



الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي والبحث العلمي
جامعة وهران للعلوم والتكنولوجيا محمد بوضياف
كلية الرياضيات و الاعلام الالي

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur Et de la Recherche Scientifique
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF
Faculté des Mathématiques et Informatique

Département : Informatique

Mémoire de fin d'études

**Ordonnancement des workflows
scientifiques sur le cloud avec
optimisation de l'énergie**

Pour l'obtention du diplôme
de **Master**

Domaine : **Mathématiques – Informatique**

Filière : **Informatique**

Spécialité : **Système d'Information et Données**

Présenté le :

Par :

-TOUHAMI
Abdelghani Chabane

-CHERIEF
Houcine Abdelkader

Jury	Nom et Prénom	Grade	Université
Président			
Encadrant	BENDOUKHA Hayat	MCB	USTO-MB
Examineur	BELAID Mohamed Said	MCB	USTO-MB
Examineur	Guerid Hicham	MCB	USTO-MB
Examineur			

2022/2023

Remerciements

bla bla bla

Contents

General introduction	9
1 The Cloud Computing	11
1.1 Introduction	12
1.2 Definition of Cloud Computing	12
1.3 Characteristics of Cloud Computing	12
1.4 Service Models	13
1.4.1 Software as a Service (SaaS)	13
1.4.2 Platform as a Service (PaaS)	14
1.4.3 Infrastructure as a Service (IaaS)	14
1.5 Deployment Models	14
1.5.1 Private cloud	14
1.5.2 Community cloud	15
1.5.3 Public cloud	15
1.5.4 Hybrid cloud	15
1.5.5 Multi-Cloud Computing	15
1.6 Benefits of cloud computing	15
1.7 Cloud Computing's obstacles	17
1.7.1 Cloud Vulnerability	17
1.7.2 Technical Issues	17
1.7.3 Cost Creep	17
1.7.4 Financial Risk in Disaster Scenarios	18
1.7.5 Vendor Lock-In	18
1.8 Conclusion	18

2	Workflow and scientific workflow	19
2.1	Introduction	20
2.2	Workflow definition	20
2.3	Workflow Management System	20
2.3.1	Definition	20
2.3.2	Workflow engine architecture	21
2.4	Scientific workflow	22
2.4.1	Definition	22
2.4.2	Scientific workflow's life cycle[8]	22
2.4.3	Benefits of workflow for the science	23
2.4.4	Business VS scientific workflows	24
2.5	Main Requirements on scientific WfMS	24
2.6	Conclusion	26
3	The problem of scheduling the scientific workflow on a cloud platform with energy optimization	27
3.1	Introduction	28
3.2	Workflow scheduling	28
3.2.1	Definition of Task scheduling of scientific workflows	28
3.2.2	Scientific workflow scheduling	28
3.2.3	Task scheduling types	29
3.2.4	Scheduling objectives	29
3.2.4.1	Cost	29
3.2.4.2	Makespan	30
3.2.4.3	Workload Maximization	30
3.2.4.4	Maximization of VM utilization	30
3.2.4.5	Energy Consumption	30
3.2.4.6	Reliability Awareness	30
3.2.4.7	Security Awareness	30
3.3	?Related works[2days]	30
3.3.1	Energy optimization problem	30
3.3.1.1	Hard optimization	31
3.3.2	Approximate optimization algorithms	31
3.3.2.1	Accelerated Search (AS)	31
3.3.2.2	? Plain GA	32
3.3.2.3	Plain GA, CA+GA	32
3.3.2.4	PBHGA	32
3.3.2.5	NSGA-II, MOCeII, IBEA	32
3.3.2.6	HCFS	33
3.3.2.7	EAMD	33
3.3.2.8	MMF-DVFS	34
3.3.2.9	EASLA, Improved EASLA	34
3.3.2.10	QHA	35
3.3.2.11	EADAGS	35

3.3.2.12	eFLS	35
3.3.2.13	VHEST, EASA	35
3.3.2.14	EDLS	36
3.3.2.15	LESA	36
3.3.2.16	EED, EEND	36
3.3.2.17	RSMECC	36
3.3.2.18	ECS, ECS + idle	37
3.3.2.19	ESPA	37
3.3.2.20	GACSM	37
3.3.2.21	EAD, PEBD	38
3.3.2.22	WPEP	38
3.3.2.23	RMREC	38
3.3.2.24	MW-HBDCS	38
3.3.2.25	MinD + ED	38
3.3.2.26	EnReal Method	39
3.3.2.27	AVVMC	39
3.3.2.28	?FFO-EVMM	39
3.3.2.29	Adaptive GA	39
3.3.2.30	Hybrid Cultural and ACO	39
3.3.2.31	MPSO-FGA	39
3.3.2.32	FOA-SA- LB	40
3.3.2.33	REEWS	40
3.3.2.34	HUA	40
3.3.2.35	MHRA	40
3.3.2.36	EARES-D	41
3.3.2.37	EVIMA	41
3.3.2.38	EAH	41
3.3.3	Comparaison entre les approches d'ordonnement des workflows	43
3.4	Problem modeling	43
3.4.1	Workflow modelisation	43
3.4.2	Cloud datacenter model[10]	43
3.4.3	VM allocation model[10]	44
3.4.3.1	Definition 1.	45
3.4.3.2	Definition 2.	45
3.4.3.3	Definition 3.	46
3.4.4	Multi-objective optimization problem formulation[10]	46
3.4.4.1	Definition 4.	47
3.4.4.2	Definition 5.	47
3.5	?Conclusion	47
4	State of art and our proposed approach	49
4.1	Introduction	50
4.2	Studied algorithm EViMA	50
4.2.1	EViMA[10]	50

4.2.2	Highly critical task selection algorithm (HiCTSA)	51
4.2.3	Low critical task selection algorithm (LoCTSA)	53
4.2.4	VM power regulating algorithm (VM-PRA)	54
4.2.5	Slack time harvesting method (STiHaM)	54
4.3	VM placement & VM consolidation	56
4.3.1	Problem	56
4.3.2	VM Placement	56
4.3.3	VM Consolidation	56
4.4	Proposed approach	58
4.4.1	Objectives	58
4.4.2	Principle and description	59
4.4.3	Steps of our approach	59
4.5	Conclusion	60
5	Implementation, simulation and discussion of results	61
5.1	Introduction	62
5.2	Simulation tool: WorkflowSim	62
5.2.1	Description and main features	62
5.2.2	Architecture	62
5.2.3	Downloading and Installing WorkflowSim with Eclipse without GitHub	63
5.2.4	Strengths	65
5.2.5	Limitations	65
5.3	Workflows	65
5.4	Simulation parameter	66
5.5	Experimentation	67
5.6	Comparison and discussion of the solution	67
5.7	Conclusion	67
	Conclusion Générale	68
	Bibliography	69

List of Figures

1.1	Cloud service models from quisted.net	14
2.1	Workflow engine architecture[17]	21
2.2	Scientific and Business workflow's life cycle [8]	23
3.1	VM allocation model	44
4.1	Description of the Boxplot Method	57
5.1	WorkflowSim's Architecture [12]	63
5.2	Import a project	64
5.3	Set URL to be the extracted folder	64
5.4	Structure of Scientific Workflows used confluence.pegasus.isi.edu	66

List of Tables

3.3.1 Workflow scheduling algorithms, systems and optimization metrics used for them.	42
3.4.1 Symbols used in the formulas	48
5.4.1 VM types and pricing model	67

General introduction

Context Cloud computing is a distributed heterogeneous computing model providing many services through the Internet without the violation of Service level agreement (SLA). The cloud services are provided as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). The cloud computing power is supplied by a collection of data centers (DCs) that are typically installed with massive hosts (physical servers). These hosts are transparently managed by the virtualization services that allow sharing their capacities among virtual instances of servers (VMS) [Enhanced hybrid multi-objective workflow scheduling approach based artificial bee colony in cloud computing].

In addition, the advent of cloud computing has another advantage over the massive spread of scientific work. Scientific workflow refers to a series of computations that enable data analysis in a distributed and systematic way; because this workflow has a large number of works, energy consumption is a major problem [Energy and cost optimization mechanism for workflow scheduling in the cloud]. However the execution of scientific workflows needs a huge amount of data or allocates enormous computing resources and big scalability.

Problematic The energy consumption of cloud data centers is an important challenge. Energy consumption is increasing day by day due to a non-energy aware strategy in resource management. About 0.5% of energy consumption in worldwide is related to cloud data centers, while this is envisaged to fourfold in 2020 [12].

In general, workflow application oriented energy-aware scheduling in clouds involves mapping tasks to resources, arranging tasks' execution order, and assigning appropriate task execution time, such optimizing energy consumption and satisfying complex constraints. It is a NP-complete problem, and has been researched extensively in the literature [Energy-aware intelligent scheduling for deadline-constrained workflows in sustainable cloud computing].

Objectives

Contribution

Outline

CHAPTER 1

The Cloud Computing

1.1 Introduction

Nowadays, cloud computing is an emerging technology due to virtualization and providing low price services on pay-as per-use basis and also it is the most popular computing system paradigm in the academic research and industry and it provides cloud services on-demand anywhere and anytime[12].

In this chapter, we will present the most convivial execution infrastructure which is the cloud computing model, its characteristics, its service and deployment models and finally its advantages and obstacles.

1.2 Definition of Cloud Computing

The modern literature provides several definitions of Cloud Computing. Some definitions of Cloud Computing refer to services provided by datacenter, while others define it as a service provided by the Internet. Still, the most quoted definition of Cloud Computing is the NIST definition.

The National Institute of Standards and Technology (NIST) which is an agency of the United States Department of Commerce whose mission is to promote American innovation and industrial competitiveness, defines cloud computing as follows:

“Cloud computing is a model for enabling ubiquitous, 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”[13].

1.3 Characteristics of Cloud Computing

The National Institute of Standards and Technology’s definition[13] of cloud computing identifies "five essential characteristics":

On-demand self-service Consumers of cloud computing services, such as server time and network storage, as needed automatically nearly instant access to resources. To support this expectation, clouds must allow self-service access so that customers can request, customize, pay and use services without intervention of human operators.

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, tablets, laptops and workstations).

Resource pooling The provider’s computing resources are pooled to serve multiple consumers using a multi-tenant 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 location at a higher level of abstraction (e.g., country, state, or datacenter). Examples of resources include storage, processing, memory and network bandwidth.

Rapid elasticity Cloud computing gives the illusion of unlimited computing resources available on demand in any quantity at any time. Therefore, users expect clouds to rapidly provide resources in any quantity at any time. In particular, it is expected that the additional resources can be (a) provisioned, possibly automatically, when an application load increases and (b) released when load decreases (scale up and down).

Measured service Services must be priced on a short term basis (e.g., by the hour), allowing users to release (and not pay for) resources as soon as they are not needed. For these reasons, clouds must implement features to allow efficient trading of service such as pricing, accounting and billing.

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.

Cloud computing has a lot of other characteristics[2]:

1. Performances are being monitored by IT experts from the service provider end.
2. Resources are abstracted or virtualized.
3. Availability improves with the use of multiple redundant sites
4. Customizable: resources rented from the cloud must be highly customizable.
5. Device and location independence which means no maintenance is required.

1.4 Service Models

The National Institute of Standards and Technology (NIST) has defined 3 services models in their definition[13]:

1.4.1 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.

1.4.2 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.

1.4.3 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).

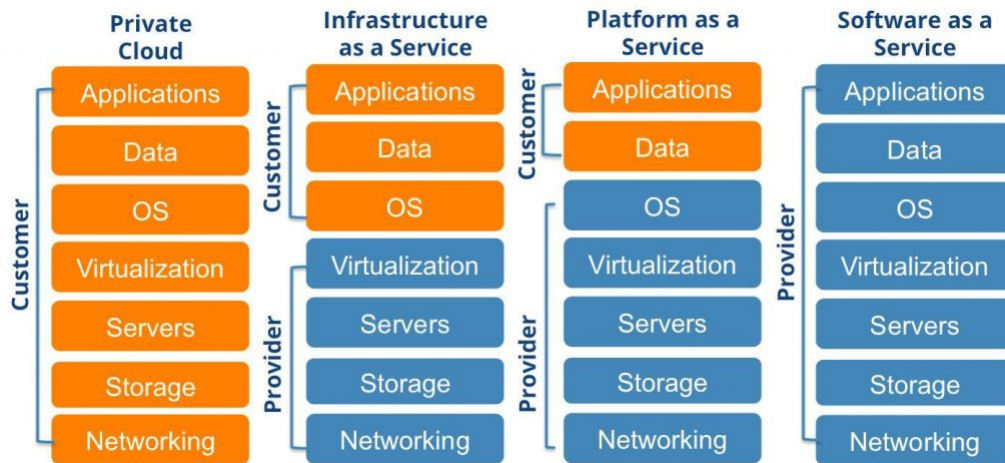


Figure 1.1: Cloud service models from quisted.net

1.5 Deployment Models

There are 5 deployment models of Cloud Computing[13][1]:

1.5.1 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.

1.5.2 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.

1.5.3 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.

1.5.4 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).

1.5.5 Multi-Cloud Computing

Multi-Cloud = Any Combination of Native Public Clouds and Private Clouds

Some companies can't use a single public cloud. Their developers may like Google Cloud, while IT operates in Azure both on-premise and in the cloud. Meanwhile, a startup they just acquired runs their killer app in Amazon Web Services (AWS).

This is the multi-cloud model. It's the ultimate flexibility to choose the cloud that best fits the application or business need. It's no wonder organizations overwhelming go the multi-cloud route, often with the inclusion of a hybrid cloud.

The main advantage of this model compared to the public model is avoiding vendor lock-in which will be defined later.

1.6 Benefits of cloud computing

Cloud Computing is fueled in part by its perceived benefits[11][2]. The ideal Cloud Computing infrastructure would possess the following desirable characteristics:

- **Reliability:** Backup and recovery of data are less expensive and extremely fast for business continuity.
- **Scalability:** We can increase or decrease the requirement of resources according to the business requirements.

- **Cost:** Cloud Computing requires significantly lesser capital expenditures to get up and running, it reduces the huge capital costs of buying hardware and software.
- **Speed:** resources can be accessed in minutes, typically within a few clicks, so no more waiting months or years and spending millions of dollars before anyone gets to log into your new solution.
- **Productivity:** Cloud computing technology allows on-the-fly, point-and-click customization and report generation for business users, so IT doesn't spend significant time on such tasks.
- **SLA-driven:** The system is dynamically managed by service-level agreements that define policies such as how quickly responses to requests need to be delivered.
- **Multi-tenancy:** The system is built in a way that allows several customers to share infrastructure with mutual opacity and without compromising privacy and security.
- **Service-oriented:** The system allows composing applications out of discrete, reusable, loosely-coupled services. Changes to, or failure of, a service will not disrupt other services.
- **Virtualized:** Applications are decoupled from the underlying hardware. An application may rely on Grid Computing and multiple applications could run on a single computer.
- **Data, Data, Data:** The key to many of the above desirable characteristics is the management of data: its distribution, partitioning, security and synchronization using technologies like Amazon's SimpleDB (large-scale relational databases) and in-memory data grids. According to premier Cloud Computing vendor, Salesforce.com, given a properly-implemented Cloud Computing solution, a client should experience some or all of the following benefits:
- **Proven Web-services integration:** Cloud Computing technology is much easier and quicker to integrate with other enterprise applications.
- **Support for deep customizations:** The Cloud Computing infrastructure not only allows deep customization and application configuration, but it also preserves all those customizations even during upgrades and with evolving needs, thus freeing up organizational IT resources.
- **Pre-built, pre-integrated apps:** Hundreds of prebuilt applications and application exchange capabilities are either preintegrated into larger, off-the-shelf applications or available for quick integration to form new applications.
- **Security** can improve due to the centralization of data.

1.7 Cloud Computing's obstacles

There are many disadvantages of Cloud Computing. The document [16] list many disadvantages:

1.7.1 Cloud Vulnerability

The primary challenge posed to data management in the cloud is content security. Due to the ease of access to content over the internet, concerns have been raised due to data vulnerability. In a review of cloud security issues, funded by IBM, 382 organizations disclosed cloud security issues in 2016 across 16 different industries. The security issues faced by cloud environments include; the misuse of cloud content, malicious hacking, failure of vendor security and shared technology vulnerabilities. While improve cloud security is an area currently undergoing considerable research and it was observed that the security of cloud content is vulnerable on several levels. On a "communications level" risk arises from the sharing of sensitive information (e.g. access passwords). On a "security level" the cloud is vulnerable to password cracking, cookie poisoning and CAPTCHA breaking among other threats. Moreover, cloud content is subject to Service Level Agreements (SLAs) such as legal or confidentiality requirements which may compromise data security. Due to the multi-tiered vulnerability of cloud architecture, current innovations aiming to improve cloud security are unlikely to remove all levels of threat.

1.7.2 Technical Issues

Although Cloud Computing offers improved efficiency by means of "on-demand" content access, the cloud can also be prone to technical malfunctions such as outages. Even the most established cloud vendors do encounter unexpected technical problems despite generally having high standards of maintenance. It is estimated that cloud services are unavailable for 7.5 hours per year (99.9% reliability). While, at a glance, this figure may seem impressive, it does not reach the expected reliability quotient of 99.999% for sensitive data management.

1.7.3 Cost Creep

At first glance migrating to a "pay-as-you-go" pricing model as employed by many cloud providers may seem financially attractive. However, it has been well documented that organizations that migrate to a cloud environment are often charged with additional hidden usage charges. Additional usage charges can accumulate to be a deterrent to cloud adoption, particularly for smaller enterprises.

1.7.4 Financial Risk in Disaster Scenarios

Cloud users can incur significant costs in cases of a cloud outage, security breach or vendor lock-in (see below). According to the Penomen Institute, for example, in 2016, the average distributed cost incurred by a datacenter outage was almost \$750, 000. In the same year data breaches, depending on their nature, are estimated to have cost between around 130\$ and 170\$ to resolve per record. The cost in these scenarios may fall on either the cloud utilize or the cloud vendor, depending on the predefined contract between the two parties.

1.7.5 Vendor Lock-In

Vendor lock-in problem in cloud computing is characterized by expensive and time-consuming migration of applications and data to alternative providers. Cloud software vendors lock in customers in several ways:

- by designing a system incompatible with software developed by other vendors
- by using proprietary standards or closed architectures that lack interoperability with other applications
- by licensing the software under exclusive conditions.

1.8 Conclusion

In this chapter, we have presented Cloud Computing and its characteristics, then we have listed the most commonly used services models IaaS, PaaS and SaaS. Also, we have presented several deployment models. Finally, the benefits and obstacles of this computing model were listed.

Cloud computing provides virtualized cloud resources as a service, on demand and a pay-per-use basis. The characteristics of cloud computing such as elasticity and flexibility make this environment a major trend for computation and storage services. These characteristics motivate to execution of scientific applications in the cloud environment[6].

However, processing large scientific applications in the cloud computing environment generates a huge amount of data and hence is very expensive and energy-consuming. Since such complexity in processing these applications can affect the quality of service delivery, scientists have designed a step-by-step sequence (workflow) through which large-scale applications can be executed. A workflow is a sequence of scientific activities that are performed or should be performed to achieve a scientific goal[10]. In the next chapter, we will present workflow and scientific workflows.

CHAPTER 2

Workflow and scientific workflow

2.1 Introduction

Workflows are a systematic way for describing and executing a series of computational and data manipulation steps, or tasks, in a specific application domain.

In this chapter, we present workflow and the Workflow Management System (WfMS) with its architecture. We will also talk about scientific workflows, their life cycle and their types.

2.2 Workflow definition

The WfMC is a consortium formed to define standards for the interoperability of workflow management systems. WfMC define workflow as following:

“Workflow is the sequence of tasks, steps and decisions that need to be followed to complete a specific process. It can be thought of as a set of instructions that describe how a process should be performed, including the order in which tasks are to be completed, who is responsible for performing each task and what should happen next based on the outcome of each task.

Workflows can be used to automate a wide range of business processes, such as invoicing, order fulfillment, human resources and project management. The goal of a workflow is to make sure that tasks are completed in the correct order and by the right people and to ensure that the process is efficient and accurate.”[17].

2.3 Workflow Management System

Every workflow is executed on a workflow management system (WfMS)[8].

2.3.1 Definition

Workflow Management System is defined by WfMC [17] as following:

“A software application that is designed to help organizations automate and manage their business processes. A WfMS allows an organization to define and implement a workflow – a series of tasks, steps and decisions – that need to be followed to complete a specific process. The system can then be used to track the progress of the workflow, manage the flow of information and documents and ensure that tasks are completed in the correct order and by the right people.

A WfMS can be used to automate a wide range of business processes, such as invoicing, order fulfillment, human resources and project management. It can also be used to integrate different systems, such as email, calendar and customer relationship management (CRM) software. WfMS allows you to streamline the process and make it more efficient, reduce errors and delays and give you real-time visibility into the status of your processes, also it’s common to have a built-in reporting and analytics tools to measure the performance of the process”.

2.3.2 Workflow engine architecture

Workflows are compositions of tasks (also referred to as activities) by means of causal or data dependencies that are carried out on a computer. They are executed on a workflow management system (WfMS). A workflow that utilizes Web services (WSs) as implementations of tasks is usually called service composition. Web services are the most prominent implementation of the service oriented architecture (SOA). Web Service technology is an approach to provide and request services in distributed environments independent of programming languages, platforms and operating systems. It is applied in a very wide range of applications where integration of heterogeneous systems is a must[8].

The WfMC has published an architectural Reference Model which provides the general architectural framework for the work of the WfMC. It identifies “interfaces” covering, broadly, five areas of functionality between a workflow management system and its environment[17].

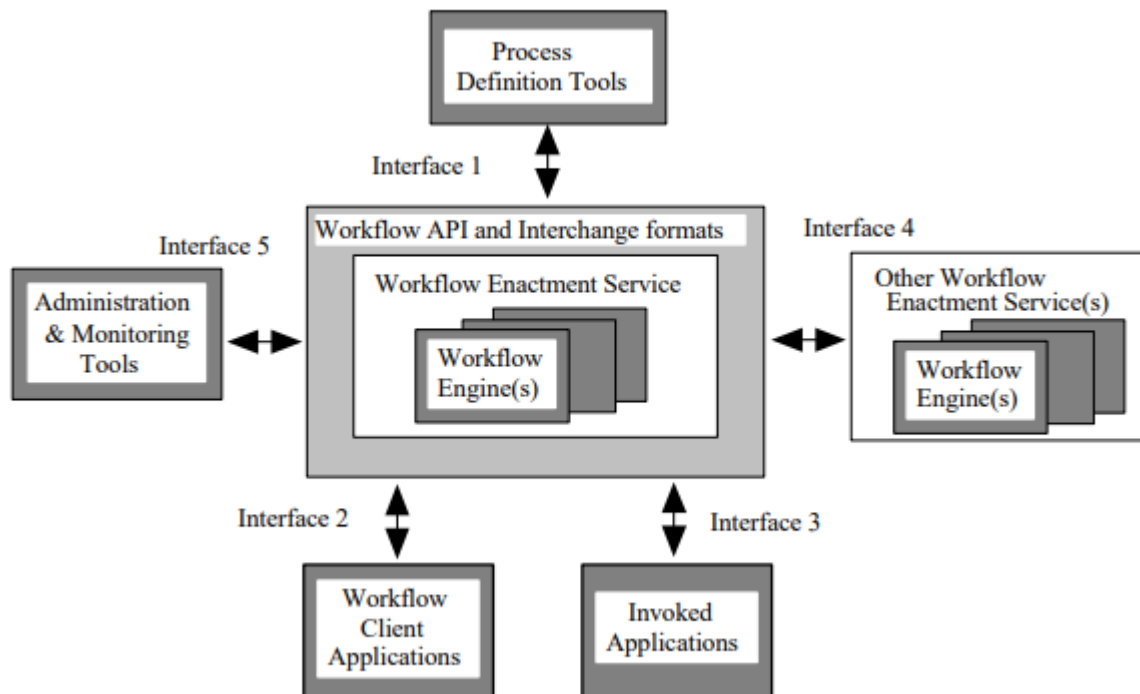


Figure 2.1: Workflow engine architecture[17]

There are 5 interfaces[19] which are:

Interface 1 Web Service (process definition tools) The 1st interface service provides the link between the tools designed for creating and modifying workflow definitions and the workflow enactment service. It controls the system connection. In addition, it inquiries the workflow for process definitions and resource classifications.

Interface 2 (workflow client applications) Through the 2nd interface service, users have access to a work list handler service and enactment services. In addition, the service produces states of cases and work items. The states are also controlled through this service (i.e. start, interruption, hang-up, executing, etc).

Interface 3 (invoked applications) When some tasks of a process need to invoke an application outside of the workflow engine, they directly activate the workflow enactment service through the 3rd interface service and open interactive applications with the current work item of the work list handler.

Interface 4 (other workflow enactment services) The interoperability and communication between workflows become the bottleneck to solve with the recommendations and standards of WfMC. The 4th interface service handles the basic operations to exchange between autonomous workflow systems. Through this interface, workflows transfer cases, resources and work items.

Interface 5 (administration and monitoring tools) The 5th interface manages the administration and the monitoring tools of the workflow system. Through this service, the executing process should be managed, well-supervised and the events should be recorded in log files.

2.4 Scientific workflow

Workflow technology is established in the business domain for several years. This fact suggests the need for detailed investigations in the qualification of conventional workflow technology for the evolving application domain of e-Science.[8].

Scientific workflows are specialized forms of workflows that are designed for data-intensive or knowledge-discovery applications in various scientific disciplines. Scientific workflows enable researchers to formulate, share, reuse and validate complex methods that involve different data sources and computational platforms.

2.4.1 Definition

Scientific workflows are data-intensive applications representing distributed data sources and complex computations in various domains, i.e. astronomy, engineering sciences and bioinformatics. In distributed environments various sensors and experimental processes generate a large volume of data that need collection and processing within specific time constraints[9].

2.4.2 Scientific workflow's life cycle[8]

The life cycle of scientific workflows heavily distinguishes from its business counterpart (see Figure 2.2). We inferred the life cycle from observations about the way scientists

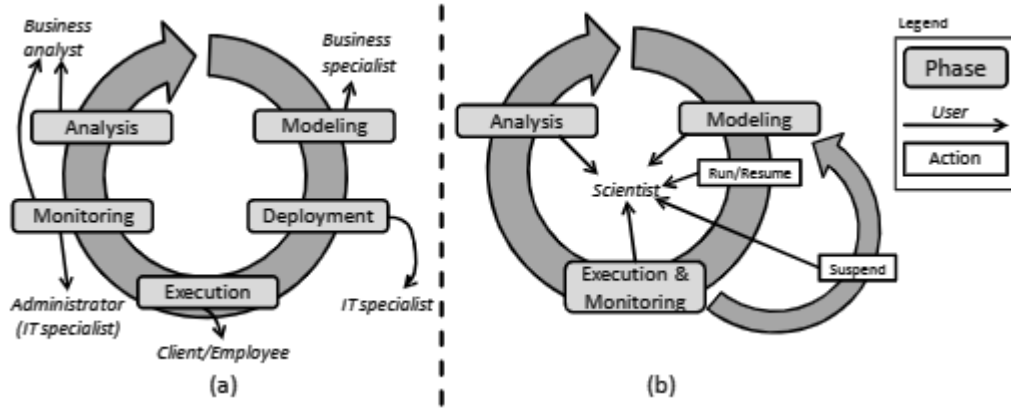


Figure 2.2: Scientific and Business workflow's life cycle [8]

create and conduct experiments and from known properties of scientific simulations and computations.

Typically, there is only one user group, the scientists, playing the roles of a modeler, user, administrator and analyst. The focus of their work is usually on single workflow instances.

To be precise, scientists do not distinguish between workflow models and instances or are not aware of the difference between models and instances, respectively. They set up a simulation and execute it once at a time. Because scientists typically develop their workflow in a trial-and-error manner, modeling and execution phases are not arranged in a strict sequence.

In fact, they can be carried out alternately. An additional cycle, therefore, leads from execution back to the modeling phase with the help of a “suspend” operation on the workflow instance, which remains hidden from the scientists. Technical details are transparent for scientists altogether.

For instance, conventional workflow adaptation is experienced as scientific workflow modeling; the deployment (of parts of a workflow) is part of the “run/resume” operation, which is also hidden for the scientist. Workflow execution starts immediately after modeling.

The traditional execution and monitoring phases are merged into a single phase in the scientific workflow life cycle because from a scientist's point of view monitoring only visualizes a running workflow (instance).

After the execution, a scientist can analyze the computed results and may re-model and re-execute the workflow possibly with different parameters.

2.4.3 Benefits of workflow for the science

Workflows in science provide different benefits[8]:

- they contribute to sharing knowledge by being available as services for collaborating scientists.

- with the help of workflows a community-based analysis of results is supported.
- workflows are able to deal with huge amounts of data, e.g. collected by sensors or calculated by scientific algorithms.
- workflows are capable of running in distributed and highly heterogeneous environments—a common scenario in scientific computations where a great variety of platforms and programming languages is usually employed.
- the automation of steps during workflow design and execution allows scientists to concentrate on solving their main scientific problems.
- workflows can be utilized to conduct scientific simulations in a parallel and automated manner.

2.4.4 Business VS scientific workflows

There are significant differences between business workflows and scientific workflows, which prevent the direct application of the developed adaptation methods[20]:

- Business workflows aim at automating organizational processes that are mainly executed by humans, while scientific workflows aim at validating the hypotheses of a researcher based on available data.
- Business workflows have a predetermined goal, while scientific workflows have experimental and exploratory goals that vary more frequently.
- Business workflows are executed under human control, while scientific workflows are executed fully automatically by a computer.
- Business workflows depend on certain resources including application programs, while scientific workflows depend more strongly on proper parameter settings.

2.5 Main Requirements on scientific WfMS

Workflows in the scientific area make use of a data-centric approach. Typically, huge amounts of data have to be processed, e.g. 5 GB data per day is transmitted by the Hubble telescope and probably need to be processed by a workflow. The modeling language for scientific workflows is primarily data-driven, sometimes with support of a few control structures (e.g. in Taverna, Triana). That means, for accommodating the scientists' needs the main focus of the modeling language should be on data flow while the control flow is considered secondary[8].

There are many requirements listed in [8]:

- **usability:** since the majority of users of WfMS are not computer scientists. A scientist needs the support of easy-to-use tools, automation as far as possible and maximal flexibility in the usage of the WfMS. This means that scientists want to:

1. model their workflows in a convenient.
 2. run their workflows and store their data on resources that are specified by the user himself or automatically chosen by the WfMS.
 3. same support for services used in traditional workflows.
- **flexible**: it means the ability of a system to react to changes in its environment. Approaches to the flexibility of workflows can be divided into two groups:
 1. workflow can be modified according to the changed situation (known as adaptation). Such modifications can address different workflow dimensions (logical, functional and organizational dimension and may be applied to workflow models or single workflow instances.
 2. workflow can be modeled in a way that avoids its modification even in the presence of a changing environment (known as avoid change).
 - **monitoring**: WfMS should support the ability to monitor the status of the workflow execution. For example, the scientist is interested in knowledge about running, finished or faulted activities, allocated resources and the dynamical choice of a service implementation. Inspecting the produced data is also a need.
 - **reproducibility**: is of utmost importance for different reasons. Scientists who are not involved in the execution of the workflow need to be able to retrace the simulation, have to review the findings, or need to use the results. The operating scientist may want to repeat a workflow or parts of it to draw conclusions and to prove statements and assumptions. Additionally, the scientist may share the results with people he collaborates with. While following and reproducing workflow runs and their (intermediate) results scientists use provenance information. Finally, provenance information should be displayed in a way that (non-computer) scientists understand the execution of workflows including the derivation of data.
 - **robustness**: this is an important issue since scientific workflows are long-running. The term robustness denotes the ability to be error-resistant. The needed flexibility mechanisms mentioned above are a way to improve the robustness of a system. But additional approaches are needed to protect the execution progress from being lost in case of unforeseeable failures, to reach a consistent system state even in the presence of failures and to proceed with a simulation/experiment after a failure.
 - **scalability**: scientific workflows should scale with the number of users, number of utilized services, data or calculation resources and involved participants. Today, typical scientific workflows are mostly executed in a central manner on a single machine. A decentralized workflow enactment can help to scale via distributed process execution. It can be achieved, for example, by parallel execution in distributed and heterogeneous execution environments using provisioning techniques.

- **specific requirements:** there are specific requirements for each specific scientific domain, e.g. life science, medical science, chemistry or mechanical engineering. Specific domain-related requirements and the fact that it is hard to cover all scientific domains in one WfMS create the need to extend WfMS and the workflow models and possibly adapt these models and their instances. This includes, for example, domain-specific services, result displays or meta-models for semantic annotations.

2.6 Conclusion

In this chapter, we have presented workflow and scientific workflows, their lifecycle, their benefits and the differences between them and business workflows. We have also presented a workflow management system (WfMS) and its architecture and explained how the foreseen system meets the general requirements of scientists and scientific workflows.

Reducing the energy consumption and execution makespan of scientific workflows while keeping the execution cost of workflow tasks within budget is an effective way to address the workflow scheduling problem[10].

In the next chapter, we will introduce the problem of scheduling the scientific workflow on a cloud platform with energy optimization.

CHAPTER 3

The problem of scheduling the scientific workflow on a cloud platform with energy optimization

3.1 Introduction

Large-scale complex scientific applications/workflow are executed and analyzed in multi-disciplinary areas of research such as astronomy and physics. The workflow contains a large number of mutually dependent tasks which are executed according to their dependency constraint. Due to dependency constraints, the child task can start its execution only when the parent task finishes its execution. A directed acyclic graph (DAG) is used to represent workflows. These workflows often have disparate requirements (such as storage and CPU) and constraints (dependency) that need to be accounted for during their execution[6].

In the cloud environment, effective management of scientific workflows is the main challenge since there are numerous virtual machines and many user tasks that should be scheduled by considering dependencies and various objectives[12].

In this chapter, we will introduce the scientific workflow scheduling problem, firstly, we model it, then we list some related works that optimize the energy consumption and we finish this chapter with a comparative table.

3.2 Workflow scheduling

we need somthing here

3.2.1 Definition of Task scheduling of scientific workflows

Task scheduling of scientific workflow is extremely important. Task scheduling algorithms are based on appropriate policies to deploy scientific workflow to the physical nodes in the cloud computing system. Because various physical nodes are heterogeneous and their running status is dynamically changed over time, at the same time, the system resources required by users and the request types of users are different, so the design of the scheduling algorithms of scientific workflow is complex in the cloud computing system. A bad scheduling policy can lead to a decline in system performance, a long time span of request feedback, a decrease in user satisfaction and a decrease in system throughput. In general, the goal of tasks scheduling of scientific workflow in the cloud environment is to optimize the allocation of scientific workflow, maximize user satisfaction and maximize the economic benefits of operators and minimize operating costs[7].

3.2.2 Scientific workflow scheduling

Workflow scheduling calls for high computation and communication costs, especially scientific workflows that have to do with areas such as astronomy and biology are among the most commonly used cloud applications. In cloud computing, a service provider provides users with various capacities at various costs. Usually, faster resources are more expensive than slower ones. Thus, the scheduler with varied resources for workflow will have different runtimes and costs; therefore, there will be various restrictions by the user. The time restriction guarantees that the workflow is done within the time specified

by the user. The cost constraint guarantees that the cost does not exceed the budget established by the user. The optimal response will create a balance between these two criteria[14].

3.2.3 Task scheduling types

Task scheduling can be categorized, as offline scheduling and online scheduling[14].

- Offline scheduling which means the execution time is calculated and mapping tasks shall be examined at pre-scheduled times. This scheduling approach is static. This strategy helps us prepare a supply plan for tasks, allocate resources to them and then execute the tasks according to this supply plan[14]. Static scheduling generates a scheduling plan (SP) that allocates all executable tasks to compute nodes before execution and the WMS strictly adheres to the scheduling plan throughout the duration of the scientific workflow execution. Since it is prior to execution, static scheduling generates little overhead during execution. It is efficient if the WMS can accurately predict the execution time of each task, when the execution environment varies little during workflow execution and when the WMS has sufficient information about the computing and storage capabilities of the corresponding computers. However, when the execution environment undergoes dynamic changes, it is very difficult to achieve load balance. Static task scheduling algorithms have two types of processor selection methods: **heuristic-based** and **guided random search-based**. The heuristic-based method schedules tasks according to a predefined rule while the random search-based method schedules tasks randomly[4].
- The second approach is online task scheduling, which means dynamic scheduling tasks, where one needs to use and consider available resources rather than operating time. The tasks will be performed without delay, using the available resources[14]. Dynamic scheduling produces scheduling plans that distribute and allocate executable tasks to compute nodes during workflow execution. This type of scheduling is suitable for scientific workflows, in which the workload of tasks is difficult to estimate, or for environments where computer capabilities vary greatly during execution. It should be noted that dynamic scheduling requires time to dynamically generate scheduling plans during execution. Dynamic scheduling algorithms can be based on queueing techniques in a publish/subscribe model with different strategies such as First In First Out (FIFO), adaptive, etc[4].

3.2.4 Scheduling objectives

There are many scheduling objectives[15]:

3.2.4.1 Cost

The minimization of the execution cost or the respect of user-defined budget: for the user, reducing execution cost is of capital importance. Users sometimes have a cap on

the amount of money spent on resources (i.e. budget). Minimizing execution costs and taking into account user-defined budgets in a scheduling strategy is crucial.

3.2.4.2 Makespan

The minimization of the execution time or the respect of user-defined deadline as in the case of the execution cost, execution time and deadline are of capital importance for cloud users. Therefore, scheduling strategies must strive to reduce the makespan while mapping workflow tasks on cloud resources and be aware of the user-defined deadline.

3.2.4.3 Workload Maximization

The maximization of the workload aims at maximizing the amount of work done, that is, the number of workflows executed.

3.2.4.4 Maximization of VM utilization

Idle time slots in provisioned VMs are deemed as a waste of money as they were paid for but not utilized and as a result, algorithms try to avoid them in their schedules. Minimizing idle time slots and maximizing the utilization of resources helps reduce execution costs as well as energy consumption, engraving profit to providers.

3.2.4.5 Energy Consumption

The current industries, organizations and governments are really concerned about the reduction of energy and to reduce the carbon footprint. For cloud computing providers, energy consumption reduction is key to upgrading their return on investment (ROI). Energy-efficient scheduling on the servers is one of the challenging problems in a dynamic environment such as the cloud domain for reducing energy consumption.

3.2.4.6 Reliability Awareness

Scheduling algorithms considering reliability as part of their objectives must put mechanisms in place to ensure that the workflow execution is completed within the users' QoS constraints even if resource or task failures occur. Or simply minimize possible failure during workflow execution.

3.2.4.7 Security Awareness

3.3 ?Related works[2days]

3.3.1 Energy optimization problem

Scientific workflows are the constitution of distinct tasks with complex dependencies. Resource provisioning and the order in which workflow tasks are executed are challenging problems. The inefficient utilization of resources while executing the workflows wastes

a tremendous number of resources. The inefficient utilization of resources increases the number of unused provisioned resources. These unused resources increase energy consumption without performing any useful operation. Resource utilization can be increased by efficient resource provisioning. An energy-efficient scheduling algorithm can be used to manage the resources that are required by the task while executing these scientific workflow tasks. In the literature, numerous workflow scheduling algorithms have been proposed. These scheduling algorithms focus on diminishing makespan and cost with inadequate resources. The selection and designing of a competent and operative workflow scheduling algorithm are also challenging tasks. The energy-aware scheduling algorithm must be selected which can provide a proper resource from the offered resources which are efficient enough to complete the workflow tasks within their deadline constriction and it can decrease the energy consumption[6].

3.3.1.1 Hard optimization

A multi-objective optimization approach is suggested here for scientific workflow task-scheduling problems in cloud computing. More frequently, scientific workflow involves a large number of tasks. It requires more resources to perform all these tasks. Such a large amount of computing power can be supported only by cloud infrastructure. To implement complex science applications, more computing energy is expended, so the use of cloud virtual machines in an energy-saving way is essential. However, even today, it is a difficult challenge to conduct a scientific workflow in an energy-aware cloud platform. The hardness of this problem increases even more with several contradictory goals. Most of the existing research does not consider the essential characteristic of the cloud and significant issues, such as energy variation and throughput besides makespan and cost[14].

3.3.2 Approximate optimization algorithms

Workflow scheduling in the cloud environment is an NP-hard optimization problem with rich literature over the last decade. Previous works have presented many meta-heuristic and heuristic-based methods to solve this problem[14]. We list here some of the related works which optimize energy consumption.

3.3.2.1 Accelerated Search (AS)

Accelerated Search (AS) algorithm based on Dynamic Programming (DP) to obtain a combination of various task schemes which can be completed in a given time with the minimum possible energy by introducing the guaranteed probability and data migration energy.

<https://www.sciencedirect.com/science/article/abs/pii/S0743731516301745?via%3Dihub>

3.3.2.2 ? Plain GA

3.3.2.3 Plain GA, CA+GA

plain genetic (Plain GA), (ii) cellular automata supported by genetic approach (CA + GA) and (iii) heuristic, giving preferences to high-efficiency machines in allocation (EAH), based on a typical FIFO algorithm The used fitness function is based on energy minimization only.

3.3.2.4 PBHGA

The Parallel bi-objective hybrid genetic algorithm (PBHGA) is a parallel evolutionary algorithm for solving multi-objective optimization problems. It combines several optimization techniques including genetic algorithm (GA) and local search.

The algorithm works by using an initial population of random solutions. Individuals in this population are rated based on their goals and their fitness is assessed using a fitness function. Then, individuals are selected for breeding using a rank-based or roulette selection operator.

Reproduction is done using genetic operators such as mutation and crossover. Mutation involves modifying a single gene of the individual while crossbreeding involves creating new individuals by combining the characteristics of two different parents.

The process of selection and reproduction is repeated for several generations until an optimal solution is found. During the whole optimization process, PBHGA maintains a population of solutions to efficiently explore the search space.

The parallel approach of the algorithm helps to speed up the optimization process by running multiple search tasks simultaneously on different processors. It also makes it possible to efficiently explore regions of the search space that would otherwise be difficult to reach.

Finally, the hybrid genetic algorithm uses local search techniques to improve the quality of solutions by applying perturbation operations to existing solutions. These perturbation operations are designed to slightly perturb existing solutions to bring them closer to an optimal solution.

https://d1wqtxts1xzle7.cloudfront.net/30720910/meta20100_submission_167-libre.pdf?13918334&content-disposition=inline%3B+filename%3DA_Parallel_Bi_objective_Hybrid_Genetic_A.pdf&Expires=53B8bU-gDc9JpCPoP0vTk-q5bnriImBXuoaeVmakjOKmkzlTFQMaHtsQQfstc0tckuqSSdJnflS~bNJcUg5HnJ3kFM8a1MViJwGUSVrTwUAtH5zLaZn17D9nE~BCB92y5IEAQY40IXQdffofKusk-c6TgUitAlVd6rGIIRZemuBI4536iEc0s70ys5IvfR-vqJGKhu4T84ZHLSoIW4rml2WBbZGdqH7ZZVQbtAD3ppGZr-pagXt-A__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

3.3.2.5 NSGA-II, MOCell, IBEA

NSGA-II, MOCell and IBEA are popular multi-objective evolutionary algorithms for solving multi-objective optimization problems.

NSGA-II (Non-dominated Sorting Genetic Algorithm II) uses a non-dominated sorting approach to assess the quality of population solutions. Solutions are ranked according to their dominance over other solutions and non-dominated solutions are grouped into Pareto fronts. The Pareto fronts are then ranked in order of dominance to create a new population of parent solutions. This parent population is used to generate a new population of child solutions by crossover and mutation.

MOCeII (Multi-Objective Cellular Genetic Algorithm) uses a grid approach to divide the search space into cells and each cell is treated as a separate optimization problem. The solutions are then evaluated locally and the best individuals are selected to reproduce within the cell. Then, the solutions are exchanged between cells to avoid local convergence.

IBEA (Indicator-Based Evolutionary Algorithm) uses quality indicators to assess the quality of solutions. It uses an indicator function that measures the distance of each solution to an approximation of the true Pareto front. The most distant solutions are selected for reproduction in order to favor the diversity of the population.

These algorithms all use an evolutionary approach to optimize multiple goals at the same time. Candidate solutions are evaluated against several optimization criteria and the best solutions are selected for replication. The algorithms then use crossover and mutation operators to generate a new population of candidate solutions. The process is repeated until solutions of sufficient quality are found.

One of the advantages of these algorithms is that they allow to find several optimal solutions which are equivalent in terms of quality for different criteria. These solutions can then be used to explore the search space and help make informed decisions.

<https://ieeexplore.ieee.org/abstract/document/6606595>

3.3.2.6 HCFS

The Heuristics with Continuous Frequency Scaling (HCFS) algorithm is an optimization technique that solves scheduling problems using a heuristic approach. It is particularly suitable for solving task scheduling problems that have complex time dependencies and resource constraints.

<https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5396>

3.3.2.7 EAMD

new energy-aware scheduling algorithm called Energy Aware Scheduling by Minimizing Duplication(EAMD) which considers the energy consumption as well as the makespan of applications. It adopts a subtle energy-aware method to determine and delete the abundant task copies in the schedules generated by duplication-based algorithms, which is easier to operate than DVFS and produces no extra time and energy consumption. This algorithm can reduce large amount of energy consumption while having the same makespan compared with duplication-based algorithms without energy awareness.

<https://ieeexplore.ieee.org/document/6319162>

3.3.2.8 MMF-DVFS

The MMF-DVFS (Multi-Objective Multi-Frequency Dynamic Voltage and Frequency Scaling) algorithm is an optimization algorithm for power management in embedded systems. It aims to simultaneously minimize power consumption and execution time of multiple tasks with quality of service (QoS) constraints such as timeliness and reliability.

The MMF-DVFS algorithm uses an approach based on multi-objective optimization which seeks to find a compromise between the objectives of minimizing energy consumption and execution time. The algorithm uses a technique called "Dynamic Voltage and Frequency Scaling" (DVFS) which allows the frequency and voltage of the processor to be dynamically adjusted according to performance and power consumption requirements.

The MMF-DVFS algorithm is also able to take into account QoS constraints using a decomposition approach into several sub-problems. The algorithm decomposes the global problem into several sub-problems for which it individually optimizes the objectives while keeping the other objectives at an acceptable level.

The MMF-DVFS algorithm uses local and global search methods to find the best possible solution. Local search methods are used to improve local solutions while global search methods are used to explore new regions of the search space.

<https://ieeexplore.ieee.org/abstract/document/5493460>

3.3.2.9 EASLA, Improved EASLA

EASLA (Efficient Antlion Optimizer with a Small Local Attraction) is a metaheuristic optimization algorithm inspired by the hunting behavior of antlions. The algorithm is used for solving various optimization problems, including function optimization and feature selection.

The Improved EASLA algorithm is an enhanced version of EASLA, which incorporates some modifications to improve its performance. The key modifications include the incorporation of a mutation operator to explore the search space and the use of a dynamic population size to adapt to the search problem.

The basic idea behind EASLA is to simulate the hunting behavior of antlions, which use a combination of ambush and trap techniques to capture their prey. In EASLA, the antlion represents the best solution found so far and the ants represent potential solutions in the search space.

The algorithm starts by randomly generating a population of ants and placing them in the search space. The ants move toward the antlion based on a probability function that takes into account both the distance to the antlion and the local attraction of the area. The antlion is then updated to the best solution found among the ants.

The Improved EASLA algorithm extends this basic framework by introducing a mutation operator that randomly modifies the position of some ants, which can help to explore new regions of the search space. The population size is also dynamically adjusted during the search to balance exploration and exploitation.

Overall, EASLA and its improved variant aim to balance exploration and exploitation in the search space, using a combination of local and global search strategies to find good solutions to optimization problems.

Seyedali Mirjalili and Seyed Mohammad Mirjalili, "Efficient ant optimizer with small local attraction", *Advances in Engineering Software*, vol. 102, p. 24-45, 2016.
Seyedali Mirjalili and Seyed Mohammad Mirjalili, "Improved Ant Optimizer: A New Metaheuristic Algorithm for Solving Optimization Problems", *Applied Soft Computing*, vol. 41, p. 407-423, 2016.

3.3.2.10 QHA

The Quantum-Inspired Harmony Search Algorithm (QHA) is a quantum-inspired optimization method for solving optimization problems. The algorithm starts by initializing a set of candidate solutions for the optimization problem. Each candidate solution is represented by a vector of parameter values.

<https://ieeexplore.ieee.org/document/7173041>

3.3.2.11 EADAGS

Scheduling (EADAGS) on heterogeneous processors that can run on discrete operating voltages Such processors can scale down their voltages and slow down to reduce energy whenever they idle due to task dependencies. EADAGS combines dynamic voltage scaling (DVS) with Decisive Path Scheduling (DPS) to achieve the twin objectives

<https://link.springer.com/article/10.1007/s10586-009-0119-6>

3.3.2.12 eFLS

The eFLS (enhanced fuzzy logic system) algorithm is a modeling method based on fuzzy logic for decision making in uncertain environments. The goal of the eFLS algorithm is to create a robust decision system that can adapt to changing situations using fuzzy decision rules.

Roeder, J.; Rouxel, B.; Altmeyer, S.; Grellck, C. Energy-aware scheduling of multi-version tasks on heterogeneous real-time systems. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, Gwangju, Republic of Korea, 22–26 March 2021; pp. 501–510.

3.3.2.13 VHEST, EASA

VHEST stands for Variable Hessian Estimation Search Technique. It is a type of derivative-based optimization algorithm that uses the Hessian matrix of the objective function to find the optimal solution. The algorithm uses an estimation of the Hessian matrix that is updated during the search process. VHEST has been used for various optimization problems, including parameter estimation and design optimization EASA stands for Efficient and Accurate Simulated Annealing. It is a metaheuristic optimization algorithm that is inspired by the annealing process in metallurgy. EASA starts

with a high-temperature search process, during which the algorithm can accept worse solutions to escape from local optima. As the temperature is lowered, the algorithm becomes more selective and only accepts better solutions. EASA has been applied to various optimization problems, including parameter estimation, design optimization and clustering.

Both VHEST and EASA are known for their efficiency and accuracy in solving optimization problems and they have been applied to various real-world problems in engineering, science and other fields.

3.3.2.14 EDLS

The Exponential Differencing Least Squares (EDLS) algorithm is a linear regression estimation algorithm that is used to model data that has exponential trends or time-varying growth rates. It is widely used in finance to model interest rates, exchange rates and asset prices.

The EDLS algorithm works by minimizing the sum of the squares of the deviations between the observed values and the values predicted by the model. Unlike classical linear regression methods which assume that errors have a normal distribution, the EDLS algorithm assumes that errors have an exponential distribution. This approach makes it possible to take into account exponential trends in the data and to better model variations in growth rates.

article intitulé "A Two-Step Estimation Procedure for Dynamic Panels with Non-Stationary Common Factors", publié dans le Journal of Econometrics en 2005.

3.3.2.15 LESA

GACSM: (GACSM), to address task scheduling on heterogeneous multiprocessor systems using Dynamic Voltage and Frequency Scaling (DVFS).

<https://ieeexplore.ieee.org/document/8974273>

3.3.2.16 EED, EEND

algorithms, namely, Energy-Efficiency with Duplication (EED) and Energy-Efficiency with Non Duplication (EEND) Both algorithms, in contrast to their counterparts in the literature, strive to make a balance across the energy consumption, the schedule length and the number of processors used. Synthetic benchmarks and real-world applications are used to evaluate the performance of our algorithms.

<https://ieeexplore.ieee.org/document/7300601>

3.3.2.17 RSMECC

The RSMECC (Reed-Solomon-MDPC Code-based Encryption and Compression) algorithm is a data encryption and compression algorithm that uses Reed-Solomon codes and MDPC (Moderate Density Parity-Check) codes. The algorithm begins by generating a

public key and a private key for data encryption. The public key is shared with the sender of the data while the private key is kept by the receiver.

<https://link.springer.com/article/10.1007/s00521-019-04415-2>

3.3.2.18 ECS, ECS + idle

ECS is a task scheduling algorithm that considers both the processing time and energy consumption of tasks when scheduling them on processors. The algorithm aims to find an optimal schedule that minimizes the energy consumption of the system while meeting the performance requirements. ECS uses a list scheduling approach to find a feasible schedule and then applies a genetic algorithm to optimize the schedule, ECS+Idle is an extension of ECS that exploits the idle time of processors to further reduce energy consumption. The algorithm schedules tasks in a way that allows processors to enter idle mode as much as possible without violating the performance requirements. ECS+Idle uses a greedy approach to find an initial schedule and then applies a genetic algorithm to optimize the schedule by exploiting the idle time of processors. Both ECS and ECS+Idle have been shown to be effective in reducing energy consumption in computing systems while maintaining performance requirements. These algorithms have been applied to various scheduling problems, including task scheduling on heterogeneous systems, scheduling in cloud computing environments and scheduling in real-time systems.

<https://www.hindawi.com/journals/sp/2016/5612039/>

3.3.2.19 ESPA

ESPA (Enhanced Scheduling Priority Algorithm) is a scheduling algorithm used in real-time systems that aims to improve the scheduling performance compared to traditional priority-based scheduling algorithms such as Earliest Deadline First (EDF) or Rate Monotonic (RM) scheduling. The ESPA algorithm works by assigning a scheduling priority to each task based on its execution history and the current system state. The algorithm uses a set of rules to determine the priority of a task Miyake, M., & Shimizu, T. (1996). A new task scheduling method for real-time systems: enhanced scheduling priority algorithm (ESPA). In Proceedings of the 3rd International Workshop on Real-Time Ada Issues, Volume II: Ada in Real-Time, Real-Time Systems (pp. 169-176).

3.3.2.20 GACSM

GACSM stands for "Genetic Algorithm-based Clustering for Service Composition Management". It is an algorithm that aims to optimize the service composition process in a service-oriented architecture by clustering together services that are frequently used together.

<https://ieeexplore.ieee.org/abstract/document/8974273>

3.3.2.21 EAD, PEBD

algorithms-Energy-Aware Duplication (EAD) scheduling and Performance-Energy Balanced Duplication (PEBD) scheduling. Existing duplication-based scheduling algorithms replicate all possible tasks to shorten schedule length without reducing energy consumption caused by duplication. . Our algorithms, in contrast, strive to balance schedule lengths and energy savings by judiciously replicating predecessors of a task if the duplication can aid in performance without degrading energy efficiency. To illustrate the effectiveness of EAD and PEBD.

<https://ieeexplore.ieee.org/document/5611491>

3.3.2.22 WPEP

The Workflow performance evaluation platforms (WPEP) algorithm is not an optimization algorithm, but rather a method for evaluating the performance of workflows, i.e. automated work processes in which tasks are executed sequentially or in parallel. The first step is to model the workflow, identifying the tasks, transitions and resources needed for each task.

<https://ieeexplore.ieee.org/abstract/document/9393895>

3.3.2.23 RMREC

The "Reducing energy consumption using remapping of critical tasks" algorithm is an optimization method to minimize the power consumption of computing applications by remapping critical tasks to the most energy-efficient cores. Critical tasks are identified using performance profiling techniques to determine which tasks consume the most processing time.

<https://ieeexplore.ieee.org/abstract/document/9253551>

3.3.2.24 MW-HBDCS

is a scheduling method that makes it possible to efficiently plan the execution of several workflows on a set of heterogeneous resources while respecting budget and deadline constraints. The problem is modeled using a formal representation of workflows and resources, as well as budget and time constraints.

<https://link.springer.com/article/10.1007/s00500-018-3229-3>

3.3.2.25 MinD + ED

The "Minimum Dependencies Energy-efficient DAG (MinD+ED)" algorithm is a scheduling method for workflows represented in the form of a DAG (Directed Acyclic Graph) which aims to minimize energy consumption while guaranteeing the completion of the workflow on time. The workflow is represented as a DAG, where nodes represent tasks and arcs represent dependencies between tasks.

<https://ieeexplore.ieee.org/abstract/document/8109158>

3.3.2.26 EnReal Method

The EnReal Method (Energy and Reliability aware Method) algorithm is a scheduling method for scientific workflows that aims to minimize energy consumption while ensuring high reliability of results. The workflow is represented as a DAG (Directed Acyclic Graph), where the nodes represent the tasks and the arcs represent the dependencies between the tasks.

<https://ieeexplore.ieee.org/abstract/document/7276993>

3.3.2.27 AVVMC

the AVVMC VM consolidation scheme that focuses on balanced resource utilization of servers across different computing resources (CPU, memory and network I/O) with the goal of minimizing power consumption and resource wastage, adaptation and integration of the Ant Colony Optimization (ACO) metaheuristic with balanced usage of computing resources based on vector algebra.

<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=63826b7681bb1ff8195612ef75>

3.3.2.28 ?FFO-EVMM

3.3.2.29 Adaptive GA

The Adaptive GA (Genetic Algorithm) algorithm is an optimization method inspired by Darwin's theory of evolution The Adaptive GA algorithm begins by initializing a population of candidate solutions, which are sets of values representing the parameters to be optimized. These candidate solutions are often randomly generated , For each candidate solution in the population, the algorithm evaluates the quality of the solution using an objective function, also called a fitness function. This function assigns a numerical value to each solution based on the quality of that solution.

<https://www.tandfonline.com/doi/abs/10.1080/00221680209499862>

3.3.2.30 Hybrid Cultural and ACO

The Hybrid Cultural and ACO algorithm combines two powerful optimization techniques: Cultural Algorithms (CA) and Ant Colony Optimization (ACO) to solve complex optimization problems. The combination of CA and ACO allows the algorithm to explore the solution space efficiently and effectively. The ACO approach is used to construct high-quality solutions, while the CA approach is used to improve the diversity of the population and prevent premature convergence.

<https://www.igi-global.com/article/an-energy-aware-task-scheduling-in-the-cloud-computing-using-a-hybrid-cultural-and-ant-colony-optimization-algorithm/188661>

3.3.2.31 MPSO-FGA

The MPSO-FGA (Multi-Parent Particle Swarm Optimization with Fitness Gradient Adaptation) algorithm is a hybrid optimization method combining Particle Swarm Opti-

mization (PSO) and Evolutionary Optimization (EA). The MPSO-FGA algorithm begins by initializing a population of candidate solutions, which are sets of values representing the parameters to be optimized. These candidate solutions are often randomly generated. For each candidate solution in the population, the algorithm evaluates the quality of the solution using an objective function, also called a fitness function. This function assigns a numeric value to each solution based on the quality of that solution

<https://ieeexplore.ieee.org/abstract/document/8015589>

3.3.2.32 FOA-SA- LB

The FOA-SA-LB (Fruit Fly Optimization Algorithm with Simulated Annealing and Levy Flight Backtracking) algorithm is a hybrid optimization method that combines the Fruit Fly Swarm Optimization (FOA) algorithm, the simulated annealing algorithm (SA) and Lévy's jump algorithm with backtracking (LB). The FOA-SA-LB algorithm begins by initializing a population of candidate solutions, which are sets of values representing the parameters to be optimized. These candidate solutions are often randomly generated.

<https://www.sciencedirect.com/science/article/pii/S2352914817300187>

3.3.2.33 REEWS

The Real-coded Estimation of Distribution Algorithm with Re-Evaluation and Windowing Strategy (REEWS) algorithm is an optimization method that uses a distribution-based approach to find the optimal solution to a given optimization problem.

University of Technology of Belfort-Montbéliard in France

3.3.2.34 HUA

The HUA (Hybridization of Harmony Search Algorithm and Univariate Marginal Distribution Algorithm) is an evolutionary hybrid algorithm that combines the advantages of two popular algorithms, namely the Harmony Search Algorithm and the Distribution Algorithm. Univariate Marginal Distribution Algorithm The algorithm starts by generating a population of candidate solutions randomly.

The HUA algorithm (Hybridization of Harmony Search Algorithm and Univariate Marginal Distribution Algorithm) was proposed by researchers S. B. Park and S. M. Lee in their article entitled "A hybridization of harmony search algorithm and univariate marginal distribution algorithm for optimization problems" published in the scientific journal "Expert Systems with Applications" in 2010.

3.3.2.35 MHRA

The MHRA algorithm (Multi-Objective Harmony Search Algorithm with Reference-Point-Based Nondominated Sorting Approach) is an evolutionary algorithm used to solve multi-objective optimization problems. The algorithm starts by generating a population of candidate solutions randomly. Each candidate solution is evaluated using an objective function that measures its quality. This function assigns a vector of values

<https://ieeexplore.ieee.org/abstract/document/4289081>

3.3.2.36 EARES-D

The EARES-D (Enhanced Adaptive Range Extended Search with Diversity) algorithm is a global optimization algorithm that has been proposed to solve complex optimization problems. The algorithm starts by generating a population of candidate solutions randomly in a given search space.

<https://www.sciencedirect.com/science/article/pii/S1084804522000595>

3.3.2.37 EViMA

L'algorithme EViMA (Elastic Virtual Machine Allocation) est un algorithme de gestion de la charge de travail pour les applications basées sur le Cloud Computing. Il est utilisé pour allouer des machines virtuelles (VM) aux charges de travail afin de minimiser le coût d'exploitation et d'optimiser les performances. Le fonctionnement de l'algorithme EViMA est basé sur un processus de prédiction de la charge de travail future. Il utilise des techniques de régression pour prédire la charge de travail future en fonction des données historiques de charge de travail et des facteurs externes tels que les tendances du marché, les événements saisonniers, etc. Une fois que la charge de travail future a été prédite, l'algorithme EViMA alloue les ressources nécessaires pour répondre aux besoins de la charge de travail. Il utilise une stratégie d'allocation élastique, qui permet de modifier la taille de la VM allouée à une charge de travail en fonction de ses besoins actuels. L'algorithme EViMA prend également en compte les coûts de location des VM et les délais de provisionnement associés. Il vise à minimiser le coût total d'exploitation tout en garantissant que les exigences de performance de la charge de travail sont satisfaites. En résumé, l'algorithme EViMA utilise des techniques de prédiction de la charge de travail et une stratégie d'allocation élastique pour optimiser l'allocation des ressources de VM et minimiser le coût total d'exploitation pour les charges de travail basées sur le Cloud. And so on It is composed by Par quatre sous algorithme (HiCTSA) And LoCTSA and VM-PRA and STiHaM

3.3.2.38 EAH

The EAH (Exploratory Adaptation Hyper-heuristics) algorithm is a hyper-heuristic optimization method for solving combinatorial optimization problems. The goal of the EAH algorithm is to solve combinatorial optimization problems using a heuristic approach that can adapt its behavior depending on the characteristics of the problem.

<https://ieeexplore.ieee.org/document/6778110>

Algorithm type	Year	Algorithm	Metrics, QoS	
Dynamic programming		Accelerated Search	EC (O), Probability of execution (C), ExecT (C)	
Evolutionary algorithms		Plain GA	EC (O)	
Evolutionary algorithms		Plain GA, CA+GA	EC (O)	
Evolutionary algorithms		PBHG A	Pareto front (O)	
Evolutionary algorithms		NSGA-II, MOCe ll, IBEA	Pareto front (O)	
		HCFS	EC (O), ExecT (O, C), Reliability (O, C)	
		EAMD	EC (O), ExecT (O)	
		MMF-DVFS	EC (O), ExecT (C)	
		EASLA, Improved EASLA	EC (O), ExecT (C)	
		QHA	EC (O, C), ExecT (O, C)	
		EADAGS	EC (O), ExecT (O)	
		eFLS	EC (O), ExecT (C)	
		VHEST, EASA	Utilization (O), ExecT (O)	
		EDLS	EC (O), ExecT (O)	
		LESA	ExecT (O), EC (C)	
		EED, EEND	EC (O), ExecT (C)	
		RSMECC	AST (O), AFT (O), ExecT (O), EC (C)	
		ECS, ECS + idle	EC (O), ExecT (O)	
		ESPA	EC (O), ExecT (O)	
		GACSM	EC (O), ExecT (C)	
		EAD, PEBD	EC (O), ExecT (O)	
	2021	WPEP	Resource scheduling, fault-tolerant, energy efficient approaches	
	2020	RMREC	Lower energy consumption without compromising budget constraint	
	2018	MW-HBDCS	reliability and energy	
	2016	MinD + ED		
	2015	EnReal Method	Resource Utilization and energy consumption	
	2014	AVVMC	Power consumption and resource wastage	
	2016	FFO-EVMM	Energy consumption	
	2016	Adaptive GA	Energy consumption and Time	
	2017	Hybrid Cultural and ACO	Makespan and energy consumption	
	2017	MPSO-FGA	Energy consumption and makespan	
	2017	FOA-SA- LB	Makespan and energy consumption	
	2019	REEWS	Reliability and energy consumption	
	2017	HUA	Energy consumption	
	2018	MHRA	Energy consumption and makespan	
	2014	EARES-D	Energy, completing time and resource utilization	
	2022	EViMA	Energy consumption, Makespan and Cost	

Table 3.3.1: Workflow scheduling algorithms, systems and optimization metrics used for them.

3.3.3 Comparaison entre les approches d'ordonnement des workflows

3.4 Problem modeling

3.4.1 Workflow modelisation

In the workflow scheduling problem, applications are defined by the directed acyclic graphs (DAGs). A workflow includes a set of interdependent tasks that are bounded together through data or functional dependencies[12].

We consider workflow W as a graph $G = (T, D)$, where $T = \{T_0, T_1, \dots, T_n\}$ indicates n tasks and shows the data flow dependencies among them. The data-flow dependency (T_i, T_j) means that task T_i is an immediate predecessor of task T_j and task T_j is an immediate successor of task T_i .

- In some cases, there are several predecessors and successors and so $Pr(T_i)$ and $Su(T_i)$ are defined to indicate the set of immediate predecessors and successors for task T_i [12].

$$Pr(T_i) = \{T_j | (T_j, T_i) \in D\} \quad (3.4.1)$$

$$Su(T_i) = \{T_j | (T_i, T_j) \in D\} \quad (3.4.2)$$

T_{entry} and T_{exit} are tasks without any predecessors and a task without any successors, respectively.

- The degree of dependency of workflow task on wti can be calculated using this equation.

$$ddi = \left(\sum_{T_p \in Pr(T_i)} e(T_p, T_i) + \sum_{T_c \in Su(T_i)} e(T_i, T_c) \right) \quad (3.4.3)$$

where $\sum_{T_p \in Pr(T_i)} e(T_p, T_i)$ is the summation of all the inward edges of workflow task T_i .

and $\sum_{T_c \in Su(T_i)} e(T_i, T_c)$ is the summation of all the outward edges of workflow task T_i .

$T_{\{entry\}}$ and $T_{\{exit\}}$ is a task without any predecessors and a task without any successors, respectively.

3.4.2 Cloud datacenter model[10]

The cloud datacenter is a network of computing resources that enables application and data sharing.

We assume that a data center has a set of M heterogeneous physical machines (PM) that are used to configure a set of k VMs for workflow task scheduling.

Let $\mathbf{p} = (p_1, p_2, p_3 \dots p_n)$, denotes the different types of PMs available in a cloud data center. These PMs have different Processing capacities such as CPUs, memory,

bandwidth, network I/O and the size of the storage which are measured in million instructions per second (MIPS).

VMs are deployed on PMs and each PM can host k VMs. Let k be a set of VM types ($vm_1, vm_2, vm_3, \dots, vm_k$) provided to users at an hourly-based pricing model. We assume that the VMs are infinite and each VM type k is capable of providing an infinite number of VMs for workflow tasks execution.

3.4.3 VM allocation model[10]

Reducing energy consumption in a cloud data center has recently become a major concern for many researchers. This is because the cloud environment has become a trusted platform that individuals, organizations and the government rely on to conduct business. So, if the energy consumption in the cloud data center is not reduced, it will have a direct or indirect impact on all the stakeholders.

A VM allocation model has been proposed (see Fig.3.1), which aims at managing cloud resources to reduce energy consumption in the cloud data center. In workflow scheduling, a scheduler is developed to assign the right tasks to the right VMs to improve the scheduling objectives. However, the biggest challenge is how to manage the cloud resources effectively to achieve the set goals. This is because cloud users turn in a large number of requests that require resources to schedule. This becomes even more obvious if these requests (tasks) are to be scheduled within certain deadlines. This challenge can be well managed if the scheduler is implemented with a well-defined resource management plan.

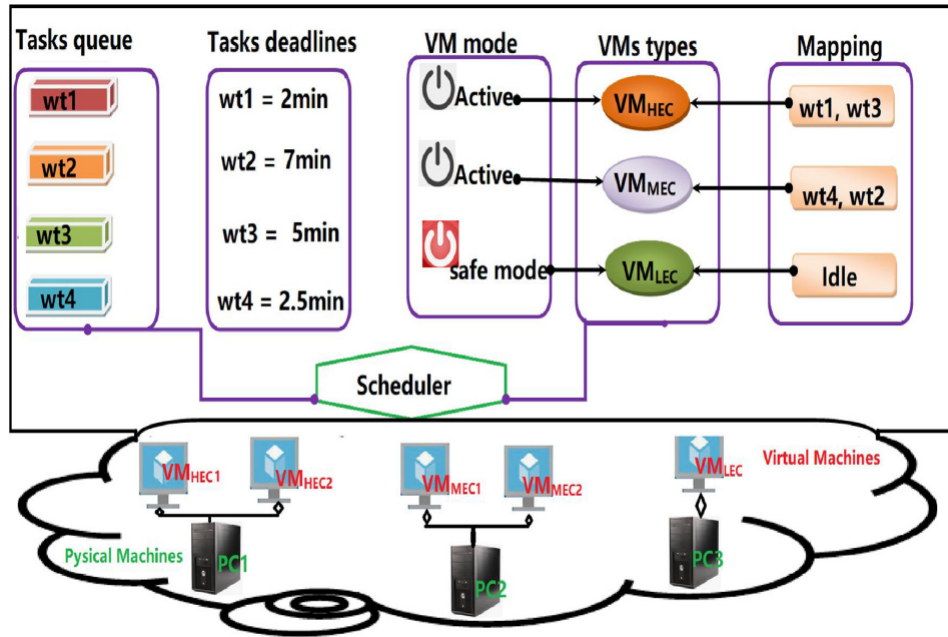


Figure 3.1: VM allocation model

In the model, we assume that there are only three types of VMs (VM_{HEC} , VM_{MEC}

and VM_{LEC}) in the cloud data center, which are specifically defined herein:

3.4.3.1 Definition 1.

The VMs are deployed based on their energy consumption level. There are three (3) physical machines (PMs) hosting five VMs (three different types). Each VM type \mathbf{k} has an infinite number of instances to handle multiple workflow tasks. The VMs are defined by specifications as follows:

$$VM\ Type = \begin{cases} VM_{HEC} & (HEC_1), (HEC_2), (HEC_3).....HEC_k \\ VM_{MEC} & (MEC_1), (MEC_2), (MEC_3).....MEC_k \\ VM_{LEC} & (LEC_1), (LEC_2), (LEC_3).....LEC_k \end{cases}$$

- Where VM_{HEC} is a list of virtual machines with high processing speed (VMs with high energy consumption).
- Where VM_{MEC} represents a set of VMs with medium processing speed (VMs with medium energy consumption).
- Where VM_{LEC} is a group of virtual machines with lower processing speed (VMs with lower energy consumption).

3.4.3.2 Definition 2.

Each workflow contains several sets of workflow tasks representing ($Wt1, Wt2, Wt3, Wt4, \dots, Wtn$); \mathbf{Wt} is a set of ' \mathbf{n} ' workflow tasks in a scientific workflow application. We assume there are three compositions of workflow tasks which are defined herein:

$$Tasks\ grouping = \begin{cases} HCT & (HCT1), (HCT2), (HCT3).....HCTn \\ MCT & (MCT1), (MCT2), (MCT3).....MCTn \\ LCT & (LCT1), (LCT2), (LCT3).....LCTn \end{cases}$$

- Where HCT is a set of tasks in the tasks ready queue that is defined as highly critical tasks that are at risk of missing their deadlines if they are not migrated to a high-speed procession VM immediately.
- Where MCT represents a group of queued workflow tasks that are classified as medium critical. This group of tasks is less critical compared to HCT, but more critical than LCT. These tasks can be migrated to medium-speed VMs for execution.
- Where LCT is a group of tasks in the ready queue defined as less critical tasks. This category of tasks does not need to be processed on high-speed VMs for scheduling. The LCTs can be allocated on lower power VMs (VMs with lower processing speed) because their expected finish time (EFT) is far from their deadline.

The EFT of workflow task $_i$ on VM type \mathbf{k} can be calculated using this equation which represents the expected schedule.

$$EFT_{wt_i, VM_k} = \frac{S(wt_i)}{EC(VM_k)} \Longleftrightarrow ExpectedSchedule \quad (3.4.4)$$

- Where $S(wt_i)$ is the size of workflow task i ,
- $EC(VM_k)$ is the execution capacity of VM type k .

3.4.3.3 Definition 3.

Every workflow task has a deadline. A deadline is a time-based scheduling constraint that requires a workflow task to be scheduled within a specified time period. These deadlines are set by users. A task can have a shorter/earliest deadline or a longer deadline.

- A shorter/earliest deadline task is a task that is close to its specified scheduling time. Such tasks are assigned the highest priorities.
- Tasks with long deadlines are tasks that do not require higher priority. This is because such tasks are not close to their predefined deadlines and therefore can be assigned to resources after the higher priority tasks (shorter deadline tasks) have been scheduled.

Thus, each task must be executed before the specified deadline. If the execution of a workflow task T exceeds its deadline, it means that the deadline constraint set by the user has been violated.

As shown in Figure 3.1, we have four workflow tasks that have arrived and are queued to be mapped to VMs. The arrived tasks are $wt1$, $wt2$, $wt3$ and $wt4$. We also have three types of VMs (VM_{HEC} , VM_{MEC} and VM_{LEC}). These tasks have different deadlines and we need to map them to VMs based on their deadlines. $wt1$ belongs to the HCT group because its deadline is 2 min, which is too tight and if it is not allocated immediately, it might miss its deadline. To prevent $wt1$ from missing its deadline, we quickly mapped it to VM_{HEC} . The reason is that VM_{HEC} is a high-speed VM that can finish executing $wt1$ before or on its deadline. The next task closer to its deadline is $wt4$ and since VM_{HEC} is currently executing $wt1$, we mapped $wt4$ to VM_{MEC} . The projections show that by the time $wt1$ will meet its deadline, $wt3$ will have 3 min left to also meet its deadline (5min-2 min). Therefore, $wt3$ is classified as an HCT at that time and if there is no other HCT in the waiting queue at that time, then $wt3$ is mapped to VM_{HEC} and $wt2$ is mapped to VM_{MEC} . In this case, VM_{LEC} is idle. The idle VM is turned off or put into safe mode to save power.

3.4.4 Multi-objective optimization problem formulation[10]

The goal of workflow scheduling is to find the best optimal solution from various optimization methods that can give good results for the optimization problem. The optimization problem can be a single-objective or multi-objective optimization problem. Single-objective optimization involves optimizing only one scheduling objective, such as execution makespan or cost. Multi-objective optimization, on the other hand, focuses on optimizing two or more objectives.

The problem of multi-objective optimization varies from user to user and must be managed accordingly. Some users may want a reduced execution makespan and execution

cost while ensuring that QoS constraints are not violated, others may want to reduce energy consumption and task execution time while ensuring effective utilization of cloud resources. In this work, a multi-objective scheduling problem has been considered.

We consider a workflow application as DAG. It is modeled with data dependencies representing precedence constraints between tasks. These tasks represent requests from the users to be processed on cloud resources (VM_{HEC} , VM_{MEC} , VM_{LEC}) so that the energy consumption, makespan and execution cost are reduced without violating the deadlines of the tasks. However, reducing energy consumption increases the execution cost of scientific workflow applications. This is because energy consumption and execution cost are conflicting metrics.

Our goal is to reduce the energy consumption, makespan and execution cost of workflow tasks with dependency constraints.

3.4.4.1 Definition 4.

Since we aim at improving resource management to reduce energy consumption in cloud data centers at a lower cost without violating scheduling constraints, energy consumption can be derived from the next equation. Here, TE_E is the total energy consumed for executing workflows, ED_E is the energy consumed for transferring all data and the sum of energy consumed for scheduling all tasks on all VM.

$$TE_E = ED_E + \sum_{En}^{Ek} ? \quad (3.4.5)$$

3.4.4.2 Definition 5.

The metric for the execution cost is calculated using the following equation:

$$TC = (P_j * PC + (\sum_{f \in Finj} Size(f) + \sum_{f \in Foutj} Size(f))) * TrC \quad (3.4.6)$$

where,

- TC is the total cost.
- P_j is the processing time of job j
- PC is the processing cost
- TrC is the cost of transferring the input files (F_{inj}) and the output files (F_{outj}).

3.5 Conclusion

In this chapter, we have do \todo...

CHAPTER 3. THE PROBLEM OF SCHEDULING THE SCIENTIFIC
WORKFLOW ON A CLOUD PLATFORM WITH ENERGY OPTIMIZATION

Symbol	Definition	Unity
wt_i	workflow task	-
VM_k	Virtual Machine type k	-
EFT_{wt_i, VM_k}	Expected Finish Time of task W_{ti} executed on VM_k	Second
$S(wt_i)$	Size of wt_i	MI
$EC(VM_k)$	Execution Capacity of VM_k	MIPS
TE_E	Total Energy consumed for executing workflows	kW/h
E_{DT}	Energy consumed for Data Transferring	kW/h
TC	Total Cost	\$
P_j	Processing time of job j	\$
PC	Processing Cost	\$
TrC	Cost of Transferring the input files (F_{inj}) and the output files (F_{outj}).	\$

Table 3.4.1: Symbols used in the formulas

Cloud data centers generate massive energy consumption on a global scope. It needs to reduce the overall energy consumption of cloud data centers under the premise of satisfying QoS[5].

In the next chapter we will suggest our proposed approach.

CHAPTER 4

State of art and our proposed approach

4.1 Introduction

In recent years, the vast energy consumption generated by cloud data centers has attracted wide attention from society. Relevant survey data show that electricity demand in US data centers has increased from 29 billion kWh in 2000 to nearly 73 billion kWh in 2020. The computing device will also increase the thermal load while using electric energy. The thermal load of a single rack will reach 50 kW in 2025. In addition, it also leads to the decline of quality of service (QoS) and environmental deterioration, which runs counter to the concept of green sustainable development. Therefore, achieving energy conservation and emission reduction is the key to developing the next generation of green data centers[5].

We will present in this chapter the EViMA algorithm which is used for improving resource management in the cloud data center to obtain the best task schedules that reduce energy consumption, execution makespan and cost without violating deadline constraints. Then will introduce the VM placement and VM consolidation problem and its 4 sub-problems and for each subproblem some of the most used methods to find the solutions. And finally, we will present the objectives, the concept of our approach and also its description and explication.

4.2 Studied algorithm EViMA

4.2.1 EViMA[10]

Energy-efficient virtual-machine mapping algorithm (EViMA) focuses on improving resource management in the cloud data center to obtain the best task schedules that reduce energy consumption, execution makespan and execution cost without violating deadline constraints.

The EViMA algorithm is supported by 4 other algorithms, namely:

- Highly Critical Workflow Task Selection Algorithm (HiCTSA)
- Low Critical Task Selection Algorithm (LoCTSA)
- VM Power Regulating Algorithm (VM-PRA),
- Slack Time Harvesting Algorithm (STiHA).

The algorithm takes as input a set of workflow tasks represented by $wt = (wt1, wt2, wt3, \dots, wtn)$ and a set of instances (VMs) represented by $(VM_{HEC}, VM_{MEC}$ and $VM_{LEC})$. The “n” workflow tasks can be any type of task in a queue that requires resources for scheduling.

1. The algorithm starts by identifying the arrived tasks and computing their expected finishing times (EFT) to determine the task execution order.

2. Then, a task priority queue is created in which the tasks are arranged in three execution orders based on their critical times (HCT, MCT and LCT). To determine the execution order of the tasks, their expected finishing times are calculated using Eq.(4). The tasks that are close to their deadline and need to be mapped to VMs at or before their deadline are given higher priority.
3. To generate cost-efficient and energy-optimized schedules, a VM assignment order is also introduced (Line 5). Once both the tasks execution order and VM assignment order is established, we employ EViMA to explore through the various workflow tasks to find the overall critical tasks (HCTs), (tasks that are approaching their deadline) from the ready queue and immediately assign them to a high speed procession VM (VM_{HEC}). This is done to find high-quality solutions to the workflow scheduling problem. For example, if wt_i through the searching process is found to belong to the HCT queue, Algorithm 4.2 is called to assign it to a VM with high processing speed (VM_{HEC}), as described in line 8. However, if the task belongs to the LCT group, Algorithm 4.3 is called to process it, as described in line 9.
4. On the other hand, if the expected finishing time of the task is less than or equal to the harvested slack time of VMk, Algorithm 4.5 is applied to schedule the task as in lines 15 and 16.
5. If the expected end time is equal to the deadline, assign the task directly to VM_{MEC} as in lines 11 and 12. Otherwise, apply Algorithm 4.4 to shut down the VM when one of the VMs is idle to save power (line 18).

4.2.2 Highly critical task selection algorithm (HiCTSA)

Highly Critical Task Selection Algorithm (HiCTSA), presented as Algorithm 4.2, solve the problem of which workflow tasks should be selected first and to which VM and ensure that all workflow tasks meet their deadlines.

HiCTSA works at both the VM and task levels to ensure that tasks are selected and assigned to resources for execution to avoid delays. We consider a set of workflow tasks $WT = (wt1, tw2, tw3, \dots, wtn)$ in a workflow application. These tasks have different deadlines and therefore users expect them to be executed before their deadlines. Also, these tasks require some resources to start the scheduling process.

Algorithm 2 starts by checking the criticality of the tasks as in line 1. Then compares the expected finishing time of each workflow task in the ready queue (line 2). If the expected finishing time of workflow task $wt1$ approaches its deadline, this task is marked as an HCT task (line 4). A mapping is then created for all HCT (line 5). However, if two or more tasks have the same deadline, another high speed VM is deployed to map those tasks (line 7 and 8)[10].

Algorithm 4.1 EViMA[10]

Require: Workflow, set of VMs and set of VM types ($VM_{HEC}, VM_{MEC}, VM_{LEC}$)

```

1:  $wt_{ReadyPool} = clustered(wt_1, wt_2, wt_3, \dots, wt_n)$ 
2: while  $wt_{ReadyPool} \neq \phi$  do
3:   Compute EFT of each tasks
4:   Group  $Tasks \Rightarrow HCT, MCT, LCT$ 
5:   Group  $VMs \Rightarrow (VM_{HEC}, VM_{MEC}, VM_{LEC})$ 
6:   foreach  $wt$  in  $wt_{ReadyPool}$  do
7:     if  $wt_i$  in HCT and  $wt_j$  in LCT then
8:       apply algorithm 2 to execute HCT
9:       apply algorithm 3 to execute LCT
10:    else
11:      if EFT of  $wt_i = D1$  of  $wt_i$  then
12:         $wt_i \mapsto VM_{MEC}$ 
13:        Update  $wt_{ReadyPool}$ 
14:      else
15:        if EFT of  $wt_i \leq ST$  then
16:          apply algorithm 5
17:        else
18:          apply algorithm 4 to save mood the idle VM
19:          Update  $wt_{ReadyPool}$ 
20:        end if
21:      end if
22:    end if
23:  end foreach
24: end while

```

Algorithm 4.2 HiCTSA[10]

Input : • Workflow DAG, $G=(W,T)$ where $W= (wt_1 ,wt_2 ,wt_3 ,\dots,wt_n)$
 • A set of cloud resources with different CPU configuration and different pricing
Output:
 1 Check tasks criticality in $wt_{ReadyPool}$
 2 Compare EFT of wt_1 on VM_k to its Dl
 3 **if** *EFT of $wt_1 > Dl$ of wt_1* **then**
 4 | Call wt_1 HCT
 5 | Map all HCT to VM_{HEC}
 6 **else**
 7 | **if** $DLwt_i == DLwt_j$
 8 | *deploy VM_{HEC2}*
 9 | **If** *EFT of $wt_i = Dl$ of wt_i* , **then**
 10 | Call the task MCT
 11 | Update $wt_{ReadyPool}$
 12 | **while** $wt_{ReadyPool} \neq \phi$ **do**
 13 | | Repeat step 3 to 8
 14 | **end**
 15 | **else**
 16 **end**

4.2.3 Low critical task selection algorithm (LoCTSA)

Low Critical Task Selection Algorithm (LoCTSA) proposed in Algorithm 4.3, reduces the execution cost and consists of two phases[10]:

1. The first phase is the task identification and grouping phase, in which less critical tasks are identified and queued for execution. In the first phase, the algorithm compares the expected finishing time of each workflow task with its deadline. If a task's EFT is greater than its deadline, the task is moved to a new queue called the LCT queue (lines 1, 2 and 3). This is done to avoid mapping less critical tasks to HEC VM.
2. The second phase is the VM allocation phase. In the second phase, the quality of the solution is improved by ensuring that all less critical tasks are assigned to VMs with lower processing speed (VMs with lower cost and lower energy consumption) instead of assigning them to VMs with higher execution costs. This is done to reduce the cost of running workflow tasks.

Algorithm 4.3 LoCTSA[10]

```

1 Compare EFT of  $wt_1$  on  $VM_k$  to its  $Dl$ 
2 if EFT of  $wt_1 < Dl$  of  $wt_1$  then
3   | Call  $wt_1$  LCT
4   | Map all LCT to  $VM_{LEC}$ 
5 else
6   | if EFT of  $wt_i = Dl$  of  $wt_i$ , then
7     | move  $wt_i$  into MCT queue
8     | Update  $wt_{ReadyPool}$ 
9   | end
10 end

```

4.2.4 VM power regulating algorithm (VM-PRA)

VM-PRA (see Algorithm 4) checks the status of all VMs used to execute workflow tasks. Sometimes, underutilized VMs are not shut down, which can increase the power consumption of a data center. VM-PRA overcomes this drawback by introducing Idle “off” Busy “on” techniques that identify the status of all VMs.

The method starts by defining the status of all VMs that are used for workflow task execution. The VMs are divided into two stages, namely a busy stage and an idle stage. A VM is said to be busy when it is currently executing or processing a workflow task. Idle VMs, on the other hand, are VMs that have not yet been assigned any job or have completed the execution process and are waiting for more jobs to be executed. The algorithm checks, if there are some idle VMs and if so, it switches all idle VMs to the memory state. For example, when VM_{HEC} and VM_{LEC} are running and VM_{MEC} is idle, the VM-PRA algorithm quickly switches VM_{MEC} into power saving mode to save some energy. In reality, VMs do not consume power, but when the VM is powered off or put into power-saving mode, it reduces the power consumption of its PM and hence a reduction in data center power consumption.

4.2.5 Slack time harvesting method (STiHaM)

Slack time is a time interval that occurs when a particular task completes its execution before its actual scheduled completion time. In other words, it is a time difference between a task’s scheduled completion time and its actual completion time. The slack time is used to accumulate the slack times for each VM type in the VM pool as in line 5 of Algorithm 4.5. A slack time hub is attached to each VM to store the slack times of each VM (see line 6). If a slack time of VM_k is equal to the EFT of wt_i , the scheduler is asked to map wt_i to VM_k to use the slack time as in lines 7 and 8 of Algorithm 4.5. The slack time is used to further reduce the execution cost of the workflows. Since wt_i can be scheduled on VM_k using the obtained slack time, the cost of provisioning a new VM to schedule the tasks is saved.

Algorithm 4.4 VM-PRA[10]

```

1  VMPool = (VMHEC, VMMEC, VMLEC,.....VMk)
2  foreach VM in the VMPool do
3      Check VMs status (busy or idle)
4      if VMHEC is idle then
5          Put VMHEC on save mood
6      else
7          if VMMEC and VMLEC are idle then
8              Put VMMEC and VMLEC on save mood
9              Update VMPool
10         end
11     end
12 end

```

Algorithm 4.5 STiHaM[10]

Input : • Most workflow tasks have slack times
Output:

```

1  WtReadPool = (wt1, wt2, tw3,.....wtn)
2  VMPool = (VMa, VMb, VMc,.....VMK)
3  foreach VM in the VMPool do
4      Generate the EFT and ST of all tasks
5      Harvest the slack time of each VM
6      Store the slack times of each VM in STHub
7      Prompt the scheduler when EFT of wti,j = ST of VMk
8      Allocate wti,j to utilize the slack time
9      Update VMPool
10 end

```

4.3 VM placement & VM consolidation

4.3.1 Problem

Information and communications technology equipment (ICT) and cooling equipment consume most of the energy in the data centers. The server is the most critical ICT equipment. In most cases, the average CPU utilization rate of the server is only 15% to 20% and there are a large number of servers in an idle state which consumes much energy. Therefore, VMs can be relocated into as few physical machines (PM) as possible through VM consolidation technology and the low-load PM can be switched off. No-load PM can be switched to a sleep state to reduce the number of PMs during activities and avoid energy waste[5].

In addition, the problem of low server utilization is appearing to narrow the dynamic power ranges of servers: even completely idle servers still consume about 70% of their peak power. Therefore, keeping servers underutilized is highly inefficient from the energy consumption perspective[3].

4.3.2 VM Placement

It is the process of mapping VMs to PMs.

4.3.3 VM Consolidation

It refers to the replacement process using a fewer number of PMs, thus contributing to energy conservation[3]. Most of the existing methods are committed to consolidating VMs to a small number of servers to improve the utilization of server resources and reduce total energy consumption while preventing hot-spots[5].

Dynamic VM consolidation follows a distributed model, where the problem is divided into 4 sub-problems[3]:

1. **Host under-load detection:** Determining if a host is considered to be under-loaded so that all VMs should be migrated and the host should be switched to a low-power mode. There are 2 methods:
 - Static Thresholding: fixed value is set to a utilization threshold and below this value, the PM is considered under-loaded.
 - Dynamic Thresholding: Static Thresholding is unsuitable for a cloud computing environment that has dynamic and unpredictable workloads. The system should be able to automatically adjust its behavior depending on the workload patterns, there are 2 patterns:
 - Adaptive Thresholding (AT): This study proposes an Adaptive Thresholding (AT) technique based on statistical analysis of historical data, collected during the lifetime of VMs. AT adjusts the value of the CPU utilization threshold depending on:

- * Mean Adaptive Thresholding: depending on the mean of the values of the minimum PMs utilization in the data center, an adaptive value is set to a utilization threshold. This value is changed frequently due to the dynamic and unpredictable workloads of the cloud environment.
- * Median Adaptive Thresholding: it depends on the median. An adaptive value is set to a utilization threshold. This value is also changed frequently in a cloud environment.
- Boxplot: It is one of the most frequently used graphical techniques for analyzing data sets, see Figure 4.1. To create a boxplot, the elements of any data set are arranged in ascending order. Then, the median value of the arranged numbers, (called in this method Q2), is calculated. Q2 divides the data into two subsets. To divide the data into quarters, the medians of these two subsets are calculated too. The median of the left subset is called Q1, while the median of the right one is called Q3. If the data set has an even number of elements, Q1 will be the average of the two middle elements. This works with the two subsets as well, if they have an even number of elements.

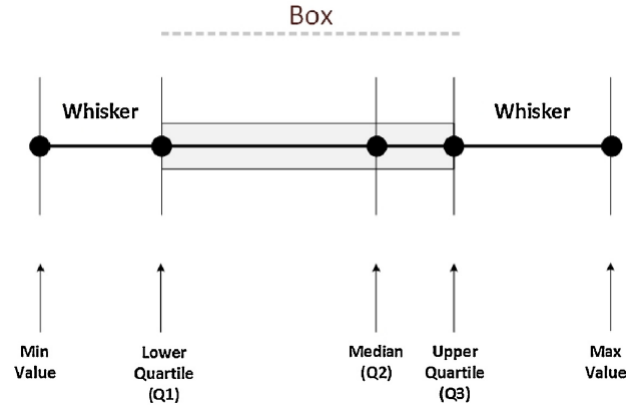


Figure 4.1: Description of the Boxplot Method

2. **Host overload detection:** Determining if a host is considered to be overloaded, so that some VMs should be migrated to other active, or reactivated hosts, to meet the QoS requirements. There are 2 methods which were explicated before:
 - Static Thresholding.
 - Dynamic Thresholding: there are 2 patterns:
 - Adaptive Thresholding (AT).
 - * Mean Adaptive Thresholding: depending on the mean of the values of the maximum PMs utilization in the data center, an adaptive value is set to a utilization threshold.

- * Median Adaptive Thresholding: it depends on the median of the values of the maximum PMs utilization in the data center.
 - Boxplot (we will use this method in our approach).
3. **VM selection:** Selecting VMs that would be migrated from under/ overloaded hosts. There are many policies:
- Random Sampling (RS) Policy
 - Systematic Sampling (SS) Policy
 - Minimum Time Cost Migration (MTCM) Policy
 - Minimum Energy Cost Migration (MECM) Policy (we will use this method in our approach, see Algorithm. 4.6)

Algorithm 4.6 Algorithm of MECM

<i>Input:</i>	<i>Set of VMs = {VM₁, VM₂, ..., VM_n}</i>
<i>Output:</i>	<i>Selected element VM_s</i>
1	for $i = 1$ to n do
2	<i>calculate</i> $VM_i^{Mig_energy}$
3	<i>sort (VM) such that</i> $VM_i^{Mig_energy} < VM_{i+1}^{Mig_energy}$
4	$VM_s = VM[1]$
5	return VM_s

4. **VM placement:** Placing VMs selected for migration to other active, or reactivated hosts. To enhance the energy efficiency in virtualized cloud data centers, it could be noted that the approaches with conditional VM placement are much better compared with random VM placement approaches. The conditional VM placement approaches result in better energy efficiency[3]. We will use VM algorithm placement based on KP(see Algorithm 4.7).

4.4 Proposed approach

4.4.1 Objectives

The objective of our project is to provide a workflow scheduling mechanism in Cloud Computing. We propose an approach that consists of managing the available resources within a Cloud Computing system in the most efficient way possible and properly assigning tasks to virtual machines in order to minimize cost, makespan and energy without violating the deadline constraints established for each task by the user, in order to satisfy QoS.

So, this approach considers the following metrics and constraint:

- Energy consumption

Algorithm 4.7 VM placement based on KP

<i>Input</i>	<i>Set of VMs ($VMs = \{VM_1, VM_2, \dots, VM_n\}$), set of the available PMs ($PMs = \{PM_1, PM_2, \dots, PM_m\}$)</i>
<i>Output</i>	<i>VM placement</i>
1	sort PMs based on capacity / availability in descending order
2	$j=1$
3	repeat for all VMs
4	pick PM_j
5	select a VM which results minimum energy on PM_j
6	if the requirement of the selected $VM_{(s)}$ \leq PM_j available capacity then
7	place $VM_{(s)}$ on PM_j
8	$j=j+1$
9	until no more VMs
10	return VMs to PMs placement result

- Cost
- Maskespan

4.4.2 Principle and description

Our proposed approach is a version of the EViMA algorithm with the addition of a VM placement algorithm.

- We suggest doing workflow partitioning and grouping tasks in the same partition as just one task, this can reduce the scheduling time.
- We suggest starting the EViMA algorithm first for scheduling scientific workflows.
- We think that when the PM is under-loaded or overloaded, it will consume more energy this is why we think that if we apply during the execution some of the virtual machine migration strategies and policies as in this article[3], may be we will avoid this problem and so reducing energy consumed by PM. After seeing the results of experiments and test performance[3], so we decide to apply the best policies which are:
 - Boxplot: for both **host overload and under-load detection**
 - MECM for **VM selectionW**
 - We will use the classical Knapsack Problem (KP) for **VM placement**

4.4.3 Steps of our approach

First, we apply the EViMA approach which focuses on improving resource management in the cloud data center to obtain the best task schedules that reduces energy consumption,

Algorithm 4.8 Our approach

Require: Workflow, set of VMs and set of VM types ($VM_{HEC}, VM_{MEC}, VM_{LEC}$), PMs

- 1: $wt_{ReadyPool} = clustered(wt_1, wt_2, wt_3, \dots, wt_n)$
 - 2: apply algorithm EViMA
 - 3: $VM_{candidateList} = \phi$
 - 4: add all VMs that are in under-loaded PMs to $VM_{candidateList}$
 - 5: add all VMs that are in over-loaded PMs to $VM_{candidateList}$
 - 6: While $VM_{candidateList}$ is not empty
 - 7: apply algorithm 6 to select one VM from $VM_{candidateList}$
 - 8: apply algorithm 7 to determine emplacement of this VM
 - 9: End while
-

execution makespan and execution cost without violating deadline constraints.

Then, we initialize the VM candidate list to an empty set and we add all VMs that are in under/over-loaded physical machines. Physical machines are considered under-loaded if they are 1st quarter(Q1) using the boxplot method (see 4.1) and are overloaded if they are great than 4th quarter(Q4).

After that, using algorithm MECM calculate for each VM the expected migration energy and sort them in ascending order.

And finally, we map the selected VMs to suitable physical machines to do a VM consolidation, which will allow us to switch off the PMs which are not used.

4.5 Conclusion

This chapter was devoted to presenting our approach, based on the EViMA algorithm and VM placement and consolidation, to scheduling workflow tasks in a cloud computing environment.

So in the next chapter, we will present our experiment and simulation in order to evaluate the performance of the solutions generated by the proposed algorithm.

CHAPTER 5

Implementation, simulation and discussion of results

5.1 Introduction

Over the past few years, cloud computing has gained tremendous popularity since it presents a flexible and efficient solution for many services on the Internet. Cloud is a large and complex system since it is composed of several users, service providers, physical machines, service brokers, task scheduling algorithms, bandwidth, internet latency and storage technology, etc. On the other hand, all cloud-based implementations need various configurations. Therefore, it is very difficult for researchers to evaluate the performance of their policies in a real cloud environment[12].

In this chapter we will present the simulation tool, workflows used, experiments parameters, performance metrics and outcomes of the measurement will be presented.

5.2 Simulation tool: WorkflowSim

The simulator can play an important role in reducing cost, efficiency, infrastructure complexity and security risks before a solution can be deployed on a real infrastructure. By focusing on issues related to the quality of a particular component under various scenarios, cloud simulators enable performance analysts to monitor the behaviors of the system[12].

5.2.1 Description and main features

WorkflowSim was introduced for modeling Scientific Workflows in a cloud environment. Workflows in heterogeneous distributed systems show different levels of overheads that are explained based on computational operations and miscellaneous works.

Most simulators such as CloudSim do not consider fine granularity simulations of workflows and task clustering. Task clustering reduces the number of jobs to be executed and so execution overhead is decreased.

Generally, a job may have a high risk of suffering from failures since it consists of several tasks. Researchers with WorkflowSim can study the impact of job failures runtime performance of workflows for different clustering methods[12].

5.2.2 Architecture

Figure 5.1 shows the structure of WorkflowSim which is composed of several components:

1. Workflow Mapper that maps abstract workflows to concrete workflows.
2. Workflow Engine that controls the data dependencies.
3. Workflow Scheduler assigns jobs to resources.
4. Clustering Engine that groups small tasks into a large jobs.
5. Failure Generator that injects task failures for each execution site.

6. Failure Monitor that stores the failure information such as resource id and task id.

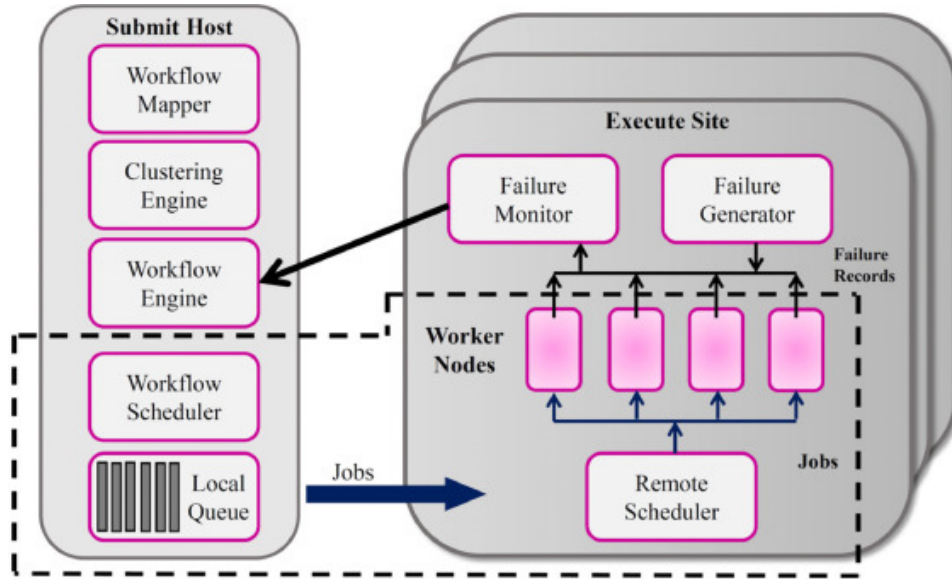


Figure 5.1: WorkflowSim's Architecture [12]

5.2.3 Downloading and Installing WorkflowSim with Eclipse without GitHub

To make the first WorkflowSim project, follow this steps[18]:

1. **Download Source Files** directly from <https://github.com/WorkflowSim/WorkflowSim-1.0/archive/master.zip> and unzip it to *your_repo_root*.
2. **Check out your source codes:**
 - (a) create a new Java project called 'WorkflowSim' (it doesn't have to be 'WorkflowSim').
 - (b) Right-click your project and choose 'Import'. Click 'General'->' Projects from Folder or Archive'.(see Figure 5.2)
 - (c) Set the URL to be your extracted folder (wait until Finish button will be enabled as in Figure 5.3) .
3. **Run an Example:** Run an example i.e. `org.workflowsim.examples.WorkflowSimBasicExample1.java`
 - (a) Open `WorkflowSimBasicExample1.java`
 - (b) replace the String `daxPath = "/Users/chenweiwei/Work/WorkflowSim-1.0/config/dax/Monta` with your real physical file path.
 - (c) Right-click on `WorkflowSimBasicExample1.java` and choose 'Run File'. You should be able to see some output.

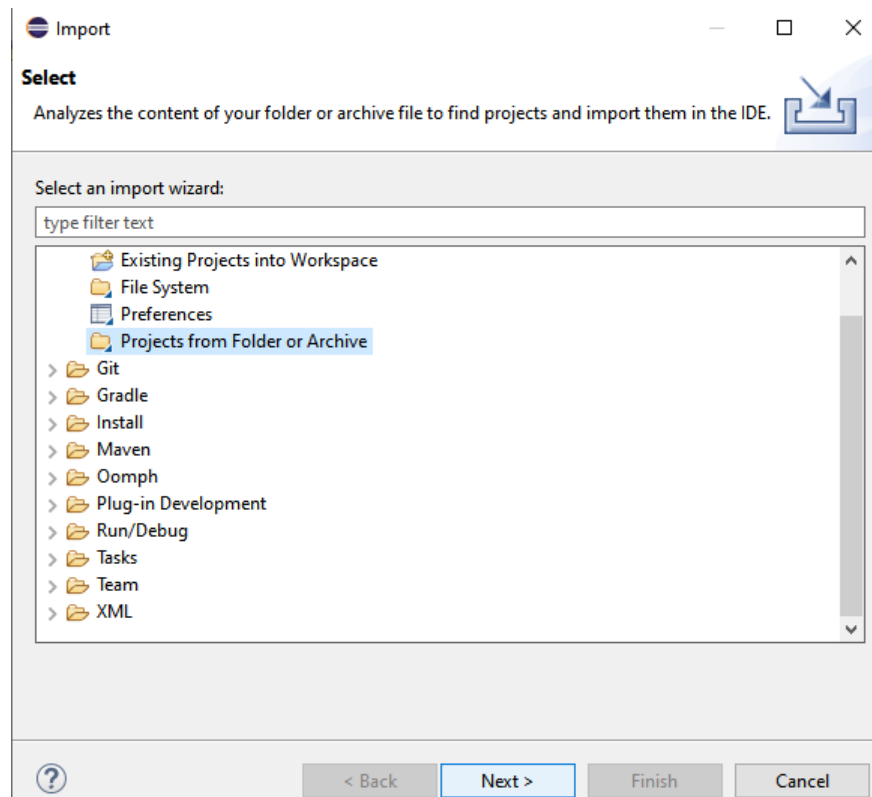


Figure 5.2: Import a project

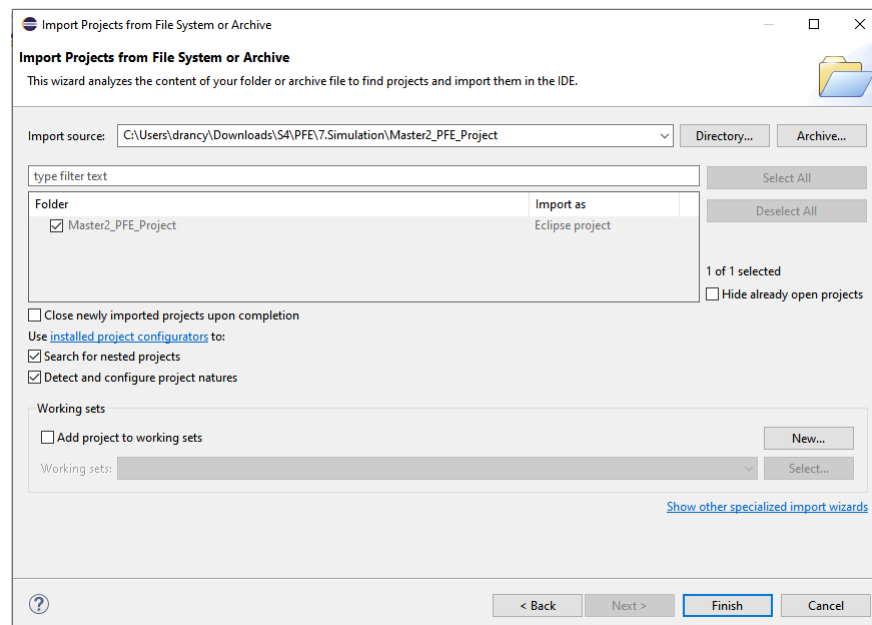


Figure 5.3: Set URL to be the extracted folder

5.2.4 Strengths

WorkflowSim has many advantages[12]:

1. It supports a stack of workflow parser and workflow engine delay to workflow optimization techniques with better accuracy are implemented.
2. It implements several workflow-scheduling methods such as HEFT, Min-Min and Max-Min and so developers can compare their algorithms with them in a simple way.
3. It considers task clustering and layered overhead in the workflow simulation. Therefore, developer scan and test various strategies based on a real trace of overheads.
4. It models failures for two layers (i.e., task/job) by an interface and hence developers can design fault-tolerant techniques.

5.2.5 Limitations

WorkflowSim has some limitations [12]:

1. It includes limited types of failures. Therefore, developers cannot simulate the situation when a task is not successfully sent due to network problems or workflow scheduler issues.
2. It does not consider the performance characteristic of file I/O. Therefore, developers cannot obtain suitable simulations for data-intensive applications since these applications involve reading or writing huge data files.
3. It supports only simple workflow techniques and so developers cannot use other important approaches like workflow partitioning in their implementations.

5.3 Workflows

We use five different types of benchmark workflows (Montage, Cybershake, LIGO, Epigenomics and Sipht) available on the Pegasus website.

These workflows typically involve complex data of various sizes and complicated computer simulations. They need high computation power and the availability of large infrastructures that cloud computing environments provide with different quality of service levels. The quality of schedules obtained by various algorithms is also compared using these workflows. Information about the task dependencies, the amount of data transmitted and the execution time estimates are presented in DAX format. DAX files include a single time estimate for each task. To simulate the case of different time constraints provided by users, the start time and a deadline for each workflow are randomly assigned [14].

Each of these workflow types has four different workloads. For example, the Cybershake workflow has four workloads, including 30, 50, 100 and 1000 [10].

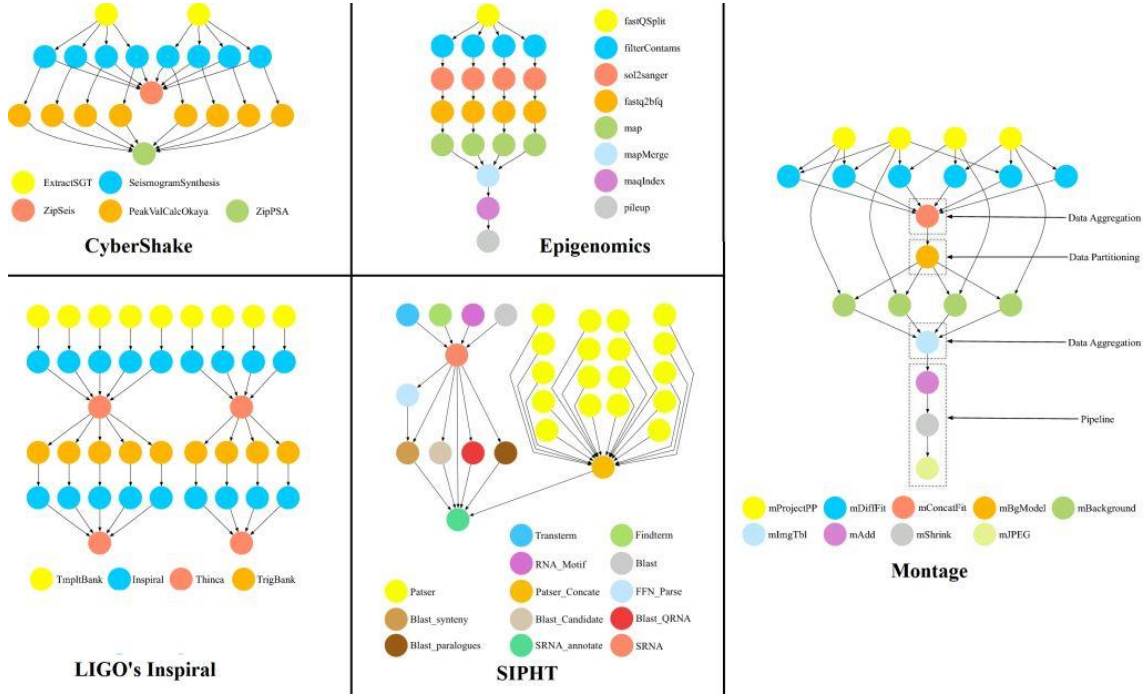


Figure 5.4: Structure of Scientific Workflows used confluence.pegasus.isi.edu

5.4 Simulation parameter

The inputs to the simulation environment are as follows:

- The average bandwidth between resources is 20 MBps.
- The processing matrix for each VM is measured in Million Instruction Per Second (MIPS).
- The task lengths are set in Million Instruction (MI).

In this work, we assumed that there are only three types of VMs (VM_{HEC} , VM_{MEC} and VM_{LEC}).

We further assumed that there are only 3 types of VMs in the cloud data center, each with an infinite number of instances. The computational capacities and the prices per hour for each VM type are set as the work. The instant capacity selection and pricing model are based on Amazon EC2 (see Table 5.4.1)[10].

Virtual Machine	type	Capability (CU)	Price (\$/h)
LEC	1	1.0	0.12
LEC	2	1.5	0.195
MEC	3	2.0	0.28
MEC	4	2.5	0.375
HEC	5	3.0	0.48
HEC	6	3.5	0.595

Table 5.4.1: VM types and pricing model

5.5 Experimentation

5.6 Comparison and discussion of the solution

5.7 Conclusion

We have simulated the proposed approach using the WorkflowSim simulator. We compared the outcomes of five separate scientific workflows with our proposed algorithm.

General conclusion

Discussion

Limits

Perspectives

Bibliography

- [1] So many clouds-what's the difference?, May 2020.
- [2] Digital notes on cloud computing (r18a0523). Technical report, DEPARTMENT OF INFORMATION TECHNOLOGY, MALLA REDDY COLLEGE OF ENGINEERING & TECHNOLOGY (Autonomous Institution - UGC, Govt. of India), Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad - 500100, Telangana State, India, 2021.
- [3] Auday Al-Dulaimy, Wassim Itani, Rached Zantout, and Ahmed Zekri. Type-aware virtual machine management for energy efficient cloud data centers. *Sustainable Computing: Informatics and Systems*, 19:185–203, 2018.
- [4] Mohammed BENHAMMOUDA et al. *Contribution a l'optimisation d'ordonnancement de workflows dans un environnement cloud*. PhD thesis, 2021.
- [5] Rui Chen, Bo Liu, WeiWei Lin, JianPeng Lin, HuiWen Cheng, and KeQin Li. Power and thermal-aware virtual machine scheduling optimization in cloud data center. *Future Generation Computer Systems*, 2023.
- [6] Neha Garg, Manish Raj, Indrajeet Gupta, Vinay Kumar, GR Sinha, et al. Energy-efficient scientific workflow scheduling algorithm in cloud environment. *Wireless Communications and Mobile Computing*, 2022, 2022.
- [7] Xiaozhong Geng, Yingshuang Mao, Mingyuan Xiong, and Yang Liu. An improved task scheduling algorithm for scientific workflow in cloud computing environment. *Cluster Computing*, 22:7539–7548, 2019.
- [8] Katharina Görlach, Mirko Sonntag, Dimka Karastoyanova, Frank Leymann, and Michael Reiter. Conventional workflow technology for scientific simulation. *Guide*

- to *e-Science: Next Generation Scientific Research and Discovery*, pages 323–352, 2011.
- [9] Mandeep Kaur and Rajni Aron. An energy-efficient load balancing approach for scientific workflows in fog computing. *Wireless Personal Communications*, 125(4), 2022.
- [10] J Kok Konjaang, John Murphy, and Liam Murphy. Energy-efficient virtual-machine mapping algorithm (evima) for workflow tasks with deadlines in a cloud environment. *Journal of Network and Computer Applications*, 203:103400, 2022.
- [11] Anita Lee-Post and Ram Pakath. *Cloud Computing: A Comprehensive Introduction*, pages 25–26. 01 2014.
- [12] Najme Mansouri, R Ghafari, and B Mohammad Hasani Zade. Cloud computing simulators: A comprehensive review. *Simulation Modelling Practice and Theory*, 104:102144, 2020.
- [13] Peter Mell and Timothy Grance. The nist definition of cloud computing, 2011-09-28 2011.
- [14] Ali Mohammadzadeh, Mohammad Masdari, and Farhad Soleimanian Gharehchogh. Energy and cost-aware workflow scheduling in cloud computing data centers using a multi-objective optimization algorithm. *Journal of Network and Systems Management*, 29:1–34, 2021.
- [15] Jean Etienne Ndamlabin Mboula. *Energy-efficient Workflow Scheduling with Budget and Deadline constraints in a Cloud Datacenter*. Theses, Faculty of Science, University of Ngaoundere, Cameroon, September 2021.
- [16] Eoin Scollard. Introduction to the concepts of cloud computing. 2019.
- [17] WfMC. *Glossary*, 2009.
- [18] WorkflowSim.org. Workflowsim-1.0#readme, 2012.
- [19] BADR Youakim. Service-oriented workflow. *Journal of Digital Information Management*, 6(1):119, 2008.
- [20] Christian Zeyen, Lukas Malburg, and Ralph Bergmann. Adaptation of scientific workflows by means of process-oriented case-based reasoning. In Kerstin Bach and Cindy Marling, editors, *Case-Based Reasoning Research and Development*, Cham, 2019. Springer International Publishing.

Résumé

bla bla bla

Mots clés :

Abstract

bla bla bla

Keywords :
