous SLAs

Cloud data centers need to provide strict QoS guarantees, which are documented in the form of SLAs. Resource provisioning within a data center directly influences whether the SLAs are met. Current Cloud data centers host applications from clients distributed globally. These clients have very different requirements, which may also vary over time. For example, an organization using Cloud services may require tighter response time guarantees in day time than in night time. To address the problem, it is necessary to develop algorithms that exploit time variation in SLAs of the users to minimize the number of physical servers required.

Currently, to simplify management, resources in Cloud data centers are allocated to clients depending only on that client's SLAs, regardless of the SLAs of other users. The intrinsic differences among various SLAs can make huge differences in the amount of resources allocated to each user. Although heterogeneous requirements of users make scheduling and VM consolidation algorithms complex, they can be exploited to improve energy-efficiency. It is important to devise and analyze algorithms that make use of such heterogeneity. Moreover, to meet the requirements of large-scale Cloud data centers, it is necessary to design a solution scalable to handling thousands of users.

Thermal-Aware Dynamic VM Consolidation

A significant part of electrical energy consumed by computing resources is transformed into heat. High temperature leads to a number of problems, such as the reduced system reliability and availability, as well as the decreased life time of the hardware. In order to keep the system components within their safe operating temperature and prevent failures and crashes, the emitted heat must be dissipated.

The cooling problem becomes extremely important for modern blade and 1U rack servers that pack computing resources with high density and complicate heat dissipation. By laws of physics, all the power has to be convert into heat. According to [Carlson 2012] a pair of microprocessors mounted on a single motherboard can draw 200-400 watts or more of power. According to a 2008 Gartner report [Wang 2008], 50 percent of data centers will soon have insufficient power and cooling capacity to meet the demands of high-density equipment. The cooling cost is one of the major contributors of the total electricity bill of large data centers. Therefore, apart from the hardware improvements, it is essential to address the problem of optimizing the cooling system operation from the resource management side. One of the ways to minimize the cooling operating costs is to continuously monitor thermal state of physical nodes and reallocate VMs from a node when it becomes overheated. In this case, the cooling system of the offloaded node can be slowed down allowing natural heat dissipation. Moreover, there has been research work on modeling the thermal topology of a data center that can lead to more efficient workload placement [Johnson 2009]. Therefore, it is necessary to investigate how and when to reallocate VMs to minimize the power drawn by the cooling system, while preserving safe temperature of the resources and minimizing the migration overhead and performance degradation.

Part II

Part II Selected chapters :French version

Résumé

La multiplication de l'informatique en nuage (Cloud) a abouti à la création de centres de données dans le monde entier. Le Cloud contient des milliers de nœuds de calcul. Cependant, les centres de données consomment d'énorme quantités d'énergie à travers le monde estimées à plus de 1,5% de la consommation mondiale d'électricité et devrait continuer à croître. Une problématique habituellement étudiée dans les systèmes distribués est de répartir équitablement la charge. Mais lorsque l'objectif est de réduire la consommation électrique, ce type d'algorithmes peut mener à avoir des serveurs fortement sous chargés et donc à consommer de l'énergie inutilement. Cette thèse présente de nouvelles techniques, des algorithmes et des logiciels pour la consolidation dynamique et distribuée de machines virtuelles (VM) dans le Cloud. L'objectif principal de cette thèse est de proposer des stratégies d'ordonnancement tenant compte de l'énergie dans le Cloud pour les économies d'énergie. Pour atteindre cet objectif, nous utilisons des approches centralisées et décentralisées. Les contributions à ce niveau méthodologique sont présentées sur ces deux axes. L'objectif de notre démarche est de réduire la consommation de l'énergie totale du centre de données en contrôlant la consommation globale d'énergie des applications tout en assurant les contrats de service pour l'exécution des applications. La consommation d'énergie est réduite en désactivant et réactivant dynamiquement les nœuds physiques pour répondre à la demande des ressources. Les principales contributions sont les suivantes:

- Ici on s'intéressera à la problématique contraire de l'équilibrage de charge. Il s'agit d'une technique appelée Anti Load-Balancing pour concentrer la charge sur un nombre minimal de nœuds. Le but est de pouvoir éteindre les nœuds libérés et donc de minimiser la consommation énergétique du système.

- Ensuite une approche centralisée a été proposée et fonctionne en associant une valeur de crédit à chaque nœud. Le crédit d'un nœud dépend de son affinité pour ses tâches, sa charge de travail actuelle et sa façon d'effectuer ses communications. Les économies d'énergie sont atteintes par la consolidation continue des machines virtuelles en fonction de l'utilisation actuelle des ressources, les

topologies de réseaux virtuels établis entre les machines virtuelles et l'état thermique de nœuds de calcul. Les résultats de l'expérience sur une extension de CloudSim (EnerSim) montrent que l'énergie consommée par les applications du Cloud et l'efficacité énergétique ont été améliorées.

- Le troisième axe est consacré à l'examen d'une approche appelée "Cooperative scheduling Anti load-balancing Algorithm for cloud". Il s'agit d'une approche décentralisée permettant la coopération entre les différents sites. Pour valider cet algorithme, nous avons étendu le simulateur MaGateSim.

Avec une large évaluation expérimentale d'un ensemble de données réelles, nous sommes arrivés à la conclusion que l'approche à la fois en utilisant des algorithmes centralisés et décentralisés peut réduire l'énergie consommée des centres de données.

Mots clés: Energie, Heuristique, Machine virtuelle, Cloud, Migration, consolidation

Chapter 1

Introduction

Contents

Ce chapitre présente le contexte de la recherche exploré dans cette thèse. Il commence avec les motivations fondamentales sur les organisations décentralisées et coordonnées des systèmes Grille/Informatique en nuage (Grille/Cloud), y compris les systèmes d'allocation des ressources. Le chapitre fournit ensuite une discussion sur quelques problèmes de la recherche, les objectifs de ce travail et la méthodologie.

1.1 Motivation

Le cloud computing a émergé comme un nouveau modèle d'affaires de calcul et de stockage basé sur le modèle sans forfait (pay-as-you-go model), d'accès à des quantités potentiellement importantes de capacités de centres de données distants. La facturation des clients dépend des services qu'ils ont consommées jusqu'à présent. Un services particulier de cloud offert par les fournisseurs de cloud, qui a gagné beaucoup d'attraction au cours des dernières années est l'infrastructure comme un service (IaaS). Dans les IaaS de cloud, des ressources de calcul et de stockage sont provisionnées sur demande par les fournisseurs de cloud. De nombreux fournisseurs de cloud comme Amazon, Google, Rackspace, SalesForce sont apparus et offrent maintenant une énorme quantité de services tels que la capacité de calcul et de

stockage de données à la demande. Afin de soutenir de plus en plus les demandes de services des clients, les fournisseurs de cloud ont déployé un nombre croissant de centres de données à grande échelle. La gestion de tels centres de données impose aux fournisseurs de cloud de résoudre un certain nombre de défis. En particulier, les fournisseurs de cloud conçoivent et mettent en oeuvre maintenant des IaaS à grande échelle. Comme l'infrastructure de cloud consomme beaucoup d'énergie entraînant des coûts opérationnels qui dépassent le coût de l'infrastructure en quelques années, de nombreuses recherches ont été menées sur l'amélioration de l'efficacité énergétique.

Dans le cloud computing, les techniques existantes pour les économies d'énergie peuvent être divisées en deux catégories : (1) d'abord, Dynamic Voltage / Fréquence Scaling (DVFS). Les économies d'énergie sont obtenues par ajustement de l'horloge de fonctionnement pour réduire les tensions d'alimentation des circuits. Bien que cette approche peut obtenir une réduction significative de la consommation d'énergie, elle dépend des réglages des composants matériels pour effectuer des tâches de mise à l'échelle; (2) Ensuite, l'autre technique visant à réduire l'énergie consommée par un centre de données de Cloud est d'utiliser la consolidation. Il s'agit de la fusion de plusieurs choses en une seule. Appliqué au monde de l'informatique, cela consiste à rassembler plusieurs ressources en une seule. La virtualisation est principalement utilisée dans les projets de consolidation de noeuds. La consolidation de noeuds vise à minimiser le nombre de noeuds physiques nécessaires pour accueillir un groupe de machines virtuelles (VM). Compte tenu de l'importance des économies d'énergie, la gestion de l'efficacité énergétique des IaaS doit être conçue. Plusieurs tentatives ont été faites au cours des dernières années pour concevoir et mettre en oeuvre des systèmes de gestion de IaaS de cloud pour faciliter la création de IaaS de cloud privés.

Étant donné l'augmentation du nombre de centres de données, ils sont confrontés à des défis en termes d'évolutivité, d'autonomie et d'efficacité énergétique. Cependant, la plupart des tentatives existantes pour concevoir et mettre en oeuvre des IaaS pour les clouds privés sont toujours basées sur des architectures centralisées, disposant d'une autonomie, et d'une absence de mécanismes d'économie d'énergie.

1.2 Problèmes de recherche et Objectifs

L'objectif de cette thèse est de concevoir, mettre en oeuvre et évaluer des algorithmes d'ordonnancement qui tiennent compte de l'énergie dans les systèmes distribués. Pour atteindre cet objectif principal, cette thèse étudie les problèmes de recherche suivants :

- **Comment définir un algorithme d'ordonnancement.** Les stratégies d'ordonnancement de tâches ont été étudiées dans une variété de scénarios liés à la complexité des processus d'affaires et de calcul scientifique. L'objectif de l'ordonnancement des tâches est de répondre aux exigences de performance, et obtenir la plus faible consommation d'énergie totale pour l'exécution de toutes les tâches. La gestion de l'énergie dans les stratégies d'ordonnancement dans les systèmes distribués doit prendre en compte les aspects suivants : cyclique, la continuité, la communication, la corrélation de données, l'hétérogénéité des noeuds et le temps. Dans le cas de l'énergie consommée dans les centres de données, la consolidation est l'une des multiples solutions utilisées. Avec des algorithmes d'ordonnancement distribués, il est recommandé de choisir des techniques de coopération;

- **Structure du réseau .** Le type d'algorithme défini dans la section précédente dépend de la façon dont est structuré le réseau: centralisé, distribué ou hybride;

- **Migration :**

  - Quand migrer les machines virtuelles. La consolidation dynamique des machines virtuelles (VMs) comprend deux processus fondamentaux: (1) la migration des machines virtuelles des noeuds surchargés pour éviter la dégradation des performances et (2) la migration des machines virtuelles à partir de noeuds sous-chargés pour améliorer l'utilisation des ressources et de minimiser la consommation d'énergie. Une décision importante qui doit être faite dans les deux cas est de déterminer quand la migration des

machines virtuelles afin de minimiser la consommation d'énergie, tout en respectant les contraintes de qualité de service (QoS) définis;

- Quelles VMs migrer. Une fois la décision de migrer les machines virtuelles d'un noeud sous-chargé ou surchargé est faite, il est nécessaire de choisir une ou plusieurs machines virtuelles à partir de l'ensemble des machines virtuelles attribuées au noeud, qui doivent être réaffectées à d'autres noeuds. Le problème consiste à déterminer le meilleur sous-ensemble de machines virtuelles à migrer vers des noeuds qui ne sont pas dans la même situation, qui fournira la reconfiguration du système le plus bénéfique.

- Où migrer les machines virtuelles sélectionnées pour la migration. Il s'agit de déterminer le meilleur placement des machines virtuelles sélectionnées sur le noeud pour la migration vers d'autres noeuds. Il s'agit d'un autre aspect essentiel qui affecte la qualité de la consolidation de VM et la consommation d'énergie du système.

- **Consolidation :**

  - Quand et où les nœuds physiques à éteindre ou allumer. Pour minimiser la consommation d'énergie par le système et éviter les violations des exigences de qualité de service, il est nécessaire de déterminer de manière efficace, quand et quels nœuds physiques doivent être désactivés ou réactivés pour gérer l'augmentation de la demande pour les ressources;

  - Comment concevoir des algorithmes distribués de consolidation dynamique de VMs. Pour fournir une évolutivité et éliminer les points de défaillance uniques, il est nécessaire d'utiliser une approche décentralisée pour les algorithmes de consolidation dynamique de VMs. Le problème est que les algorithmes de gestion de ressources sont traditionnellement centralisées. Par conséquent, une bonne approche est de proposer un système distribué de consolidation dynamique de VMs.

Pour faire face aux défis liés aux problèmes de recherche ci-dessus, les objectifs suivants pour améliorer l'efficacité énergétique totale de Cloud en contrôlant de la consommation globale d'énergie tout en respectant les contrats de niveau de service (SLAs) des applications de Cloud, sont :

- Explorer, analyser et classer la recherche dans le domaine de l'efficacité énergétique afin d'acquérir une compréhension systématique des techniques et des approches existantes;
- Développer des algorithmes distribués, pour l'efficacité énergétique, de consolidation dynamique de VM pour les environnements de cloud satisfaisant des contraintes de QoS;
- Conception et mise en oeuvre d'un simulateur de consolidation dynamique de VM qui peut être utilisé pour évaluer les algorithmes proposés;
- Facilité de gestion des VMs : nous concevons un système qui est suffisamment souple pour permettre la création et le retrait dynamiques de serveurs. Comme les composants du système peuvent tomber en panne à tout moment, il est souhaitable pour un système de réparer les pannes ou de réallouer les tâches vers d'autres serveurs sans intervention humaine.
  - Conservation du contexte d'exécution: Il doit être possible d'arrêter le processus d'exécution de la VM et de le redémarrer là où il s'est arrêté. Le temps d'exécution peut être réduite lorsque la tâche migre vers un nœud plus puissant;
  - La prévention de ralentissement : le ralentissement d'une tâche augmente le temps d'exécution et l'énergie consommée par les hôtes et a un impact conséquent sur les utilisateurs.
- Efficacité énergétique : l'un de nos objectifs est de proposer des algorithmes de gestion de VMs qui sont capables de créer des moments d'inactivité de noeuds, amenant ces derniers vers l'état d'économie d'énergie et de les réveiller une fois nécessaire (par exemple lorsque la charge augmente).

1.3 Méthodologie

Nous étudions différentes techniques utilisées dans le fonctionnement du Cloud et des centres de données. Pour tester les algorithmes proposés, des simulateurs gérant l'éfficace énergétique pour l'environnement de Cloud ont été développés. Ces simulateurs opèrent au niveau centralisé et décentralisé. Nous avons utilisé des techniques comme la coopération, la collaboration et la virtualisation. Afin de satisfaire tout le sens de la durabilité qui peut être construit sur les algorithmes d'ordonnancement d'économes d'énergie proposés, nous analysont des solutions existantes et les comparons pour en choisir une que nous améliorerons. Ainsi nous montrons comment la consolidation est capable de diminuer la consommation d'énergie par rapport à la réduction des coûts en énergie en tenant compte des contraintes de QoS.

1.4 Organisation de la thèse

Ce résumé de thèse est organisé comme suit :

- le chapitre 2 présente un résumé des contributions . D'abord, un algorithme d'ordonnancement nommé Anti-équilibrage de charge algorithme est présenté . Ensuite, un algorithme decentralisé coopératif qui tient compte de l'énergie dans le cloud utilisant l'algorithme d'anti-équlibrage de charge est présenté. Afin d'améliorer son évolutivité , un système de consolidation VM entièrement décentralisée basée sur un réseau P2P non de machines physiques (MP) est proposé . Ce chapitre présente les résultats de l'algorithme centralisé de consolidation VM et l'algorithme entièrement décentralisée de consolidation VM . Enfin nous parlons des simulateurs.

- le chapitre 3 conclut ce manuscrit en résumant nos contributions et de présenter les orientations futures de la recherche.

Chapter 2

Résumé des contributions et des résultats

Contents

Les systèmes de gestion de cloud existants sont principalement basés sur des architectures centralisées et des mécanismes de gestion de l'énergie souffrent actuellement de plusieurs limites. Pour répondre à ces limitations, une de nos contributions sera de concevoir, de mettre en œuvre et d'évaluer un nouveau système de gestion de cloud. Ce système offre une solution de gestion globale de VM intégrant des mécanismes avancés de gestion de VM tels qu'éviter des PM sous-chargées donc la consolidation de VMs, et la gestion de l'énérgie. Nous vous proposons trois algorithmes de gestion d'énergie dans les clouds.

Ce chapitre présente les contributions et les résultats obtenus. Il est structuré comme suit : section 2.1 présente l'algorithme Anti-équilibrage de charge (ALBA); section 2.2 présente l'approche centralisée; section 2.3 traite de l'approche décentralisée; la section 2.4 résume les travaux sur les simulateurs; et enfin, la section 2.5 résume les résultats.

2.1 Anti load-balancing Algorithm (ALBA).

Cette contribution est une technique d'ordonnancement d'anti-équilibrage de charge appelée Anti load-Balancing (ALBA). Un problème souvent étudié est de savoir comment répartir uniformément la charge de travail. Mais lorsque l'objectif est de réduire la consommation d'énergie, ce type d'algorithmes peut conduire à avoir des noeuds largement sous-chargés et donc de consommer inutilement de l'énergie. Ici, nous allons étudier le problème opposé : concentrer la charge sur un nombre minimum de machines. L'objectif est d'éteindre les noeuds libérés et donc de minimiser la consommation d'énergie du système.

La technique ALBA est la base de nos deux approches centralisée et décentralisée que nous allons présenter dans la suite de ce chapitre. Cet algorithme permet de migrer des VMs en cours d'exécution. L'algorithme $ALBA$ utilise principalement deux seuils (seuil bas et seuil haut) pour la charge des noeuds. La charge d'un noeud est comparée à ces seuils. Si elle est inférieure au seuil bas les machines virtuelles sont migrées puis le noeud est éteint. Si la charge est supérieure au seuil haut, une partie de cette charge est répartie sur d'autres noeuds afin de limiter le ralentissement des VMs ou le surchauffe du noeud.

Dans cette section, nous présentons notre algorithme ALBA. Les scénarios sont composés de quatre étapes :

- **Etape 1 :** déterminer quand un noeud est considéré comme étant surchargé; ceci entrainant la migration d'un ou plusieurs machines virtuelles à partir de ce nœud. Si la décision est d'ajouter un ou plusieurs nœuds, l'algorithme doit déterminer la partie de la de charge qui doit être déplacée vers d'autre nœuds ajoutés.

- **Étape 2:** déterminer si un noeud est sous-chargé afin de migrer toutes les machines virtuelles à partir de ce nœud et le mettre en veille.

- **Étape 3:** sélection de machines virtuelles qui doivent être migrées à partir d'un nœud surchargé.

- **Étape 4 :** trouver un nouveau noeud pour les machines virtuelles sélectionnées pour la migration.

Cet algorithme a deux avantages majeurs par rapport aux algorithmes traditionnelles de consolidation de VMs : (1) diviser le problème simplifie le traitement analytique des étapes; et (2) la méthode peut être mise en œuvre de manière répartie par l'exécution des procédures de détection de sous-charge et de surcharge, les algorithmes de sélection de VMs et les algorithmes de placement de VMs.

Afin de maximiser leur retour sur investissement (ROI), les fournisseurs de Cloud doivent appliquer des stratégies de gestion énergétique efficace des ressources, telle que la consolidation dynamique de VMs en mettant les nœuds sous-chargés en veille. Toutefois, cette consolidation n'est pas triviale, car il peut entraîner des violations de SLAs négociés avec les clients.

2.2 Energy aware clouds scheduling using anti load-balancing algorithm (EACAB).

Cette contribution présente un ordonnancement de cloud tenant compte de l'énergie, utilisant l'algorithme Anti load-balancing (EACAB). L'algorithme EACAB est basé sur la technique de calcul de crédit ($\sigma_{i,j}$) de chaque noeud j (dans le site i) utilisée dans Comet [Jeon 2010].

Comet est basé sur le calcul du crédit de chaque agent mobile. Chaque agent de Comet tente de maximiser son propre crédit en se déplaçant entre les hôtes. Un agent a_i utilise la formule suivante:

$$C_i = -x_1 w_i + x_2 h_i - x_3 g_i$$

. Où w_i représent la charge de calcul de l'hôte exécutant l'agent a_i, h_i et g_i correspondent à la charge de communication à l'intérieur et à l'extérieur de l'agent a_i, et où X_1, X_2 et x_3 sont des coefficients de type réel, positifs qui constituent la dépendance attribué à chaque agent à sa création servant d'estimer son affinité par rapport à leur hôte. Ainsi un agent se déplace vers un nouvel hôte si la charge

attribuée par l'hôte actuel est inférieure est à celle sollicitée, ou si sa communication externe est réduite ou si la communication interne augmente.

De la même manière, l'algorithme EACAB proposé fonctionne en associant une valeur de crédit à chaque noeud. Le crédit d'un noeud dépend du noeud, sa charge de travail actuelle, ses communications et l'historique de l'exécution des VMs. Quand un noeud est sous-chargé (charge $<$ $gamma$), toutes ses VMs sont migrent vers un noeud relativement plus chargé. La formule est alors :

$$\sigma_{i,j} = c_{i,j} - r_{i,j}s_{i,j} + \varepsilon - \gamma$$

Où $c_{i,j}$ est la charge actuelle du noeud j dans le site i, $r_{i,j}$ est sa charge de calcul demandé, $s_{i,j}$ est la satisfaction des VMs, et γ et ε sont respectivement le seuil de sous-charge et de surcharge.

Chaque noeud dispose de son propre *crédit*. Plus le crédit d'un noeud est élevé, plus ses ses VMs ont la chance d'y rester. Le crédit d'un noeud augmente si :

- Sa charge de travail ou le nombre de VMs dans le noeud augmente;
- ses communications entre ses VMs et d'autres noeuds augmente;
- sa charge augmente, tout en restant entre le seuil de sous-charge γ et le seuil de surcharge ε.

Au contraire, le crédit d'un noeud diminue dans les cas suivants:

- Sa charge de travail ou le nombre de VMs diminue;
- le noeud a juste envoyé ou reçu un message de l'ordonnanceur qui indique que le noeud va probablement devenir vide dans un temps court.

Le crédit d'un noeud sera utilisé dans la politique de sélection: le noeud qui a le crédit le plus faible est sélectionné pour la migration des VMs. La politique de localisation identifie le noeud distant qui a le plus grand crédit et qui est en mesure de recevoir les VMs sélectionnées par la politique de sélection sans être trop chargé.

L'algorithme est présenté au chapitre 3 de la partie 1. Les économies d'énergie sont atteintes par la consolidation continue de machines virtuelles en fonction de l'utilisation actuelle des noeuds.

2.3 Cooperative scheduling Anti load-balancing Algorithm for cloud (CSAAC).

L'ordonnancement dynamique d'un grand nombre de machines virtuelles dans le cadre des infrastructures largement distribuées est soumis à d'importants problèmes d'évolutivité, de réactivité, et de tolérance aux pannes dans une stratégie de contrôle centralisé.

Dans cette section, nous étudions si une approche décentralisée peut aider à résoudre les problèmes notés dans les sytèmes centralisés. La principale contribution de cette section est l'introduction d'une approche distribuée qui permet d'optimiser des performances et de réaliser des gains d'énergie sur le cloud global. Dans cet approche d'ordonnancement dynamique décentralisée nous avons proposé un algorithme intitulée Cooperative scheduling Anti load-balancing Algorithm for cloud (CSAAC). Il est composée d'une phase de migration, d'une phase de soumission de VMs et d'une phase d'ordonnancement dynamique.

La phase de soumission constitue la première phase de l'algorithme. Chaque fois qu'un noeud j reçoit une $VM_{i,j,k}$ soumise par un utilisateur local, celui-ci (noeud j) se comporte comme noeud demandeur $H^{req}_{i,j}$ et génère un message de demande d'exécution $M^{req}_{i,j,k}$. Ce message contient les informations sur les caractéristiques de la VM $VM_{i,j,k}$, notamment le temps d'exécution estimé $D_{i,j,k}$ et le nombre de processeur (PE) demandé $VM^{pe}_{i,j,k}$. Le message de requête est reproduit et diffusé. Tous les noeuds recevant le message $M^{req}_{i,j,k}$, y compris la noeud demandeur j, sont considérés comme des noeuds répondeurs. Chaque noeud répondeur $H^{resp}_{i',j'}$ doit envoyer un message d'acceptation $M^{ack}_{i',j',k}$ indiquant que le noeud j' est capable et prêt à exécuter la VM reçu $VM_{i,j,k}$. Chaque candidat , considéré comme noeud répondeur, en fontion de son statut et de ses ressources, calcule un temps d'éxécution estimée de la VM et surtout l'énergie consommée nécessaire. Ces informations sont

envoyées au au noeud demandeur sous forme de message d'acceptation $M^{ack}_{i',j',k}$. Le noeud demandeur recueille tous les messages reçu $M^{ack}_{i',j',k}$, et génère un message $M^{ass}_{i',j',k}$ et l'envoyer au noeud sélectionné $H^{ass}_{i',j'}$ (noeud attribué).

La phase d'ordonnancement dynamique permet par exemple de prendre une décision de redistribution pour une machine virtuelle qui se trouve dans une liste longue et donc ne peut pas être exécutée immédiatement par le noeud. Périodiquement la phase de migration est exécutée pour résoudre les problèmes de surcharge et de sous charge des noeuds.

La motivation principale de cet approche est de permettre l'ordonnancement efficient qui minimise la consommation d'énergie dans le cloud global. Contrairement aux solutions d'ordonnancement classique dans les cloud, cet algorithme est conçu pour offrir l'ordonnancement d'événements pertinents, tels que la soumission de VM. De plus, l'algorithme permet le rééchelonnement dynamique afin de s'adapter aux caractéristiques des centres de données de cloud telles que l'instantanéité et de la volatilité. L'algorithme, composé de trois phases, utilise également la consolidation afin de réduire l'énergie consommée.

2.4 Simulateurs.

Pour évaluer des algorithmes d'ordonnancement dans un environnement de cloud computing mais surtout la consommation d'énergie des infrastructures de cloud nous avons besoin d'un simulateur qui a les fonctionnalités suivantes : (1) prendre en compte l'énergie consommée par le système et le temps d'exécution des tâches; (2) utilisation des techniques de virtualisation, de migration et surtout la possibilité d'éteindre ou d'allumer des nœuds physiques; (3) paramétrage facile pour simuler de nouveaux algorithmes et (4) simuler des algorithmes décentralisés dynamiques et coopératifs dans un environnement de cloud.

Ainsi nous avons développé un simulateur Enersim 1 qui étend ALEA et CloudSim. CloudSim gère des machines virtuelles et étend GridSim. Les principales limites de CloudSim sont le manque de coopération entre les entités et les nœuds qui ne tiennent pas compte de l'énergie. La plupart de ces fonctions sont directement util-

isées dans Enersim 1. C'est un outil utilisé pour simuler l'algorithme proposé dans l'approche centralisée, EACAB. EnerSim 1 améliore de manière significative Alea et CloudSim et prend en compte la consommation d'énergie des infrastructures de cloud.

Ainsi un nouveau type de ressource nommée Enerhost a été ajouté. La classe Enerhost.java contient des nouvelles méthodes qui calculent de façon approximative l'énergie consommée par les ressources. Nous avons défini pour chaque nœud j d'un site i deux caractéristiques : la puissance minimum ($p_{i,j}^{min}$) et la puissance maximale ($p_{i,j}^{max}$).

De même, une nouvelle classe abstraite appelée ReplayPolicy hérité des classes DatacenterBroker de Alea et AllocPolicy de CloudSim est également ajoutée. Cette classe permet de sélectionner au hasard un ensemble de nœuds avec une configuration donnée, et d'y exécuter des machines virtuelles. Les machines virtuelles sont introduites afin de simuler des environnements de cloud. Nous avons ajouté des paramètres $p_{i,j}^{min}$ et $p_{i,j}^{max}$ sur les machines virtuelles. La classe VM dans EnerSim 1 prend en compte l'énergie.

D'autre classes telles que EnerJob, Migration et OverReservation ont été ajoutées. Enersim 1 permet également aux utilisateurs d'exporter les résultats sous le format cvs ou xls et facilite ainsi la génération de graphiques. Enersim 1 a une interface graphique complète qui peut être utilisé pour configurer la simulation. Une limite de Enersim est qu'il ne peut que simuler des algorithmes centralisés.

Ensuite, EnerSim 1 a été étendue avec la possibilité de simuler des algorithmes décentralisés et surtout dans le cas des ordonnancements coopératifs. Le simulateur Enersim 2 étend MagateSim et Enersim 1. Le simulateur MagateSim dont apporte les propriétés décentralisées et coopératives. MaGateSim étendu avec l'ajout de propriétés qui lui permettent de tenir compte de l'énergie et de la migration. EnerSim 2 étend MaGateSim en lui permettant de gérer des VM mais surtout l'énergie consommer par les ressources. Nous avons ajouté l'interface IJob qui est implémente la classe de MagateSim Job. Ainsi, nous créons la classe EnerJobII de qui héritent de la classe de EnerJob de Enersim 1 et implémente l'interface IJob. La nouvelle classe simResourceInfo donne des informations sur les ressources. La classe

AdvancedPolicy étend la classe AllocPolicy de Enersim 1.

2.5 Résultats.

Nous avons évalué EACAB et CSAAC qui sont des systèmes de gestion d'énergie efficace pour les IaaS de cloud dans les centres de données virtualisés à grande échelle .

La simulation a été réalisée dans un environnement de 100 clusters contenant 100 nœuds chacun. 1000 tâches sont générées aléatoirement avec une durée d'exécution entre 10 et 40 ans. Nous supposons qu'un nœuds j dans le site i a deux états : Allumé et éteint. Lorsqu'un nœud est allumé, sa consommation d'énergie est fonction de la charge et est entre $P_{i,j}^{min}$ et $P_{i,j}^{max}$. Les expériences ont montré que l'algorithme produit un gain d'énergie qui peut atteindre 50% par rapport à un algorithme standard First Fit (FF) [Borgetto 2012]. Une observation est que cet algorithme est capable de réduire la consommation d'énergie de 15% à 35% lorsque le nombre de tâches augmente 350-1000. EACAB vérifier régulièrement l'existence de nœuds surchargés ou sous-chargés pour migre leurs machines virtuelles afin d'éviter le ralentissement de certains. FF ne vérifie pas la surcharge des nœuds. Quand le nombre de tâches augmente le nombre de noeuds allumés augmente également, ce qui conduit à une consommation d'énergie plus élevée. Cela est particulièrement vrai s'il n'y a pas de migration après le placement initial des tâches. Le gain d'énergie de notre algorithme augmente avec le nombre de tâches car la migration est activée. Le nombre de migrations est faible dans les deux EACAB et FF au début de l'expérience. Les bons résultats de EACAB viennent du fait de l'augmentation des possibilités pour faire migrer des tâches quand le nombre de tâches augmente. Il répartit mieux les tâches sur les nœuds, et réduit le nombre de nœuds allumé. Avec le seuil de sous charge définit, il est possible de réduire encore le nombre de noeuds sous tension, mais ceci peut entrainer la surcharge de noeuds restants. Ces nœuds seraient devenus des points chauffages et auraient un impact négatif sur le refroidissement. C'est pourquoi nous avons défini un autre seuil de surcharge. Dans l'ensemble, l'EACAB peut allouer efficacement les VMs avec un gain d'énergie

importante.

Le CSAAC a été largement évalué en utilisant des tâches de Grid'5000 et a montré son évolutivité et son efficacité énergétique. En particulier, nos résultats expérimentaux ont montré que : (1) le CSAAC est capable d'obtenir un placement des tâches à faible consommation d'énergie en un temps optimimum; (2) la performance des applications n'est pas affectée en effectuant des migrations avec l'utilisation de seuils de sous-charge ou de surcharge; (3) le système s'adapte parfaitement avec un nombre croissant de ressources le rendant ainsi apte à gérer à grande échelle des centres de données virtualisés ; (4) La technique d'anti-équilibrage de charge utilisé par les deux approches permet de réaliser des économies d'énergie substantielles; (5) quand le nombre de $t\hat{a}ches > 300$, le CSAAC est capable de réduire la consommation d'énergie moyenne d'environ 10 à 80% . (5) le temps d'exécution des travaux est également réduite de 5 à 25% lorsque le nombre de $t\hat{a}ches > 300$, par rapport à l'algorithme CASA

Enfin, nous avons comparé les deux algorithmes et pouvons obtenir la conclusion évidente que EACAB et CSAAC peuvent réduire l'énergie consommée des centres de données.

Chapter 3

Conclusion

Contents

Cette thèse examine l'ordonnancement de tâches dans le cloud en développant des approches de solution au problème de consommation d'énergie dans le cloud. Notre objectif est de concevoir l'ordonnancement dans le cloud en tenant compte de l'énergie utilisant l'algorithme anti équilibrage de charge capable de coordonner les comportements de l'ordonnancement des entités indépendantes dans le cloud. La solution développée vise principalement à réduire l'énergie consommée. Elle est basée sur l'ordonnancement centralisée et décentralisée qui est adéquate pour le cloud.

Cette thèse essaie d'améliorer la compréhension de la modélisation du problème d'ordonnancement dans l'environnement Cloud pour réduire la consommation d'énergie. Une orientation était d'étudier les modèles pour y inclure l'allocation des ressources en tenant compte de l'énergie. Il y a une demande croissante de la puissance de calcul de l'industrie et des universitaires, qui a conduit à la consommation de puissance extrême. Nombres d'initiatives ont été prises dans le développement de matériel éconergétique. La consommation globale d'énergie cependant, continue de croître en raison des exigences écrasantes pour les ressources informatiques et les centres de données. L'utilisation de la consommation de la puissance d'une manière inefficace finira par conduire à des problèmes critiques tels que l'insuffisance ou le dysfonctionnement du système de refroidissement. La forte consommation d'énergie conduit à générer une quantité substantielle de dioxyde de carbone. L'architecture

proposée dans cette thèse considère la décision d'ordonnancement basée sur l'utilisation des ressources et de l'énergie consommée. Nous avons étendu le modèle d'ordonnancement pour l'allocation des ressources en tenant compte à la fois de la consolidation (pour éteindre noeuds) et des propriétés des tâches en vue de réduire la consommation totale d'énergie. Dans le chapitre 3 de la partie 1, les algorithmes de consolidation dynamique de VM ont été analysés. Le chapitre a conclu par une discussion sur les avantages potentiels d'algorithmes d'ordonnancement décentralisées. Cette thèse a proposé et étudié une série de nouvelles techniques pour la gestion distribuée de la consolidation dynamique de VM dans les IaaS de cloud sous contraintes de QoS. L'approche proposée améliore l'utilisation des ressources des centres de données et réduit la consommation d'énergie, tout en satisfaisant les exigences de qualité de service définis.

Ce chapitre récapitule les contributions principales de ce travail et présente des orientations futures de recherche.

3.1 Résumé contributions

Le Cloud est devenu un autre mot à la mode après le Web 2.0. Cependant, il y a des dizaines de définitions différentes pour le Cloud Computing et il semble y avoir aucun consensus sur ce qu'est un cloud. D'autre part, le Cloud n'est pas un concept entièrement nouveau; il dispose d'une connexion complexe au paradigme grille de calcul relativement nouveau, et d'autres technologies pertinentes, telles que l'informatique utilitaire, les clusters, et les systèmes distribués en général. Le Cloud a récemment émergé comme un nouveau paradigme informatique qui permet aux clients de louer des services basés sur le modèle "pay-as-you-go". Les clients sont facturés pour seulement ce qu'ils utilisent. Pour supporter les demandes de services croissant des clients, les fournisseurs de cloud construisent actuellement un nombre croissant de centres de données à grande échelle. La gestion de tels centres de données est une tâche stimulante. Elle implique la conception de nouvelles bases de gestion du cloud et des algorithmes qui peuvent baisser la consommation d'énergie de centre de données pendant les périodes de basse utilisation de ressource.

Cette thèse nous a permis de concevoir, mettre en oeuvre et évaluer un système de placement des tâches à haut rendement énergétique pour les cloud.

- **Algorithme "Anti load-Balacing"** . Dans cette thèse, nous avons analysé et développé un modèle d'ordonnancement qui est l'inverse de l'équilibrage de charge. Avec cet algorithme les VMs sont concentrées dans un minimum de noeuds. Cette concentration de la charge ou l'anti-équilibrage de charge permet d'économiser de l'énergie consommée par les nœuds, mais peut dégrader les performances des nœuds et potentiellement augmenter leur la consommation d'énergie.

- **Algorithm d'ordonnancement "Anti load-Balacing" dans le Cloud tenant compte de l'énergie.** Un algorithme centralisée de consolidation dynamique de VM a été proposé et permetd'obtenir un gain d'énergie significative. L'algorithme est basé sur un modèle de crédit défini dans Comet. Nous pouvons affirmer que les économies d'énergie sont atteintes par la consolidation continue de machines virtuelles en fonction de l'utilisation actuelle des ressources.

- **Algorithm cooperatif d'ordonnancement "Anti load-Balacing" dans le Cloud tenant compte de l'énergie**. Une consolidation dynamique distribuée de VMs consistant à diviser le problème en 4 sous-problèmes: (1) la détection de sous charge de noeud; (2)détection de surcharge de noeud; (3) la sélection VM, et (4) placement VM. Les gestionnaires locaux éliminent le point de défaillance unique et rendent le système totalement distribué et décentralisé. Comparé avec d'autres travaux de recherche, notre algorithme permet de réaliser des économies d'énergie tout en menant à bien l'exécution des tâches.

  Nous avons distribué cet algorithme, de sorte que chaque cluster puisse échanger des tâches, en fonction de leurs charges respectives. Nous avons présenté et évalué cette approche décentralisée pour le cloud. Il prend en compte l'énergie et fournit une amélioration de l'efficacité énergétique par rapport aux algorithmes de déséquilibrage de charge classiques. Notre principal problème était

d'optimiser la consommation d'énergie suivant des contraintes de performance des VMs. La consommation d'énergie est à prendre au sens large en essayant d'éviter des points chauds pour réduire l'impact sur le refroidissement. Dans l'ensemble, l'approche proposée permet de placer efficacement les VMs avec un gain d'énergie importante.

Ainsi, nous avons évalué CSAAC en utilisant la simulation. Les résultats de la simulation ont montré que notre algorithme est capable d'effectuer des ordonnancements à faible consommation d'énergie en utilisant moins de temps. En particulier, nous avons montré que dans le cas où le nombre de tâche est supérieur à 300, notre algorithme est capable de réduire la consommation d'énergie moyenne d'environ 10% à 80%. Dans le même temps, le temps d'exécution des tâches est également réduit de 5% à 25% lorsque le nombre de tâches est supérieur à 300.

Les conclusions tirées de ces résultats indiquent que les approches proposées d'ordonnancement nécessitent plus de travail, mais sont par ailleurs prometteuses en ce qui concerne l'efficacité énergétique.

- Nous avons développé pour l'approche centralisée un simulateur EnerSim 1 cité ci-dessus. EnerSim 1 étend ALEA et CloudSim. Afin de simuler l'ordonnancement coopérative, nous avons étendu MaGateSim et EnerSim 1 par l'ajout de propriétés permettent de tenir compte de l'énergie et de la migration.

  En outre, nous avons effectué des expériences consistant à comparer l'approche centralisée et la consolidation VM dynamique distribué. Des expériences menées sur une topologie de Grid5000 ont montré que la consolidation dynamique VM est capable de réduire la consommation d'énergie par les nœuds de calcul jusqu'à 30%.

3.2 Perspectives

Comme nous venons de le voir dans la section 3.1 , les résultats obtenus nous ont donné une motivation suffisante pour envisager la suite en tant que futurs sujets de recherche à étudier dans le prolongement de cette thèse.

Nous prévoyons d'étendre l'hétérogénéité de notre modèle de performance à d'autres parties de l'IaaS de cloud. Nous tenons à soutenir pleinement l'hétérogénéité des machines virtuelles, les machines physiques (PMs) et les tâches des utilisateurs, plus précisément, les machines virtuelles sont variée en termes de mémoire, disque, et de CPU; les PMs peuvent différer en terme de capacité de calcul, de réseau, de mémoire, de disque, Graphics Processing Unit (GPU) et d'hyperviseur (VMM); les tâches d'un utilisateur peuvent demander plusieurs ressources différentes pour différentes quantités de temps.

Malgré d'importantes contributions permettant d'augmenter l'efficacité énergétique, il y a un certain nombre de défis en matière de recherche ouverts qui doivent être abordés afin de faire progresser le domaine.

- **Topologies de réseaux de VM**

  Dans les centres de données virtualisés les VMs communiquent souvent entre elles, créant des topologies de réseaux virtuels. Toutefois, en raison des migrations de machines virtuelles ou l'allocation non optimale, les machines virtuelles communiquant peuvent se retrouver hébergés sur des nœuds physiques logiquement éloignés conduisant à des transferts de données coûteuses entre eux. Si les machines virtuelles communiquant sont allouées à des machines dans différents racks ou boîtiers, la communication réseau peut impliquer des commutateurs qui consomment des quantités importantes d'énergie. Pour éliminer ce transfert de données supplémentaires et de réduire la consommation d'énergie, il est nécessaire d'observer la communication entre les machines virtuelles et placer les machines virtuelles qui communiquent sur les mêmes nœuds ou nœuds étroitement situés. En particulier, pour fournir des réallocations efficaces, il est nécessaire de développer des modèles de consommation d'énergie des périphériques réseau et estimer le coût de transfert de données

en fonction du volume de trafic. En outre, les migrations de VM consomment de l'énergie supplémentaire et ont un impact négatif sur la performance. L'algorithme de placement dde VMs doit veiller à ce que le coût de la migration ne dépasse pas les bénéfices.

- **Seuil dynamique**

  Les valeurs fixes du seuil d'utilisation ne sont pas adaptées pour un environnement avec des charges de travail dynamiques et imprévisibles, dont les différents types d'applications peuvent partager un nœud physique. Le système devrait être capable d'ajuster automatiquement son comportement en fonction des modèles de charge de travail exposés par les applications. Il sera interéssant de présenter un algorithme pour l'auto-ajustement du seuil.

- **SLAs dynamiques et hétérogènes**

  Les centres de données de cloud doivent fournir des garanties de qualité de service strictes, qui sont documentés sous la forme de SLA. L'approvisionnement des ressources au sein d'un centre de données influe directement si les SLA sont remplies. Les clients ont des exigences très différentes, qui peuvent également varier dans le temps. Par exemple, une organisation qui utilise les services de Cloud peut exiger des garanties plus strictes de temps de réponse dans la journée que dans la nuit. Pour résoudre le problème, il est nécessaire de développer des algorithmes qui exploitent la variation du temps de SLA de l'utilisateur afin de minimiser le nombre de serveurs physiques requises.

  Actuellement, pour simplifier la gestion, les ressources des centres de données de cloud sont attribués aux clients en fonction uniquement des SLAs de ce client, quels que soient les SLAs d'autres utilisateurs. Les différences intrinsèques entre les différents SLA peuvent faire de grandes différences dans le nombre de ressources affectées à chaque utilisateur. Il est important de concevoir et analyser des algorithmes qui font usage de cette hétérogénéité. En outre, pour répondre aux besoins des centres de données de cloud à grande échelle, il est nécessaire de concevoir une solution évolutive pour gérer des milliers d'utilisateurs.

- **Consolidation dynamique de VM tenant compte de la température**
  Une partie importante de l'énergie électrique consommée par les ressources informatiques est transformée en chaleur. Une température élevée conduit à un certain nombre de problèmes, tels que la fiabilité du système et réduit la disponibilité, ainsi que la diminution de la durée de vie du matériel. Afin de garder les composants du système au sein de leur température de fonctionnement et éviter les pannes et les accidents, la chaleur émise doit être dissipée. Le problème de refroidissement devient extrêmement important pour les serveurs qui emballent des ressources informatiques à haute densité et compliquent la dissipation de chaleur. Par conséquent, en plus des améliorations de matériel, il est indispensable de traiter le problème de l'optimisation du refroidissement le fonctionnement du système. Une des façons de minimiser les coûts d'exploitation du refroidissement est de surveiller en permanence l'état thermique de nœuds physiques et de réaffecter les machines virtuelles d'un noeud quand il devient surchauffé. Dans ce cas, le système de refroidissement du noeud déchargé peut être ralenti permettant la dissipation de la chaleur naturelle. En outre, il y a eu des travaux de recherche sur la modélisation de la topologie thermique d'un centre de données qui peuvent conduire au placement de la charge de travail plus efficace [Johnson 2009]. Par conséquent, il est nécessaire de savoir comment et quand réaffecter les machines virtuelles afin de minimiser la puissance absorbée par le système de refroidissement, tout en préservant la température en toute sécurité des ressources et en minimisant les frais de la migration et de la dégradation de la performance.

Appendix A

Annex

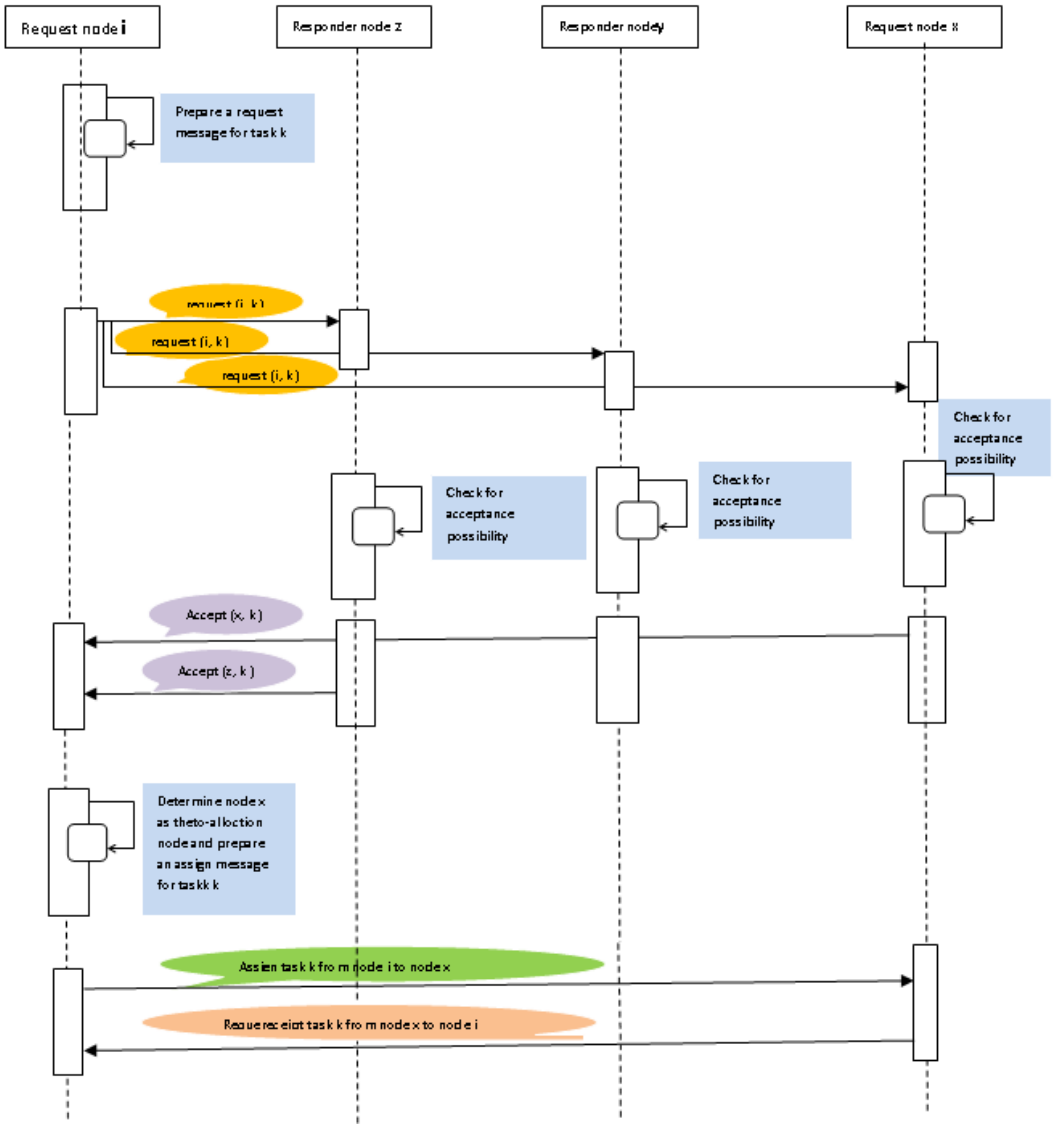

Figure A.1: Submission phase : Task distribution process

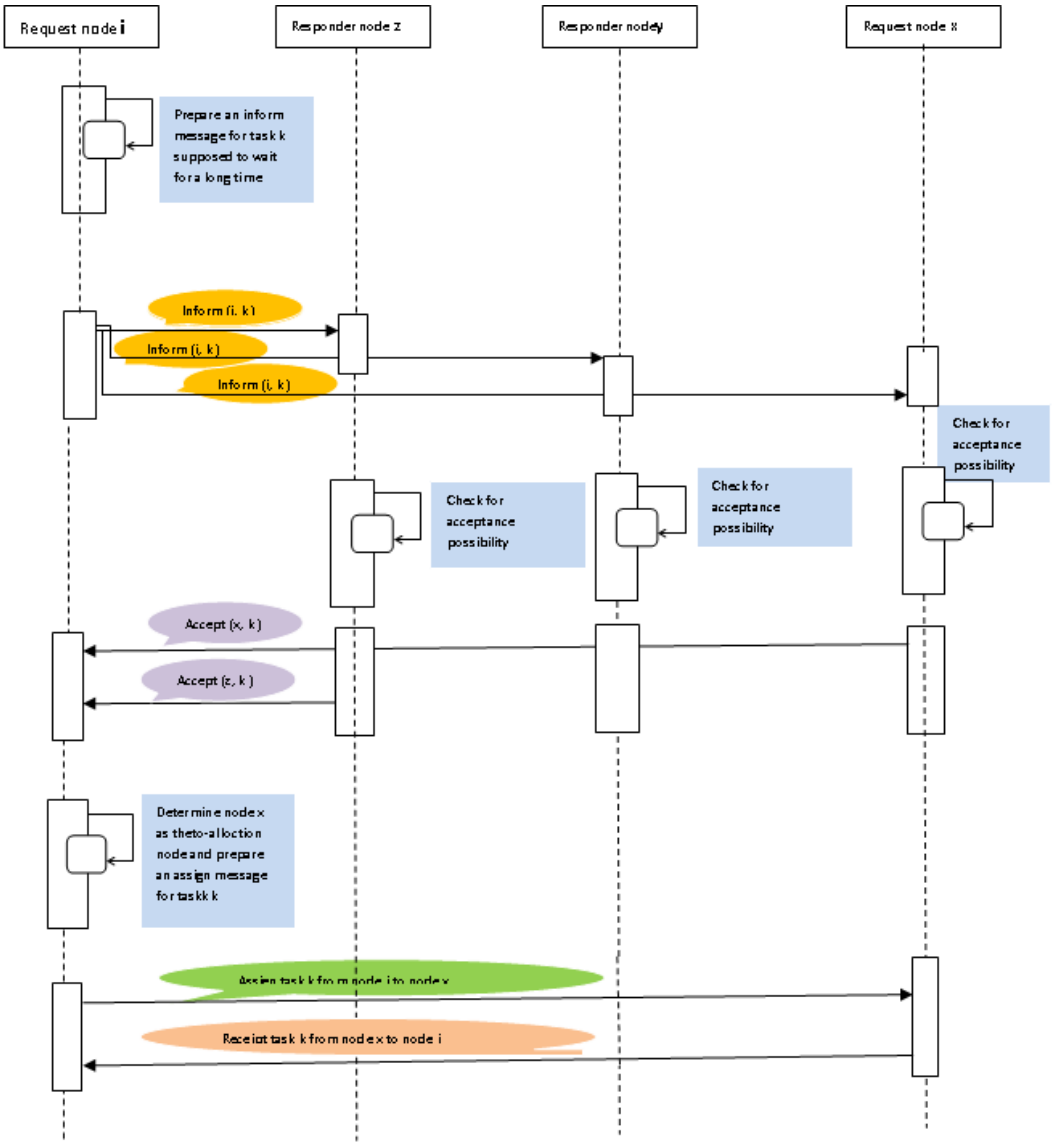

Figure A.2: Dynamic scheduling phase : Task distribution process

Publications

- Books parts
  - **Pragati Agrawal, Damien Borgetto, Carmela Comito, Georges Da Costa, Jean-Marc Pierson, Payal Prakash, Shrisha Rao, Domenico Talia, Cheikhou Thiam, Paolo Trufio**. *Chap. 8 Scheduling and Resource Allocation. Dans : Large-Scale Distributed Systems and Energy Efficiency: A holistic view.* Jean-Marc Pierson (Eds.), **in Wiley**, 8, 2014 (à paraître).
- International conferences articles
  - **Cheikhou Thiam, Georges Da Costa, Jean-Marc Pierson**. *Energy aware clouds scheduling using anti-load balancing algorithm : EACAB* (short paper). Dans : International Conference on Smart Grids and Green IT Systems (SMARTGREENS 2014), Barcelona, SPAIN, 03/04/2014-04/04/2014, INSTICC, (Eds.), **INSTICC - Institute for Systems and Technologies of Information, Control and Communication**, 2014 (paru).
  - **Cheikhou Thiam, Georges Da Costa, Jean-Marc Pierson.** *Cooperative Scheduling Anti-load balancing Algorithm for Cloud : CSAAC* (short paper). Dans : IEEE International Conference on Cloud Computing Technology and Science, Bristol, UK, 02/12/2013-05/12/2013, (paru).
  - **Cheikhou Thiam, Georges Da Costa**. *Anti-Load Balancing to Reduce Energy Consumption* (student paper). Dans : International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering, Ajaccio-Corsica-France, 13/01/2011-15/04/2011, P Iványi, B H V Topping (Eds.), Civil-Comp Proceedings, (en ligne), avril 2011. Accès : **http://www.ctresources.info/ccp/paper.html?id=6278**

Bibliography

[Adiga 2002] Narasimha R Adiga, George Almási, George S Almasi, Y Aridor, Rajkishore Barik, D Beece, R Bellofatto, G Bhanot, R Bickford, M Blumrich*et al.* *An overview of the BlueGene/L supercomputer.* In Supercomputing, ACM/IEEE 2002 Conference, pages 60–60. IEEE, 2002. (Cited on page 63.)

[AlEnawy 2005] Tarek A AlEnawy and Hakan Aydin. *Energy-aware task allocation for rate monotonic scheduling.* In Real Time and Embedded Technology and Applications Symposium, 2005. RTAS 2005. 11th IEEE, pages 213–223. IEEE, 2005. (Cited on pages 61 and 67.)

[Amazon 2010] EC Amazon. *Amazon elastic compute cloud (Amazon EC2).* Amazon Elastic Compute Cloud (Amazon EC2), 2010. (Cited on page 16.)

[Armbrust 2010] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica*et al.* *A view of cloud computing.* Communications of the ACM, vol. 53, no. 4, pages 50–58, 2010. (Cited on page 28.)

[Armstrong 1998] Robert Armstrong, Debra Hensgen and Taylor Kidd. *The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions.* In Heterogeneous Computing Workshop, 1998.(HCW 98) Proceedings. 1998 Seventh, pages 79–87. IEEE, 1998. (Cited on page 41.)

[Arora 2002] Manish Arora, Sajal K Das and Rupak Biswas. *A de-centralized scheduling and load balancing algorithm for heterogeneous grid environments.* In Parallel Processing Workshops, 2002. Proceedings. International Conference on, pages 499–505. IEEE, 2002. (Cited on pages 45 and 67.)

[Aupy 2012] Guillaume Aupy, Anne Benoit and Yves Robert. *Energy-aware scheduling under reliability and makespan constraints.* In High Performance

Computing (HiPC), 2012 19th International Conference on, pages 1–10. IEEE, 2012. (Cited on pages 63 and 67.)

[Axelrod 1984] Rcjbert Axelrod. *THE EVOLUTION 0P COOPERATION*. 1984. (Cited on page 18.)

[Azevedo 2012] Dan Azevedo, Disney Alan French and Emerson Network Power. *PUE^{TM}: A COMPREHENSIVE EXAMINATION OF THE METRIC*. 2012. (Cited on page 58.)

[Baliga 2011] Jayant Baliga, Robert WA Ayre, Kerry Hinton and Rodney S Tucker. *Green cloud computing: Balancing energy in processing, storage, and transport*. Proceedings of the IEEE, vol. 99, no. 1, pages 149–167, 2011. (Cited on page 61.)

[Barham 2003] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt and Andrew Warfield. *Xen and the art of virtualization*. ACM SIGOPS Operating Systems Review, vol. 37, no. 5, pages 164–177, 2003. (Cited on page 16.)

[Bartolini 2013] Andrea Bartolini, Matteo Cacciari, Andrea Tilli and Luca Benini. *Thermal and energy management of high-performance multicores: Distributed and self-calibrating model-predictive controller*. 2013. (Cited on page 61.)

[Bein 2010] Doina Bein, Wolfgang Bein and Shashi Phoha. *Efficient data centers, cloud computing in the future of distributed computing*. In Information Technology: New Generations (ITNG), 2010 Seventh International Conference on, pages 70–75. IEEE, 2010. (Cited on pages 52 and 67.)

[Bejerano 2004] Yigal Bejerano, Seung-Jae Han and Li Erran Li. *Fairness and load balancing in wireless LANs using association control*. In Proceedings of the 10th annual international conference on Mobile computing and networking, pages 315–329. ACM, 2004. (Cited on page 18.)

[Beloglazov 2012] Anton Beloglazov, Jemal Abawajy and Rajkumar Buyya. *Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing.* Future Generation Computer Systems, vol. 28, no. 5, pages 755–768, 2012. (Cited on pages 51 and 67.)

[Berkeley 2013] Berkeley. *The Energy Efficiency Potential of Cloud-Based Software: A U.S. Case Study.* 2013. (Cited on page 59.)

[Bianchini 2004] Ricardo Bianchini and Ramakrishnan Rajamony. *Power and energy management for server systems.* Computer, vol. 37, no. 11, pages 68–76, 2004. (Cited on page 54.)

[Borgetto 2012] Damien Borgetto, Michael Maurer, Georges Da-Costa, Jean-Marc Pierson and Ivona Brandic. *Energy-efficient and sla-aware management of iaas clouds.* In Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet, page 25. ACM, 2012. (Cited on pages 115 and 156.)

[Braun 2001] Tracy D Braun, Howard Jay Siegel, Noah Beck, Ladislau L Bölöni, Muthucumaru Maheswaran, Albert I Reuther, James P Robertson, Mitchell D Theys, Bin Yao, Debra Hensgen*et al. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems.* Journal of Parallel and Distributed computing, vol. 61, no. 6, pages 810–837, 2001. (Cited on page 50.)

[Brown 2008] Richard Brown*et al. Report to congress on server and data center energy efficiency: Public law 109-431.* 2008. (Cited on page 56.)

[Buyya 2002a] Rajkumar Buyya, David Abramson, Jonathan Giddy and Heinz Stockinger. *Economic models for resource management and scheduling in grid computing.* Concurrency and computation: practice and experience, vol. 14, no. 13-15, pages 1507–1542, 2002. (Cited on pages 49 and 67.)

[Buyya 2002b] Rajkumar Buyya and Manzur Murshed. *Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling*

for grid computing. volume 14, pages 1175–1220. Wiley Online Library, 2002. (Cited on page 105.)

[Buyya 2005] Rajkumar Buyya, David Abramson and Srikumar Venugopal. *The grid economy.* Proceedings of the IEEE, vol. 93, no. 3, pages 698–714, 2005. (Cited on page 49.)

[Calheiros 2011] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose and Rajkumar Buyya. *CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms.* Software: Practice and Experience, vol. 41, no. 1, pages 23–50, 2011. (Cited on pages 72, 105 and 106.)

[Cantú-Paz 1998] Erick Cantú-Paz. *A survey of parallel genetic algorithms.* Calculateurs paralleles, reseaux et systems repartis, vol. 10, no. 2, pages 141–171, 1998. (Cited on page 33.)

[Carlson 2012] Andrew B Carlson. *Data center cooling*, February 14 2012. US Patent 8,113,010. (Cited on page 137.)

[Casanova 2008] Henri Casanova, Arnaud Legrand and Martin Quinson. *Simgrid: A generic framework for large-scale distributed experiments.* In Computer Modeling and Simulation, 2008. UKSIM 2008. Tenth International Conference on, pages 126–131. IEEE, 2008. (Cited on page 105.)

[Chase 2001] Jeffrey S Chase, Darrell C Anderson, Prachi N Thakar, Amin M Vahdat and Ronald P Doyle. *Managing energy and server resources in hosting centers.* In ACM SIGOPS Operating Systems Review, volume 35, pages 103–116. ACM, 2001. (Cited on page 62.)

[Chedid 2002] Wissam Chedid and Chansu Yu. *Survey on power management techniques for energy efficient computer systems.* Laboratory Report. Mobile Computing Research Lab, 2002. (Cited on page 64.)

[Chen 2011] Dong Chen, Noel A Eisley, Philip Heidelberger, Robert M Senger, Yutaka Sugawara, Sameer Kumar, Valentina Salapura, David L Satterfield,

Burkhard Steinmacher-Burow and Jeffrey J Parker. *The IBM Blue Gene/Q interconnection network and message unit.* In High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for, pages 1–10. IEEE, 2011. (Cited on page 64.)

[Chow 2002] Ka-Po Chow and Yu-Kwong Kwok. *On load balancing for distributed multiagent computing.* Parallel and Distributed Systems, IEEE Transactions on, vol. 13, no. 8, pages 787–801, 2002. (Cited on page 82.)

[Christodoulou 2004] George Christodoulou, Elias Koutsoupias and Akash Nanavati. *Coordination mechanisms.* In Automata, Languages and Programming, pages 345–357. Springer, 2004. (Cited on page 17.)

[Chryssolouris 2001] George Chryssolouris and Velusamy Subramaniam. *Dynamic scheduling of manufacturing job shops using genetic algorithms.* Journal of Intelligent Manufacturing, vol. 12, no. 3, pages 281–293, 2001. (Cited on page 40.)

[Clarke 2002] Ian Clarke, Scott G Miller, Theodore W Hong, Oskar Sandberg and Brandon Wiley. *Protecting free expression online with Freenet.* Internet Computing, IEEE, vol. 6, no. 1, pages 40–49, 2002. (Cited on pages 21 and 22.)

[Comito 2011] Carmela Comito, Deborah Falcone, Domenico Talia and Paolo Trunfio. *Energy Efficient Task Allocation over Mobile Networks.* In Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on, pages 380–387. IEEE, 2011. (Cited on pages 65 and 67.)

[Condor2013 2013] Condor2013. *Condor2013.* 2013. (Cited on page 48.)

[dcd 2013] dcd. *THE DCDI 2013 CENSUS – UK FIGURES.* 2013. (Cited on page 59.)

[dcd 2014] dcd. *15% GROWTH FORECAST FOR NORTH AMERICA COLOCATION MARKET 2014.* 2014. (Cited on page 59.)

[Demers 1989] Alan Demers, Srinivasan Keshav and Scott Shenker. *Analysis and simulation of a fair queueing algorithm.* In ACM SIGCOMM Computer Communication Review, volume 19, pages 1–12. ACM, 1989. (Cited on page 43.)

[Denning 1965] Peter J Denning. *Queueing models for file memory operation.* Rapport technique, DTIC Document, 1965. (Cited on page 42.)

[Dias de Assunção 2008] Marcos Dias de Assunção, Rajkumar Buyya and Srikumar Venugopal. *InterGrid: A case for internetworking islands of Grids.* Concurrency and Computation: Practice and Experience, vol. 20, no. 8, pages 997–1024, 2008. (Cited on page 65.)

[Duy 2010] Truong Vinh Truong Duy, Yukinori Sato and Yasushi Inoguchi. *Performance evaluation of a green scheduling algorithm for energy savings in cloud computing.* In Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on, pages 1–8. IEEE, 2010. (Cited on page 55.)

[Ellis 1999] Carla Schlatter Ellis. *The case for higher-level power management.* In Hot Topics in Operating Systems, 1999. Proceedings of the Seventh Workshop on, pages 162–167. IEEE, 1999. (Cited on page 63.)

[Emeneker 2007] Wesley Emeneker and Dan Stanzione. *Efficient virtual machine caching in dynamic virtual clusters.* In SRMPDS Workshop, ICAPDS 2007 Conference, 2007. (Cited on pages 51 and 67.)

[Fallenbeck 2006] Niels Fallenbeck, Hans-Joachim Picht, Matthew Smith and Bernd Freisleben. *Xen and the art of cluster scheduling.* In Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing, page 4. IEEE Computer Society, 2006. (Cited on pages 52 and 67.)

[Feitelson 1994] Dror G Feitelson. A survey of scheduling in multiprogrammed parallel systems. IBM TJ Watson Research Center, 1994. (Cited on page 43.)

[Feitelson 1995] Dror G Feitelson and Larry Rudolph. Job scheduling strategies for parallel processing: Ipps'95 workshop, santa barbara, ca, usa, april 25, 1995. proceedings. Springer, 1995. (Cited on page 42.)

[Feller 2010] Eugen Feller, Louis Rilling, Christine Morin, Renaud Lottiaux and Daniel Leprince. *Snooze: a scalable, fault-tolerant and distributed consolidation manager for large-scale clusters.* In Proceedings of the 2010 IEEE/ACM Int'l Conference on Green Computing and Communications & Int'l Conference on Cyber, Physical and Social Computing, pages 125–132. IEEE Computer Society, 2010. (Cited on page 89.)

[Flautner 2001] Krisztián Flautner, Steve Reinhardt and Trevor Mudge. *Automatic performance setting for dynamic voltage scaling.* In Proceedings of the 7th annual international conference on Mobile computing and networking, pages 260–271. ACM, 2001. (Cited on page 63.)

[Foster 2008] Ian Foster, Yong Zhao, Ioan Raicu and Shiyong Lu. *Cloud computing and grid computing 360-degree compared.* In Grid Computing Environments Workshop, 2008. GCE'08, pages 1–10. Ieee, 2008. (Cited on page 27.)

[Freund 1998] Richard F Freund, Michael Gherrity, Stephen Ambrosius, Mark Campbell, Mike Halderman, Debra Hensgen, Elaine Keith, Taylor Kidd, Matt Kussow, John D Lima*et al. Scheduling resources in multi-user, heterogeneous, computing environments with SmartNet.* In Heterogeneous Computing Workshop, 1998.(HCW 98) Proceedings. 1998 Seventh, pages 184–199. IEEE, 1998. (Cited on page 41.)

[Fujimoto 2003] Noriyuki Fujimoto and Kenichi Hagihara. *Near-optimal dynamic task scheduling of independent coarse-grained tasks onto a computational grid.* In Parallel Processing, 2003. Proceedings. 2003 International Conference on, pages 391–398. IEEE, 2003. (Cited on page 45.)

[García 2005] Pedro García, Carles Pairot, Rubén Mondéjar, Jordi Pujol, Helio Tejedor and Robert Rallo. *Planetsim: A new overlay network simula-*

tion framework. In Software engineering and middleware, pages 123–136. Springer, 2005. (Cited on page 105.)

[Garg 2011] Saurabh Kumar Garg, Chee Shin Yeo, Arun Anandasivam and Rajkumar Buyya. *Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers*. Journal of Parallel and Distributed Computing, vol. 71, no. 6, pages 732–749, 2011. (Cited on page 52.)

[Geelan 2009] Jeremy Geelan. *Twenty-one experts define cloud computing*. Cloud Computing Journal, vol. 2, pages 1–5, 2009. (Cited on page 27.)

[Goldberg 1989] David Goldberg. *Genetic Algorithms in optimization, search and machine learning*. Addison Wesley, New York. Eiben AE, Smith JE (2003) Introduction to Evolutionary Computing. Springer. Jacq J, Roux C (1995) Registration of non-segmented images using a genetic algorithm. Lecture notes in computer science, vol. 905, pages 205–211, 1989. (Cited on page 28.)

[Gomez 1997] Faustino Gomez and Risto Miikkulainen. *Incremental evolution of complex general behavior*. Adaptive Behavior, vol. 5, no. 3-4, pages 317–342, 1997. (Cited on page 20.)

[Gomez 1999] Faustino J Gomez and Risto Miikkulainen. *Solving non-Markovian control tasks with neuroevolution*. In IJCAI, volume 99, pages 1356–1361, 1999. (Cited on page 20.)

[GoogleAppEngine 2013] GoogleAppEngine. *GoogleAppEngine*. 2013. (Cited on pages 16 and 28.)

[Green500List2013 2013] Green500List2013. *Green500List2013*. 2013. (Cited on page 64.)

[Grid50002010 2010] Grid50002010. *Grid50002010*. 2010. (Cited on pages 115 and 116.)

[Guerout 2013] Tom Guerout and Mahdi Ben Alaya. *Autonomic energy-aware tasks scheduling*. In Enabling Technologies: Infrastructure for Collaborative En-

terprises (WETICE), 2013 IEEE 22nd International Workshop on, pages 119–124. IEEE, 2013. (Cited on page 33.)

[Hamscher 2000] Volker Hamscher, Uwe Schwiegelshohn, Achim Streit and Ramin Yahyapour. *Evaluation of job-scheduling strategies for grid computing.* In Grid Computing—GRID 2000, pages 191–202. Springer, 2000. (Cited on pages 64 and 67.)

[Hazewinkel 2001] Michiel Hazewinkel. *Greedy algorithm.* In Encyclopedia of Mathematics (2001). Springer, 2001. (Cited on page 41.)

[Helmbold 1996] David P Helmbold, Darrell DE Long and Bruce Sherrod. *A dynamic disk spin-down technique for mobile computing.* In Proceedings of the 2nd annual international conference on Mobile computing and networking, pages 130–142. ACM, 1996. (Cited on page 63.)

[Hermenier 2009] Fabien Hermenier, Xavier Lorca, Jean-Marc Menaud, Gilles Muller and Julia Lawall. *Entropy: a consolidation manager for clusters.* In Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pages 41–50. ACM, 2009. (Cited on page 88.)

[Hintemann 2010] Ralph Hintemann and Holger Skurk. *Energieeffizienz im Rechenzentrum.* In Green-IT, Virtualisierung und Thin Clients, pages 19–56. Springer, 2010. (Cited on page 62.)

[Hofbauer 1998] Josef Hofbauer and Karl Sigmund. Evolutionary games and population dynamics. Cambridge University Press, 1998. (Cited on page 19.)

[Hoffa 2008] Christina Hoffa, Gaurang Mehta, Timothy Freeman, Ewa Deelman, Kate Keahey, Bruce Berriman and John Good. *On the use of cloud computing for scientific workflows.* In eScience, 2008. eScience'08. IEEE Fourth International Conference on, pages 640–645. IEEE, 2008. (Cited on page 88.)

[Hosting 2014] AT&T Synaptic Hosting. *AT&T Synaptic Hosting.* 2014. (Cited on page 16.)

[Howell 1998] Fred Howell and Ross McNab. *SimJava: A discrete event simulation library for java.* Simulation Series, vol. 30, pages 51–56, 1998. (Cited on page 106.)

[Huai 2007] Jinpeng Huai, Qin Li and Chunming Hu. *CIVIC: a hypervisor based virtual computing environment.* In Parallel Processing Workshops, 2007. ICPPW 2007. International Conference on, pages 51–51. IEEE, 2007. (Cited on page 14.)

[Huang 2009] Ye Huang, Amos Brocco, Michele Courant, Beat Hirsbrunner and Pierre Kuonen. *MaGate Simulator: a simulation environment for a decentralized grid scheduler.* In Advanced Parallel Processing Technologies, pages 273–287. Springer, 2009. (Cited on pages 66 and 105.)

[Huang 2013a] Lu Huang, Hai-shan Chen and Ting-ting Hu. *Survey on Resource Allocation Policy and Job Scheduling Algorithms of Cloud Computing.* Journal of Software (1796217X), vol. 8, no. 2, 2013. (Cited on page 33.)

[Huang 2013b] Ye Huang, Nik Bessis, Peter Norrington, Pierre Kuonen and Beat Hirsbrunner. *Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm.* Future Generation Computer Systems, vol. 29, no. 1, pages 402–415, 2013. (Cited on pages 10, 53, 66 and 67.)

[Izal 2004] Mikel Izal, Guillaume Urvoy-Keller, Ernst W Biersack, Pascal A Felber, Anwar Al Hamra and Luis Garces-Erice. *Dissecting bittorrent: Five months in a torrent's lifetime.* In Passive and Active Network Measurement, pages 1–11. Springer, 2004. (Cited on page 22.)

[Janssen 2013] Cory Janssen. *Techopedia.* 2013. (Cited on page 26.)

[Jeon 2010] Hyeran Jeon, Woo Hyong Lee and Sung Woo Chung. *Load Unbalancing Strategy for Multicore Embedded Processors.* Computers, IEEE Transactions on, vol. 59, no. 10, pages 1434–1440, 2010. (Cited on pages 85 and 151.)

[Jiang 2007] Congfeng Jiang, Cheng Wang, Xiaohu Liu and Yinghui Zhao. *A survey of job scheduling in grids.* In Advances in Data and Web Management, pages 419–427. Springer, 2007. (Cited on page 33.)

[Johnson 2009] Peter Johnson and Tony Marker. *Data centre energy efficiency product profile.* Pitt & Sherry, report to equipment energy efficiency committee (E3) of The Australian Government Department of the Environment, Water, Heritage and the Arts (DEWHA), 2009. (Cited on pages 137 and 165.)

[Kailasam 2010] Sriram Kailasam, Nathan Gnanasambandam, Janakiram Dharanipragada and Naveen Sharma. *Optimizing service level agreements for autonomic cloud bursting schedulers.* In Parallel Processing Workshops (ICPPW), 2010 39th International Conference on, pages 285–294. IEEE, 2010. (Cited on page 52.)

[Keller 2000] Axel Keller and Alexander Reinefeld. Anatomy of a resource management system for hpc clusters. ZIB, 2000. (Cited on page 43.)

[Kim 2011] Kyong Hoon Kim, Anton Beloglazov and Rajkumar Buyya. *Power-aware provisioning of virtual machines for real-time Cloud services.* Concurrency and Computation: Practice and Experience, vol. 23, no. 13, pages 1491–1505, 2011. (Cited on pages 51 and 67.)

[Kliazovich 2012] Dzmitry Kliazovich, Pascal Bouvry and Samee Ullah Khan. *GreenCloud: a packet-level simulator of energy-aware cloud computing data centers.* The Journal of Supercomputing, vol. 62, no. 3, pages 1263–1283, 2012. (Cited on page 61.)

[Klusáček 2008] Dalibor Klusáček, Luděk Matyska and Hana Rudová. *Alea–Grid scheduling simulation environment.* In Parallel Processing and Applied Mathematics, pages 1029–1038. Springer, 2008. (Cited on page 106.)

[Klusáček 2010] Dalibor Klusáček and Hana Rudová. *Alea 3: job scheduling simulator.* In Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, page 61. ICST (Institute for Computer Sciences,

Social-Informatics and Telecommunications Engineering), 2010. (Cited on pages 72, 104 and 105.)

[Konishi 2006] Junjiro Konishi, Naoki Wakamiya and Masayuki Murata. *Proposal and evaluation of a cooperative mechanism for pure P2P file sharing networks.* In Biologically Inspired Approaches to Advanced Information Technology, pages 33–47. Springer, 2006. (Cited on page 24.)

[Kozuch 2002] Michael Kozuch and Mahadev Satyanarayanan. *Internet suspend/resume.* In Mobile Computing Systems and Applications, 2002. Proceedings Fourth IEEE Workshop on, pages 40–46. IEEE, 2002. (Cited on page 16.)

[Krashinsky 2005] Ronny Krashinsky and Hari Balakrishnan. *Minimizing energy for wireless web access with bounded slowdown.* Wireless Networks, vol. 11, no. 1-2, pages 135–148, 2005. (Cited on page 63.)

[Kurowski 2013] K Kurowski, A Oleksiak, W Piątek, T Piontek, A Przybyszewski and J Węglarz. *DCworms–A tool for simulation of energy efficiency in distributed computing infrastructures.* Simulation Modelling Practice and Theory, vol. 39, pages 135–151, 2013. (Cited on page 105.)

[Lai 2008] Kuan-Chou Lai and Chao-Tung Yang. *A dominant predecessor duplication scheduling algorithm for heterogeneous systems.* The Journal of Supercomputing, vol. 44, no. 2, pages 126–145, 2008. (Cited on pages 40 and 67.)

[Li 2010] Jiayin Li, Meikang Qiu, Jianwei Niu, Wenzhong Gao, Ziliang Zong and Xiao Qin. *Feedback dynamic algorithms for preemptable job scheduling in cloud systems.* In Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on, volume 1, pages 561–564. IEEE, 2010. (Cited on page 52.)

[Lifka 1995] David A Lifka. *The anl/ibm sp scheduling system.* In Job Scheduling Strategies for Parallel Processing, pages 295–303. Springer, 1995. (Cited on page 42.)

[Maheswaran 1999] Muthucumaru Maheswaran, Shoukat Ali, HJ Siegal, Debra Hensgen and Richard F Freund. *Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems.* In Heterogeneous Computing Workshop, 1999.(HCW'99) Proceedings. Eighth, pages 30–44. IEEE, 1999. (Cited on pages 41 and 42.)

[Mandal 2005] Anirban Mandal, Ken Kennedy, Charles Koelbel, Gabriel Marin, John Mellor-Crummey, Bo Liu and Lennart Johnsson. *Scheduling strategies for mapping application workflows onto the grid.* In High Performance Distributed Computing, 2005. HPDC-14. Proceedings. 14th IEEE International Symposium on, pages 125–134. IEEE, 2005. (Cited on page 42.)

[Mell 2011] Peter Mell and Timothy Grance. *The NIST definition of cloud computing (draft).* NIST special publication, vol. 800, no. 145, page 7, 2011. (Cited on page 29.)

[Mergen 2006] Mark F Mergen, Volkmar Uhlig, Orran Krieger and Jimi Xenidis. *Virtualization for high-performance computing.* ACM SIGOPS Operating Systems Review, vol. 40, no. 2, pages 8–11, 2006. (Cited on page 12.)

[MicrosoftAzure 2013] MicrosoftAzure. *A view of cloud computing.* 2013. (Cited on page 28.)

[Moab 2012] Moab. *Adaptive Computing, Moab, Moab Adaptive Computing Suite.* 2012. (Cited on page 36.)

[Montresor 2009] Alberto Montresor and Márk Jelasity. *PeerSim: A scalable P2P simulator.* In Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on, pages 99–100. IEEE, 2009. (Cited on page 105.)

[Mtibaa 2007] Abdellatif Mtibaa, Bouraoui Ouni and Mohamed Abid. *An efficient list scheduling algorithm for time placement problem.* Computers & Electrical Engineering, vol. 33, no. 4, pages 285–298, 2007. (Cited on page 39.)

[Mu'alem 2001] Ahuva W. Mu'alem and Dror G. Feitelson. *Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with*

backfilling. Parallel and Distributed Systems, IEEE Transactions on, vol. 12, no. 6, pages 529–543, 2001. (Cited on page 43.)

[Nagothu 2010] KranthiManoj Nagothu, Brian Kelley, Jeff Prevost and Mo Jamshidi. *Ultra low energy cloud computing using adaptive load prediction*. In World Automation Congress (WAC), 2010, pages 1–7. IEEE, 2010. (Cited on pages 61 and 67.)

[Nurmi 2009] Daniel Nurmi, Richard Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff and Dmitrii Zagorodnov. *The eucalyptus open-source cloud-computing system*. In Cluster Computing and the Grid, 2009. CCGRID'09. 9th IEEE/ACM International Symposium on, pages 124–131. IEEE, 2009. (Cited on page 88.)

[ORGERIE 2010] Anne-Cécile ORGERIE and Laurent LEFÈVRE. *When Clouds become Green: the Green Open Cloud Architecture*. 2010. (Cited on pages 62 and 67.)

[Padala 2007] Pradeep Padala, Xiaoyun Zhu, Zhikui Wang, Sharad Singhal, Kang G Shin*et al.* *Performance evaluation of virtualization technologies for server consolidation*. HP Labs Tec. Report, 2007. (Cited on page 15.)

[Pan 2005] Feng Pan, Vincent W Freeh and Daniel M Smith. *Exploring the energy-time tradeoff in high-performance computing*. In Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International, pages 9–pp. IEEE, 2005. (Cited on page 63.)

[Perea 2011] Federico Perea and Huub W De Waard. *Greedy and-Greedy Algorithms for Multidimensional Data Association*. Aerospace and Electronic Systems, IEEE Transactions on, vol. 47, no. 3, pages 1915–1925, 2011. (Cited on page 41.)

[pikeresearch 2013] pikeresearch. *Market research*. 2013. (Cited on page 59.)

[Pinheiro 2001] Eduardo Pinheiro, Ricardo Bianchini, Enrique V Carrera and Taliver Heath. *Load balancing and unbalancing for power and performance in*

cluster-based systems. In Workshop on compilers and operating systems for low power, volume 180, pages 182–195. Barcelona, Spain, 2001. (Cited on pages 62 and 67.)

[Pinheiro 2003] Eduardo Pinheiro, Ricardo Bianchini, Enrique V Carrera and Taliver Heath. *Dynamic cluster reconfiguration for power and performance*. In Compilers and operating systems for low power, pages 75–93. Springer, 2003. (Cited on page 62.)

[Quesnel 2012] Flavien Quesnel and Adrien Lèbre. *Cooperative dynamic scheduling of virtual machines in distributed systems*. In Euro-Par 2011: Parallel Processing Workshops, pages 457–466. Springer, 2012. (Cited on page 89.)

[Quétier 2007] Benjamin Quétier, Vincent Neri and Franck Cappello. *Scalability comparison of four host virtualization tools*. Journal of Grid Computing, vol. 5, no. 1, pages 83–98, 2007. (Cited on page 15.)

[Rahman 2011] Mustafizur Rahman, Rajiv Ranjan, Rajkumar Buyya and Boualem Benatallah. *A taxonomy and survey on autonomic management of applications in grid computing environments*. Concurrency and Computation: Practice and Experience, vol. 23, no. 16, pages 1990–2019, 2011. (Cited on page 33.)

[Ranjan 2006] Rajiv Ranjan, Aaron Harwood and Rajkumar Buyya. *SLA-based coordinated superscheduling scheme for computational Grids*. In Cluster Computing, 2006 IEEE International Conference on, pages 1–8. IEEE, 2006. (Cited on pages 64 and 67.)

[Ranjan 2008] Rajiv Ranjan, Mustafizur Rahman and Rajkumar Buyya. *A decentralized and cooperative workflow scheduling algorithm*. In Cluster Computing and the Grid, 2008. CCGRID'08. 8th IEEE International Symposium on, pages 1–8. IEEE, 2008. (Cited on pages 46 and 67.)

[Ratnasamy 2001] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp and Scott Shenker. A scalable content-addressable network, volume 31. ACM, 2001. (Cited on page 22.)

[Rhea 2001] Sean Rhea, Chris Wells, Patrick Eaton, Dennis Geels, Ben Zhao, Hakim Weatherspoon and John Kubiatowicz. *Maintenance-free global data storage*. Internet Computing, IEEE, vol. 5, no. 5, pages 40–49, 2001. (Cited on page 22.)

[Ripeanu 2001] Matei Ripeanu. *Peer-to-peer architecture case study: Gnutella network*. In Peer-to-Peer Computing, 2001. Proceedings. First International Conference on, pages 99–100. IEEE, 2001. (Cited on pages 21 and 22.)

[Ruth 2005] Paul Ruth, Phil McGachey and Dongyan Xu. *Viocluster: Virtualization for dynamic computational domains*. In Cluster Computing, 2005. IEEE International, pages 1–10. IEEE, 2005. (Cited on page 52.)

[Ruth 2006] Paul Ruth, Junghwan Rhee, Dongyan Xu, Rick Kennell and Sebastien Goasguen. *Autonomic live adaptation of virtual computational environments in a multi-domain infrastructure*. In Autonomic Computing, 2006. ICAC'06. IEEE International Conference on, pages 5–14. IEEE, 2006. (Cited on page 52.)

[Sabin 2003] Gerald Sabin, Rajkumar Kettimuthu, Arun Rajan and Ponnuswamy Sadayappan. *Scheduling of parallel jobs in a heterogeneous multi-site environment*. In Job Scheduling Strategies for Parallel Processing, pages 87–104. Springer, 2003. (Cited on page 45.)

[Sadhasivam 2009] Sudha Sadhasivam, N Nagaveni, R Jayarani and R Vasanth Ram. *Design and implementation of an efficient two-level scheduler for cloud computing environment*. In Advances in Recent Technologies in Communication and Computing, 2009. ARTCom'09. International Conference on, pages 884–886. IEEE, 2009. (Cited on page 52.)

[Salesforce.com 2014] Salesforce.com. *Salesforce.com*. 2014. (Cited on page 16.)

[Scheduler 2013] Maui Scheduler. *Maui Scheduler*. 2013. (Cited on page 36.)

[Schwiegelshohn 1998] Uwe Schwiegelshohn and Ramin Yahyapour. *Analysis of first-come-first-serve parallel job scheduling*. In Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms, pages 629–638. Society for Industrial and Applied Mathematics, 1998. (Cited on page 42.)

[Seiden 2000] Steven S Seiden. *A guessing game and randomized online algorithms*. In Proceedings of the thirty-second annual ACM symposium on Theory of computing, pages 592–601. ACM, 2000. (Cited on page 42.)

[Serban 2013] T Serban, M Danelutto and P Kilpatrick. *Autonomic scheduling of tasks from data parallel patterns to CPU/GPU core mixes*. In High Performance Computing and Simulation (HPCS), 2013 International Conference on, pages 72–79. IEEE, 2013. (Cited on page 33.)

[Sharma 2010] Raksha Sharma, V Kant Soni, M Kumar Mishra and Prachet Bhuyan. *A survey of job scheduling and resource management in Grid Computing*. World Academy of Science, Engineering and Technology, vol. 64, pages 461–466, 2010. (Cited on page 33.)

[Shenai 2012] Sudhir Shenai*et al. Survey on scheduling issues in cloud computing*. Procedia Engineering, vol. 38, pages 2881–2888, 2012. (Cited on page 33.)

[Shin 2006] Seungwon Shin, Jaeyeon Jung and Hari Balakrishnan. *Malware prevalence in the KaZaA file-sharing network*. In Proceedings of the 6th ACM SIGCOMM conference on Internet measurement, pages 333–338. ACM, 2006. (Cited on page 22.)

[Shivle 2004] Sameer Shivle, Ralph Castain, Howard Jay Siegel, Anthony A Maciejewski, Tarun Banka, Kiran Chindam, Steve Dussinger, Prakash Pichumani, Praveen Satyasekaran, William Saylor*et al. Static mapping of subtasks in a heterogeneous ad hoc grid environment*. In Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, page 110. IEEE, 2004. (Cited on page 52.)

[Sih 1993] Gilbert C Sih and Edward A Lee. *A compile-time scheduling heuristic for interconnection-constrained heterogeneous processor architectures.* Parallel and Distributed Systems, IEEE Transactions on, vol. 4, no. 2, pages 175–187, 1993. (Cited on page 40.)

[Skovira 1996] Joseph Skovira, Waiman Chan, Honbo Zhou and David Lifka. *The EASY—LoadLeveler API Project.* In Job Scheduling Strategies for Parallel Processing, pages 41–47. Springer, 1996. (Cited on page 43.)

[Smith 1980] R Smith. *Communication and Control in Problem Solver.* IEEE Transactions on computers, vol. 29, page 12, 1980. (Cited on page 65.)

[Smith 1993] John Maynard Smith. Evolution and the theory of games. Springer, 1993. (Cited on page 19.)

[Soltesz 2007] Stephen Soltesz, Herbert Pötzl, Marc E Fiuczynski, Andy Bavier and Larry Peterson. *Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors.* In ACM SIGOPS Operating Systems Review, volume 41, pages 275–287. ACM, 2007. (Cited on page 15.)

[Sotomayor 2009a] Borja Sotomayor, Rubén S Montero, Ignacio M Llorente and Ian Foster. *Virtual infrastructure management in private and hybrid clouds.* Internet Computing, IEEE, vol. 13, no. 5, pages 14–22, 2009. (Cited on page 88.)

[Sotomayor 2009b] Borja Sotomayor, Rubén S Montero, Ignacio Martín Llorente and Ian Foster. *Resource leasing and the art of suspending virtual machines.* In High Performance Computing and Communications, 2009. HPCC'09. 11th IEEE International Conference on, pages 59–68. IEEE, 2009. (Cited on pages 52 and 67.)

[Srinivasan 2004] Jayanth Srinivasan, Sarita V Adve, Pradip Bose and Jude A Rivers. *The case for lifetime reliability-aware microprocessors.* In ACM SIGARCH Computer Architecture News, volume 32, page 276. IEEE Computer Society, 2004. (Cited on page 61.)

[Stoica 2001] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek and Hari Balakrishnan. *Chord: A scalable peer-to-peer lookup service for internet applications*. In ACM SIGCOMM Computer Communication Review, volume 31, pages 149–160. ACM, 2001. (Cited on pages 21 and 22.)

[Stoica 2002] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker and Sonesh Surana. *Internet indirection infrastructure*. In ACM SIGCOMM Computer Communication Review, volume 32, pages 73–86. ACM, 2002. (Cited on page 22.)

[Sun 2011] Dawei Sun, Guiran Chang, Qiang Guo and Xingwei Wang. *Modelling, evaluating and designing virtual machine scheduling by a clustering mechanism in cloud computing environments*. International Journal of Wireless and Mobile Computing, vol. 5, no. 1, pages 70–76, 2011. (Cited on pages 53 and 67.)

[symform 2013] symform. *Cooperative Storage Cloud*. 2013. (Cited on page 26.)

[Taylor 1978] Peter D Taylor and Leo B Jonker. *Evolutionary stable strategies and game dynamics*. Mathematical biosciences, vol. 40, no. 1, pages 145–156, 1978. (Cited on page 19.)

[Tian 2013] Wenhong Tian, Ruini Xue, Jun Cao, Qin Xiong and Yunjun Hu. *An Energy-Efficient Online Parallel Scheduling Algorithm for Cloud Data Centers*. In Services (SERVICES), 203 IEEE Ninth World Congress on, pages 397–402. IEEE, 2013. (Cited on pages 51 and 67.)

[Vaquero 2008] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres and Maik Lindner. *A break in the clouds: towards a cloud definition*. ACM SIGCOMM Computer Communication Review, vol. 39, no. 1, pages 50–55, 2008. (Cited on page 29.)

[View 2007] IBM Virtualizatin View. *Virtualization can help power efficiency*, 2007. (Cited on page 62.)

[virtualizationVmware 2013] virtualizationVmware. *virtualizationVmware*. 2013. (Cited on page 15.)

[Von Laszewski 2000] Gregor Von Laszewski, Ian Foster and Jarek Gawor. *CoG kits: a bridge between commodity distributed computing and high-performance grids*. In Proceedings of the ACM 2000 conference on Java Grande, pages 97–106. ACM, 2000. (Cited on page 48.)

[Voorsluys 2009] William Voorsluys, James Broberg, Srikumar Venugopal and Rajkumar Buyya. *Cost of virtual machine live migration in clouds: A performance evaluation*. In Cloud Computing, pages 254–265. Springer, 2009. (Cited on page 61.)

[Wang 2008] David Wang. *Meeting green computing challenges*. In Electronics Packaging Technology Conference, 2008. EPTC 2008. 10th, pages 121–126. IEEE, 2008. (Cited on page 137.)

[Wang 2010] Lizhe Wang, Jie Tao, Gregor von Laszewski and Dan Chen. *Power aware scheduling for parallel tasks via task clustering*. In Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on, pages 629–634. IEEE, 2010. (Cited on page 39.)

[Ward 2002] Brian Ward. The book of vmware: the complete guide to vmware workstation. No Starch Press, 2002. (Cited on page 16.)

[Warren 2002] Michael Warren, Eric Weigle and W Feng. *High-density computing: A 240-node beowulf in one cubic meter*. In Supercomputing 2002, 2002. (Cited on page 63.)

[Wu 2000] Min-You Wu, Wei Shu and Hong Zhang. *Segmented min-min: A static mapping algorithm for meta-tasks on heterogeneous computing systems*. In Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th, pages 375–385. IEEE, 2000. (Cited on page 48.)

[Xian 2007] Changjiu Xian, Yung-Hsiang Lu and Zhiyuan Li. *Energy-aware scheduling for real-time multiprocessor systems with uncertain task execu-*

tion time. In Proceedings of the 44th annual Design Automation Conference, pages 664–669. ACM, 2007. (Cited on pages 60 and 67.)

[Yazir 2010] Yagiz Onat Yazir, Chris Matthews, Roozbeh Farahbod, Stephen Neville, Adel Guitouni, Sudhakar Ganti and Yvonne Coady. *Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis*. In Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on, pages 91–98. Ieee, 2010. (Cited on page 89.)

[Yong 2001] Chern Han Yong and Risto Miikkulainen. *Cooperative coevolution of multi-agent systems*. University of Texas at Austin, Austin, TX, 2001. (Cited on page 20.)

[Yu 2008] Kun-Ming Yu and Cheng-Kwan Chen. *An adaptive scheduling algorithm for scheduling tasks in computational grid*. In Grid and Cooperative Computing, 2008. GCC'08. Seventh International Conference on, pages 185–189. IEEE, 2008. (Cited on page 44.)

[Yu 2009] Jia Yu and Rajkumar Buyya. *Gridbus workflow enactment engine*. Grid Computing: Infrastructure, Service, and Applications, L. Wang, W. Jie, and J. Chen Eds, CRC Press, Boca Raton, FL, USA, pages 119–146, 2009. (Cited on pages 46, 64, 65 and 67.)

[Zhang 2010] Qi Zhang, Lu Cheng and Raouf Boutaba. *Cloud computing: state-of-the-art and research challenges*. Journal of Internet Services and Applications, vol. 1, no. 1, pages 7–18, 2010. (Cited on page 29.)

[Zhao 2013] Dali Zhao, Houman Homayoun and Alexander V Veidenbaum. *Temperature aware thread migration in 3D architecture with stacked DRAM*. In ISQED, pages 80–87, 2013. (Cited on page 61.)

[Zong 2011] Ziliang Zong, Adam Manzanares, Xiaojun Ruan and Xiao Qin. *EAD and PEBD: two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters*. Computers, IEEE Transactions on, vol. 60, no. 3, pages 360–374, 2011. (Cited on pages 40 and 67.)