PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

MOHAMED BOUDIAF UNIVERSITY - M'SILA

FACULTY : MATHEMATICS AND COMPUTER SCIENCE

DEPARTMENT : COMPUTER SCIENCE

N°:……

DOMAIN : MATHEMATICS AND COMPUTER SCIENCE

DEPARTEMENT: COMPUTER

SCIENCE OPTION : IDO

Thesis submitted for obtaining

Academic Master's degree

By: Hamza Sarra

Entitled

Job scheduling in cloud computing environment

Defended before the jury composed of:

Dr. GUEMOUGUI Abdessattar	University of M'sila	President
Dr. Barkat Abdelbasset	University of M'sila	Reporter
Dr. BOUZAAROURA Ahlam	University of M'sila	Examiner

Academic year: 2019/2020

Dedication

«To my husband, and my sons
Chihab Eddine, and Siradje Eddine »

Acknowledgment

Foremost, great thank for ALLAH, and I thank my Parents my sisters and brothers who supported me during this academic journey and my whole life.

Then I would like to thank my thesis advisor Dr. Barkat Abdelbasset for the continuous support of me, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

Many sincere thanks also go to mathematics and computer science Faculty members at Mohamed Boudiaf University of M'sila for their insightful conversations and hard questions, thank you for teaching me how to be a dedicated researcher.

Million thanks go to my responsible at work the vice rector of pedagogy Pr. Bouguerra Rabah for his support and encouragement, and kindness during my study.

Special thanks to my uncle Younes for all his help to complete this work

Thank you very much.

HAMZA Sarra

August/2020

Abstract

Cloud computing provides a pool of virtualized computing resources and adopts pay-per-use model. Task Scheduler is one of the most important cloud computing problems. And that cannot be scheduled using single criteria, but according to many criteria and rules. These rules and criteria to be agreed upon by the service provider and the user.

In this report, we present a particle swarm optimization (PSO) based scheduling algorithm to minimize the completion execution time (Makespan). Experiment was conducted by varying computation of tasks, number of particles and (Makespan) in fitness function. Then we modify the proposed PSO by adding a method for sorting the tasks by thier length to execute them using the shortest job first (SJF).

The two proposed algorithms are implemented by using the CloudSim simulator. By comparing the two proposed algorithms (PSO and SJF-PSO) the result given by (SJF-PSO) algorithm has the lower Makespan.

Keywords:

Cloud Computing, scheduling algorithms, Particle Swarm Optimization (PSO), Shortest Job First (SJF), Makespan, CloudSim.

TABLE OF CONTENTS

Chapter II: The Proposed PSO Scheduling Algorithm

Chapter III: Implementation and Experiment Result

LIST OF FIGURES

LIST OF TABLE

Introduction

INTRODUCTION

On Google, a search for the term “cloud computing” at june 2020 yields more than 400 million hits, this is an indication of the importance of this technology in recent times, the Cloud computing is derived from distributed computing, and it allows individuals and organizations to access computing resources as services remotely through the Internet.

Cloud computing provides a pool of abstracted, virtualized resources, including computing power, storage, platforms and software applications over the Internet based on users demand. Due to its many characteristics such as elastic, scalable resource provision and cost-effectiveness, cloud computing has been adopted by more and more users.

Cloud computing offers a great variety of services. Based on the level of services, there are three categories generally. They are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IaaS puts servers, storage, networks, and data center fabrics together as demanded by users. Cloud users can then install operating system and deploy their own applications on the cloud. PaaS, on the other hand, provides middleware, database and development tools. It enables users to deploy applications onto a virtualized cloud platform. Finally, in SaaS the complete operating environment, along with applications, management, and user interfaces, are provided to cloud users. Since all cloud services can be access by subscription and run with a pay per-use model, cloud computing leverages many attractive features to users, including low cost and simple management. There are many technical challenges faced by cloud providers, such as maintaining high utilization while delivering services that are low cost, short delay, and dynamic deployability. It is critical for cloud providers to maintain an optimal tasks scheduling and management system to meet these challenges.

Just like any other system, cloud computing is not without issues. One of the most challenging issues in cloud computing is task scheduling. Users submit their tasks to the cloud service

provider at any given time and it is up to the cloud provider to execute. Cloud service providers employ scheduling algorithms to ensure proper and efficient resource utilization on their end and attempt to achieve customer satisfaction as well. The scheduler decides which resources will be used, as well as which tasks will be executed on each of those resources, and allocate tasks to the resources. The tasks scheduling problem, like general scheduling problems, is NP-complete.

Tasks scheduling algorithms often utilize heuristics and metaheuristics, including soft computing techniques, to obtain approximated solutions. In this report, we adopt a task scheduling strategy using Particle Swarm Optimization (PSO). PSO, an applied soft computing method developed by Kennedy and Eberhart [], is one of the most advance evolutionary algorithms driven from nature. PSO approximates an optimal solution by iteratively improving a group of candidate solutions, called particles. Each particle is modified iteratively by the best information from both the individual and the entire group. By collecting the cumulative intelligence of whole group, the group is expected to move toward the most optimal solutions. PSO works well on most global optimal problems. In addition, it is simple, effective, and of low computational cost.

Makespan is a key performance measurement criterion assigned by cloud users and considered by task schedulers. Makespan is the time from the beginning till the completion of the sequence of tasks. Different application schedulers may use different policies with different objectives. Some algorithms are designed to achieve minimum cost while others strive for minimum Makespan or for load balance. Most existing algorithms focus on achieving a single optimal criterion. In this report, a job scheduling strategy to attain a minimal Makespan is implemented.

The extensive research on all issues in real environment is extremely difficult because it requires developers to consider network infrastructure and the environment, which may be

beyond the control. Therefore, simulators are developed. To understand and apply better the state-of-the-art of cloud computing simulators.

CloudSim is a simulator tool used to implement the task scheduling algorithms in a cloud environment, using this framework, the cloud computing can be analyzed, designed, modeled, and implemented for result analysis.

This thesis contains three main chapters; Chapter one presents the general idea of cloud computing, its characteristics, services, deployment models and the architecture.

Will the Virtualization technology is the main foundation of Cloud Computing, for this reason we will look in this chapter an overview of the virtualization, definition, Approaches, Hypervisor, and types. Also, the section includes What defines Data Center and how cloud computing uses it.

Chapter two explains the main concepts of scheduling cloud tasks, techniques on it. In addition to, it presents samples of related work studies of various task scheduling algorithms in cloud environment, namely: Biogeographic based optimization (BBO), Tournament Selection genetic algorithm (TS-GA), and the Ant colony optimization (ACO). Then we discuss the proposed PSO algorithm in details.

Chapter three we present a Simulation of proposed PSO algorithm for job scheduling in a cloud environment using CloudSim. we declare the experimental settings and results.

Finally, we summarize the conclusion of this thesis and proposes some ideas for future work.

Chapter I

Background and Literature Review

I.1. Overview

Cloud Computing is one of the hottest terms in computing. Cloud computing is a leading technique for bringing up a pool of computational resources with extendibility and employment of these techniques by the usage of internet. It provides vast computing capability to big companies which involve in dealing with enormous data produced daily on a pay and use model.[15]

The key aim of this chapter is to conduct a comprehensive evaluation of the relevant literature relating to the emerging field of Cloud Computing, focusing on three key areas of research development in general: Cloud Computing technology and its characteristics; the virtualization technology; and the data center.

The first section provides the definition of Cloud Computing, characteristics, service, deployment models and the architecture of cloud computing.

The Virtualization technology is the main foundation of Cloud Computing, for this reason we provide an overview of the virtualization, definition, Approaches, Hypervisor, and types.

The third section includes What defines Data Center and how cloud computing uses it.

I.2. Cloud Computing

Many practitioners in the commercial and academic spheres have attempted to define exactly what "cloud computing" is and what unique characteristics it presents. Wikipedia have defined it as follows: "*Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. The term is generally used to describe data centers available to many users over the Internet. Large clouds, predominant today, often have functions distributed over multiple locations from central servers. If the connection to the user is relatively close, it may be designated an edge server".*

In Amazon Cloud computing is the on-demand delivery of IT resources over the Internet with pay-as-you-go pricing. Instead of buying, owning, and maintaining physical data centers and servers, you can access technology services, such as computing power, storage, and databases, on an as-needed basis from a cloud provider like Amazon Web Services (AWS).

The definition of cloud computing provided by the National Institute of Standards and Technology NIST has gained significant traction within the IT industry. According to this definition: Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.[1]A note to the definition says that "Cloud computing is still an evolving paradigm. Its definitions, use cases, underlying technologies, issues, risks, and benefits will be refined in a spirited debate by the public and private sectors. These definitions, attributes, and characteristics will evolve and change over time." This is undoubtedly true. Nevertheless, at this point in time, the NIST definition is widely accepted and is increasingly regarded as authoritative [5]. The (Fig I.1) shown an overview of cloud computing, with typical types of applications supported by that computing model.

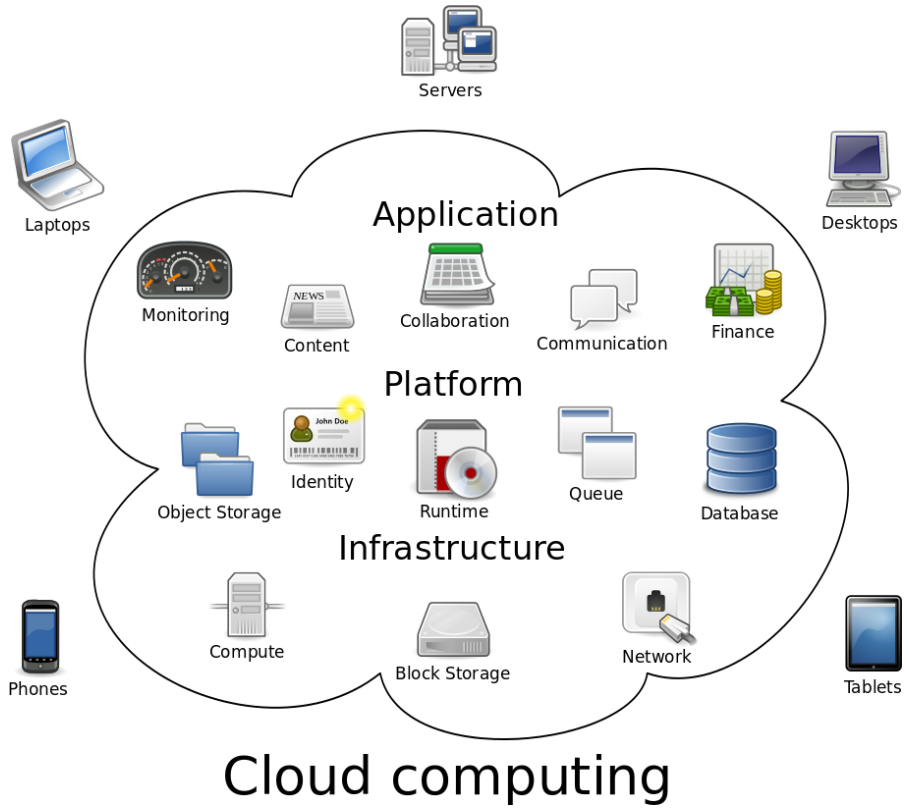

Fig I.1: Diagram showing overview of cloud computing, with typical types of applications supported by that computing model.

I.2.1. Cloud Computing characteristics

The five essential characteristics of the NIST definition are shown below (in Fig I.2) and described in the following sections.

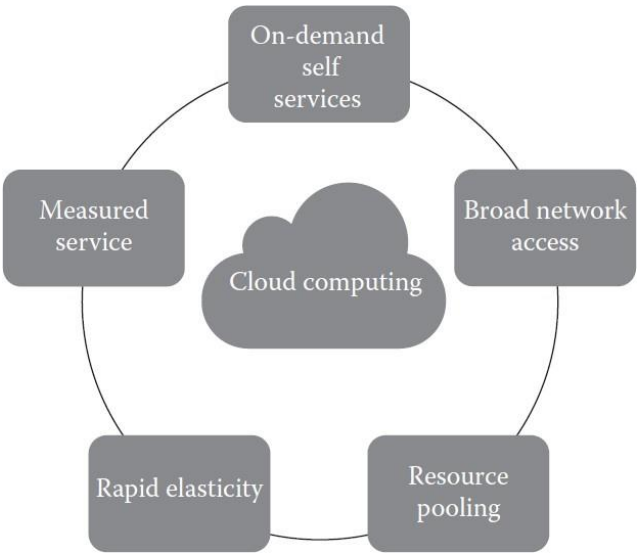

Fig I.2: The essential characteristics of cloud computing.

a. **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. [4]
b. **Broad network access:** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and personal digital assistants). [4]
c. **Elastic resource pooling:** The provider's computing resources are pooled to serve multiple consumers using a multitenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify the location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, and network bandwidth. [4]

d. **Rapid elasticity:** Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time. [4]
e. **Measured service:** Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported providing transparency for both the provider and consumer of the utilized service.[4]

I.2.2. Cloud Computing Service Models

The following figure (Fig I.3) shown The three main deferent types of service models for cloud computing defined in [1]:

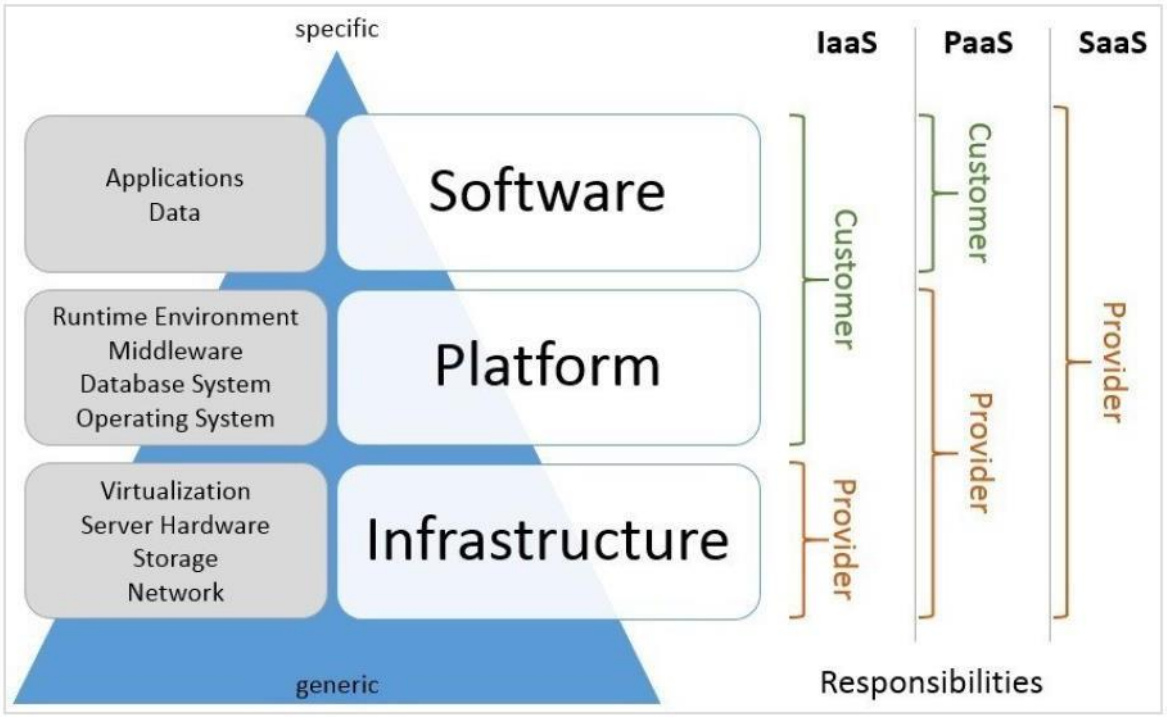

Fig I.3: Layers of the cloud service model and their respective responsibilities.[6]

I.2.2.1. Software as a Service (SaaS)

The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings. [1]

I.2.2.2. Platform as a Service (PaaS)

The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment. [1]

I.2.2.3. Infrastructure as a Service (IaaS)

The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls). [1]

I.2.3. Deployment Models

Apart from the service models, NIST [1] defined the following deploying models for the cloud:

I.2.3.1. Private cloud

The cloud infrastructure is provisioned for exclusive use by a single organization comprising multiple consumers (e.g., business units). It may be owned, managed, and operated by the organization, a third party, or some combination of them, and it may exist on or off premises. [1]

I.2.3.2. Community cloud

The cloud infrastructure is provisioned for exclusive use by a specific community of consumers from organizations that have shared concerns (e.g., mission, security requirements, policy, and compliance considerations). It may be owned, managed, and operated by one or more of the organizations in the community, a third party, or some combination of them, and it may exist on or off premises.[1]

I.2.3.3. Public cloud

The cloud infrastructure is provisioned for open use by the general public. It may be owned, managed, and operated by a business, academic, or government organization, or some combination of them. It exists on the premises of the cloud provider.[1]

I.2.3.4. Hybrid cloud

The cloud infrastructure is a composition of two or more distinct cloud infrastructures (private, community, or public) that remain unique entities, but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load balancing between clouds).[1]

The figure (Fig I.4) summarize the deferent types of cloud deployment models.

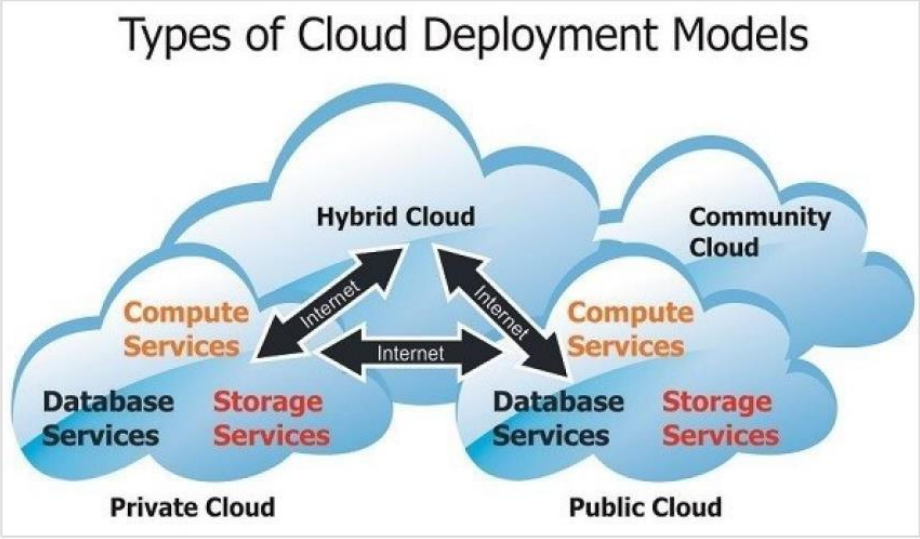

Fig I.4: Types of Cloud Deployment Models [7]

I.2.4. Cloud Computing Architecture

The cloud architecture consists of five separate components which work together to provide on-demand services.

Figure I.5 is taken from National Institute of Standards and Technology (NIST) Cloud computing reference architecture [2]. defines five major cloud actors: cloud provider, cloud consumer, cloud carrier, cloud auditor and cloud broker. These are briefly described as further.

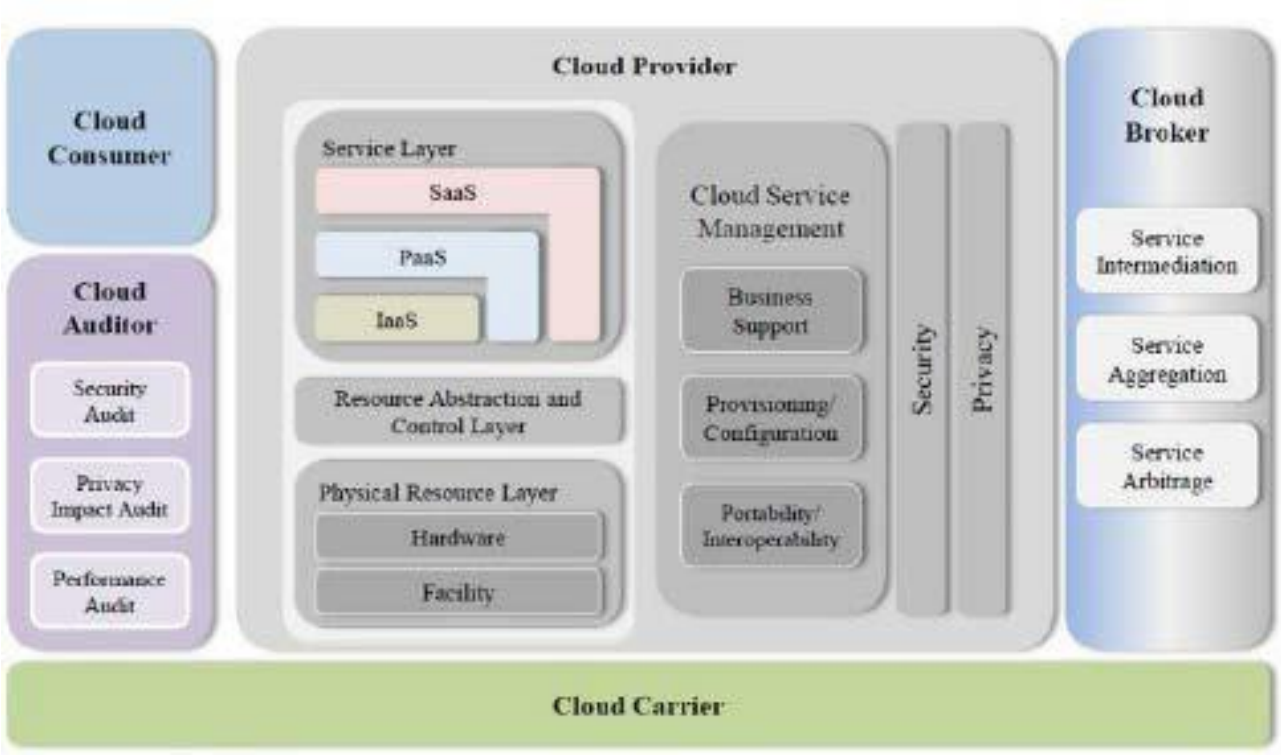

Fig I.5: NIST Cloud Computing Reference Architecture (CCRA) [2]

I.2.4.1. Cloud Provider

The cloud provider are organizations that provides cloud services. The cloud provider has control over the IT infrastructure and manages the technical outages if planned or unplanned. Cloud provider also ensures that agreed Service Level Agreements (SLA's) are achieved. [3]

I.2.4.2. Cloud Consumer

A cloud consumer is a person(s) or an organization using cloud service(s) and having an agreement with cloud provider or cloud broker. [3]

I.2.4.3. Cloud Carrier

The cloud carriers are networks and telecommunication companies, which ensure that services from cloud provider are available to cloud consumers. Cloud carrier work closely with cloud provider to meet agreed SLA's. [3]

I.2.4.4. Cloud Broker

The cloud brokers are third party companies, which work closely with both cloud providers and cloud consumers. Generally, these are consulting companies and so they can easily sell various cloud solutions to their existing customers as well as to new customers. [3]

I.2.4.5. Cloud Auditor

The cloud auditors are third parties who are specialized in independent assessment of cloud services offered by cloud providers. A cloud auditor can audit various areas such as security, privacy, performance, licensing, operations and other areas to highlight the gaps against various operations and data privacy standards. [3]

I.3. Virtualization

I.3.1. Definition

Virtualization could be defined as separation of logical operations from their physical environment. To give an easy example on virtualization, we can use virtual memory. Virtual memory is an extension to system memory in a computer, which is used from the hard drive, rather than from the RAM memory.[8]

If the system memory runs out, this virtual extension can be used to keep the system running. As shown in

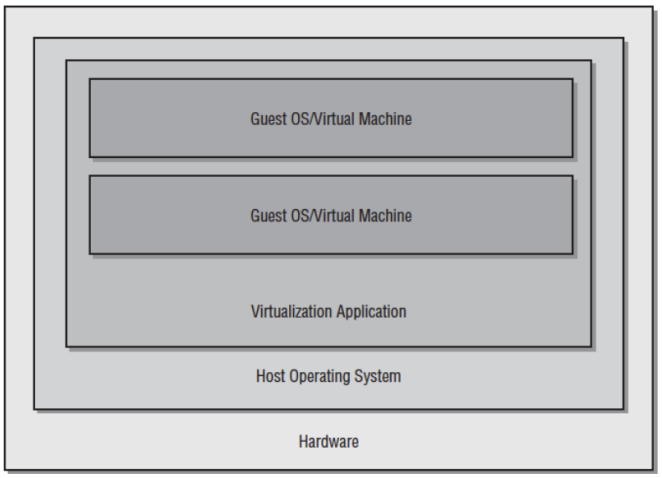

Fig I.6: Virtualization overview.[8]

I.3.2. Approaches in Virtualization

There have been many approaches adopted in the implementation of virtualization technology. Some of the important approaches are discussed in the following subsections.[4]

I.3.2.1. Full Virtualization

Full virtualization uses a special kind of software called a hypervisor. The hypervisor interacts directly with the physical server's hardware resources, such as the CPU and storage space, and acts as a platform for the virtual server's OSs. It helps to keep each virtual server completely independent and unaware of the other virtual servers running on the physical machine.

Each guest server or the virtual machine (VM) is able to run its own OS. That means one virtual server could be running on Linux and the other one could be running on Windows. Examples include VMWare ESX and VirtualBox. In the full virtualization, the guest OS is unaware of the underlying hardware infrastructure. That means the guest OS is not aware of the fact that it is running on a virtualized platform and of the feeling that it is running on the real hardware. In this case, the guest OS cannot communicate directly to the underlying physical infrastructure.

The OS needs the help of virtualization software hypervisors to communicate with the underlying infrastructure. The advantages of the full virtualization include isolation among the various VMs, isolation between the VMs and the hypervisor, concurrent execution of multiple OSs, and no change required in the guest OS. A disadvantage is that the overall system performance may be affected due to binary translation. [4]

I.3.2.2. Paravirtualization

In this case, VMs do not simulate the underlying hardware, and this uses a special API that a modified guest OS must use. Examples include Xen and VMWare ESX server. In this type of virtualization, partial simulation of the underlying hardware infrastructure is achieved. This is also known as partial virtualization or OS-assisted virtualization. This virtualization is different from the full virtualization in that, here, the guest OS is aware of the fact that it is running in a virtualized environment. In this case, hypercalls are used for the direct communication between the guest OS and the hypervisor.

In paravirtualization, a modified or paravirtualized guest OS is required. 107 Technological Drivers for Cloud Computing An advantage of this approach is that it improves the overall system performance by eliminating the overhead of binary translation. A disadvantage could be that a modification of the guest OS is required. [4]

I.3.2.3. Hardware-Assisted Virtualization

In this type of virtualization, hardware products supporting the virtualization are used. Hardware vendors like Intel and AMD have developed processors supporting the virtualization through the hardware extension. Intel has released its processor with its virtualization technology VT-x, and AMD have released its processor with its virtualization technology AMD-v to support the virtualization. An advantage of this approach could be that it eliminates the overhead of binary translation and paravirtualization. A disadvantage includes the lack of support from all vendors. [4]

I.3.3. Hypervisor and Its Role

The concept of using VMs increases the resource utilization in a cloud computing environment. Hypervisors are software tools used to create the VMs, and they produce the virtualization of various hardware resources such as CPU, storage, and networking devices. They are also called virtual machine monitor (VMM) or virtualization managers. They help in the virtualization of cloud data centers (DCs). The various hypervisors used are VMware, Xen, Hyper-V, KVM, etc.

Hypervisors help to run multiple OSs concurrently on a physical system sharing its hardware. Thus, a hypervisor allows multiple OSs to share a single hardware host. In this case, every OS appears to have the host's processor, memory, and other resources allocated solely to it. However, the hypervisor is actually controlling the host processor and resources and in turn allocates what is needed to each OS.

The hypervisor also makes sure that the guest OSs (called VMs) do not interrupt each other. In virtualization technology, hypervisor manages multiple OSs or multiple instances of the same OS on a single physical computer system. Hypervisors are designed to suit a specific processor, and they are also called virtualization managers. [4]

I.3.3.1. Types of hypervisor

Type 1 hypervisor: This type of hypervisor runs directly on the host computer's hardware in order to control the hardware resources and also to manage the guest OSs. This is also known as native or bare-metal hypervisors. Examples include VMware ESXi, Citrix XenServer, and Microsoft Hyper-V hypervisor. [4]

Type 2 hypervisor: This type of hypervisor runs within a formal OS environment. In this type, the hypervisor runs as a distinct second layer while the guest OS runs as a third layer above the hardware. This is also known as the hosted hypervisors. Examples include VMware Workstation and VirtualBox.108 Essentials of Cloud Computing. [4]

I.3.4. Types of Virtualization

Depending on the resources virtualized, the process of virtualization can be classified into the following types.

I.3.4.1. OS Virtualization

In OS virtualization, a desktop's main OS is moved into a virtual environment. The computer that is used by the service consumers remains on their desk, but the OS is hosted on a server elsewhere. Usually, there is one version of the OS on the server, and copies of that individual OS are given to the individual user. Various users can then modify the OS as they wish, without affecting the other users. [4]

I.3.4.2. Server Virtualization

In server virtualization, existing physical servers are moved into a virtual environment, which is then hosted on a physical server. Modern servers can host more than one server simultaneously, which allows the users to reduce the number of servers to be reserved for various purposes. Hence, IT and administrative expenditures are reduced. Server virtualization can use the virtual processors created from the real hardware processor present in the host system. The physical processor can be abstracted into a collection of virtual processors that could be shared by the VMs created. [4]

I.3.4.3. Memory Virtualization

In main memory virtualization, the virtual main memory that is abstracted from the physical memory is allocated to various VMs to meet their memory requirements. The mapping of physical to virtual memory is performed by the hypervisor software. The main memory virtualization support is provided with the modern x86 processors. Also, the main memory consolidation in the virtualized cloud DCs could be performed by the hypervisor by aggregating the free memory segments of various servers to create a virtual memory pool that could be utilized by the VMs. [4]

I.3.4.4. Storage Virtualization

In storage virtualization, multiple physical hard drives are combined into a single virtualized storage environment. To various users, this is simply called cloud storage, and it could be a private storage, such that it is hosted by a company, or a public storage, such that it is hosted outside of a company like DropBox, or a mixed approach of the two.

In the case of storage virtualization, physical storage disks are abstracted to a virtual storage media. In cloud DCs, high availability and backup of the user's data are achieved 109 Technological Drivers for Cloud Computing through storage virtualization technology. Modern hypervisors help in achieving storage virtualization. The concept of storage virtualization is implemented in advance storage techniques such as storage area networks (SANs) and network-attached storage (NAS). [4]

I.3.4.5. Network Virtualization

In network virtualization (NV), logical virtual networks are created from the underlying physical network. The physical networking components such as the router, switch, or network interface card could be virtualized by the hypervisor to create logical equivalent components. Multiple virtual networks can be created by using the same physical network components that could be used for various purposes. NV can also be achieved by combining various network components from multiple networks. [4]

I.3.4.6. Application Virtualization

In application virtualization, the single application installed on the central server is virtualized and the various virtualized components of the application will be given to the users requesting the services. In this case, the application is given its own copy of components such as own registry files and global objects that are not shared with others. The virtual environment prevents conflicts in the resource usage. An example is the Java Virtual Machine (JVM).

In the cloud computing environment, the CSPs deliver the SaaS model through the application virtualization technology. In the case of application virtualization, the cloud users are not required to install the required applications on their individual systems. They can, in turn, get the virtualized copy of the application, and customize and use it for their own purposes. [4]

I.3.5. Virtualization and Cloud Computing

The Virtualization technology is the main foundation of Cloud Computing offerings [9].

This concept refers to the set of hardware and software tools enabling the abstraction of a physical resource into several logical units, that can be used separately by different Operating Systems (OS) and applications [10]. Resources Virtualization is a key feature for Cloud providers to build efficient, flexible and cost-effective computing models satisfying the Cloud market challenges. It enables simple resource management and dynamic resource resizing, reduced hardware costs due to resource sharing, isolation and fairness between tenants, increased availability and quick recovery through easy backups and rapid migrations [11, 12].

Virtualization of servers on a shared cluster can consolidate the web services. As the virtual machines are the container of cloud services, the provisioning tools will first find the corresponding physical machines and deploy the virtual machines to those nodes before scheduling the service to run on the virtual nodes. In cloud computing, virtualization is not only means the system virtualization platforms are uses but also some fundamental services are provided.[16]

In addition, in cloud computing, virtualization also means the resources and fundamental

infrastructure is virtualized. From the user point of view, the user will not care about the computing resources that are used for providing the services. Cloud users do not need to know and have no way to discover physical resources that are involved while processing a service request. And also, from the developer point of view, the application developers do not care about some infrastructure issues such as scalability, fault tolerance i.e. they are virtualized. Application developers focus on the service logic.[16]

System Virtualization: In many cloud computing systems, system virtualization software is used. System virtualization software is a special kind of software which simulates the execution of hardware and run even the unmodified operating systems. Cloud computing systems use the virtualization software as the running environment for legacy software such as old operating systems and unusual applications. Virtualization software is also used as the platform for developing new cloud applications that the developers can use any operating systems and programming environments as they like. The development environment and deployment environment can now be the same which eliminates some runtime troubles. The system virtualization supported is illustrated in Fig I.6 with a virtualization infrastructure.[16]

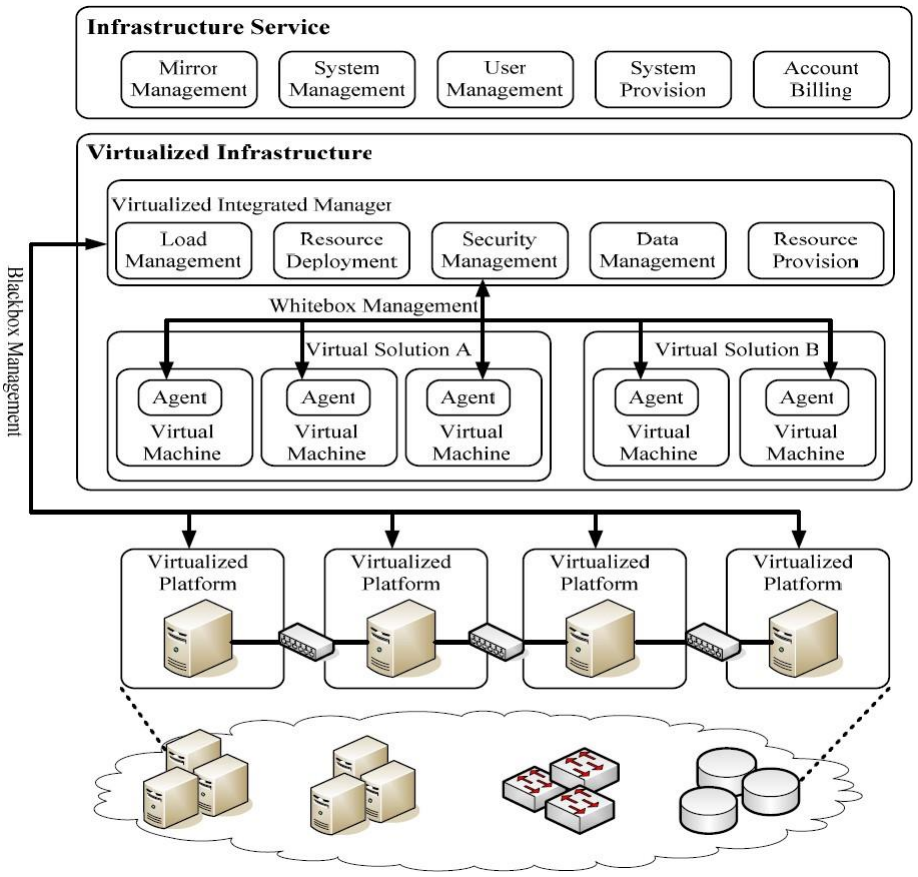

Fig I.6: Virtualized servers, storage, and network for cloud platform construction.[16]

I.4. Datacenter

I.4.1. What is a data center?

Data center can be defined as (Rihards Balodis and Inara Opmane): A data center is a physical environment facility intended for housing computer systems and associated components. Data centers comprise the above-mentioned computer systems and staff that maintains them. The necessary physical environment facility encompasses power supplies with the possibility to ensure backup power. necessary communication equipment and redundant communication cabling systems, air conditioning, fire suppression and physical security devices for staff entrances.

Another way data center can be defined as Data center infrastructure comprises a large number of servers interconnected by a network.[13]

I.4.2. Information Technology Infrastructure of data center

Data center IT equipment may be classified as servers, storage, and networking devices. Servers are mounted within racks and consist of hardware resources (such as CPUs, NICs, and RAMs) hosting applications like video streaming and big data.

All the data generated by these applications are stored in storage systems. Data center storage consists of high capacity (around Terabytes) disk drives or flash devices. The storage tiers are connected to the servers either directly or through networking devices (managed by a global distributed file system), forming a NAS (Network Attached Storage). The RAID (Redundant Array of Independent Disks) storage technology can be used to provide high availability, redundancy, and increase fault tolerance.

Networking equipment manages the internal communication between servers and storage systems as well as all the input/output data flow from/to the data center. Typically, a data center network is designed based on three hierarchical levels: core, distribution, and edge. It is through the core level that all data center traffic (ingress/egress) to the outside world will be managed. Distribution is a level between the edge and core levels which aggregates the edge switches. Its main goal is simply network management and cabling reduction. Finally, the edge level passes data from devices (generating or consuming data) to the distribution or core levels.

Manage all these components is a great challenge because hardware clusters have a deep and complex memory/storage hierarchy, the system is composed of heterogeneous components, and there are many failure-prone components. It is necessary to use an additional software layer to provide an abstraction of this variety of components. According to, this software layer has three levels: platform level, cluster level, and application level. The platform-level software is composed of firmware, operational system, and kernel that provide an abstraction of hardware of a single machine in the cluster and provide server-level services. The cluster-level software is related to any software that managed resources at cluster level, such as distributed file systems, management resource systems, and monitoring systems. The application level software is composed of software that implements a particular service. The set of all software used to monitor, measure, manage and control the data center behavior is named DCIM (Data Center Infrastructure Management).[17]

I.4.3. Cloud computing and data center

The term “cloud” started emerging around the same time wireless handheld devices started to become popular in the marketplace. When accessing the web via a wireless handheld device, it seems like you are pulling data out of the clouds. It is natural, then, that the data centers providing this information should be called cloud data centers. Today, it appears that everyone is jumping on the “cloud” bandwagon with all kinds of cloud companies, cloud products, and cloud services entering the market.

Cloud data centers are being rapidly deployed around the world. Since these installations must support up to hundreds of thousands of servers, data center efficiency and cost of operations have become critical. Because of this, some cloud data centers have been erected near cheap electrical power sources, such as hydroelectric dams, or in colder climates to help reduce cooling costs. Some companies, such as Microsoft®, are building modular data centers using pods, which are self-contained server storage and networking modules the size of a shipping container. These modules are trucked in, stacked up, and connected to power, cooling, and networking. Other data centers use server racks as the basic building block and contain rows and rows of these racks. No matter what the structure, networking is an important part of these large cloud data center networks.[14]

I.5. Conclusion

In this chapter the fundamental aspects and the concepts which are present, while facing the fields of cloud computing and virtualization are the object of giving an illustration of the recent techniques and the new concepts towards which the data processing is directed. But always the optimization will remain the first objective, searching to optimize the exploitation of the available resources in a cloud or more precisely searching to tasks scheduling in a cloud is an important research topic in cloud computing, a lot of work which consists to implement optimization algorithms was realized. In the next chapter, we first present some work and then we will try to demonstrate and implement a proposed PSO algorithm using the CloudSim platform.

Chapter II

The Proposed PSO Scheduling Algorithm

II.1. Overview

Cloud Computing provides on demand access to some shared resources and services. These services are provided as a service over a network, and can be accessed over the internet.

Cloud computing has become an interesting and beneficial way of changing the whole computing. in the cloud computing the users submit their tasks so as to be executed as soon as possible in the available shared pool of resource. In each cloud provider datacenter, there is a component that control and manage the tasks scheduling in different VMs. Schedulers for cloud computing determine on which processing resource jobs of a workflow should be allotted.

Task scheduling in cloud computing is an NP-complete problem [18]. There are no specific methods to get polynomial time solutions for these problems. The meta-heuristics-based technique have gained attention to get near optimal solutions for these complex problems.

This chapter is devised in three main section, in the first one we describe the tasks scheduling in cloud computing, and his different types , then the second section we presents a review study of various task scheduling algorithms in cloud environment, based on meta-heuristics techniques namely: biogeographic based optimization (BBO), Tournament Selection genetic algorithm (TS-GA), and the Ant colony optimization (ACO), in the last section a case study on Practices swarm optimization (PSO) is designed. The PSO algorithm has been tested using CloudSim toolkit.

II.2. Tasks scheduling in Cloud Computing

Scheduling is a process to determine the order resource which is mapped to be executed. The main advantage of scheduling algorithm represented a high performance. It is one of the important activities performed in grid and cloud computing environments, as its objective is to deliver available computing resources as services over networks [18]. On the other hand, the task scheduling algorithm is considered a complex process because it must schedule a large number of tasks into the available resources. Also, there are many parameters that should be taken into consideration to develop a task scheduling algorithm. Some of these parameters are important from the Cloud user perspective (i.e., tasks compilation time, cost, and response time). Other parameters are important from the Cloud provider perspective (i.e., resource utilization, fault tolerant, and power consumption). [19]

Tasks scheduling It consists of assigning a set of Tasks T to all available Resources R in Datacenters, in order to complete all tasks under constraints imposed, effectively satisfy user U requests and maximize the revenue of the provider [20], as shown in Fig II.1.

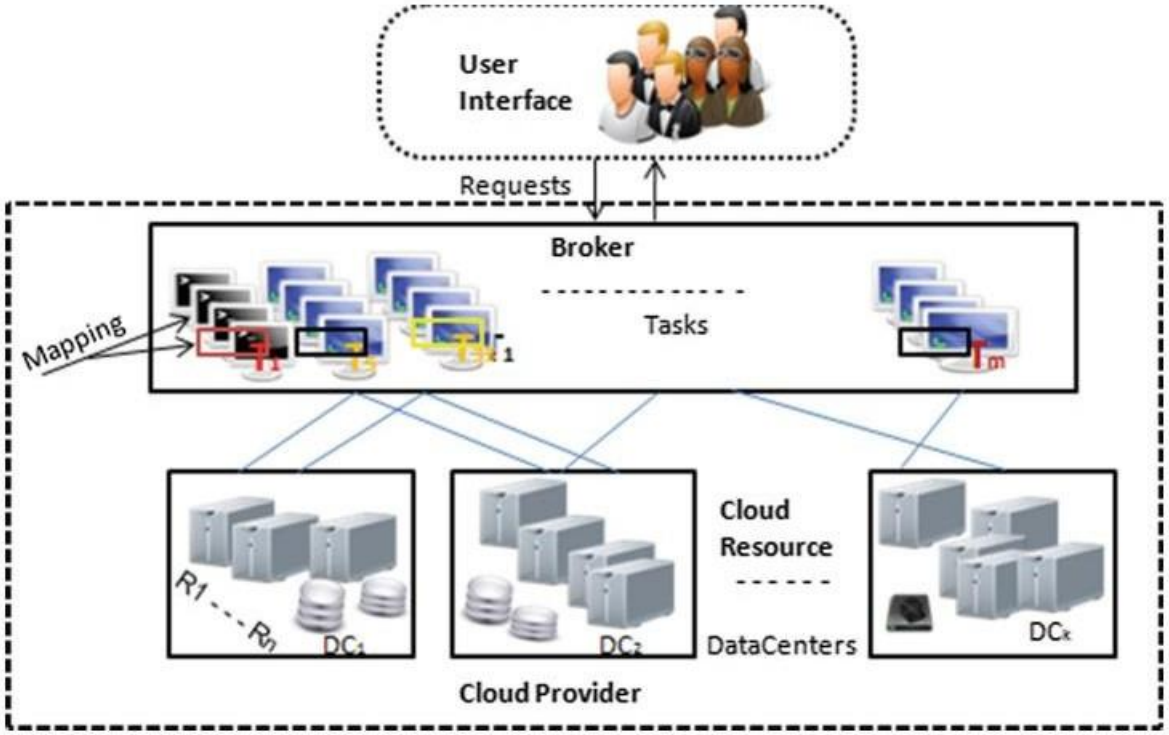

Fig II.1: Resource allocation for task scheduling problem in cloud computing [20]

The broker can be a manageable server or a specific datacenter in which the scheduling policy locates. Whereas, the resources are available at the datacenter level. The Cloud provides services to the user using three main steps as illustrated in the Fig II.1:

- The user sends requests.
- The broker receives and analyzes these requests.
- Application of the scheduling approaches to affect each task to the appropriate resource.[20]

II.2.1. Task Scheduling Types in Cloud Computing

The diversity of services providing (application, storage, CPU, etc.) and the necessity to make it available to users, as well as the need for a better management of datacenters, led to a multiplicity of task scheduling [20] as we illustrate in Fig II.2.

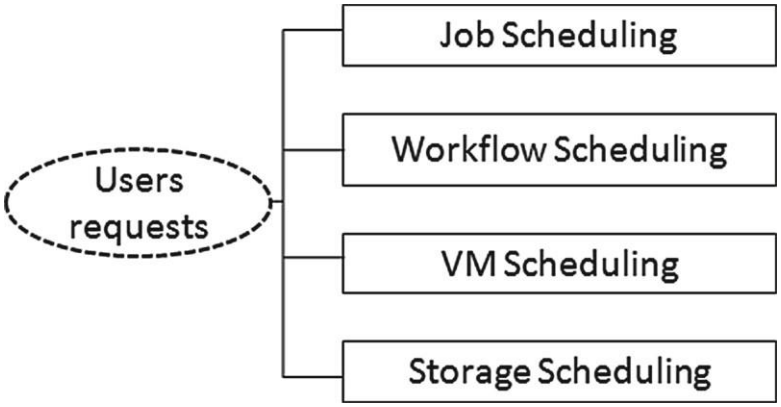

Fig II.2: Task Scheduling Types in Cloud Computing. [20]

II.2.1.1. Job Scheduling

Usually, job calls on a set of independent tasks [3]. Most approaches begin by evaluating the weight of the tasks in its pool according to (waiting time, deadline, arrival time, execution time, etc.) to determine the priority of each task in order to estimate the best mapping.[20]

II.2.1.2. Workflow Scheduling

Set of dependent tasks, represented with Directed Acyclic Graph (DAG) where each node indicates the tasks and each edge represents the communication between dependency tasks. The task can only be executed once all its parent tasks have been completed. [20]

II.2.1.3. VM Scheduling

Clouds providers create groups of VMs in a resource pool, then each VM should assign to a host. It is one of the main challenges in implementing IaaS because there are many constraints defined in a Service Level Agreement (SLA) document which explain how to associate the VMs with each client. This latter problem can be solved by a VM migration technique. [20]

II.2.1.4. Storage Scheduling

Mean a set of data block, often with large capacity and heterogeneous type (image, text, sound, etc.) which are stored in clusters that exist in different geographical places. Also, its size increases continually as is the case in the bigdata. Furthermore, to obtain a good optimization, we require a high processing method for storing and retrieving data, so scheduling is considered as helpful solution to deal with this type of problems. For this purpose, it is generally necessary to create a partition of data, based on the available information about storage sites and resources.[20]

The following table (Tab II.1) show the four types of task scheduling

Task type	Main scheduling phases	Main parameters	Example of queries
Job	• Resource allocation phase(allocated pool of tasks to set of resources)	• Weight tasks or size	• Job request • Web applications
Workflow	• Tasks sorting phase • Resource allocation phase(allocated pool of tasks to set of resources)	• Data transmission • Bandwidth • Sub-deadline of tasks	• Workflow application
VM	• VMs placement (allocated set of VMs to set pf physical resources)	• Consolidation of VMs • Migration of VMs	• VM requests
Storage	• Distribution (location) of data (allocated blocks of data to cluster resources)	• Chunk[a]	• Big data application • Storage requests • Data mining application

Tab II.1: Comparison between different tasks scheduling types in cloud computing [20]

II.2.2. Related works

The job scheduling problem for the optimization objectives of minimizing the maximum completion time has been proved to be NP-complete problems. So for this reason it is basically handled by various known heuristic methods which give solutions for problem instances basically in restricted manner. [21]

Bio inspired algorithm help us to solve the problems of optimization. Many researchers did enormous work in this area from the past few decades. However, still there is a large more scope for bio inspired algorithm in exploring new application and opportunities in cloud computing, and they were used to tackle various challenges faced in Cloud Computing tasks scheduling. [22]

The first interest work is [18] where the authors have used the biogeographic based optimization (BBO), they formalize the problem as Follow:

A cloud node (CN) is a set of computational resources with limited computation speed on the cloud, CN = (C1,C2,….Cm), each CN has corresponding calculating speed.

The set of CN speed is S =(s_1, s_2, . . . , s_m). The unit of s_i =(1, 2, . . . ,m) is the number of Cycles Per Unit of Time s_i is the speed of CN Ci. A set of jobs J = (J_1,J_2,….J_n), each job has a length L = (l_1,l_2,….l_n). The authors introduce the matrix A=(a_{ij}) (i = 1, 2, . . . , m; j = 1, 2, . . . , n) as the

time it takes CN C_i to complete job J_j. the time taking by a node i to execute a job j, $a_{ij} = \frac{L_j}{S_i}$

They define the decision matrix X = (x_{ij}) (i = 1, 2, . . . ; m; j = 1; 2; . . . n) as :

$$x_{ij} = \begin{cases} 1 & \textit{if } J_j \textit{ is assigned to cloud node } C_i \\ 0 & \textit{otherwise} \end{cases}$$

For a job assignment on the cloud nodes, two essential parameters are calculated to measure the performance of this assignment; The first is the flowtime for each cloud node which is equal to the time taken by this cloud node to execute all of the tasks assigned to it, for example for the cloud node C_i the flowtime

$$SF(C_i) = \sum_{j=1}^{n} a_{ij} * x_{ij}$$

The second parameter is the makespan (MS), which is equal to the maximum flowtime of cloud nodes

$$\boldsymbol{MS = max_{i\in 1..m} SF(C_i)}$$

Based on this modeling, the authors propose an optimization based on the biogeography algorithm for optimal planning of tasks in the cloud, where the objective here is to minimize the makespan of the X matrix.

The main goal of scheduling in the second article [19] is to enhance the completion time for executing all tasks on the VMs, in the same time, minimize the total cost of usage the resource and maximize utilization of the resource. For this reason, the authors propose the Tournament Selection-Genetic Algorithm (TS-GA) which is task scheduling algorithm based on the default GA with some modifications. According to these modifications the parents will be considered in each population beside the produced child after the crossover process. Also, the Tournament Selection is used to select the best chromosomes to overcome the limitation of the population size. So the main idea of this proposed algorithm is that after each selection in the population, there is a solution that might satisfy good fitness function, but it is not selected to crossover process.

By the proposed algorithm, this solution is not removed from the population, but it is chosen and added to the population when next iteration is started. This step is considered as a good step as some of the iterations can generate the best solution.

The representation of solutions in task scheduling for each chromosome consists of VM ID and ID for each task to be executed on these VM. Each VM and the executed tasks on it are encoded into the binary bit.

The completion time of task Ti on VMj as is defined using equation Completion time=CTmax[i,j]

where

$$i \in T, \quad i = 1, 2, 3, \ldots n$$

$$j \in VM \quad , \quad j = 1, 2, 3, \ldots m$$

Therefore, to reduce the completion time which can be denoted as CTmax, the execution time of each task for each virtual machine must be calculated for the scheduling purpose. If the

processing speed of virtual machine VMj is PSj, then the processing time for task Pi can be calculated by equation

$$Pij = \frac{Ci}{Psj}$$

Where, Pij the processing time for task Pi on VMj and Ci computational complexity of task Pi

The processing time of each task in the virtual machine can be calculated by equation

$$Pj = \sum_{i=1}^{n} Pij$$

Tournament Selection is used to overcome the limitation of the population size. Two individuals are chosen at random from the population. A random number r is then chosen between 0 and 1. If r < k (where k is a parameter, for example, 0.75), the fitter of the two individuals is selected to be a parent; otherwise the less fit individual is selected. The non-chosen individuals are then returned to the original population and could be selected again.

Then a new crossover has been used differently from the used crossover in the original GA. Therefore, two chromosomes which are selected to crossover process to generate two offspring will be considered as offspring also. So four children produced. After that, the two best children are chosen from these.

In this modified TS-GA algorithm the new populations after crossover are added into old populations (parents).

There is a solution that might satisfy good fitness function, but it is not selected during the crossover process. By the proposed TS-GA algorithm, this solution is not removed from the population, but it is chosen and added to the population when next iteration is started. This step is considered as good step as some of the iterations can generate the best solution.

In [27] the authors proposed a Multi-objective Optimization Algorithm for Cloud Computing Task Scheduling Based on Improved Ant Colony Algorithm (MO-ACO) to find the optimal resource allocation for each task in the dynamic cloud System which minimizes the makespan and costs of tasks on the entire system, and balance the entire system load.

In the algorithm, each ant looks for the appropriate virtual machine for the corresponding task. The ants finally find an optimal matching scheme according to the optimized target. The ants find the optimal solution in parallel, and communicate with each other through information.

Then adjusting the pheromone on the task with the virtual machine path, which affects the selection of the ant for the other ant and the next iteration for the ant. In this article, the initial pheromone, heuristic function and pheromone update rule of ant colony algorithm are improved by combining the objective function to be optimized.

Experimental results compared to Ant Colony Optimization (ACO) and Min-Min showed the MO-ACO algorithm satisfies expectation.

II.3. PSO algorithm (particle swarm optimization)

Particle Swarm Optimization (PSO) algorithm, As GA and ACO is one of the most well-regarded algorithms in the literature of stochastic optimization approaches. It belongs to the family of swarm-based techniques and is a population-based algorithm. [23]

The PSO algorithm simulates the navigation mechanism of birds in nature. The main inspiration Is the simple equations proposed by Raynold in 1987 And it was developed by Kennedy and Eberhart in 1995, in which both researchers got inspired by social and computer sciences [24].

For simulating the interaction of individuals in flying swarms. This swarm of particles simultaneously explore a problem's search space with the goal of finding the global optimum configuration [25]. Oriented to the movements of their neighbors and the best positions in solution space found so far, particles move in solution space with velocities and randomness. This plays the role of the explorative mechanism. The orientation to the best so far found positions is the exploitative counterpart.[23]

In PSO algorithm, candidate solutions (fishes, birds,.etc.) or particles travel through the search space to find an optimal solution, by interacting and sharing information with neighbor particles, namely their individual best solution (local best) and computing the neighborhood best. Also, in each step of the procedure, the global best solution obtained in the entire swarm is updated. Using all of this information, particles realize the locations of the search space where success was obtained, and are guided by these successes. In each step of the algorithm, a fitness function is used to evaluate the particle success.

To model the swarm, in iteration t, each particle n moves in a multidimensional space according to position ($x_n[t]$) and velocity ($v_n[t]$) values which are highly dependent on local best ($x_{1n}[t]$), also known as the cognitive component, and global best ($x_{2n}[t]$), typically known as the social component, as follows.

$$v_n[t+1] = \omega v_n[t] + \sum_{i=1}^{2} \rho_i r_i (x_{in}[t] - x_n[t])$$

$$x_n[t+1] = x_n[t] + v_n[t+1]$$

Coefficients ω, x1; and x2 assign weights to the inertial influence, the local best, and the global best when determining the new velocity $v_n[t+1]$, respectively. Typically, the inertial influence is set to a value slightly less than 1.

$\rho_1$and ρ_1 are constant integer values, which represent “cognitive” and “social” components. However, different results can be obtained by assigning different influences for each component. For example, some works additionally consider a component called neighborhood best (which in this case would be a p3).

The parameters r1 and r2 are random vectors with each component generally a uniform random number between0 and 1. The intent is to multiply a new random component per velocity dimension, rather than multiplying the same component with each particle’s velocity dimension. Depending on the application and the characteristics of the problem, tuning all these parameters properly may lead to better results. However, this is a problem associated with the branch of parameterized complexity and is far from being completely solved. In fact, Fig II.3 clearly illustrates balls of different sizes.

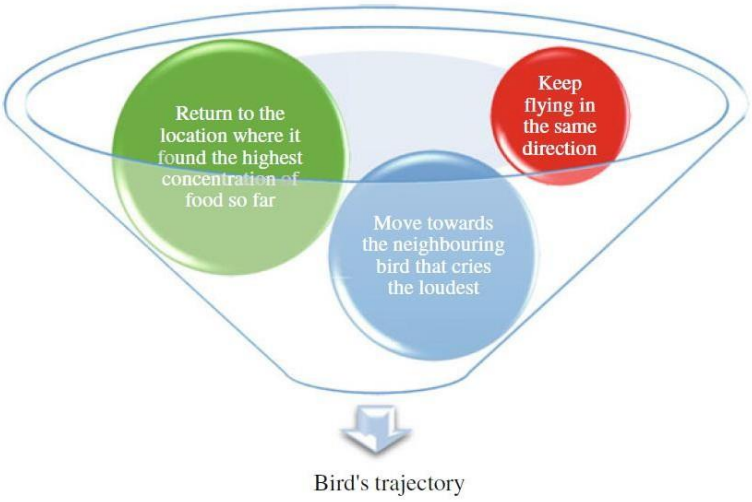

Fig II.3: Three main causes that may influence a bird’s (particle) trajectory

Where:

a) Red ball is the previous direction it was traveling.

b) Green ball is the best location it personally found thus far.

c) Blue ball is the best location the flock collectively found thus far.

This is an analogy made to the influence of each component from velocity equation namely the previous velocity of the particle by means of $\boldsymbol{\omega}$ (red ball), its individuality by means of ρ_1 (green ball), and how much it cooperates with its society by means of ρ_2 (blue ball).

In the beginning, particles' velocities are set to zero and their positions are randomly set within the boundaries of the search space. The local and global bests are initialized with the worst possible values, taking into account the nature of the problem. For instance, in a cost problem where the objective is to minimize the fitness function, particles are initialized with a large value (tending to infinity). There are a few other parameters that need to be adjusted:

• Population size: Very important to optimize to get overall good solutions inacceptable time (remember that each particle is a potential solution of the problem).

• Stopping criteria: It can be a predefined number of iterations without getting better results (known as stagnation), or other criteria depending on the problem (common examples are time or performance). [24]

II.3.1. PSO Benefits

The main advantage of using the PSO is its simple concept and ability to be implemented in a few lines of code. Furthermore, PSO also has a short-term memory, which helps the particles to fly over the local best and the global best positions. Alternatives, such genetic algorithms (GA), are more complex and, most of the time, they do not consider the previous iteration or the collective emergent performance. For instance, in GA, if a chromosome is not selected, the information contained by that individual is lost.[24]

II.4. Modeling job scheduling in Cloud Computing using PSO algorithm

We present a scheduling heuristic optimizing the cost and makespan of job scheduling using the mapping solution computed by particle swarm optimization-based algorithm. PSO is one of the latest evolutionary algorithms inspired by the social behavior of fish schooling or bird flocking. The particle corresponds to an individual bird or fish searching in a natural space.

In the following, we adopt the general model and notation used by existing works on design and implement of a scheduling strategy based on PSO algorithm [26].

a) Scheduling Problem Formulation

A job scheduling is usually represented by a set of nodes V = {t_1, …, t_n} represents the tasks in the workflow applications, and n is the total number of tasks in the scheduling.

Suppose there are a total of m resources in the cloud environment. The resources can be denoted as R = {R_1, … , R_m}. The scheduling problem is to find an optimal mapping M between tasks and resources according to some optimization objective.

b) the Fitness Function

As mentioned before, Makespan is an objective that is critical for this scheduling. In the following, we formulate several optimization objectives. Let $makespan_{total}(M)$ denote the Makespan of the job scheduling with respect to the mapping *M*:

$$\boldsymbol{makespan_{total}(M) = finish\ time\ of\ last\ task - start\ time\ of\ the\ first\ task} \qquad \textbf{(1)}$$

For the objective of optimizing Makespan (such as the work by Zhang et al [13]), the fitness function can be defined as:

$$\boldsymbol{fitnes\ function = Makespan_{total}(M)} \qquad \textbf{(2)}$$

The objective is to minimize *Fitness function.*

c) Select the Particle Swarm Model

In the PSO algorithm, each particle represents a possible solution. The flock or swarm of particles is randomly generated initially. Each particle has its own position in the space and a fitness value, and has the velocity to determine the speed and direction it flies. A group of candidate solutions (particles) moves around in the search space based on the particles' updated

position and velocity so that the PSO algorithm can get a optimized solution. Particles in the search process update themselves by tracking the local best position is the individual best known position in terms of fitness value reached so far by the particle itself. And Another the global best position is the best position in the entire population. Suppose the number of particles is *N*. The velocity and position of each particle are calculated by formulations (3) and (4).

$$v_i^{k+1} = \omega v_i^k + c_1 r_1 (p_i^k - x_i^k) + c_2 r_2 (p_g^k - x_i^k) \qquad 1 \le i \le N \qquad (3)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \qquad 1 \le i \le N \qquad (4)$$

$$p_i^k = Best(x_i^k, p_i^{k-1}) \qquad 1 \le i \le N \qquad (5)$$

$$p_g^k = Best(p_1^k, \ldots, p_N^{k-1}) \qquad (6)$$

Where:

v_i^{k+1} Velocity of particle *i* at iteration *k* + 1

v_i^k Velocity of particle *i* at iteration *k*

x_i^{k+1} Position of particle *i* at iteration *k* + 1

x_i^k Position of particle *i* at iteration *k*

p_i^k Best position of particle *i* so far at iteration *k*

p_g^k Best position of all particle*s* so far at iteration *k*

ω Inertial weight

c_1, c_2 Acceleration coefficients (positive constants)

r_1, r_2 Random numbers in [0,1]

Particle's best position is calculated by the fitness function through (5) and (6). The velocity is calculated by three factors each iteration. They are current velocity, local best position and global best position. The position is updated according to its current position and updated

velocity. These ensure the particles search around the local and global best positions and converge to a global best position in the limited iteration.

In the job scheduling problem, the particle represents a mapping between *m* resource and *n* task. Each particle position represents one schedule, and each position variable represents the job assigned to the corresponding resources. In Fig II.3, 11 possible particles for the mapping between 8 resources and 11 tasks is illustrated. The evaluation of each particle is performed based on fitness function.

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
R4	R2	R4	R5	R1	R7	R5	R3	R6	R8	R1

Fig II.4: A sample particals

d) the Termination Criteria of the Algorithm

The most commonly used termination criteria of PSO algorithm are preset a maximum number of iterations, or terminate algorithm when the fitness value of solutions have no significant changes after multiple iterations in the searching process. In our case we will choose the first termination criteria which is the number of iterations.

e) The pseudo code for PSO

Tab II.2 lists the steps of PSO algorithm. The initial step is to initialize the swarm, set randomly for each particle's his position and velocity. If iteration criterion is not met, the algorithm repeatedly does the following: firstly calculate each particle's fitness value by using the fitness function given in (2), and then update its local best position using (5); calculate global best position using (6); for each particle, update its velocity and position using (3) and (4). Finally, the global best position is the optimal mapping solution.

```
1:  Initialize particles' position and velocity randomly.
2:  While stopping criterion is not satisfied do
3:    For each particle do
4:      Calculate its fitness value using fitness function.
5:      Update its local best position
6:    End For
7:    Update global best position.
8:    For each particle do
9:      Update its velocity and position.
10:   End For
11: End While
12: Return global best position.
```

Tab II.2: The pseudo code of PSO algorithm

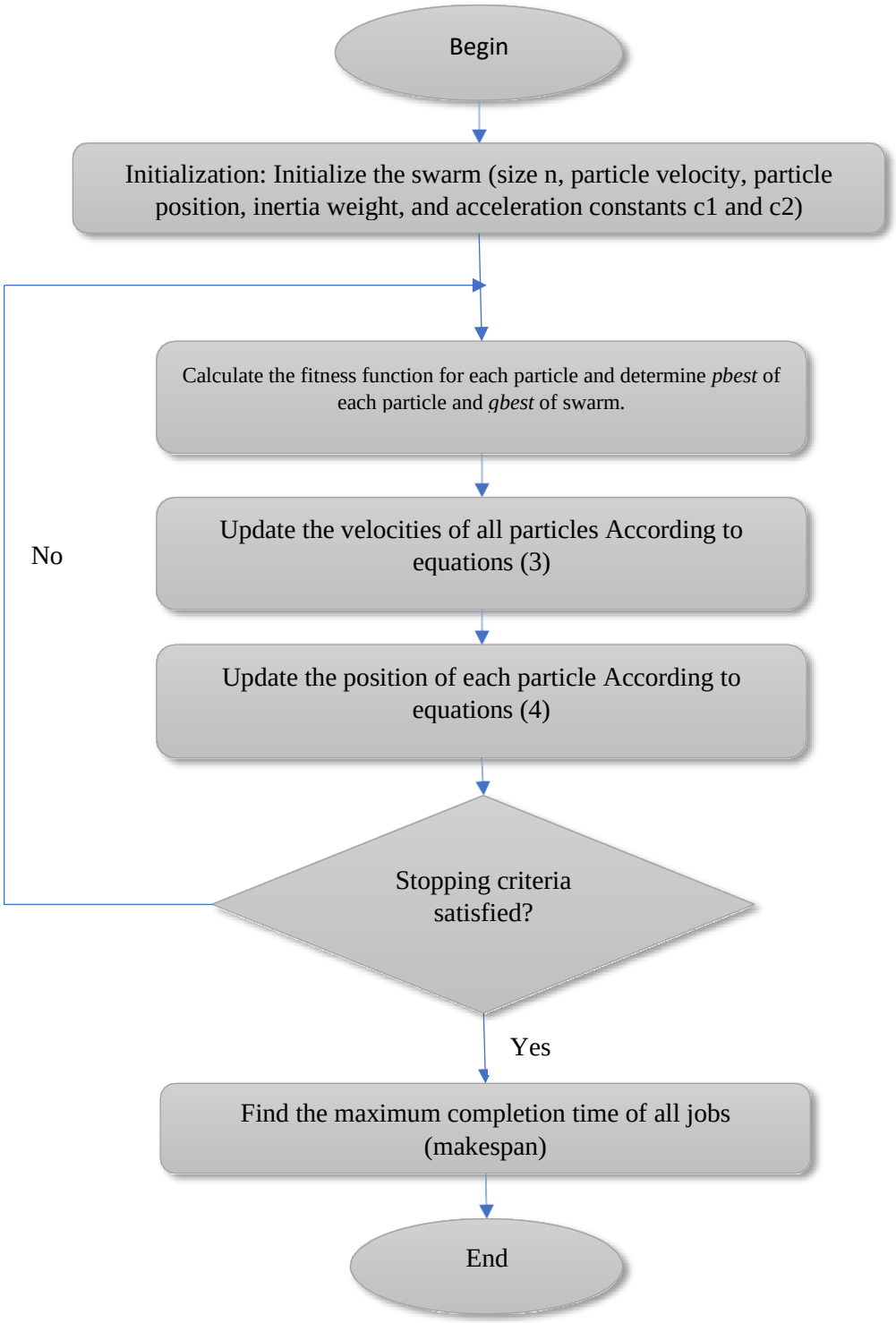

Fig II.5: The flowchart of PSO for job scheduling in cloud computing

II.5. Conclusion

Cloud computing is an emerging paradigm in the computing world, in which tasks scheduling is one of the most important areas of research. An efficient algorithm-based job scheduling is necessary to achieve user satisfaction and maximize profits for cloud computing service providers.

In this chapter on an attempt to show the recent work linked to our project and which will make the fundamental aspects as well as the main challenges that we will be tempted to materialize in the third chapter by relying on the CloudSim library and the language of java programming in order to achieve to implement the Particles swarm optimization algorithm to try to optimize the scalable partition of distributed computation.

Chapter III

Implementation and Experiment Result

III.1. Overview

Tasks scheduling is one of the keys for large-scale Cloud applications. Extensive research on all issues in real environment is extremely difficult because it requires developers to consider network infrastructure and the environment, which may be beyond the control. In addition, the network conditions cannot be controlled or predicted. Performance evaluations of workload models and Cloud provisioning algorithms in a repeatable manner under different configurations are difficult. Therefore, simulators are developed. To understand and apply better the state-of-the-art of cloud computing simulators. Many simulators are available forthe modeling the cloud environment like CloudSim, GreenCloud, CloudAnalyst, etc.

This chapter explains the computational experiments for evaluating the performance of the particle swarm optimization (PSO) with SJF-PSO using the CloudSim simulator.

III.2. Cloud computing simulation

Cloud Computing is faced many problems by many Industries which are directly or indirectly related to the Information Technology. This field is growing very fastly as many of the big players in the Information Technology field like Microsoft, Google, Amazon, SAP are investing a lot of money to get improved results. They feel that it is the future of technology.

There are variety of simulator tools for modelling and simulation of large-scale Cloud computing environments (Fig III.1). Generally, we can distinguish between two types of simulators: graphical user interface (GUI) based simulators or programming language-based simulators (like Java for example).[29]

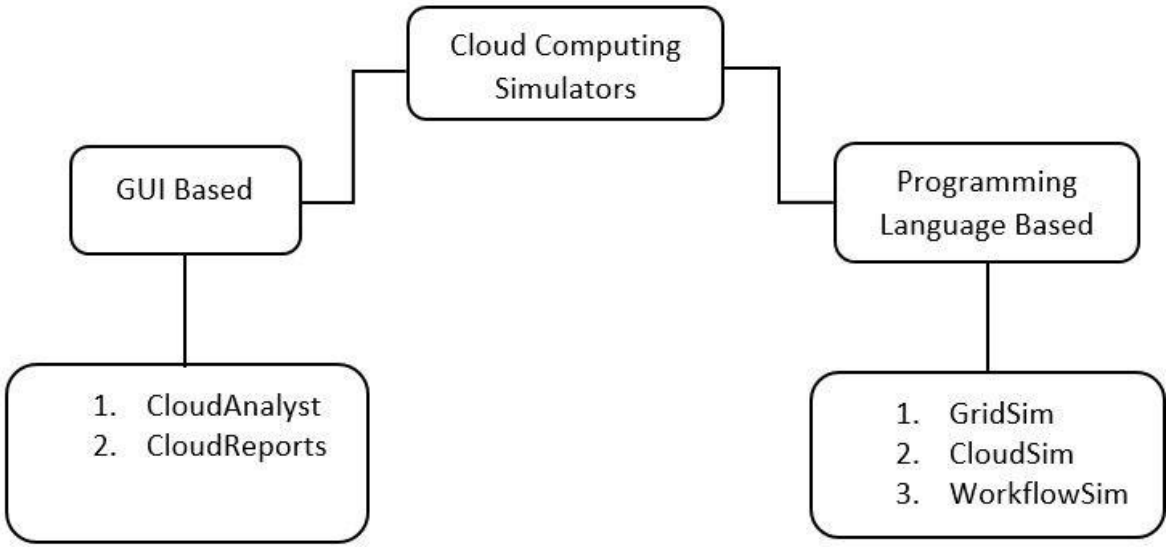

Fig III.1: Cloud Computing Simulation frameworks [29]

In this work, we chose to use the CloudSim Simulator because of my participation in on-line certified training course on “learn basics of CloudSim” from the Udemy site [33], by the trainer Anupinder singh, to simulate the proposed algorithms for job Scheduling in Cloud Computing environment. we will discuss the details of this simulator in the next section.

III.2.1. Reasons for adopting for modeling and simulation [30]

- Various cloud computing data centers
- Virtualization of server hosts, associated with customized policies for provisioning host resources to VMs
- Specific purpose software containers
- Power-aware computational resources
- Different datacenter network topologies as well as message-passing applications
- Dynamic addition or removal of simulation components
- Simulation can be stop and resume
- User-defined policies provide support for allotment of hosts to VMs and policies for allotment of host resources to VMs

III.3. Software Tools Used in the Research

III.3.1. Eclipse

Eclipse is an integrated development environment (IDE) used in computer programming. It contains a base workspace and an extensible plug-in system for customizing the environment. Eclipse is written mostly in Java and its primary use is for developing Java applications, but it may also be used to develop applications in other programming languages via plug-ins, including Ada, ABAP, C, C++, C#, Clojure, COBOL, D, Erlang, Fortran, Groovy, Haskell, Ja vaScript, Julia, Lasso, Lua, NATURAL, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Rust, Scala, and Scheme. It can also be used to develop documents with LaTeX (via a TeXlipse plug-in) and packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others.[32]

The initial codebase originated from IBM VisualAge. The Eclipse software development kit (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules.[32]

III.3.2. CloudSim

CloudSim is a simulation tool that developed in CLOUDS laboratory in University of Melbourne for cloud computing environment. It enables users to evaluate specific system issues without taking into consideration the low-level facts associated to cloud-based infrastructures and services. It means support seamless modeling, simulation and implementing on designing cloud computing framework. CloudSim is an autonomous platform that supports of large-scaled cloud platform for describing users, applications, datacenters, brokers, scheduling and provisioning policies. [30]

CloudSim is an open source software that enables users to test application in controlled and repeatable environment, find the system bottlenecks without the need of real clouds and try different configurations for developing adaptive provisioning techniques.[30]

III.3.2.1. The advantages of CloudSim Simulation [30]

- The elasticity of significant configurations
- Support for diverse cloud Environnent
- It enables modeling of application services in any environment
- cloud based application implementation in minimum time and minimum effort
- Ease of use and customization

III.3.2.2. The disadvantage of CloudSim simulation [30]

- Does not support Graphical User Interface (GUI) to generate results
- Support restricted workload transfer generator due to basic network model
- Not helpful for modeling for parallel experiments;
- Difficult to analyzing simulation results
- So researcher might need to analyze data using other tools such as Excel.

III.3.2.3. CloudSim Versions [30]

There are different versions of CloudSim available for cloud computing environment. They are

a) Version 1.0:

Version 1.0 was the first version of CloudSim, and it was released on April 7, 2009. It supports seamless scale modeling and simulation, moreover implementing on designing cloud computing framework. CloudSim is an autonomous platform that supports of large scaled cloud platform for describing users, applications, datacenters, brokers, scheduling and provisioning policies.

b) Version 2.0:

It was released on May 27, 2010. This version improved the simulation that allowed superior scalability and efficiency of simulation and addition and removal of simulation components during simulation execution.

c) Version 2.1

It was released on July 27, 2010. It has the capability to migrate using apache Maven. Maven provides various tools and plug-in for simplifies the project such as bug fixes, refactoring and removal of obsolete code.

d) Version 2.1.1:

It was released on Feb 1, 2011. It has capability to fix some bug which comes in version 2.1.

e) Version 3.0:

It was released on June 11, 2012. This version of CloudSim also updates bug fixes. It also updates in this are new VM scheduler, novel datacenter network model, new VM allocation and selection policies, new power models, new workload tracks, support for external workloads and support for user define end of simulation. It support elimination of a few classes have been made like Cloud coordinator, PowerPe, Sensor and Power.PeList. This version has capability to fix some API changes and bugs.

f) Version 4.0:

This is the latest version of CloudSim was released Jan, 2017. New applications of Cloud computing are emerging such as Internet of Things and Big Data, and new technologies are being incorporated to the fabric of Cloud data centers such as Container virtualization. It support for container virtualization and lots of bug fixes.

III.3.2.4. Architecture of CloudSim Simulation

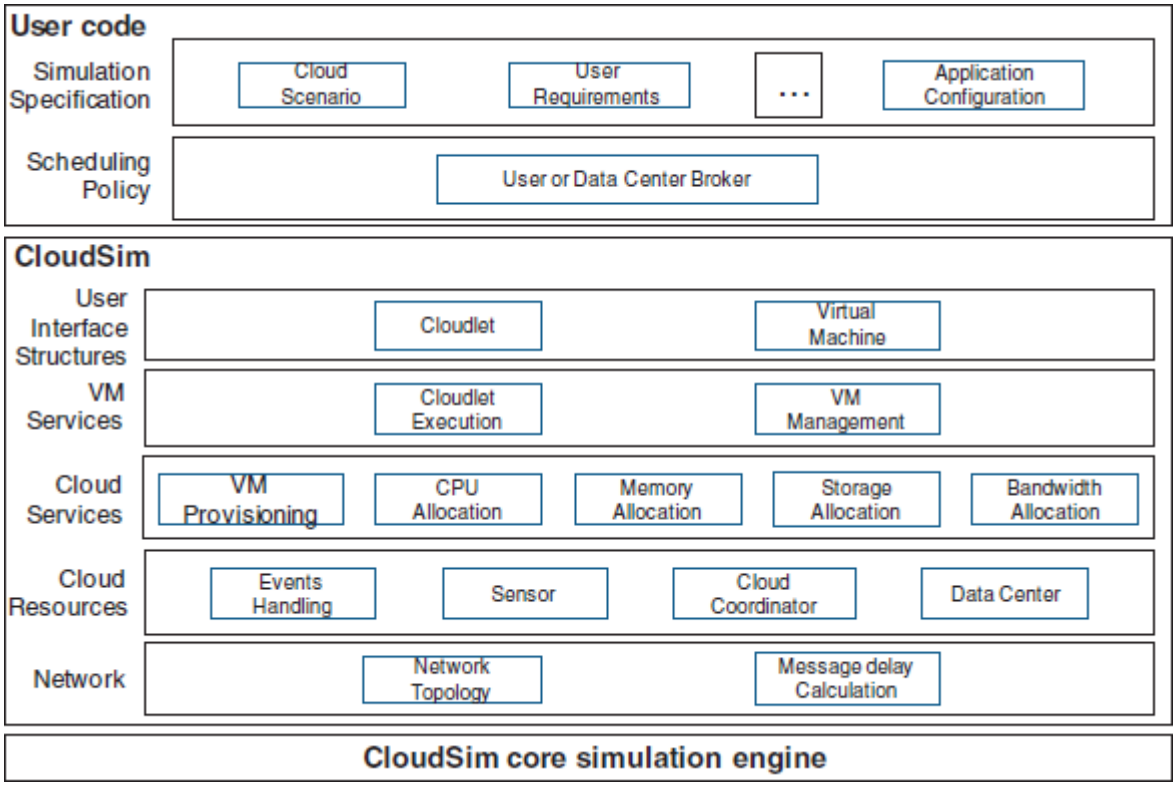

Fig III.2: Layered CloudSim architecture. [31]

The above diagram demonstrates the layered architecture of CloudSim Simulation Toolkit. The CloudSim Core simulation engine provides support for modeling and simulation of virtualized Cloud-based data center environments including queuing and processing of events, creation of cloud system entities (like data center, host, virtual machines, brokers, services, etc.) communication between components and management of the simulation clock. [31]

The CloudSim layer provides dedicated management interfaces for Virtual Machines, memory, storage, and bandwidth. Also, it manages the other fundamental issues, such as provisioning of hosts to Virtual Machines, managing application execution, and monitoring dynamic system state (e.g. Network topology, sensors, storage characteristics, etc), etc. [31]

The User Code layer is a custom layer where the user writes their own code to redefine the characteristics of the stimulating environment as per their new research findings. [31]

the fundamental classes of CloudSim, which are also the building blocks of the simulator. The overall Class design diagram for CloudSim is shown in Fig III.3

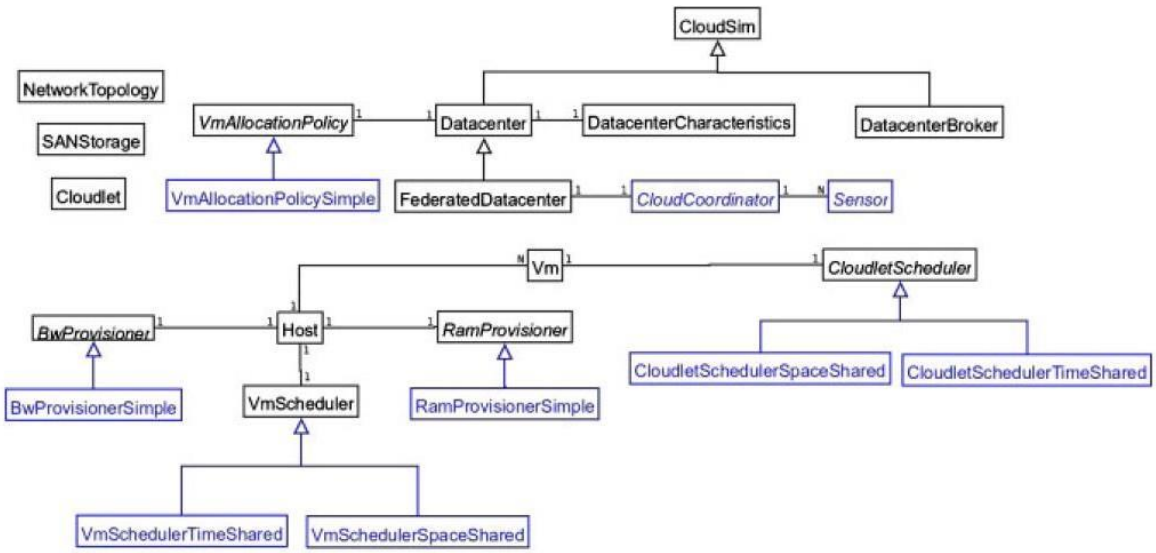

Fig III.3: CloudSim class design diagram. [31]

a) Cloudlet

This class models the Cloud-based application services (such as content delivery, social networking, and business workflow). CloudSim orchestrates the complexity of an application in terms of its computational requirements. Every application service has a pre-assigned instruction length and data transfer (both pre and post fetches) overhead that it needs to undertake during its life cycle. This class can also be extended to support modeling of other performance and composition metrics for applications such as transactions in database-oriented applications. [31]

b) Datacenter

This class models the core infrastructure-level services (hardware) that are offered by Cloud providers (Amazon, Azure, App Engine). It encapsulates a set of compute hosts that can either be homogeneous or heterogeneous with respect to their hardware configurations (memory, cores, capacity, and storage). Furthermore, every Datacenter component instantiates a generalized application provisioning component that implements a set of policies for allocating bandwidth, memory, and storage devices to hosts and VMs. [31]

c) DatacenterBroker

This class models a broker, which is responsible for mediating negotiations between SaaS and Cloud providers; and such negotiations are driven by QoS requirements. The broker acts on behalf of SaaS providers. It discovers suitable Cloud service providers by querying the CIS and undertakes online negotiations for allocation of resources/services that can meet the application's QoS needs. Researchers and system developers must extend this class for evaluating and testing custom brokering policies. The difference between the broker and the CloudCoordinator is that the former represents the customer (i.e. decisions of these components are made in order to increase user-related performance metrics), whereas the latter acts on behalf of the data center, i.e. it tries to maximize the overall performance of the data center, without considering the needs of specific customers. [31]

III.3.2.5. Basic scenario of CloudSim Simulation

The main scenario of CloudSim simulation is presented in figure III.3. This scenario may be described the detail of each component of CloudSim as follows: that Data centers (DC) provides resource such as more than one host. Host is physical machine which allocates more than one VMs. Virtual machine (VM) are logical equipment on that the cloudlet will be executed. Broker has DC characteristics that allow it to submit virtual machines to the exact host. Cloud Information Service (CIS) is responsible for the listing of resources, indexing, as well as discovering the efficiency of data centers [30].

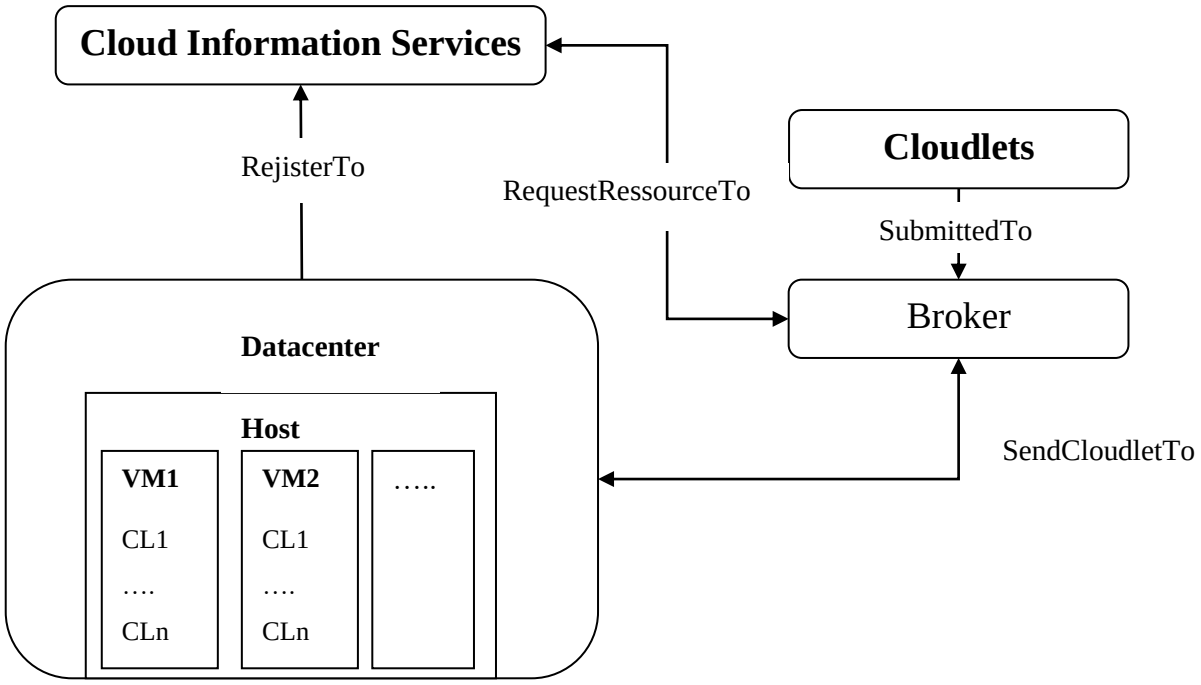

Fig III.3: Basic scenario of Simulation.[30]

CloudSim gives classes for virtual machines, Datacenters, DatacenterBroker, cloudlets and scheduling methods. Different stages of CloudSim lifecycle is shown in Fig III.4.

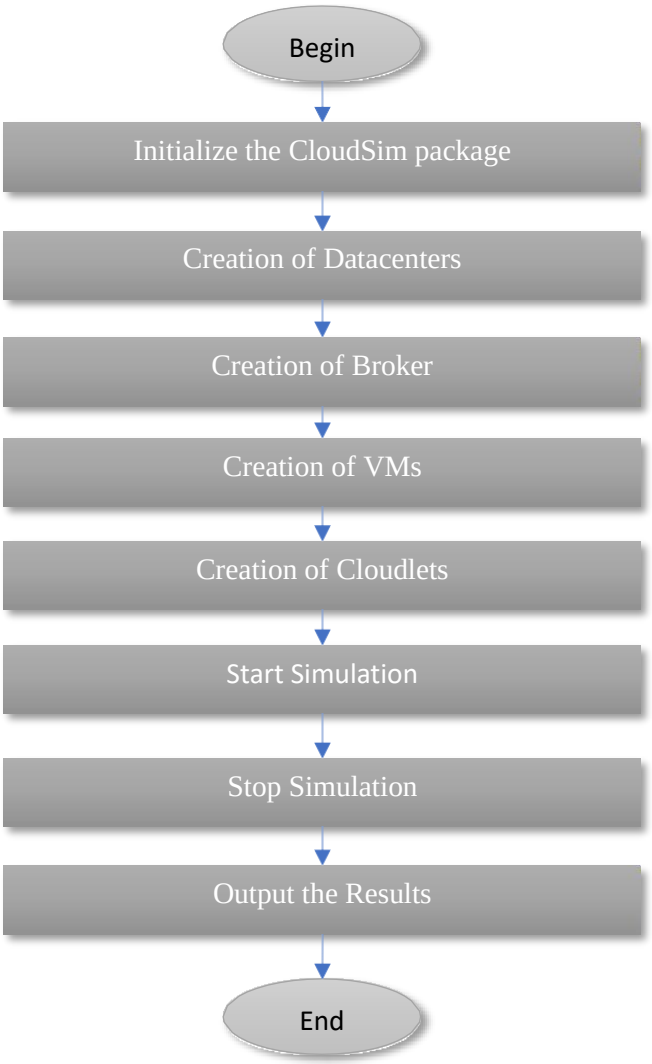

Fig III.4: CloudSim lifecycle.

III.4. Implementation and Experimented Result

The implementation code of the proposed job scheduling algorithm PSO has been written by using Eclipse Java 2019-09 this code is implemented to extract results that used to evaluate the performance of the proposed job scheduling algorithm. The computer system specification used to implement the algorithm is:

- Processor: Intel® core™ i5-8350U CPU @ 1.70GHz (8 CPUs), 1.9GHz
- Memory (RAM): 16GB
- Hard disk drive: 500 GB (SSD)
- Operating System: Win10 Professional 64bits

We use the JSwarm[17] package to conduct the simulation in PSO. The PSO class, shown in Fig III.1, aggregates Swarm class that also has Particle and FitnessFunction class. Swarm, Particle and FitnessFunction are in JSwarm package. Then I define our own SchedulerParticle class for PSO scheduling algorithm that extends the Particle class. For FitnessFunction class, I extend it to my fitness function class, which is SchedulerFitnessFunction.

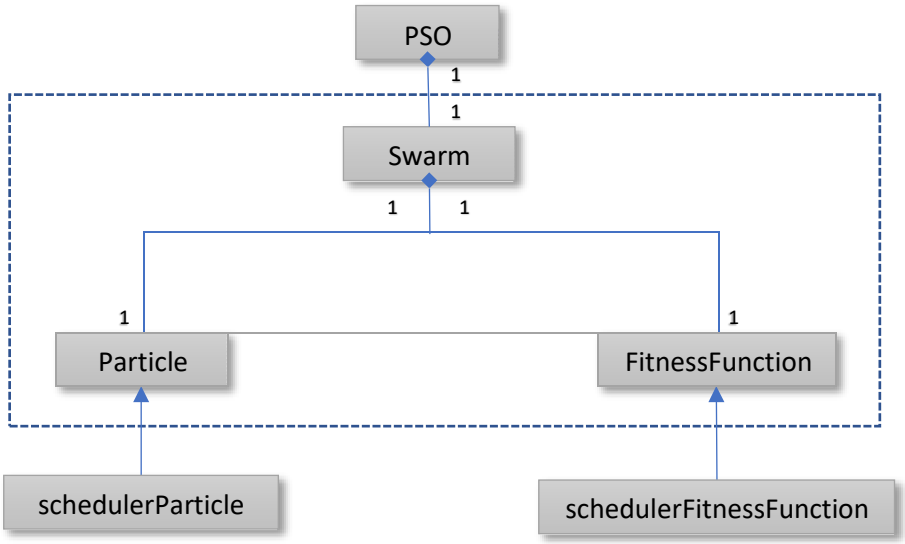

Fig III.5: Class diagram of PSO scheduling

In the experiment, I test two algorithms. PSO algorithm and SJF-PSO algorithm are the scheduling algorithms using the Makespan fitness function. The SJF-PSO proposed algorithm That use the PSO algorithm extended with sorting.

SJF or the Shortest Job First is One of the proposed tasks scheduling algorithms. The task in this algorithm has the lowest execution time, which its process having the smallest CPU burst gets a high priority and executes at the first by the CPU. The remaining tasks wait for the execution during this time, and finally the task that has the highest execution time will be executed at the end. The criteria of Shortest Job First (SJF) scheduling algorithm is that it process the smallest task first.[28]

In SJF-PSO, we have modified the CloudletSubmition method under the Datacenter Broker class Where we Calculate the cloudlets length for all cloudlets and submit them in an ascending order

we generate a text file filled with random integers; These random integers are a length for more than 1000 of cloudlets.

we write a class (InfBase) for the different basic information need in the scheduling simulation, to make it easier to control the number of Cloudlets, number of VMs, number of Datacenters, Number of Particles, and the Number of Iterations.

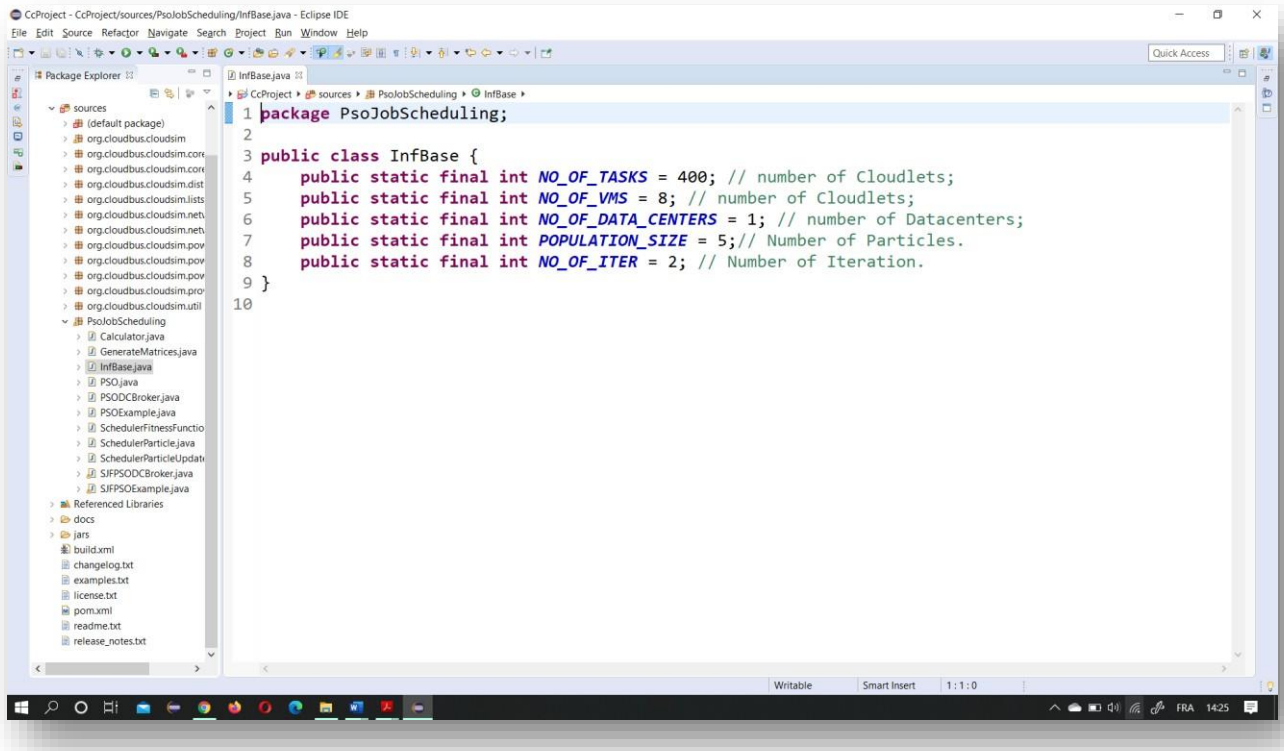

Fig III.6: The InfBase Class

Tab III.1 shown the different Simulation Parameters used in my implementation.

Parameter	Values
Number of Data-Centers	1
System Architecture	X86
Number of Hosts	4
VMM	XEN
OS	Linux
Transfer Rate	15 MB/s
Numbers of VMs	8
CPU (PEs Number)	4
RAM per VM	512 MB
Bandwidth	1000
PE‟s MIPS per VM	1000
Image Size	20000
Policy Type	Time Shared

Tab III.1: Simulation Parameters.

The data center that we use in the simulation (shown in Fig III. 9) consists of four hosts with four processing element (PEs) and each host having two VMs. They are constructed by cloud-based interface provided by CloudSim.

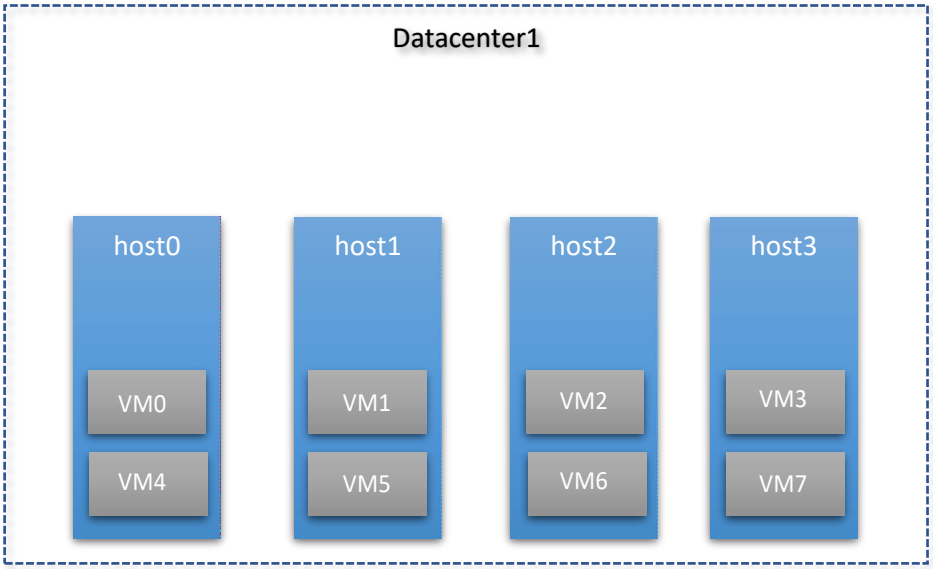

Fig III.7 Experimental datacenter infrastructure

CloudSim life cycle starts with initialization of CloudSim environment and ends with simulation results. Scheduling algorithm is applied after creation of cloudlets. The following table shown the pseudo code for the simulation steps:

Step 1: Initialize the CloudSim package by calling CloudSim.init(int numUser, Calendar cal, boolean traceFlag);
Step 2: Create Datacenters and set the needed Datacenter Characteristics;
Step 3: Create Broker;
Step 4: Create VMs and set the configuration of each VM. Add VMs to the VM list. Submit the VM list to the broker;
Step 5: Create Cloudlet. Similarly, set the properties of each Cloudlet and add it into the cloudlet list. Submit the cloudlet list to the broker;
Step 6: Implement a PSO tasks scheduling function in the DatacenterBroker;
Step 7: Bind the cloudlets to VMs by calling the function defined in DatacenterBroker to apply your own task scheduling algorithm;
Step 8: Start the simulation by calling CloudSim.startSimulation ();
Step 9: Get the results by printing them out on the console;
Step 10: Stop the simulation by calling CloudSim.stopSimulation ();

Tab III.2: pseudo code for the job scheduling simulation steps.

Once the configuration is done, we launch the simulation through the Button "Run". The simulation results presented by the following windows, including the optimal solution given by our "best fitness value" application, then the assignment of cloudlets to VMs after the creation of all the necessary entities (data center, broker, etc.).

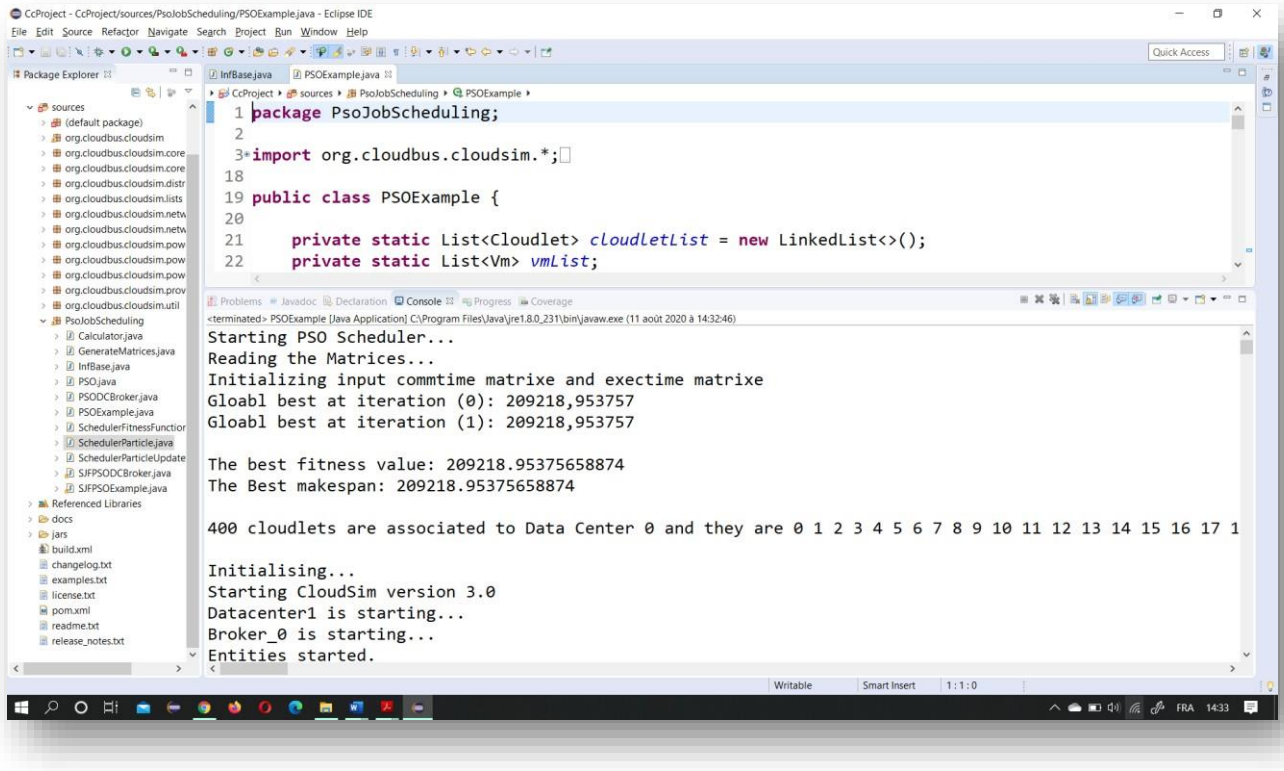

Fig III.8 Result of the Simulation shown the best makespane.

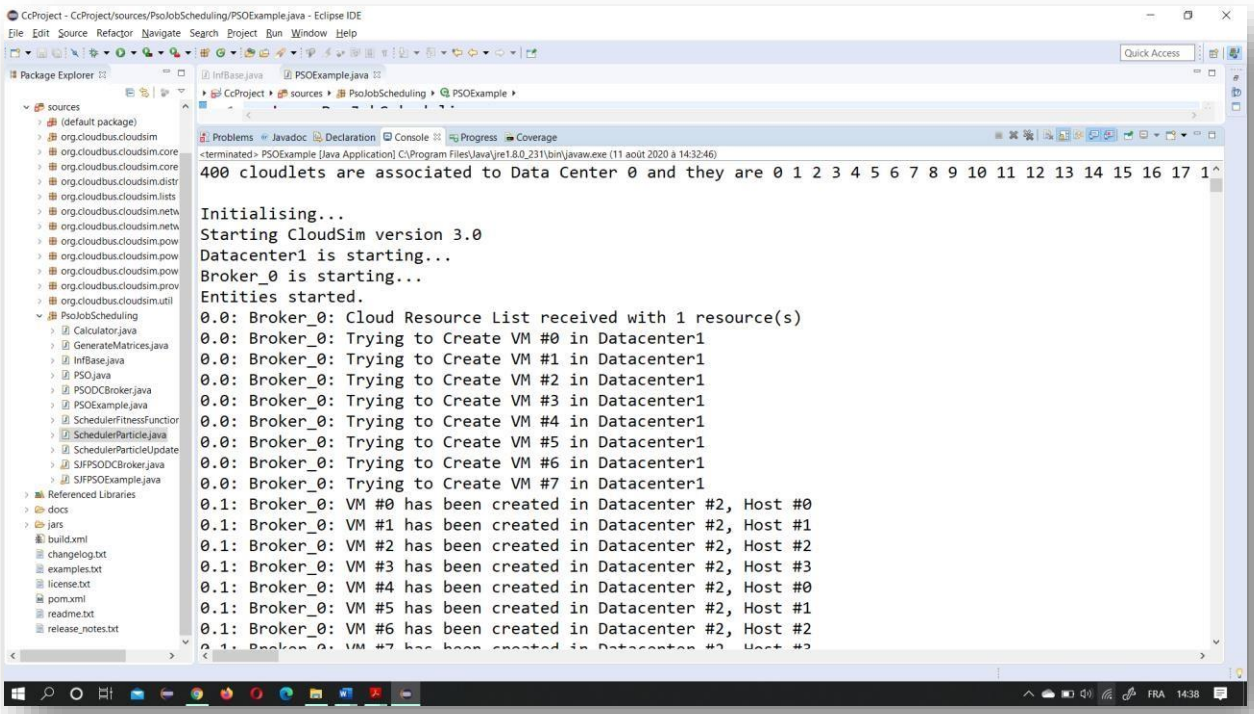

Fig III.9: creation of cloud entities.

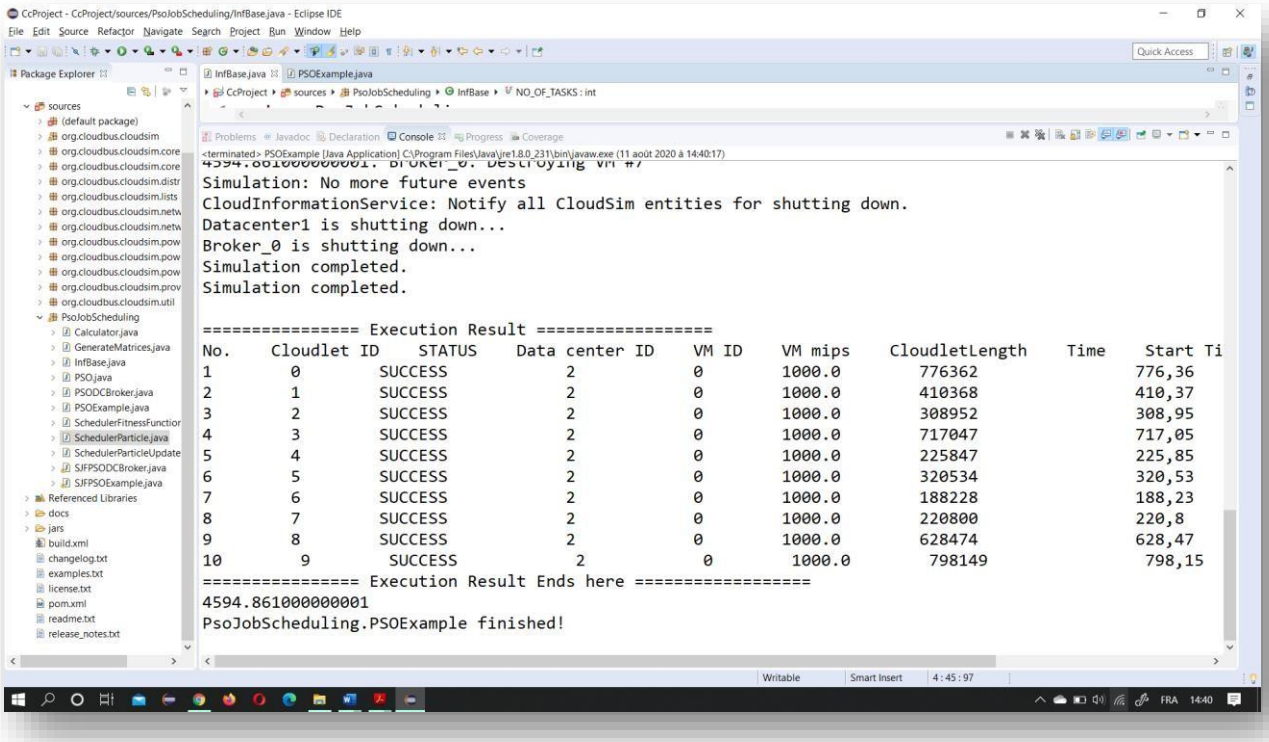

Fig III.10: execution statistics for 10 cloudlets.

The result of execution statistics for 20 cloudlets with the proposed SJF-PSO, is shown in the Fig III.11. The cloudlets 8,12,3 are executed by the VM ID=0 starting by the lowest length.

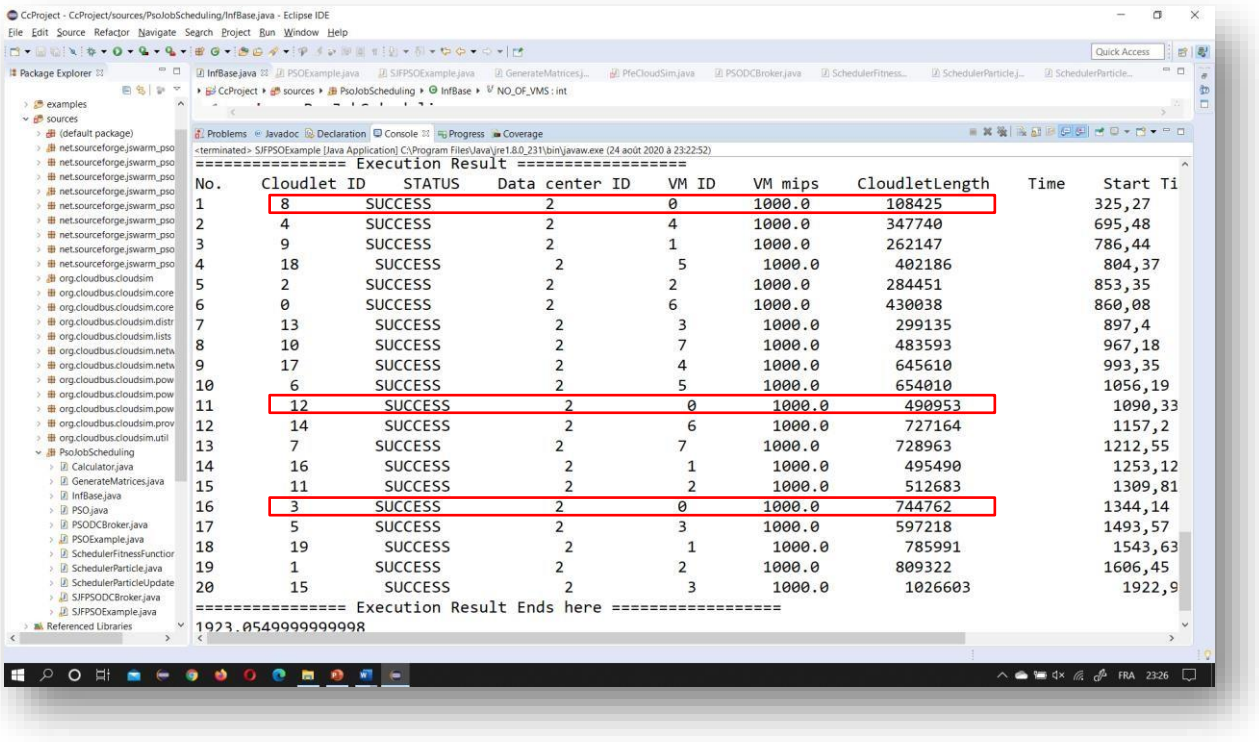

Fig III.11: execution statistics for 20 cloudlets with the proposed SJF-PSO.

For 20 cloudlets and 8 particles The following table shows the result of the best fitness value (Makespan) with respect to the number iterations:

No Of Iteration	Makespan PSO	Makespan SJF-PSO
3	11412,0	10859.3
6	11159,7	9265.0
9	11521,0	9259.9
12	9707,9	9700.2
15	8854,7	9555.9
18	9613,2	7795.0

Tab III.3: the result of the Makespane with respect to the number of iterations for PSO and SJF-PSO

For 10 Iteration and 8 Particles the following table shows the result of the best fitness value with respect to the number Cloudlets

No Of Cloudlets	Makespan PSO	Makespan SJF-PSO
100	6343,6	5445,6
200	10051,3	10976,9
300	15172,6	16048,8
400	20982,0	21016,1
500	22847,8	23308,6
600	31176,8	34001,9

Tab III.4: the result of the Makespane with respect to the number of Cloudlets for PSO and SJF-PSO

The simulation results obtained by the solution clearly demonstrate an efficient allocation using the SJF-PSO which allows a considerable optimization of the makespan of all the cloudlets at the cloud level compared to the result given by the PSO, the following two images show the results obtained.

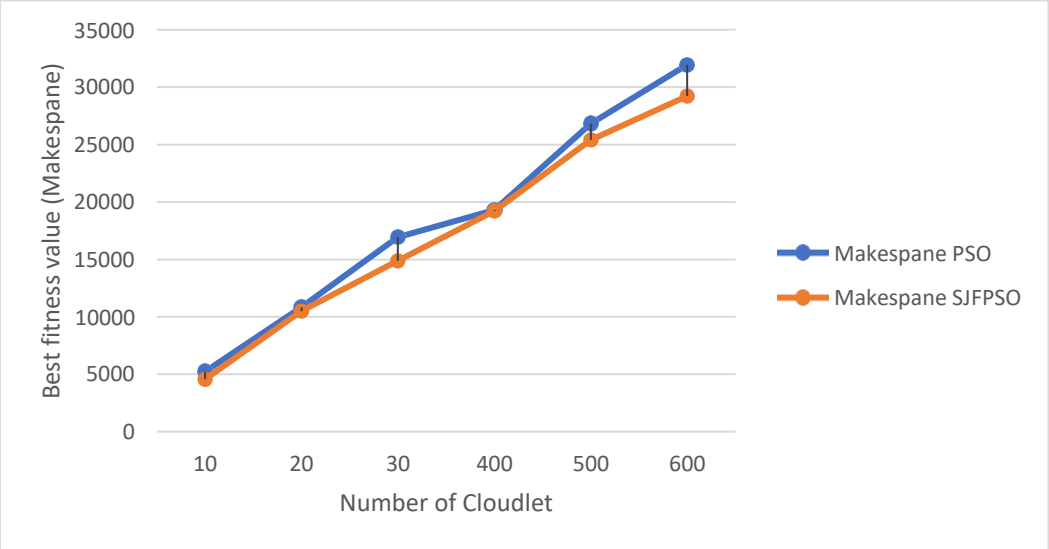

Fig III.11: Makespan comparison between PSO and SJF-PSO with respect of number of Cloudlets.

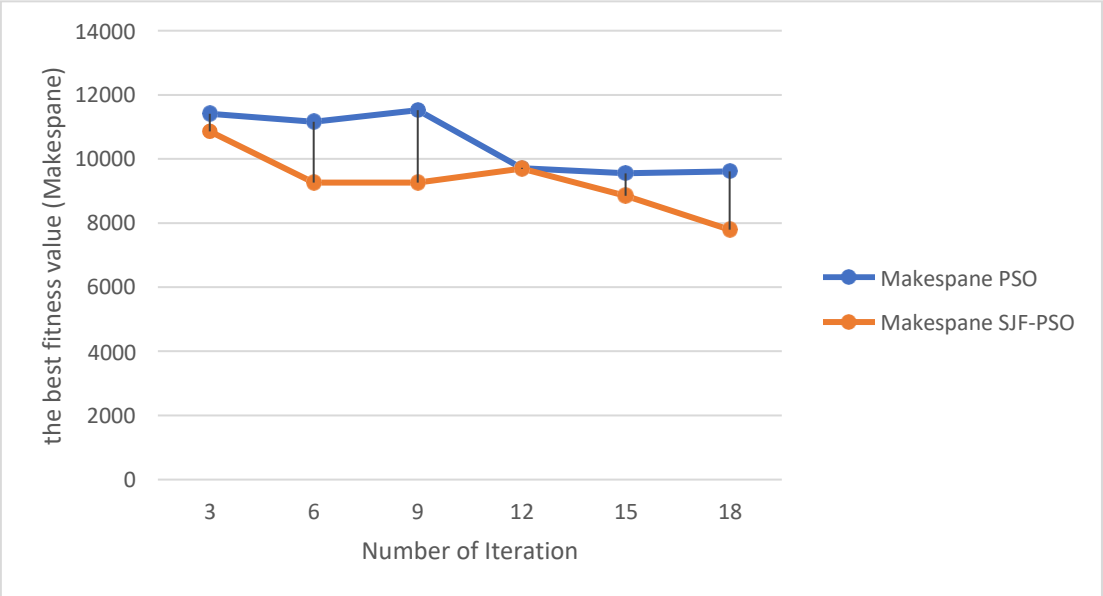

Fig III.12: Makespan comparison between PSO and SJF-PSO with respect of number of Iteration.

III.5. Conclusion

CloudSim Simulation Toolkit is one of the most used simulators for the implementation of the cloud-related research problem. The training we had did [33], helped me to follow the simulation- based approach of CloudSim and can leverage various benefits like, testing of services and scenarios in a controllable environment of cloudSim without spending a money on the real cloud services. Also simulating the small- or large-scale infrastructure to evaluate different sets of proposed algorithms as well as workload along with the resource performance. Ultimately facilitating the development, testing, and deployment of adaptive algorithms for resource provisioning techniques. And finally Optimizing the performance of various research-based algorithms with the cloudSim simulation engine before writing a research paper.

In this chapter, we have proposed our solution in the form of a java application which presents a simulation of a Cloud Computing in integration with the CloudSim library, of which we are interested in job scheduling on our Cloud by the implementation of two algorithm which are PSO and SJF-PSO

The proposed PSO scheduling considers minimizing the Makespan value of Job scheduling that cannot obtain the lowest Makespan. Therefore, we propose the SJF-PSO algorithm which firstly start by sorting the cloudlets in ascending order then we apply the same first PSO algorithm. The experimental results show that by adding sorting part into the PSO algorithm the Makespan can be reduced even further.

Conclusion

Conclusion

Cloud computing is a distributed based computer paradigm which is used by users to get good quality service with less cost. As task scheduling is a challenge in cloud computing, different algorithms have been suggested and applied to get better results regarding utilization of system resources, response time and satisfaction of user demands. In this study, two proposed algorithms for the task scheduling in cloud namely PSO, and SJF-PSO. These algorithms are implemented within the environment of CloudSim, in order to compare them.

Our results show that when using SJF-PSO for scheduling number of Cloudlets, the makespane becomes less than when using PSO, using the same numbers of cloudlets and the Cloud computing configuration. For future work, it is worth to investigate the effect of other parameters such as VMs, datacenters, memory, bandwidth for network and storage in cloud environments, and reflect that in real physical environment.

References

Articles, E-books:

[1] Peter Mell and Timothy Grance, The NIST definition of cloud computing, 2011, p (2-3).

[2] Yuri Demchenko, Marc X. Makkes, Rudolf Strijkers, Rudolf Strijkers, Intercloud Architecture for Interoperability and Integration,The 2nd NetCloud2012 Workshop on Network Infrastructure Services as part of Cloud Computing, in Proc. The 4th IEEE Conf. on Cloud Computing Technologies and Science, 2012, p (3 – 6)

[3] Keyun Ruan, Joe Carthy, Cloud Computing Reference Architecture and Its Forensic Implications: A Preliminary Analysis, Center for Cybersecurity and Cybercrime Investigation, University College Dublin, 2013, p (3-8)

[4] K. Chandrasekaran, Essentials of CLOUD COMPUTING, CRC PressTaylor & Francis Group, 2015, p (38-39)

[5] The Open Group Guide, Cloud Computing for Business,2010, p (12)

[6] Bernd Gastermanna, Markus Stoppera , Anja Kossikb , Branko Katalinicc, Secure Implementation of an On-Premises Cloud Storage Service for Small and Medium-Sized Enterprises, 25th DAAAM International Symposium on Intelligent Manufacturing and Automation, Elsevier , 2014, p (576)

[7] Chunye Gong, Jie Liu, Qiang Zhang, Haitao Chen and Zhenghu Gong, The Characteristics of Cloud Computing, 2010, In 39th International Conference on Parallel Processing Workshops, IEEE, 2010, p (275–279).

[8] Professional Xen Virtualization, Wiley, 2008, p (12).

[9] Massimo Cafaro, Giovanni Aloisio, Grids, clouds, and virtualization, SpringerLondon,

2011, p (16)

[10] James E Smith, Ravi Nair, The architecture of virtual machines, the IEEE Computer Society, 2005, p (32).

[11] Michael Nelson, Beng-Hong Lim, and Greg Hutchins, Fast transparent migration for virtual machines. In Proceedings of the Annual Conference on USENIX Annual Technical Conference, USENIX Association, 2005, p (1-4)

[12] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen†, Eric Jul†, Christian Limpach, Ian Pratt, Andrew Warfield, Live migration of virtual machines. In Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation Volume 2, USENIX Association, 2005, NSDI'05, p (273).

[13] Md. Abdur Rashid, Data Center Architecture Overview Article, National Academy for Planning and Development (NAPD), Ministry of Planning, Bangladesh, 2019, p(3)

[14] Gary Lee, Cloud Networking: Understanding Cloud-based Data Center Networks, Elsevier Inc. 2014, p (4)

[15] Linz Tom, V. R. Bindu, Task Scheduling Algorithms in Cloud Computing: A Survey, Inventive Computation Technologies, Springer Nature Switzerland AG, 2020, p (360)

[16] Kai Hwang, Geoffrey Fox, Jack Dongarra, Distributed Computing: Clusters, Grids and Clouds, Elsevier, 2012, p (221-222)

[17] Patricia Takako Endo, Glauco Estácio Gonçalves, Daniel Rosendo, Demis Gomes, Guto Leoni Santos, André Luis Cavalcanti Moreira, Judith Kelner, Djamel Sadok, Mozhgan Mahloo,Highly Available Clouds: System Modeling, Evaluations, and Open Challenges, Research Advances in Cloud Computing Sanjay Chaudhary Gaurav Somani Rajkumar Buyya , Springer Nature Singapore, 2017, p (47)

[18] Sung-Soo Kim, Ji-Hwan Byeon, Hong Yu, Hongbo Liu, Biogeography-based optimization for optimal job scheduling in cloud computing, Applied Mathematics and Computation, Elsevier Inc, 2014, p (266-280)

[19] Safwat A. Hamad, Fatma A. Omara, Genetic-Based Task Scheduling Algorithm in Cloud Computing Environment , (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 7, 2016, p (550-556)

[20] Ali Belgacem, Kadda Beghdad-Bey, Hassina Nacer, Task Scheduling in Cloud Computing Environment: A Comprehensive Analysis, Springer Nature Switzerland AG 2019, p (16-17)

[21] Raja Manish Singh, Sanchita Paul, Abhishek Kumar, Task Scheduling in Cloud Computing: Review, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (6), 2014, p (1)

[22] Gurtej Singh, Amritpal Kaur, Bio Inspired Algorithms: An Efficient Approach for Resource Scheduling in Cloud Computing, International Journal of Computer Applications Volume 116 – No. 10, April 2015, p (16)

[23] Seyedali Mirjalili, Evolutionary Algorithms and Neural Networks Theory and Applications, Studies in Computational Intelligence Volume 780, Springer Nature 2019, p15

[24] Micael Couceiro, Pedram Ghamisi, Fractional Order Darwinian Particle Swarm Optimization Applications and Evaluation of an Evolutionary Algorithm, Springer Cham Heidelberg New York Dordrecht London, 2016, p (1-3)

[25] Bastien Chopard, Marco Tomassini An Introduction to Metaheuristics for Optimization, Springer Nature Switzerland AG, 2018, p (97)

[26] Suqin Liu, Jing Wang, Xingsheng Li, Jun Shuo, and Huihui Liu, Design and Implement of a Scheduling Strategy Based on PSO Algorithm, Advances in Swarm Intelligence First International Conference ICSI 2010, Part II, 2010, p (535-538)

[27] Qiang Guoa, Task scheduling based on ant colony, optimization in cloud environment 2017 5th International Conference on Computer-Aided Design, Manufacturing, Modeling and Simulation CDMMS, 2017, p (1-11).

[28] Mokhtar A. Alworafi Atyaf Dhari Asma A. Al-Hashmi A. Basit Darem Suresha , An Improved SJF Scheduling Algorithm in Cloud Computing Environment, International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT), IEEE, 2016, p (208)

[29] Riya Joshi1 , Study and Comparison of VM Scheduling Algorithm in Cloud Computing Using CloudSim Simulator, International Journal for Research in Applied Science & Engineering Technology (IJRASET) Volume , 2018, p (1751).

[30] Mohammad Oqail Ahmad and Rafiqul Zaman Khan, Cloud Computing Modeling and Simulation using CloudSim Environment, International Journal of Recent Technology and Engineering (IJRTE), Volume-8 Issue-2, 2019, p (5439).

[31] Rodrigo N. Calheiros1, Rajiv Ranjan2, Anton Beloglazov1, C´esar A. F. De Rose3 and Rajkumar Buyya1, CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience, Wiley Press, 2011. P (30).

Web site:

[32] https://en.wikipedia.org/wiki/Eclipse_(software).

Certified Training Course :

[33] https://www.udemy.com/course/learn-basics-of-cloudsim/

Abstract

Cloud computing provides a pool of virtualized computing resources and adopts pay-per-use model. Task Scheduler is one of the most important cloud computing problems. And that cannot be scheduled using single criteria, but according to many of the criteria's and rules. These rules and criteria's to be agreed upon by the service provider and the user. In this report, we present a particle swarm optimization (PSO) based scheduling algorithm to minimize the completion execution time Makespan. Experiment is conducted by varying computation of tasks, number of particles and Makespan in fitness function. Then we modify the proposed PSO by adding a method for sorting the tasks by them length to execute them by shortest job first (SJF). The two proposed algorithms are implemented by using the CloudSim simulator. By comparing the two proposed algorithms (PSO and SJF-PSO) the result given by (SJF-PSO) algorithm is the lower Makespan.

Keywords:

Cloud Computing, scheduling algorithm, Particle Swarm Optimization (PSO), Shortest Job First (SJF), Makespan, CloudSim.

Résumé

Le cloud computing fournit un pool de ressources informatiques virtualisées et adopte un modèle de paiement à l'utilisation. L'ordonnancement de tâches est l'un des problèmes de cloud computing les plus importants. Et cela ne peut pas être planifié en utilisant des critères uniques, mais en fonction d'un grand nombre de critères et de règles. Ces règles et critères doivent être convenus par le fournisseur de services et l'utilisateur. Dans ce rapport, nous présentons un algorithme d'ordonnancement basé sur l'optimisation des essaims de particules (PSO) pour minimiser le temps d'exécution. L'expérience est menée en faisant varier le calcul des tâches, le nombre de particules et le durée dans la fonction de fitness. Ensuite, nous modifions le PSO proposé en ajoutant une méthode pour trier les tâches par leur longueur afin d'exécuter la tâches la plus courte en premier (SJF). Les deux algorithmes proposés sont implémentés en utilisant le simulateur CloudSim. En comparant les deux algorithmes proposés (PSO et SJF-PSO), le résultat donné par l'algorithme (SJF-PSO) donne des meilleurs résultas.

Mots clés:

Cloud Computing, algorithme d'ordonnancement, L'optimisation par essaims de particules (PSO), plus court processus en premier (SJF),temps d'exécution, CloudSim.

الخلاصة

توفر الحوسبة السحابية مجموعة من موارد الحوسبة الافتراضية وتعتمد على نموذج الدفع المقابل. يعد برنامج جدولة المهام أحد أهم مشكلات الحوسبة السحابية. ولا يمكن جدولة ذلك باستخدام معيار واحد، ولكن وفقًا لعدد كبير من المعايير والقواعد. يتم الاتفاق على هذه القواعد والمعايير من قبل مزود الخدمة والمستخدم. في هذه المذكرة، تقدم خوارزمية جدولة تعتمد على خوارزمية حركة الجزيئات(PSO) ، لتقليل وقت التنفيذ. يتم إجراء التجربة عن طريق حساب المهام المتفاوتة، وعدد الجسيمات والامتداد في وظيفة اللياقة البدنية. ثم نقوم بتعديل PSO المقترح عن طريق إضافة طريقة ترتيب المهام حسب طولها لتنفيذها بأقصر مهمة أولاً (SJF) يتم تنفيذ الخوارزميتين المقترحتين باستخدام محاكي CloudSim بمقارنة الخوارزميتين المقترحتين (SJF-PSO،PSO) ، وجدنا ان خوارزمية (SJF-PSO) تعطي أدنى وقت في تنفيذ المهام.

الكلمات الدالة:

الحوسبة السحابية ، خوارزمية الجدولة ، تحسين حركة الجسيمات (PSO) ، أقصر مهمة أولاً (SJF) ، وقت التنفيذ ، CloudSim