UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Computer Science

Budget-aware scheduling algorithm for scientific workflow applications across multiple clouds

A Mathematical Optimization-Based Approach

Amin Ziagham Ahwazi

INF-3990 Master's Thesis in Computer Science, May 2022

*To my **Parents**.*

Thanks for Everythings.

“Nothing can stop the man with the right mental attitude from achieving his goal; nothing on earth can help the man with the wrong mental attitude.”
–Thomas Jefferson

“The opposite of a correct statement is a false statement. But the opposite of a profound truth may well be another profound truth.”
–Niels Bohr

Abstract

Scientific workflows have become a prevailing means of achieving significant scientific advances at an ever-increasing rate. Scheduling mechanisms and approaches are vital to automating these large-scale scientific workflows efficiently. On the other hand, with the advent of cloud computing and its easier availability and lower cost of use, more attention has been paid to the execution and scheduling of scientific workflows in this new paradigm environment. For scheduling large-scale workflows, a multi-cloud environment will typically have a more significant advantage in various computing resources than a single cloud provider. Also, the scheduling makespan and cost can be reduced if the computing resources are used optimally in a multi-cloud environment. Accordingly, this thesis addressed the problem of scientific workflow scheduling in the multi-cloud environment under budget constraints to minimize associated makespan. Furthermore, this study tries to minimize costs, including fees for running VMs and data transfer, minimize the data transfer time, and fulfill budget and resource constraints in the multi-clouds scenario. To this end, we proposed Mixed-Integer Linear Programming (MILP) models that can be solved in a reasonable time by available solvers. We divided the workflow tasks into small segments, distributed them among VMs with multi-vCPU, and formulated them in mathematical programming. In the proposed mathematical model, the objective of a problem and real and physical constraints or restrictions are formulated using exact mathematical functions. We analyzed the treatment of optimal makespan under variations in budget, workflow size, and different segment sizes. The evaluation's results signify that our proposed approach has achieved logical and expected results in meeting the set objectives.

Acknowledgements

First and foremost, I would like to thank my family, whose valuable encouragement and advice have clarified my career and academic life.

I would also like to have a special thanks to my primary advisor Associate Professor Chiara Bordin, and my co-advisor, Dr. Sambeet Mishra, whose constant guidance I have completed this thesis. They not only enlightened me with the academic knowledge but also gave me valuable advice whenever I needed it the most.

Lastly, I would like to thank my friends and everyone else for all the support they have given me.

Contents

List of Figures

List of Tables

1

Introduction

Scientific workflows are collections of several structured activities and fine-grained computational tasks that enable data analysis in a structured and distributed manner. Indeed, scientific workflows contain a set of computational tasks linked via control and data dependencies. These tasks within a workflow may have different sizes and require different running times ranging from a fraction of seconds to several hours [1]. Scientific workflows have been successfully used to make significant scientific advances in various domains such as biology, medicine, planetary science, astronomy, physics, bioinformatics, and environmental science [2]. Their importance is exacerbated in today's big-data era, as they become a compelling means to process and extract knowledge from the ever-growing data produced by increasingly powerful tools such as telescopes, particle accelerators, and gravitational wave detectors. There are several examples of scientific workflows' applications, such as Montage in Astronomy, Cybershake in Earthquake science, Epigenomics, and SIPHT in Bioinformatics, and LIGO in Astrophysics). Therefore, similar to other computational systems, workflows have scheduling stag which tasks within them are allocated to the desired resources optimally. Formally expressed, scientific workflow scheduling is the analysis of application structures to optimally assign tasks to computational resources based on application characteristics and resource availability. Workflow schedulers aim to produce a satisfactory solution in a relatively short time and low cost. Hence, scheduling scientific workflow is an open challenge in this regard.

1.1 Problem Definition

The scientific workflows often involve simulations of large-scale complex applications for validating the behavior of different real-world activities. Due to their large-scale, data, and compute-intensive nature, scientific workflows require high-performance computing environments to execute in an acceptable time and cost. In past decades, workflows were scheduled on distributed systems by providing own computing infrastructures and dedicated HPCs (High-Performance Computing platforms). Nevertheless, in recent years, the emergence of the cloud computing, as a new distributed systems paradigm, brings with it tremendous opportunities to run scientific workflows at low costs without the need of owning any infrastructure. Indeed, the cloud-computing paradigm has revolutionized the way of assessing computing resources by proposing a highly versatile availability of resources through a Pay-As-You-Go model. These features have enabled users to migrate their application processing to cloud platforms. In particular, Infrastructure as a Service (IaaS) in cloud providers allows workflow management systems (or brokers) to access a virtually infinite pool of resources that can be acquired, configured, and used as needed and are charged on a pay-per-use basis. IaaS providers offer virtualized computing resources called Virtual Machines (VMs) for lease. They have a predefined CPU, memory, storage, and bandwidth capacity, and different resource bundles (i.e., VM types) are available at differing prices. They can be elastically acquired and released and are generally charged per time frame or billing period.

The adoption of cloud computing for scientific workflow deployment has led to extensive research on designing efficient scheduling algorithms capable of elastically utilizing VMs. This ability to modify the underlying execution environment is a powerful tool that allows algorithms to scale the number of resources to achieve both performance and cost efficiency. During scheduling workflow applications in the cloud, both cloud providers and users are involved with the makespan and monetary costs criteria. Makespans refer to the completion time of the entire workflow, and the price is that users need to pay due to the usage of cloud resources in the monetary cost. In cloud computing, resources of different capabilities at different prices are provided. Typically, faster computing resources are more expensive than slower ones. Thus, different workflow applications scheduling strategies using different resources may result in different makespan and monetary costs. Therefore, the problem of scheduling workflow applications in the cloud requires both time and cost constraints to be satisfied. But on the other hand, these two criteria are also in conflict. On one side, optimal execution times (makespans) converge with solutions employing the fastest and most expensive computer resources. On the other hand, full optimization of monetary cost leads to poor performance in terms of execution time. For the issues mentioned above, the scheduling of

workflows is classified as an NP-complete problem, i.e., a problem that cannot be solved within polynomial time using current computing systems [3]. Hence, a trade-off must be found between these two criteria and should be done at the right time and the right cost. Most scheduling approaches focus on minimizing the total infrastructure cost while meeting a time constraint or deadline. Only a small fraction of techniques focus on scheduling under budget constraints. Others include a deadline constraint that guides the optimization process, and the budget is only taken into consideration when deciding the feasibility of a potential schedule. Contrary to this, in this thesis, a budget-driven algorithm is proposed whose objective is to optimize the way in which the budget is spent so that the makespan (i.e., total execution time) of the application is minimized.

Scientific workflows may require more instance types of VMs than those provisioned by a cloud provider. Many workflow scheduling approaches consider a single public cloud provider for provisioning required resources in a cloud environment. However, the demand for computational resources for executing large-scale workflow applications and using a multi-cloud environment, which provides scalable services, is rising. Therefore, the problem of scheduling workflow applications in multi-cloud has been considered in recent years. Indeed, a multi-cloud environment consists of multiple IaaS cloud providers that offer diverse instance types of VMs according to price and performance. Multi-cloud environments present many advantages, such as high scalability and prevention of vendor locking [1]. They are also more cost-effective settings to schedule workflows than individual clouds because of the variety in pricing and performance of instance types available from multiple providers. In addition, in recent years, cloud providers introduced the concept of virtual CPU (vCPU). Using this concept to measure the processing power of different VM instance types will help better schedule and increase the utilization of the VMs.

In terms of finding the optimal solution for scheduling workflow applications, it can be noted that optimization-based scheduling workflow algorithms can be classified into optimal and sub-optimal approaches [4]. Most approaches focus on finding sub-optimal solutions for workflow applications scheduling problems using heuristic and meta-heuristic methods. These approaches do not scale well with the number of tasks in the workflow and produce a static schedule unable to adapt to the inherent dynamicity of cloud environments. On the other hand, valuable and practical approaches have been presented based on mathematical programming, which can find optimal solutions.

1.2 Objectives and contributions

The main goal of this thesis is to formulate the static scheduling of the scientific workflow applications in multi-cloud environments by using mathematical programming so that optimizing makespan while meeting the budget and operational constraints. The intuition behind this strategy is to finish a workflow at a given budget as soon as possible. The objective is to minimize makespan under budget restrictions. A Mixed-Integer Linear Programming (MILP) formulation is used for modeling, which can provide comprehensive, clear, and flexible descriptions for our approach. The solution achieved is optimal, and MILP formulations are widely used in the industry since they can be solved in a reasonable time through efficient algorithms.

In this thesis, we make the following contributions:

- A novel scheduling approach that adopts a multi-cloud environment to resource provisioning for scientific workflow applications.

- The use of a smaller unit called segment and an approach to calculate the power of processing of each VM based on the virtual CPU (vCPU) and the segments of the workflow.

- We are developing a linear programming model to minimize both computation and communication costs for a budget-constrained scientific workflow that runs in a multi-cloud environment.

1.3 Context

The context of this thesis is the scheduling of scientific workflow applications, mathematical optimization, and cloud computing. Indeed, our solution is an interdisciplinary approach in line with the field of two research groups at the UiT- The Arctic University of Norway, the Center for Artificial Intelligence (CAI) group, and the Arctic Green Computing (AGC) group. The Center for Artificial Intelligence, CAI, investigates new ways of building intelligent computer systems and autonomous intelligent systems, using, among many techniques, data analytics, optimization, decision-support and decision-making with reasoning. While the AGC group focuses on energy efficiency, system complexity, and dependability across mobile, embedded, and data-center systems. This thesis is considered an interdisciplinary approach in that it uses mathematical optimization to optimize the budget consumption for scheduling the scientific workflow applications. Mathematical optimization gives the algorithm the ability to make optimal decisions, a powerful technique that can be applied in the

artificial intelligence (AI) field. On the other hand, to optimize the scheduling budget and reduce the operational time and cost, we used a multi-cloud environment in the subject of data-center, which is one of the topics studied in the AGC research group.

1.4 Organization

Chapter 2 - Technical background : Summarizes the technical and theoretical knowledge required to make decisions regarding design and implementation.

Chapter3 - Related work and contribution : This chapter contains a summary of researched work related to this topic of the study and a discussion on the key contribution of the thesis.

Chapter4 - System Design and Model : Presents the proposed scheduling workflow applications' architecture, design, and mathematical model in a multi-cloud environment.

Chapter5 - Experiments : Describes the experimental setup and metrics used to evaluate the implemented system.

Chapter6 - Evaluation : Present the results of the experiments conducted.

Chapter7 - Future work : Describes some features that could be added to further improve the current model.

Chapter8 - Conclusions : Summarizes the thesis and its findings and presents concluding remarks.

2 Technical Backgrounds

This chapter describes theoretical concepts relevant to the proposed scheduling algorithm to enhance the further reading experiment. The information presented should not be interpreted as a complete description or documentation; it merely serves the purpose of explaining concepts and some of the researched elements used in the algorithm. Section 2.1 gives a brief overview of cloud computing concepts and essential other topics related to cloud computing. Section 2.2 states the structure and behavior of the scientific workflow. Finally, section 2.4 explains the concepts of mathematical programming and the model used in this thesis.

2.1 Cloud Computing

Since its inception, the Internet has undergone many changes, some of which have had a positive impact on human lifestyle, including cloud computing. This new computing paradigm has quickly gained public attention due to its features because all kinds of facilities are provided to users as a service. This technology is designed for easy use of information resources, hardware, applications, and networks, and its approach is that you no longer need to have your computing infrastructure with high processing power [5, 6]. Instead, in cloud computing, the IT resources are delivered to the user in a flexible and scalable way via the Internet based on the user's demand and based on a "PAY-AS-YOU-GO" premise [7]. Numerous different definitions of cloud computing have been

proposed in recent years in both academia and industry. They all define cloud computing in some way, but perhaps the most accurate definition is related to the National Institute of Standards and Technology (NIST). According to the NIST: "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. This cloud model is composed of five essential characteristics, three service models, and four deployment models" [8]. This definition includes cloud architectures, service models, and deployment strategies. In the following, these characteristics and models will be described.

2.1.1 Essential Characteristics

In Figure 2.1, five essential characteristics of cloud computing can be seen. In the following, these characteristics will be described

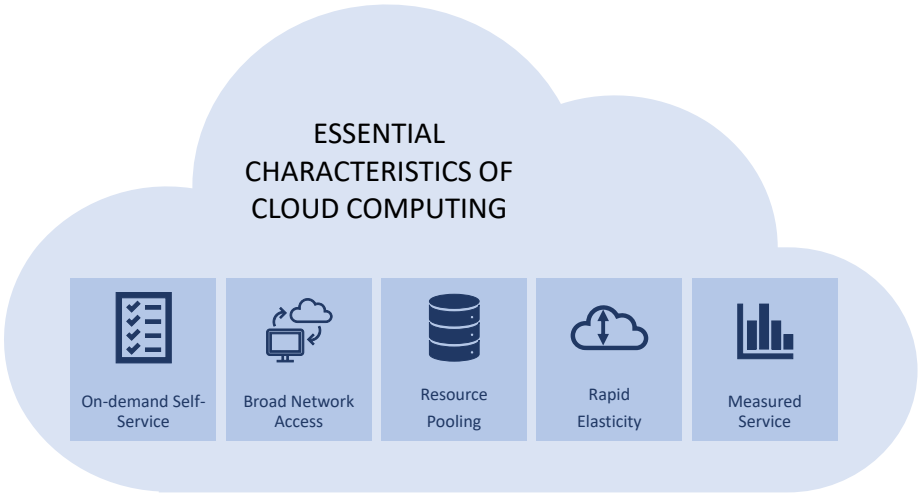


Figure 2.1: Cloud Computing Essential Characteristics

On-demand self-service: This means that a consumer with an urgent need, at a given time, can use computing resources (such as CPU, Memory, Storage, Network, software use, etc.) without resorting to human interactions, and it will be done automatically (i.e., convenient, self-service).

Broad network access: This means that cloud computing services and resources are provided and delivered to the different client platforms (e.g., mobile phones, tablets, laptops, PDA, and workstations) over the network, specifically the Internet, and through specific mechanisms and protocols.

Resource pooling: A cloud provider's computing resources are pooled to provide services to multiple consumers using a multi-tenancy model, "with different physical and virtual resources that are dynamically assigned and allocated to the consumer according to its demand." [8]. The main motivation of the resource pooling model is to create a location independence mode in which the consumer will not have any control and knowledge of the location of the provided resources. Of course, the consumer will be able to determine the location of resources at a high level (e.g., country, state, or data center).

Rapid elasticity: This means that the provided resources are flexible, and resources are allocated or released according to the consumer's demand. This feature allows the consumers to scale up their demanded resources whenever they want and release them once they finish scaling down. Also, from the consumer's perspective, resources can be provided unlimited, in any quantity, and at any time.

Measured Service: Computing resources can be pooled and shared by multiple consumers. In the meantime, the cloud providers can utilize appreciative mechanisms to calculate, monitor, and report the usage of computing resources for each consumer.

2.1.2 Service Models

In addition to the five essential characteristics mentioned regarding cloud computing, there are three service models that will be described below.

Software as a Services (SaaS): In this service model, cloud providers give consumers the ability to use the provider's software applications which are hosted and running on the cloud infrastructure. The consumer pays for it through the PAYG model if there is a cost. These applications are generally available to consumers through web browsers or through application interfaces, in which case consumers will have no control over the infrastructure used by those software applications.

Platform as a Services (PaaS): This service model allows consumers to deploy their own created applications on cloud infrastructure and the cloud provider has tools that support the entire "Software Lifecycle" and deployment process. However, in this model, the consumers have no control over the cloud infrastructures and can only manage and configure the necessary configurations for their applications.

Infrastructure as a Services (IaaS): IaaS is a solution of providing cloud computing infrastructures such as servers, storage, networks, and operating

systems (usually in terms of virtual machines). This infrastructure provides virtualized computing resources on the network as a requested service to the user or consumer. Instead of buying servers, software, data center space, or network equipment, consumers buy those cloud resources and infrastructure as a completely outsourced service. Thus, IaaS refers to online services that abstract the consumer from the details of infrastructure such as physical computing resources, location, data partitioning, scaling, security, backup, etc.

Although other service models have been introduced in the last decade and recent years (such as Database as a Service (DaaS), Containers as a Service (CaaS), Function as a Service (FaaS), etc.), these models, as mentioned earlier are the leading service models in cloud computing.

2.1.3 Deployment Models

According to the NIST definition, there are four types of deployment models. Figure 2.2 illustrates these four models, and in the following, each of them will be briefly described.

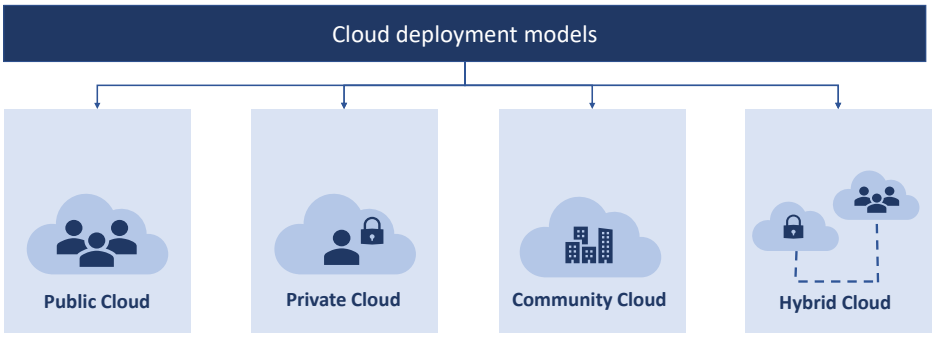


Figure 2.2: Cloud Computing Deployment Models

Private Cloud: In this deployment model, the cloud infrastructure is provided exclusively by a single organization with multiple consumers. The monopoly of these infrastructures may be in the hands of an organization, a third party, or a combination, whether it is located on-premise or off-premise.

Community Cloud: In this type of deployment model, several organizations that form a community jointly share the cloud infrastructure among themselves. These organizations generally have common concerns (e.g., security requirements, mission, policy, and compliance considerations). Cloud infrastructure can be hosted by a third party or one of the existing organizations in the community.

Public Cloud: This deployment model is the predominant and pervasive form of the current cloud computing. The public cloud is open to use by general public consumers. The cloud service provider has complete control and ownership of the public cloud with its concerns, security requirements, mission, policy, compliance considerations, profit, and charging model. There are numerous public cloud providers, including Amazon EC2, Google CE, Azure, etc.

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) [8].

2.1.4 Virtual Machines

A virtual machine (VM) is defined as "a copy of a real, physical machine, efficient and isolated." A VM software can run various operating systems and applications, connect to the network, store data, and perform other computational functions. It also needs maintenance, such as system updates and monitoring. Multiple virtual machines can be hosted on a single physical machine, often a server, and then managed using virtual machine software (Hypervisor) [9]. Figure 2.3 illustrate the simplified diagram of VM on a server. It provides flexibility for computing resources to be distributed among VMs as needed.

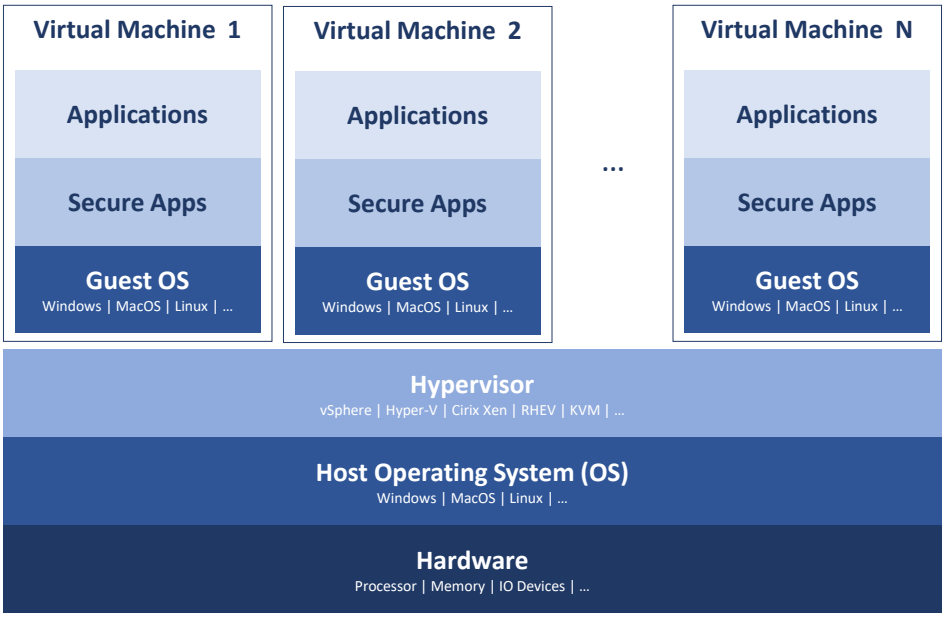


Figure 2.3: Virtual Machine in Cloud computing Diagram

There are many benefits to using a virtual machine in cloud providers, such as Lower hardware costs, Quicker Desktop Provisioning and Deployment, Smaller Footprint, Enhanced Data Security, Portability, and Improved IT Efficiency. Cloud providers define different types of these virtual machines according to the needs of consumers and clients, which are used for different applications. In other words, cloud providers have different kinds of predefined virtual machines with different specifications (e.g., CPU, Memory, Storage, Bandwidth, etc.), which are known as *Instances*. Cloud providers generally classified their instances into different groups, including General Purpose, Compute Optimized, Memory Optimized, Storage Optimized, Accelerated Computing, etc. According to its specifications, each classified instance usually has a different price. These prices are generally calculated on an hourly basis. This information, including the specifications of each instance and its price, is available on the site of each cloud provider. In addition, each instance in the cloud environment needs a specific time to prepare for the execution, which is called *Provisioning Time*. In other words, provision time is the time required to configure and be fully ready o use an instance. This time will vary depending on the instance's specification and may take 30 seconds to 10 minutes.

2.1.5 Virtual CPU (vCPU)

Virtual CPU (vCPU) is a new concept used in cloud providers to denote a virtual machine's processing power and performance. Indeed, a vCPU represents a portion or share of the underlying physical CPU assigned to a particular virtual machine. Today's CPUs support multi-core, and each CPU core can have two threads. Threads are essential to the computer's functionality because they determine how many tasks the computer can perform at any given time. Cloud providers consider each vCPU equal to one thread in the CPU core. So when it is said that a VM has two vCPU, we are dealing with a maximum of one CPU core and not the entire CPU. For example, the Intel® Xeon® E-2374G Processor has four cores and eight threads, which can have a total of 8 vCPU. This concept is shown in Figure 2.4.

As can be seen in Figure 2.4, a physical CPU with four cores can have a maximum of 8 vCPU. In other words, if we have a virtual machine with 8-vCPU, it means that in practice, we deal with a 4-core CPU. If we want to formulate these explanations, we come to the following relation:

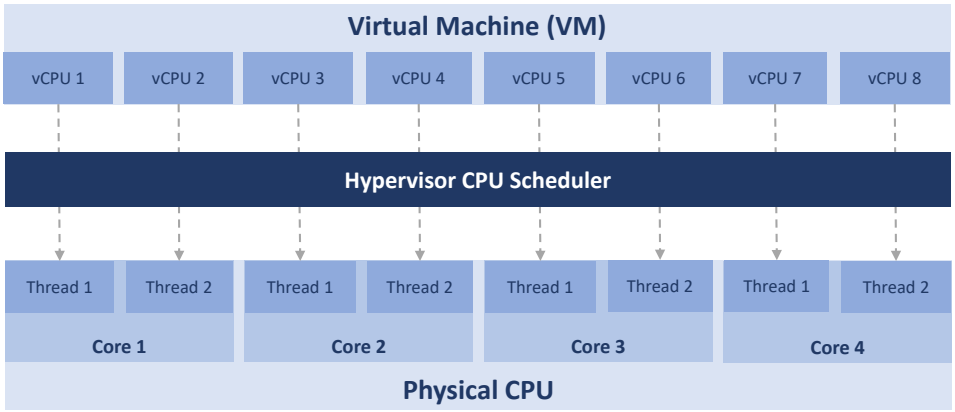


Figure 2.4: A VM with 8 vCPUs on a CPU that has 4 cores with hyper-threading enabled

2.2 Scientific Workflow Model

Scientific workflows are collections of several structured activities and fine-grained computational tasks that enable data analysis in a structured and distributed manner. Indeed, scientific workflows containing a set of computational tasks linked via control and data dependencies between them. For example, figure 2.5 represents a sample workflow with nine tasks where the output data generated by one task becomes the input data for the next one. This means that the child's task cannot be performed until the parent's task(s) are completed.

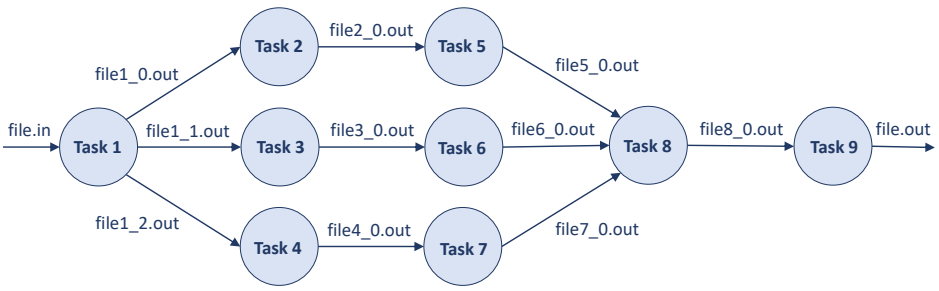


Figure 2.5: Sample workflow with nine tasks. The nodes represent computational tasks and the edges the data dependencies between these tasks

A scientific workflow application is modeled as a DAG (Directed Acyclic Graph); that is, graphs with directed edges and no cycles or conditional dependencies. Therefore, a workflow W can be defined as pair $G = (T, E)$, where T is a set of tasks $T=\{T_1, T_2, \ldots, T_n\}$, and E is a set of directed edges. An edge $e_{ij}=(t_i,t_j)$ corresponds to a dependence constraint between task t_i and t_j, in which t_i is

an immediate parent task of t_j, and t_j the immediate child task of t_i.

There are several examples of scientific workflows applications in different domains, such as Montage in Astronomy, Cybershake in Earthquake science, Epigenomics, and SIPHT in Bioinformatics LIGO in Astrophysics). In this thesis, these workflows have been used to evaluate the proposed scheduling algorithm. Figure 2.6 shows the structure of these five scientific workflows, extracted from [10]; each with a specific dependency pattern between tasks and their full characterization.

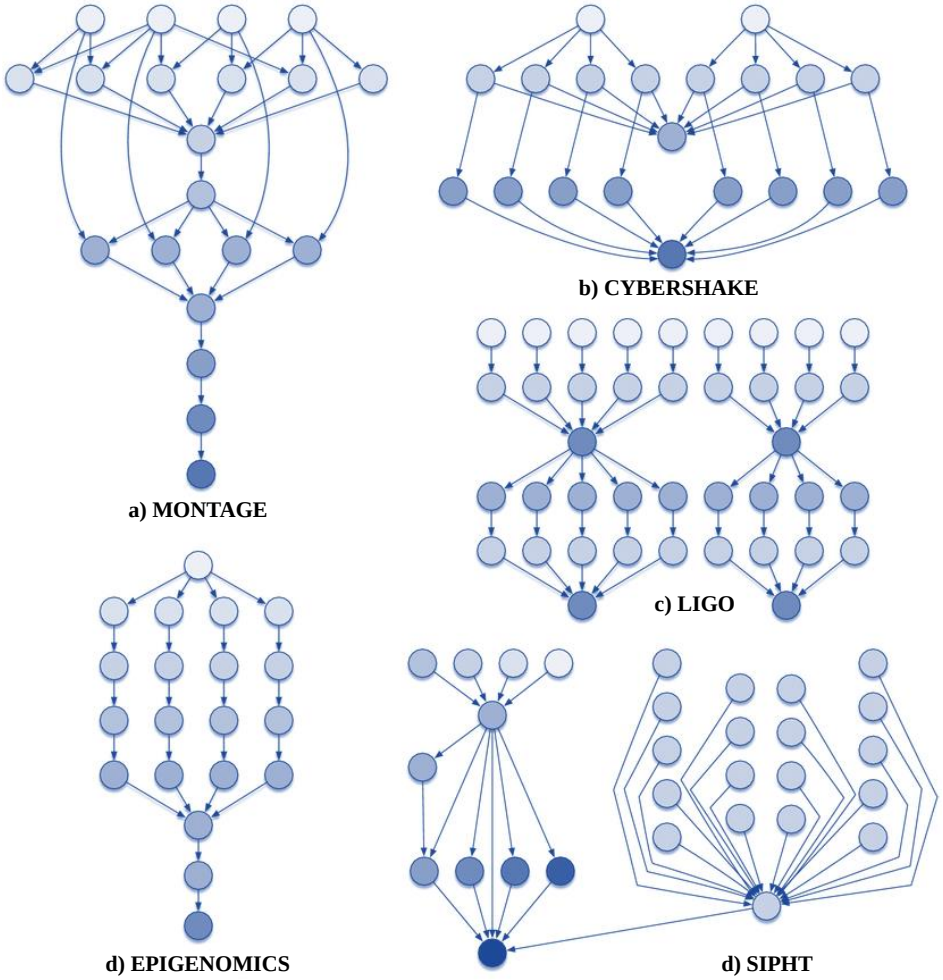


Figure 2.6: Structure of five well-known scientific workflows: Montage, Cybershake, LIGO, Epigenomics, and SIPHT

2.2.1 Bag of Tasks (BoT)

Ideally, scheduling in cloud computing aims to determine an optimal assignment of different application tasks to VMs in different clouds. However, in real-world problems, achieving such an assignment is considerably time-consuming or impossible. This challenge can be addressed by partitioning tasks and the subsequent scheduling of the partitions. Motivated by this idea, we first set tasks of the same level in the DAG graph in the same partition. Because all the tasks in a partition are the same type, we regarded them as constituting a Bag of Tasks (BoT).

Given that no data dependency exists between the tasks in a bag, they can be executed parallelly. Such partitioning enabled us to consider a few bags instead of many tasks individually. Figure 2.7 illustrates such partitioning for the Cybershake workflow. The structure of the workflow is described as a pipeline consisting of several levels, each having a bag.

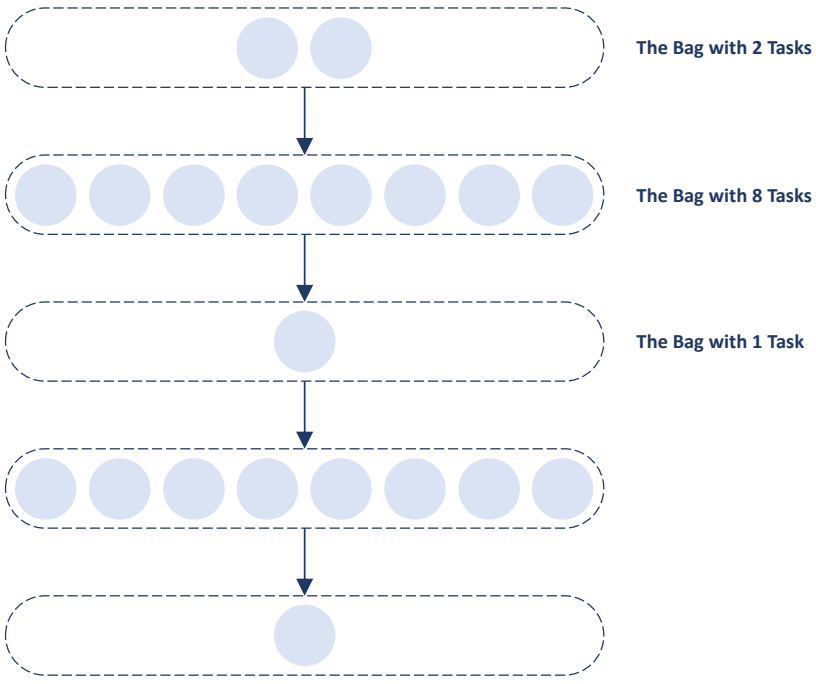


Figure 2.7: Level partitioning for Cybershake workflow

As shown in Figure 2.7, with the partitioning, a DAG workflow is converted to a pipeline workflow. This reduces the computational complexity and increases the comprehensibility of the problem.

2.3 Scheduling

Scheduling is an action that assigns the available resources to different tasks to execute. Here, "Task" can be computing elements (e.g., thread, process, or dataflow), which are scheduled to hardware resources such as the CPU, network connection, etc. Scheduling in cloud environments is similar to this definition. However, the difference is that we are dealing with virtual machines in cloud environments instead of hardware. In other words, Scheduling in the cloud also means allocating virtual machines to perform various tasks. Workflow scheduling approaches are generally implemented through two methods: Static and Dynamic Scheduling. These two will be explained in the following.

2.3.1 Static Scheduling

In static scheduling, we have complete control over the resource allocation, and the assignment of tasks is performed before starting the scheduling process. It means that all information (e.g., workflow structure, number of tasks, the execution time of each task, IO data size), as well as resources information (e.g., VM instances types, VM's vCPU, VM's price, bandwidth, etc.), is accurately and wholly obtained before the scheduling [11]. Studies showed that static scheduling outperforms dynamic scheduling. Furthermore, because of the vital information obtained before scheduling, static scheduling mechanisms perform better in terms of speed and performance. [12].

2.3.2 Dynamic Scheduling

Unlike static scheduling, dynamic scheduling mechanisms do not have complete information about resources, and this information will be collected and updated during scheduling [11]. In this kind of algorithm, two main challenges somehow increase the complexity of these algorithms, and their performance might be reduced accordingly. The first challenge is that the new tasks may arrive at any time, unpredictably. It means that the previous schedule needs to be updated. It is a vital challenge in dynamic scheduling approaches. The second challenge is that a resource needs some setup or re-configuration time to handle and execute different tasks.

2.4 Mathematical Programming (MP)

This section will overview the steps of a mathematical procedure and illustrate the general structure of mathematical models. Mathematical optimization or

mathematical programming[1] have application in many fields (e.g., mathematics, economics, management, science, and engineering) which refers to choosing the best member from a set of achievable members. In the simplest form, an attempt is made to obtain the maximum, and minimum values of an objective function by systematically selecting data from an achievable set and calculating the value of an objective function [13]. Mathematical Programming includes different classes, and each of them has its utilization and application. These classifications include Linear programming, Nonlinear Programming, Quadratic Programming, and Network Flow Programming.

In general, a mathematical procedure can proceed as follows if we want to express it orally. First, a simple form of a real problem is constructed by disregarding unnecessary details. Second, based on this simple form, a mathematical model is derived. Third, the obtained model is solved. Since this thesis utilizes the linear programming, therefore, in the following, we will describe it.

A linear programming model can generally be expressed as follow [14]:

$$\begin{aligned} max/min \quad & f\big(c_1x_1 + \ldots + c_nx_n\big) \\ subject\ to: \quad & \\ & g_i(x)\{\geq or \leq or =\}b_i, \quad i = 1, \ldots, m \\ & x := (x_1, \ldots, x_n) \in X \subseteq \{\mathbb{R}\} \end{aligned} \tag{2.1}$$

In Model (2.1), $x_1, \ldots, x_n$ are called decision variables, and f and $gi_{(i=1,\ldots,m)}$ are real valued mathematical functions of x. Constants $bi_{(i=1,\ldots,m)}$ are right hand side values which may have different interpretations (such as, the level of demands, resources,. . .) in different problems.

In model (2.1), the objective function f represents the value of cost or profit of the problem concerning the decision variables. The set of m inequalities or equalities are constraints of the model which represent the practical or physical requirements of the problem. The set X denotes some restrictions on decision variables such as a sign, divisibility, and discreteness of variables. Here, we consider a special type of Model (2.1), where f and $gi_{(i=1,\ldots,m)}$ are linear functions and $xi_{(i=1,\ldots,m)}$ are integer or binary variables ($0-1$ variables). One type of the linear programming is Mixed-Integer Linear Programming (MILP), which has been used to model our scheduling algorithm. Therefore, this model will be described below.

1. Mathematical programming is very different from computer programming. Mathematical programming is 'programming' in the sense of 'planning'

2.4.1 Mixed-Integer Linear Programming

Mixed Integer Linear Programming (MILP) is a subset of mathematical programming whose problems are similar to linear programming, except that some decision variables are integers and some of them are continuous. Like linear programming, Mixed-Integer Linear Programming aims to find the minimum or maximum value of a linear function on a space with linear constraints. The general structure of a Mixed-Integer Linear Programming model is as follows:

$$\begin{aligned} max/min \quad & c_1x_1 + \ldots + c_nx_n \\ subject\ to: \quad & \\ & a_{11}x_1 + \ldots + a_{1n}x_n \{\geq or \leq or =\} b_1 \\ & \vdots \qquad\qquad\qquad\qquad \vdots \\ & a_{m1}x_1 + \ldots + a_{mn}x_n \{\geq or \leq or =\} b_m \\ & x_1, \ldots, x_n \in \{\mathbb{R}, \mathbb{Z}, \{0,1\}\} \end{aligned} \tag{2.2}$$

Where $\mathbb{R}$ denotes the set of Real numbers, and $\mathbb{Z}$ denotes the set of Integer numbers.

3

Related Work

In this section, we will have an overview of research and work done in the field of scheduling scientific workflow in the cloud environment. Over the past two decades, many researchers have addressed the problem of scheduling scientific workflow applications [1, 12] in distributed environments such as Grid and Utlity computing. Since scheduling workflow applications is always considered an NP-hard problem, and on the other hand, the emergence of cloud computing, workflow application scheduling problems in cloud environments have received significant attention. Different algorithms and approaches to scheduling workflow applications in the cloud environment each have different goals. These goals include minimizing the total cost, minimizing the total makespan (execution time), optimizing energy consumption, etc.

As mentioned earlier, optimization-based scheduling approaches are classified into optimal and near-optimal solutions [4]. Approaches to finding near-optimal solutions for workflow scheduling problems mostly use heuristic and meta-heuristic methods, while, valuable approaches are provided based on mathematical programming that can find optimal solutions. Also, different scheduling approaches, whether optimal or near-optimal, can have a single objective or multi-objective [12]. Figure 3.1 depicts an overview of this category.

Scheduling of workflow applications with cost minimization objective in cloud has been addressed in [15, 16, 17, 18, 19, 20, 21], whereas in [22, 23, 24, 25, 26, 27], the makespan minimization is considered as objective of scheduling. Scheduling

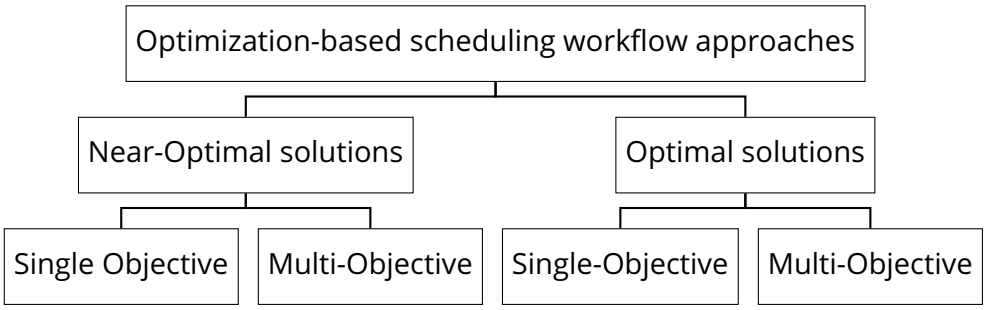


Figure 3.1: The chart of the optimization-based scheduling workflow categories

approaches in [28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38] are multi-objective which most of them consider both makespan and cost as scheduling objectives. Many workflow scheduling approaches consider a single public provider for provisioning required resources in cloud environment. Because of increasing the demand of computational resources for executing large-scale workflow applications and raising inter-cloud, which provide scalable services, the problem of workflow applications scheduling in inter-cloud models (federated cloud and multi-cloud) has been considered in recent years [17, 28, 29, 39, 40, 41, 42, 43, 44]. This research addresses the problem of workflow applications scheduling in multi-cloud environments. Some workflow applications are known as bag-of-tasks (BoT) applications which consist of a set of concurrent bags; each includes a large number of independent homogeneous tasks [45]. The problem of scheduling for BoT workflow applications in cloud environment has been addressed in [39, 45, 46, 47, 48, 49]. In the following, we will review and compare some of them with our proposed approach. Our proposed approach addresses multi-cloud environments and uses a single objective and the objective is to minimize makespan under budget constraints.

Zhu et al. [35] used multi-objective solution and applied Genetic Algorithm to minimize both cost and makespan for running a workflow in a single cloud provider. Their approach, in addition to using the near-optimal solution, is also only offered to run in a single cloud provider. While our proposed approach uses the deterministic optimal solution method and is also designed and modeled for multi-cloud environment. The solution presented in [50] adopted Particle Swarm Optimization (PSO) algorithm to scheduling scientific workflows with objective to minimize the cost under the deadline constrained. This solution is also a near-optimal solution and is modeled and designed for a single cloud environment. Talukder et al. [37] have been proposed an approach that uses differential evolution to make a trade-offs between cost and makespan. Their work is provided for running in the grid environment while the our proposed method is designed for working in cloud environments which have more benefits than Grid. The algorithm presented in [51] is a

heuristic based approach that try to minimize makespan while meet the budget constraint. The algorithm distributes the budget to all tasks in proportion to their average execution time on the available resources. Then, the resources chosen in one that allows the tasks to be accomplished early at a cost not greater than its allocated sub-budget. In our approach we use pipeline level and try to minimize makespan while meet budget. But in [51], the distribution budget for each task is a restrictive constraint that leads to increase in makespan. Moschakis et al. [49] proposed a multi-objective heuristic-based approach for multi-clouds. They adapted simulated annealing algorithm for scheduling of BoT with the performance and cost optimization objectives. Their proposed method is completely different from our model. In addition, they did not consider data transfer time and cost for consecutive bags. While, our model consider the data transfer cost and time of data transfer.

The works presented in [17, 29, 52, 39, 53, 40, 54] are optimal-based algorithms that adapt mathematical programming for their approaches. Malawski et al.[17] proposed an approach that is single-objective scheduling approach and it is modeling based on mathematical programming and also designed for multi-cloud environments. The main objective of that approach is to minimizes makespan under a deadline constraint. While our proposed model aims to reduce makespan and meet the budget constraint. Coutinho et al. [29] introduced a workflow scheduling algorithm that is multi-objective approach and try to minimize both cost and makespan. Our proposed model minimizes makespan and meanwhile fulfills the user-budget. Similar to us, they adapt multi-cloud (federated cloud) environment, but they also do not consider the data transfer time and cost. In addition they do not consider the diversity in instance types VMs. The VMs they used for modeling are single-core CPU virtual machines. However, our proposed model in addition to considering data transferring time and cost, supports the concept of segmenting and running on VMs with multi-vCPU. Abdi et al. [39] have proposed an optimization-based scheduling algorithm for hybrid-clouds environments. Their main objective is to minimize the execution cost while meeting deadline constraints. In contrast, our model tries to reduce the makespan under budget constraint.

Rodriguez et al. [54] proposed a scheduling algorithm whose objective is to optimize a workflow's execution time under a budget constraint. Their approach focuses on finer-grained pricing schemes that provide users with more flexibility and the ability to reduce the inherent wastage that results from coarser-grained ones. Their approach distributes the budget over the tasks. In some respects, this is similar to our approach (except for the finer pricing schemes and the budget distribution between tasks). In some respects, this is similar to our approach (except for the finer pricing schemes and the budget distribution between tasks). However, there are other differences. Their approach is designed to work in a single cloud environment with homogeneous

VMs instance types, while our approach is designed for working in multi-cloud environments and considering heterogeneous VM instance types.

In addition to the methods mentioned that are optimization-based approaches, we can also mention solutions that not optimization based methods. Jaikar et al [55] presented scheduling workflow approach for multi-cloud environments. It uses Matrix-based algorithm to select resources based on their distance, Cost, and Performance. Their approach, similar to our solution, adapt multi-cloud (federated cloud) environment, but they do not consider the data transfer time and cost. In addition they do not consider the diversity in instance types. While our proposed approach considers both instance type diversity and data transfer costs and time between cloud providers. Wylie et al. [56] provided a greedy approach for scheduling workflow applications on Hadoop clusters. The objective of this work is to reduce the makespan under budget constraint. This approach, in principle, only considers the execution time of the task on the VM while ignoring the data transfer time between tasks.

4

System Design and Models

This section introduces the different elements and concepts of the workflow scheduling approach adopted in this thesis. These elements are the architecture, the infrastructure model, the workflow partitioning, modeling of running tasks on VM, communication model, and scheduling approach.

4.1 Infrastructure model

In a multi-cloud environment, clients or their brokers are responsible for managing resources across multiple clouds[57]. The broker, which acts as middleware or intermediate software system, receives the request from the user or client and distributes it among different cloud IaaS resources for execution. Given that there are numerous computing resources with different specifications and prices in multi-cloud environments, managing and choosing between these resources is very difficult for the user or customer and, in some cases, very time-consuming and tedious. Therefore, it seems necessary to use an intermediate software responsible for dividing tasks within the scientific workflow and providing resources. Figure 4.1 illustrates a general concept of our scheduling approach, which is composed of a scientific workflow application, several IaaS cloud providers constituting a multi-cloud environment, and a broker who serves as the scheduler.

In Figure 4.1, first, the user submits a request to the broker to schedule the

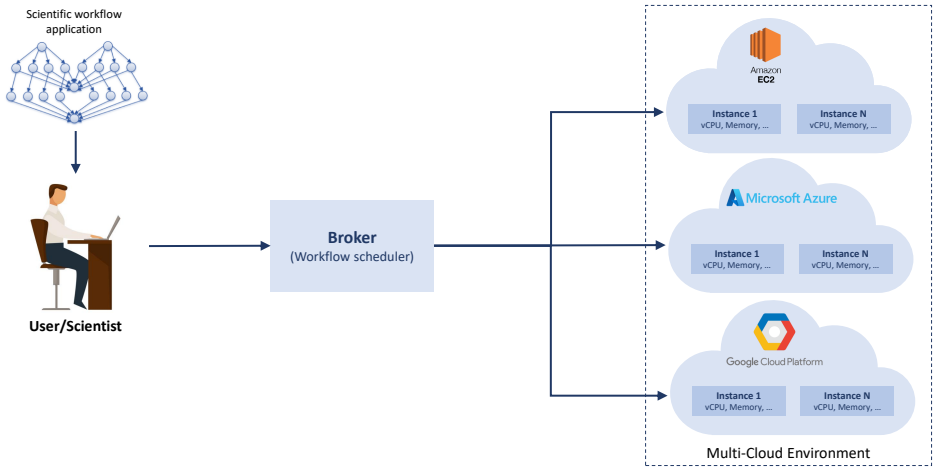


Figure 4.1: General overview of scheduling approach

scientific workflow application. This request contains information about the scientific workflow (e.g., budget, tasks execution time). After receiving the user request, the broker should complete this information with its data related to the instance types and IaaS cloud providers (e.g., price and performance, bandwidth between clouds, cost of data transfer to a specific cloud provider). After combining the information, the broker attempts to select appropriate resources for workflow scheduling so that the request is fulfilled and the objective is optimized.

In this thesis, three IaaS cloud providers are regarded as available providers in a multi-cloud environment. Nevertheless, it is not limited to these three cloud providers and can be multiple providers. These three cloud providers are Amazon EC2[1], Microsoft Azure, and Google CE[2] that are used for infrastructure modeling. Table 4.1 shows information (e.g., Price per hour, vCPU number, and related Memory) about these cloud providers. As can be seen in the table 4.1, four instance types per cloud provider have been selected. In order to provide a homogeneous multi-cloud environment, instance's specifications are considered common in terms of price and performance. Instances are all General-purpose types. In addition, the operating system of all instances is Linux Ubuntu. On the other hand, to reduce the volume of data transfer between cloud providers, we chose the location of all instances the same. The location of all instances is London, UK.

To assess the performance of each instance type VM, we used the vCPU metric. For this purpose, the clock speed of all instance processors is 2.5 GHz. On the

1. Elastic compute cloud
2. Google Computing Engine

Cloud	Instance Type	vCPU	Memory (GiB)	Price ($)/h
Amazon EC2	m5a.large	2	8	0.100
	m5a.xlarge	4	16	0.200
	m5a.2xlarge	8	32	0.400
	m5d.4xlarge	16	64	1.048
Microsoft Azure	D2as-v5	2	8	0.104
	D4as-v5	4	16	0.208
	D8as-v5	8	32	0.416
	D16-v5	16	64	0.929
Google CE	e2-standard-2	2	8	0.086
	e2-standard-4	4	16	0.173
	n2-standard-8	8	32	0.500
	n2-standard-16	16	64	1.000

Table 4.1: Instance types characterization provided by different clouds

other hand, the performance of an instance type VM is related to both processor and memory. Hence we keep the ratio between CPU and memory constant. This means that if the vCPU is doubled, the memory capacity is also doubled. This is clearly visible in table 4.1. With this definition, it is easy to conclude that the efficiency of an instance will double as the vCPU doubles.

According to the description, as mentioned above, cloud providers and their instances types can be modeled as follow. In a multi-cloud environment the cloud providers (CP) modeled as a set $CP=\{CP_1, CP_2, \ldots, CP_k\}$, which the $CP_k(k = 1, 2, \ldots)$ in CP corresponds to the set of the Cloud Providers and denotes the kth provider in CP set. Each cloud provider offers a range of instance types with different prices and configurations. Therefore, the instance types of cloud providers can be modeled as follows: $I=\{I_{11}, I_{12}, \ldots, I_{kj}\}$ which the I_{Kj} signify jth instance type of the kth cloud provider. On the other hand, the cost (price) of using each instance type per hour is also indicated by $Price_{kj}$ where k denotes the kth cloud provider, and j signifies the jth instance type in the CP_k.

Given that the transfer of processed tasks' data from one cloud provider to another is inevitable in the multi-cloud environment, the bandwidth between cloud providers is an essential factor in decision making. Because more band-

width will increase the data transfer rate, which reduces data transfer time. On the other hand, reducing data transfer time will reduce the usage time of an instance type, and using less of an instance type will reduce the cost of using it and ultimately leads to the reduction of the consumption budget. We show the average bandwidth between two cloud providers with $BW_{k,k'}$ where k and k' indicates two different cloud provider. To measure the average bandwidth between different IaaS cloud providers, the iPerf3[3] tool was used. iPerf3 is a tool for active measuring the maximum achievable bandwidth on two IP networks. It supports tuning various parameters related to buffers, timing, and protocols (e.g., SCTP, UDP, TCP) by supporting IPv4 and IPv6. iPerf reports the bandwidth, loss, and other parameters for each test it performs [58]. We installed and deployed the iperf3 tool on each of our cloud providers (Amazon EC2, Microsoft Azure, Google CE) and then, by sending test packages, from one provider to another, obtained the average bandwidth between each cloud provider. Table 4.2 shows the obtained average bandwidth between our selected cloud providers.

Cloud	Amazon EC2	Microsoft Azure	Google CE
Amazon EC2	0	3.230	3.290
Microsoft Azure	3.120	0	3.094
Google CE	3.650	3.300	0

Table 4.2: Bandwidth between different cloud providers (in Gigabyte)

In general, this table is more similar to the Hollow matrix. As a rule, as the number of cloud providers increases, so will the matrix size increases. However, data transfer to a cloud provider may also be costly. Cloud providers charge a fee for transferring data from another provider to themselves. This fee, usually, is per 1 GB for data transfer. We have indicated this fee, the cost of transferring data (CD) to a cloud provider, by the CD_k, which k denotes the kth cloud provider.

4.2 Workflows partitioning model

As stated in section 2.2.1, partitioning a scientific workflow into a few bags of tasks (BoT), can help to understand the problem better as well as reduce the complexity of the scheduling algorithm. The main reason for workflow partitioning is to reduce the time and cost of data transfer between different tasks in a workflow. According to the Figure 2.7, converting a workflow from

3. `https://iperf.fr/`

a DAG structure to the pipelines of BoT can be helpful in some ways. On the other hand, partitioning into pipelines of bags, if properly managed, can enhance the performance of the scheduling algorithm. Because processing a pipelined workflow is much easier and less expensive, from the data transfer time and cost perspective, than processing a DAG workflow. In addition, it should be noted that partitioning can not be applied to all types of scientific workflows. This means that some scientific workflows with homogeneous tasks, with no data interdependence between them, can be easily partitioned into the BoT workflow. In contrast, many scientific workflows can not be partitioned. Therefore, the main target workflows of this thesis are those workflows that can be partitioned into BoT.

A scientific workflow can include several bag of tasks that can be modeled as $B=\{B_1, B_2, \ldots, B_i\}$ which the B_i indicates the ith bag in a workflow. Each bag can also contain one or more tasks. Therefore, the number of tasks in each bag can be represented by the λ_i symbol, Where the i represents the bag index. Our scheduling model assumes that all the tasks in a bag are assigned to the same cloud for scheduling. In this way, all tasks in a bag allocated to an IaaS cloud can be run simultaneously and in parallel mode. However, since the number of tasks in a bag can be large, running all tasks simultaneously in parallel mode is not reasonable and will increase the processing time of a bag. Thus, in this thesis, we have defined a criterion that divides a bag into smaller units. We called these smaller units the segments.

These segments should be run sequentially on a VM and not parallel. Nevertheless, the tasks within each segment are executed in parallel. A bag can consist of one or more segments, which are modeled as $S=\{S_{1i}, S_{2i}, \ldots, S_{li}\}$ where l denotes the lth segment in a bag and i signify the bag index. We also show the number of segments (segment's count) in a bag with the η_i. In other words, and mathematical terms, the number of segments in a bag can be represented as $|B_i| = \eta_i$. On the other hand, each segment can also include one or more tasks that are modeled as $T=\{T_{1li}, T_{2li}, \ldots, T_{nli}\}$ where i denotes the bag index, l indicates the segment index and n signify the task index in the workflow. Each task has an execution time, which in this thesis, we will show with E_{nli}^{task}. In addition, each segment will have its own execution time, which is equal to the execution time of the most data-intensive task within that segment. We represent this criterion as $E_{li}^{segment}$ where l signify the index of segment of ith bag.

Figure 4.2 shows a simple schematic of a segment with 8 tasks. Each task also has its own execution time, which is displayed in E_{nli}^{task}. According to the explanation, the execution time of this segment is equal to the execution time of the most data-intensive task within that segment, which is task 3.

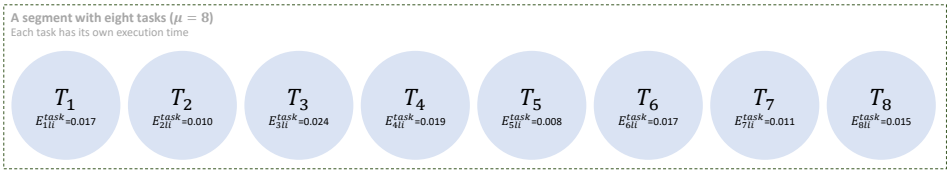


Figure 4.2: Simple schematic of a segment within a bag

Once the processing of all the segments within a bag is completed, the data generated by the tasks within the segments of that bag must be transferred to the next bag. From this point of view, each bag has a specific data size. We have shown this Bag Data (BD) size with BD_i, which i indicates the ith bag in the workflow. However, the crucial point is how many tasks should be placed within a segment? Therefore, we defined a parameter called segment size, μ, which determines exactly how many tasks should place into a segment. This parameter must be specified before scheduling. In a way, this parameter can directly affect reducing or increasing the makespan and consequently impact the execution cost. This parameter will be further discussed in the section 6 (Evaluation).

4.3 Model of running tasks on a VM

In this section, the details of modeling of running tasks on a VM will be discussed. As mentioned earlier, we use the concept of a segment, in which each segment contains one or more tasks. So instead of considering the execution time of each task, we consider the execution time of a segment. The computation time of tasks within a segment may differ as the size of their input and output data may vary. For this reason, we assume all tasks in the segment take as long to process as the most data-intensive task. That is the task that uses and produces the most amount of data out of all the ones in the segment. Equation 4.4 is the mathematical expression of the above explanations.

$$E_{li}^{segment} = max\{E_{1li}^{task}, E_{2li}^{task}, \ldots, E_{nli}^{task}\} \tag{4.1}$$

It should be noted that the execution times of each task, E_{nli}^{task}, are based on a VM with 2-vCPU. Therefore, the execution time of each segment is also based on VM with 2-vCPU. As mentioned in subsection 2.1.5, most current virtual machines support the vCPU concept on cloud providers. A VM may have one or more vCPU, so in this section, we model the execution of a bag's segments on a VM. Wu [59] shows that if several tasks are executed on VMs, with different vCPU, then the execution time of the tasks is approximately equal. The only difference is that the execution time of tasks on a VM with 1-vCPU is twice times

as long as that on a VM with 2-vCPU, and likewise four times on a VM with 4-vCPU. Therefore, it can be concluded that each running task on a VM will get the same CPU time, and it is possible to run multiple tasks simultaneously on a VM. Hence, running a segment instead of running tasks individually can increase the VM's CPU utilization and reduce the overall execution time of a bag.

We present the number of vCPU of a VM as vCPU_{kj} where kj indicates the jth VM of the kth cloud provider. In order to select the appropriate VM to run a bag with one or more segments, we need to calculate the processing power of different VM. In other words, there must be a criterion that can compare different VM in terms of processing power and speed. For this reason, we are inspired by the idea presented in [59] for the use of calculating the power of a VM. We show the processing power of a VM to execute a segment as $power^{segment}_{likj}$, which is calculated based on Equation 4.2:

$$power^{segment}_{likj} = \begin{cases} \frac{vCPU_{kj}}{\delta_{li}} & if(\delta_{li} > vCPU_{kj}) \\ 1 & otherwise. \end{cases} \tag{4.2}$$

Where δ_{li} is the count of tasks within segment l of $i$the bag. This criterion, $power^{segment}_{likj}$, indicates that if the number of tasks within a segment is equal to or less than the number of vCPUs within a VM, all tasks within a segment will be completed at the same time. Therefore, in such a case, the processing power of the VM will be considered equal to 1. However, if the number of tasks within a segment is greater than the number of VM's vCPU, then the processing power of that VM will be equal to the number of vCPU on the number of tasks within the segment. After calculating and obtaining the processing power of a VM for running segments of a particular bag, now it is time to calculate the execution time of a segment on an instance type VM. We illustrate this by τ_{likj} which is calculated as follows:

$$\tau_{likj} = \frac{E^{segment}_{li}}{power^{segment}_{likj}} \tag{4.3}$$

Therefore, after obtaining and calculating the execution time of a segment on a VM, it is now possible to calculate the execution time of a bag on a VM. The execution time of a bag is equal to the most data-intensive segment within that bag and calculated as follow:

$$E^{bag}_{ikj} = max\{\tau_{1ikj}, \tau_{2ikj}, \ldots, \tau_{likj}\} \tag{4.4}$$

where τ_{likj} indicates the execution time of the lth segment in the B_i on instance type I_{kj}.

4.4 Communication model

This section will model the cost and time of communication between different bags in different cloud providers. As mentioned earlier, a workflow consists of one or more bags, and each bag is assigned to a cloud provider. On the other hand, we know that the output data of one bag will be the input data of the next bag. So if two consecutive bags run on two different cloud providers, we will have the data transfer cost. We show this data transfer cost as $Comm^{cost}_{i,i-1,k,k'}$ which obtain as follow:

$$Comm^{cost}_{i,i-1,k,k'} = \frac{BD_i}{CD_k} \tag{4.5}$$

Where $i, i-1$ represent two consecutive bags, and k, k' represent two different cloud providers. In equation 4.5, BD_i indicates the size of an ith bag, and CD_k denotes the cost of transferring data to the kth cloud provider.

In communication modeling and the cost of data transfer, transfer time is also an important factor. Transfer time is influential in optimizing or reducing the overall execution time of the workflow. Therefore, we will model the data transfer time between consecutive bags in the following. We represent the data transfer time between two consecutive bags, on two different cloud providers, as $Comm^{time}_{i,i-1,k,k'}$ which calculated from equation 4.6.

$$Comm^{time}_{i,i-1,k,k'} = \begin{cases} 0 & k = k' \\ \frac{BD_i}{BW_{k,k'}} & k \neq k' \end{cases} \tag{4.6}$$

Where BD_i indicates the size of an ith bag, and $BW_{k,k'}$ signify the bandwidth between cloud provider kth and k'th.

4.5 Scheduling model

In this section, workflow scheduling will be modeled and explained. Nevertheless, before modeling and detailing the scheduling approach, we first list all the notations that have been introduced so far and other notations that will be used later. These notations can be seen in Table 4.3.

Notation	Definition
n	The number of bags in the workflow application
B_i	ith bag of the workflow application
λ_i	The number of tasks contained in B_i
S_i={S_{1i}, S_{2i},. . . , S_{li}}	The set of segments contained in B_i
$S_l i$	lth segment of B_i
η_i	The number of segments contained in B_i
δ_{li}	The number of tasks contained in lth segment in ith bag
μ	The segment size
E^{task}_{nli}	The execution time in hours of each task of S_{li} on VM with 2-vCPU
$E^{segment}_{li}$	The execution time in hours of lth segment on VM with 2-vCPU
E^{bag}_{ikj}	The execution time of B_i on Instance type I_{kj}
$Budget$	User defined budget for executing the workflow
m	The number of public cloud providers
CP={EC2, Azure,. . . }	The participated public cloud providers in a multi-cloud environment
CP_k	kth cloud provider
J_k={m5a.large,. . . }	The set of instance types of CP_k
I_{kj}	jth instance type of CP_k
vCPU_{kj}	The number of vCPUs in the instance type I_{kj}
$Price_{kj}$	The price in $ for running one instance type I_{kj} for an hour
BD_i	Required data size for B_i
CD_i	Cost of transferring data to CP for 1 GB
$power^{segment}_{likj}$	The current processing power of Instance type I_{kj} for lth segment in i bag
τ_{likj}	Execution time of lth segment in ith bag on an instance type I_{kj}
$BW_{k,k'}$	Bandwidth between cloud providers k, k'
$Comm^{cost}_{i,i-1,k,k'}$	Data transferring cost of B_i to B_{i-1} where they executed in CP_k and $CP_{k'}$,respectively
$Comm^{time}_{i,i-1,k,k'}$	Data transferring time of B_i to B_{i-1} where they executed in CP_k and $CP_{k'}$,respectively
y_{ik}	1 if B_i is assigned to CP_k, otherwise 0
ω_i	1 if B_i and B_{i-1} are assigned to a common cloud provider, otherwise 0
T_{ikj}	The required time to execute B_i on instance type I_{kj}
N_{ikj}	The number of B_i segments that are assigned to instance type I_{kj}
x_{ikj}	1 if the B_i is assigned to instance type I_{kj}, otherwise 0
$Cost_i$	The cost (price) in $ of execution of ith bag

Table 4.3: Notations and their description of the proposed scheduling approach

In the following, we formula the makespan minimization problem for deploying a budget-constrained scientific workflow application on the multi-clouds with different instance types VMs. Nevertheless, before giving the details, it is necessary to mention that since the proposed model is static scheduling, it is natural that some information is already available. This information includes workflow characteristics (e.g., number of bags, number of tasks per bag, and task execution time in a VM with the performance of 2-vCPU, etc.) and cloud specifications (e.g., number of cloud providers, provided instance types of each cloud, price, and performance of each instance type, etc.). The makespan mainly contains computation time and communication time (transferring time). According to the infrastructure model, partitioning model, running tasks on a VM model, and communication model defined above, the problem of minimizing the makespan of a workflow application is defined below. As mentioned before, we applied the MILP model to formulate the scheduling approach in a multi-cloud environment. A mathematical model related to a decision-making problem includes decision variables that represent the number of certain resources for use or the level of certain activities. The value of decision variables is specified during the problem-solving process. The decision variables in proposed MILP model are defined in equations 4.17, 4.18, 4.19, 4.20, 4.21, 4.22, which can be expressed as follows:

- Decision variable y_{ik} is a binary variable. It attains a value of 1 when B_i is assigned to CP_k and a value of 0 otherwise.

- Decision variable N_{ikj} is a positive integer variable. Its value indicates the number of B_i segments that are assigned to Instance type I_{kj}.

- The decision variable ω_i is a binary decision variable. It attains a value of 1 when B_i and B_{i-1} are assigned to a common cloud and a value of 0 otherwise.

- Decision variable x_{ikj} is a binary variable. It attains a value of 1 when B_i is assigned to Instance type I_{kj} and a value of 0 otherwise.

- Decision variable T_{ikj} is a positive real variable that indicates the execution time of B_i on Instance type I_{kj}.

- Variable $Cost_i$ is a positive real variable whose value represents the cost (price) of B_i.

The objective function and constraints, which are explained in detail after the

model, are as follows:

$$Objective function = min\left(\sum_{k=1}^{m}\sum_{j\in J_k}\sum_{i=1}^{n} T_{ikj}\right) \tag{4.7}$$

Subject to

$$\sum_{k=1}^{m} y_{ik} = 1 \quad \forall i \in \{1,2,\ldots,n\}, \forall k \in \{1,2,\ldots,m\}, \tag{4.8}$$

$$\omega_i \geq y_{i-1} - y_i \quad \forall i \in \{2,\ldots,n\}, \tag{4.9}$$

$$\omega_i \geq y_i - y_{i-1} \quad \forall i \in \{2,\ldots,n\}, \tag{4.10}$$

$$\sum_{j\in J_k} N_{ikj} = y_{ik} \cdot \eta_i \quad \forall i \in \{1,2,\ldots,n\}, \quad \forall k \in \{1,2,\ldots,m\}, \tag{4.11}$$

$$N_{ikj} \geq x_{ikj} \quad \forall i \in \{1,2,\ldots,n\}, \forall k \in \{1,2,\ldots,m\}, \forall j \in J_k, \tag{4.12}$$

$$N_{ikj} \leq x_{ikj} \cdot \eta_i, \quad \forall i \in \{1,2,\ldots,n\}, \forall k \in \{1,2,\ldots,m\}, \forall j \in J_k, \tag{4.13}$$

$$\left(\left(N_{ikj} \cdot E^{bag}_{ikj}\right) + \left(\omega_i \cdot Comm^{time}_{i,i-1,k,k'}\right)\right) \leq T_{ikj}, \tag{4.14}$$

$$\forall i \in \{1,2,\ldots,n\}, \forall k \in \{1,2,\ldots,m\}, \forall j \in J_k$$

$$Cost_i = Cost^{Exe}_i + Cost^{Comm}_i \tag{4.15}$$

$$Cost^{Exe}_i = \left(\left(N_{ikj} \cdot E^{bag}_{ikj}\right) + \left(\omega_i \cdot Comm^{time}_{i,i-1,k,k'}\right)\right) \cdot Price_{kj},$$

$$Cost^{Comm}_i = \omega_i \cdot Comm^{cost}_{i,i-1,k,k'},$$

$$\forall i \in \{1,2,\ldots,n\}, \forall k \in \{1,2,\ldots,m\}, \forall j \in J_k,$$

$$Budget \geq \sum_{i=1}^{n} Cost_i \tag{4.16}$$

$$y_{ik} \in \{0,1\} \quad \forall i \in \{1,2,\ldots,n\}, \forall k \in \{1,2,\ldots,m\}, \tag{4.17}$$

$$N_{ikj} \in \mathbb{Z}^+ \quad \forall i \in \{1,2,\ldots,n\}, \forall k \in \{1,2,\ldots,m\}, \forall j \in J_k, \tag{4.18}$$

$$w_i \in \{0,1\} \quad \forall i \in \{1,2,\ldots,n\}, \tag{4.19}$$

$$x_{ikj} \in \{0,1\} \quad \forall i \in \{1,2,\ldots,n\}, \forall k \in \{1,2,\ldots,m\}, \forall j \in J_k, \tag{4.20}$$

$$T_{ikj} \in \mathbb{R}^+ \quad \forall i \in \{1,2,\ldots,n\}, \forall k \in \{1,2,\ldots,m\}, \forall j \in J_k, \tag{4.21}$$

$$Cost_i \in \mathbb{R}^+ \quad \forall i \in \{1,2,\ldots,n\}, \tag{4.22}$$

Below, the roles of the constraints and objective function of the MILP model are explained in detail:

- The objective function, presented in Equation (4.7), indicates the total execution time (makespan) of a scientific workflow, including execution time and data transfer time. The MILP model minimizes this objective function.

- Constraint (4.8) denotes that each bag must be assigned to only one cloud provider.

- Constraints (4.9) and (4.10) show that if B_i and B_{i-1} are assigned to a common cloud, then $\omega_i = 0$. Thus, communication cost and data transfer time must be disregarded from analysis.

- Constraint (4.11) guarantees that all the segments in B_i are assigned and that no segment can be assigned more than once.

- Constraints (4.12) and (4.13) show that $x_{ikj} = 1$ is equivalent to positivity of N_{ikj} or that $x_{ikj} = 0$ is equivalent to a vanishing N_{ikj}.

- Constraint (4.14) determines the required time to run instance type I_{kj} in hours for executing the segments in B_i.

- Constraint (4.15) determines the cost of B_i, whose value is sum of execution cost and communication cost. The execution cost is calculated based on $\left(\left(N_{ikj} \cdot \tau_{ikj}\right) + \left(\omega_i \cdot Comm^{time}_{i,i-1,k,k'}\right)\right) \cdot Price_{kj}$. This value is a rounded-up integer value because the price for running an instance type is charged in dollars per hour. Then, this value will multiply price of the instance type I_{kj} that B_i assigned to. On the other hand, $\omega_i \cdot Comm^{cost}_{i,i-1,k,k'}$ signify the communication cost of B_i.

- Constraint (4.16) guarantees that the sum of the cost of all bags is less than the given budget.

5

Experiments

This chapter outlines the experimental setup and implementation details of the prototype of the proposed scheduling model. The prototype is used to explore how well the proposed scheduling model works and how to meet the specified needs and requirements. First, section 5.1 gives a brief overview of the programming language and tools used to implement the prototype. Next, section 5.2 outlines the pyomo tool data portal. Section 5.3 describes the data segregation way for use in the prototype. A brief description of the optimization solver tool will also be given in section 5.4. Finally, section 5.5 explains the experimental setup used to evaluate the prototype.

5.1 Programming languages and tools

To implement the prototype of the proposed scheduling model, python programming language[1] version 3.8 10 has been used. Python is an open-source programming language supported by Python Software Foundation[2]. In addition, the C# programming language[3] version 9.0, which is part of the .NET SDK (Software Development Toolkit), is also used to prepare data and workflows structures. The .NET is now available as an open-source SDK and can be run

1. https://www.python.org/
2. https://www.python.org/psf-landing/
3. https://docs.microsoft.com/en-us/dotnet/csharp/

cross-platform. Furthermore, the Ubuntu 20.04 LTS [4] OS (Operating System) used as the environment to implement this prototype. The kernel version used is 5.4.0-56-generic.

The Pyomo (Python Optimization Modeling Objects) tool was used to model the proposed mathematical optimization problem. Pyomo[5] allows users to formulate optimization problems in Python in a manner that is similar to the notation commonly used in mathematical optimization. In addition, the Pegasus Workflow Generator[6] tool has been used to generate the workflow used in the evaluation section. Indeed, Pegasus is a coding-based tool that allows the user to generate workflows by explaining high-level descriptions. The output generated by Pegasus will be an XML file that contains information about jobs (e.g., execution time, data size, etc.) and the relationships between them. For instance, Listing 5.1 shows the data generated by Pegasus for the Montage workflow with 50 tasks/jobs.

Listing 5.1: Sample Pegasus DAX file

```
<?xml version="1.0" encoding="UTF-8"?>
<adag xmlns="http://pegasus.isi.edu/schema/DAX" version="2.1">
    <job id="ID00000" namespace="Montage" name="mProjectPP" runtime="13.69">
        <uses file="region.hdr" link="input" register="true" transfer="true"
              optional="false" type="data" size="304"/>
        <uses file="2mass-jID00000.fits" link="input" register="true"
             transfer="true" optional="false" type="data" size="4222080"/>
        <uses file="p2mass-jID00000.fits" link="output" register="false"
             transfer="false" optional="false" type="data" size="4146344"/>
    </job>
    <job id="ID00001" namespace="Montage" name="mProjectPP" version="1.0"
         runtime="13.73">
        <uses file="region.hdr" link="input" register="true" transfer="true"
              optional="false" type="data" size="304"/>
        <uses file="2mass-jID00001.fits" link="input" register="true"
             transfer="true" optional="false" type="data" size="4222080"/>
        <uses file="p2mass-jID00001.fits" link="output" register="false"
             transfer="false" optional="false" type="data" size="4169076"/>
    </job>
    .
    .
    .
    <child ref="ID00008">
        <parent ref="ID00000"/>
    </child>
    <child ref="ID00009">
        <parent ref="ID00001"/>
        <parent ref="ID00000"/>
    </child>
    .
    .
    .
</adag>
```

4. https://ubuntu.com/
5. https://www.pyomo.org/
6. https://pegasus.isi.edu/

5.2 Pyomo Data Portal

As mentioned recently, pyomo is used for modeling the proposed mathematical optimization problem. Pyomo uses various data formats (e.g., TAB, CSV, JSON, XML, YAML, DAT, etc.) to define the necessary parameters and sets. By default, pyomo uses the DAT data format to define parameters and sets. However, due to its hard readability and the aim of better data segregation, an attempt has been made to use another data format. As a result, we have used JSON data format due to its higher readability and more straightforward data format. However, regardless of what data format is used, in pyomo the data required for use in the modeling must be prepared in advance as parameters or sets. For instance, if a workflow has three bags, it should explicitly define for pyomo that the workflow with three bags and each bag has a certain number of tasks.

Listing 5.2: Sample JSON data portal in pyomo

```
{
    "Bags" : [
        [1], [2], [3]
    ],
    "NumberofTaskInBags": [
        { "index": [1], "value": 2},
        { "index": [2], "value": 2},
        { "index": [3], "value": 1}
    ],
    "ExecutionTime" : [
        { "index": [1, "Task1"],  "value": 0.30},
        { "index": [1, "Task2"],  "value": 0.26},
        { "index": [2, "Task3"],  "value": 0.31},
        { "index": [2, "Task4"],  "value": 0.36},
        { "index": [2, "Task5"],  "value": 0.27},
        { "index": [3, "Task6"],  "value": 0.27}
    ]
}
```

Listing 5.2 depicts the definition of the parts of a workflow that include three bags and five tasks in the JSON data format.

5.3 Data Segregation

As mentioned, the data generated by the Pegasus Workflow Generator tool contains information such as the execution time of each task, the relationships between them, and so on. However, this information can not be used directly in pyomo. Hence, we must have a tool to convert and segregate this information

and make it usable for pyomo. For this purpose, we wrote the converter program, which allows us to convert the data generated by Pegasus into the format required by pyomo. Figure 5.1 illustrates the overview of the Converter program to convert Pegasus DAX files to the data format required by Pyomo.

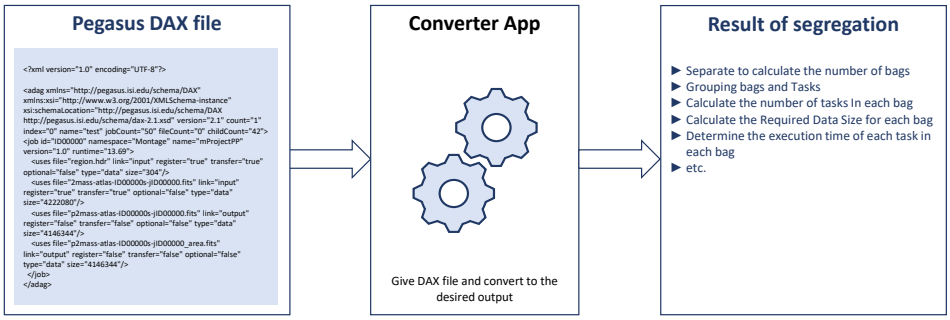


Figure 5.1: General overview of Converter program for Pegasus DAX files

The converter program is written in C# programming language with the help of LINQ[7] library.

5.4 Optimization Solver

The proposed mathematical models are solved by GLPK [8] (GNU Linear Programming Kit) version 4.32. Indeed, the GLPK package is intended for solving large-scale linear programming (LP), mixed integer programming (MIP), and other related problems. It is a set of routines written in ANSI C and organized in the form of a callable library. It is part of the GNU project.

5.5 Experimental setup

All experiments are carried out on Dell Precision 5550 with Core i7 (2.70 to 5.1 GHz) processors with 6-cores (12 Hyper Threads) and 32 GB DDR4 SD RAM. On the other hand, all workflow data and information and Iaas cloud providers are actual. In principle, this computer has been used to run the proposed model and evaluate it with actual data.

The workflows used for evaluation of the proposed model are four well-known workflows from different scientific areas, which were taken from the Pegasus

7. Language Integrated Query
8. `https://www.gnu.org/software/glpk/`

Workflow Generator, and the tasks in each of these workflows are considered between 50 and 1000 tasks as mentioned before these four well-known workflows are Montage, LIGO, SIPHT, and CyberShake. The Montage application from the astronomy field is used to generate custom mosaics of the sky based on input images. Most of its tasks are I/O intensive while not requiring much CPU processing capacity. The LIGO workflow from the astrophysics domain is used to detect gravitational waves. It is composed mainly of CPU-intensive tasks with high memory requirements. SIPHT is used in bioinformatics to automate the search for sRNA encoding genes. Most of the tasks in this workflow have high CPU and low I/O utilization, also in the bioinformatics domain. Finally, CyberShake is used to characterize earthquake hazards by generating synthetic seismograms and can be classified as a data-intensive workflow with large memory and CPU requirements. The workflows are depicted in Figure 2.6, and their full description and characterization are presented by Bharathi et al. [10]. Table 5.1 illustrates the Characteristics of these four well-known scientific workflows, including the count of tasks, average size, and average task execution time.

Type	Workflows	Bags	Avg. Size (MB)	Avg. exe time (sec)
I/O	**Montage**	9	20,6	11.34
	CyberShake	5	1156.1	51.70
CPU	**LIGO**	8	55.6	222.0
	SIPHT	7	22.02	210.27

Table 5.1: Characteristics of the real-world large-scale scientific workflows

For evaluation, three cloud providers are available (Amazon EC2, Microsoft Azure, and Google CE) as Iaas providers in a multi-cloud environment. It is assumed that each cloud provider also has 4 instance types. The Instance types configurations are based on those shown in Table 4.1. Bandwidth between different cloud providers is also considered according to Table 4.2. The costs of data transfer (CD_k) to Amazon EC2, Microsoft Azure, and Google CE were assumed to be $0.18, $0.2, and $0.2 per gigabyte, respectively. The cost and time of data transfer between two sequential bags in a workflow are regarded as 0 if the bags are assigned to a common cloud. Otherwise, the values must be calculated.

6 Evaluation

This chapter presents the evaluation of the proposed scheduling approach. The results discussed in this section aim to document the performance of the implementation through the experiments brought forth in chapter 5. Section 6.1 highlights some of the criteria used in the evaluation. Section 6.2 will then present the evaluation results. The different effects on budget and number of tasks on optimal cost and makespan will be examined.

6.1 Evaluation Criteria

In the experiments, different budget intervals were employed. The experiments were conducted using five different budgets, with β_1 being the strictest and β_5 being the most relaxed one. For each workflow, β_1 is equal to the cost of running all the tasks in a single VM (with 2-vCPU) of the cheapest instance type of each cloud provider. β_5 is the cost of running each workflow task on a different VM of the most expensive type available to each cloud provider. An interval size of $\beta_{int} = \frac{\beta_5 - \beta_1}{4}$ is then defined and used to estimate the remaining budgets: $\beta_2 = \beta_1 + \beta_{int}$, $\beta_3 = \beta_2 + \beta_{int}$, and $\beta_4 = \beta_3 + \beta_{int}$.

Also, in the evaluation to show the optimal cost ratio in different budgets, the optimal cost ratio to a more relaxed budget (β_5) has been used. By dividing the optimal cost by a more relaxed budget, we arrive at a percentage of spending

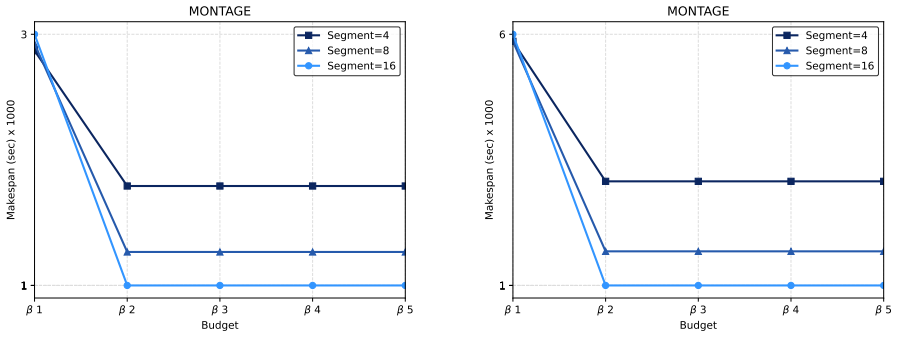
that helps us better understand the diagrams.

6.2 Evaluation Results

In this section, the results obtained from the evaluation of the proposed model will be reviewed and analyzed. Four types of evaluations will be discussed. The treatment of makespan concerning variation in the budget, makespan with respect to variation in task number, optimal cost per budget ratio in different budgets, and optimal cost per different task number.

6.2.1 Makespan with respect to variations in budget

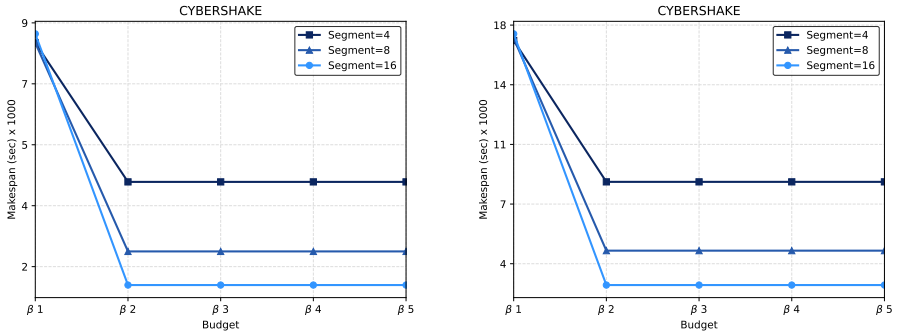
This subsection discusses the examination of the effects of changes in the budget on makespan. For this purpose, we have done this evaluation for four workflows with two different workflow sizes (500 and 1000 tasks). In addition, each workflow is evaluated with different segments size. We looked into three segment sizes (4, 8, and 16, respectively). Figure 6.1, Figure 6.2, Figure 6.3, and Figure 6.4 depicts the obtained results. In these results, the number of tasks in a workflow is 1000. Also, the budgets used are from β_1 to β_5, which are explained in section 6.1.



(a) Montage with 500 tasks **(b)** Montage with 1000 tasks

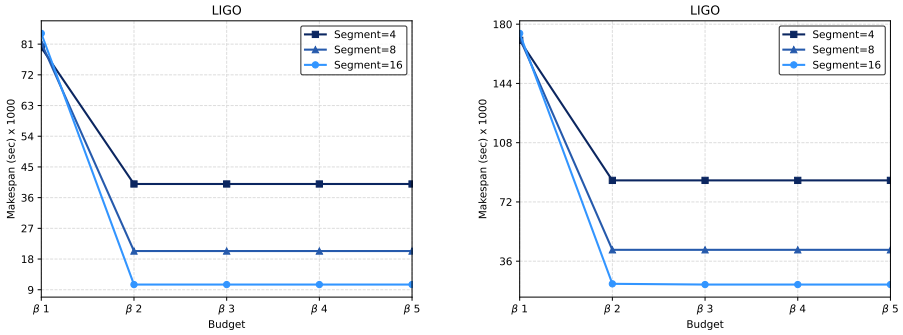
Figure 6.1: Makespan for Montage workflow grouped by budget

As you can see in Figure 6.1, Figure 6.2, Figure 6.3, and Figure 6.4, for all workflows, along with the increase in the budgets, makespan decreased as expected. This is because as the budget increases, our proposed model has more maneuverability. Therefore, it selects faster VMs (expensive VMs), thus reducing execution time and workflow's makespan. On the other hand, with lower budgets, our proposed model selects cheaper VMs (which are usually slower)



(a) CyberShake with 500 tasks **(b)** CyberShake with 1000 tasks

Figure 6.2: Makespan for CyberShake workflow grouped by budget

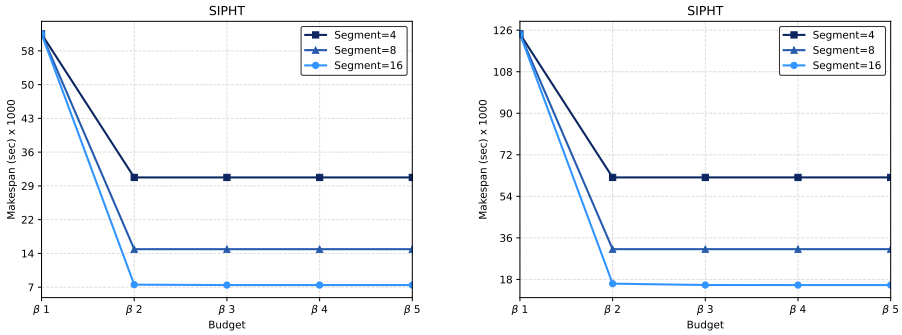


(a) LIGO with 500 tasks **(b)** LIGO with 1000 tasks

Figure 6.3: Makespan for LIGO workflow grouped by budget

to meet budget constraints. That is why in β_1, the makespan is higher. Looking at all figures (6.1, 6.2, 6.3, 6.4), it can be deduced that workflow's makespan do not decrease beyond a specific budget interval and it remains constant. One of the reasons that can be stated is that from a specific budget interval, our proposed model always selects the VMs that have the best performance (faster MVs with more vCPU) and are cost-effective.

On the other hand, by looking at the Figures (6.1, 6.2, 6.3, 6.4) [a and b], it can be concluded that as the size of the segment increases, the makespan also decreases. This is because as the size of the segment increases, so the number of tasks that can be run simultaneously and in parallel also increases. Therefore, at the same time, more tasks are completed, which ultimately leads to a reduction in the overall scheduling makespan of the workflow. Also, by comparing images a and b of each figure, it can be seen that the makespan of 1000 tasks is longer than the workflow of 500 tasks, which is normal and



(a) SIPHT with 500 tasks **(b)** SIPHT with 1000 tasks

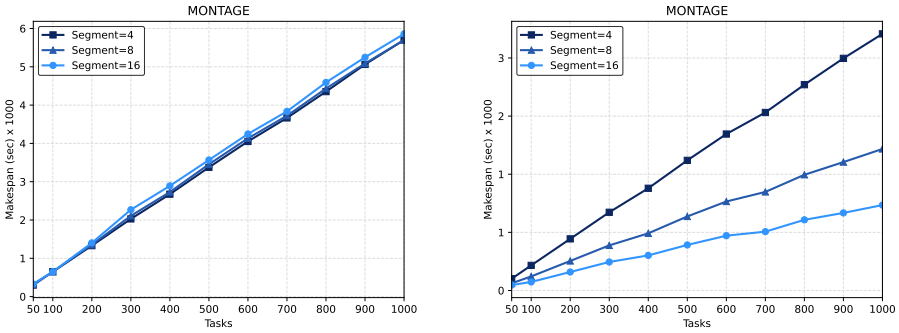
Figure 6.4: Makespan for SIPHT workflow grouped by budget

expected. It is also worth noting that the makespan of LIGO (Figure 6.3) and SIPHT (Figure 6.4) are greater than the Montage (Figure 6.1) and Cybershake (Figure 6.2). The reason for this goes back to the workflow characteristics. As mentioned earlier, according to the Table 5.1, LIGO and SIPHT workflows are CPU-intensive.This means that they usually need more time to be executed. Therefore, their execution time is longer than Montage and CyberShake workflows (I/O intensive).

6.2.2 Makespan with respect to variations in Tasks number

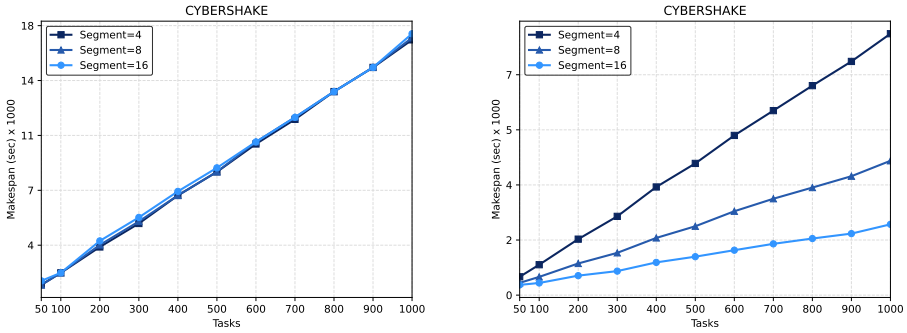
This subsection evaluates the effects of increasing the number of workflow tasks on makespan. This evaluation was performed for four workflows with two different budget sizes (β_1 and β_5). In addition, the number of tasks per workflow varies from 50 to 1000. In addition, each workflow is evaluated with different section sizes. We looked at three section sizes (4, 8, and 16, respectively). Figure 6.5, Figure 6.6, Figure 6.7, and Figure 6.8 depicts the results of this evaluation.

As seen in the figures above, as the number of tasks increases in all workflows, makespan also increases. This is quite obvious, and as the number of workflow tasks increases, more time will be required for processing, IO, and data transfer. to other cloud providers. Also, by looking closely at the figures, it can be seen that with the increasing budget, makespan decreases. This result caan be achieved by comparing diagrams a and b in all figures (Figure 6.5, Figure 6.6, Figure 6.7, and Figure 6.8). As the budget increases, our proposed model chooses higher-performance virtual machines. Performing tasks on efficient virtual machines will reduce the makespan.



(a) Makespan for β_1 **(b)** Makespan for β_5

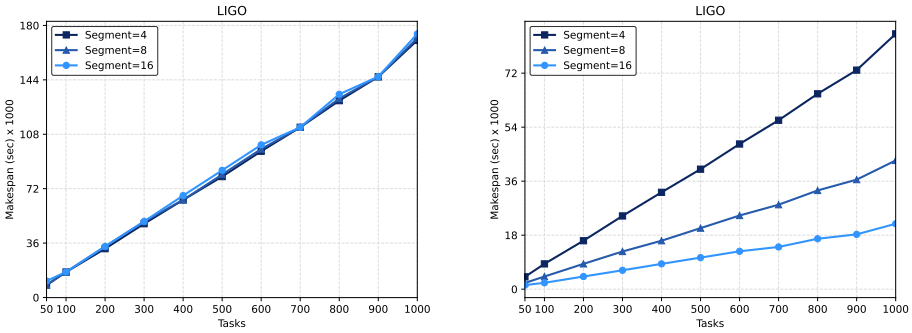
Figure 6.5: Makespan for MONTAGE workflows based on the different tasks number



(a) Makespan for β_1 **(b)** Makespan for β_5

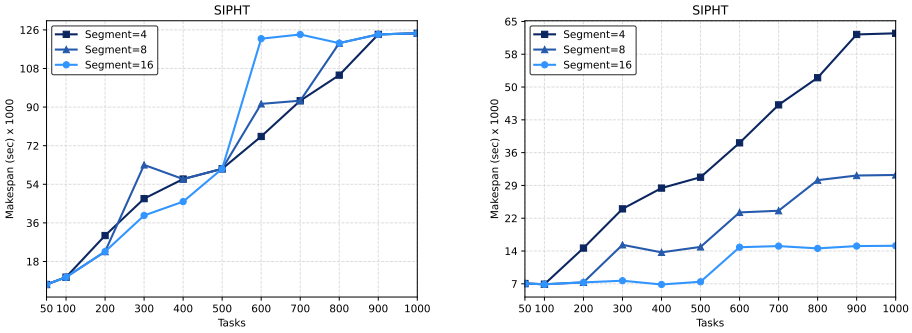
Figure 6.6: Makespan for CYBERSHAKE workflows based on the different tasks number

Another point that can be deduced is that as the size of the segments increases, the makespan decreases. This can be seen carefully in diagrams b of all workflows in Figures (6.5, 6.6, 6.7, and 6.8). As the size of the segment increases, our proposed model selects more powerful (more vCPU) and more cost-effective virtual machines. According to the user's budget, our model tries to select virtual machines that can schedule the workflow in the minimum makespan and at the lowest possible cost. Only for β_1, makespan is close together and almost equal in some workflows for all segment sizes. The reason is that with a low budget, our proposed model chooses the cheapest VMs that can meet the user's budget, which in the case of the lowest budget (β_1), always determines the same VMs.



(a) Makespan for β_1 **(b)** Makespan for β_5

Figure 6.7: Makespan for LIGO workflows based on the different tasks number



(a) Makespan for β_1 **(b)** Makespan for β_5

Figure 6.8: Makespan for SIPHT workflows based on the different tasks number

6.2.3 Optimal Cost per budget

In this subsection, the ratio of the optimal cost to the allocated budget will be examined, and its behavior with increasing the budget for the size of the different segments will be discussed. This evaluation has been done for four workflows, five budgets interval, and three different segment sizes. In addition, the evaluation is performed for all workflows with 1000 tasks. The results of this evaluation are shown in Figure 6.9 [a-d]. The results show that as the budget increases, the optimal cost decreases, and as a result, the optimal cost per budget rate naturally decreases. Looking at the Figure 6.9 [a-d], as budget intervals increase, it is easy to see a relatively downward trend for all workflows. As the budget increases, our proposed model can select the most cost-effective VMs with the proper performance. So that in addition to reducing the makespan, the budget constraint will also be met.

Looking carefully at Figure 6.9 [a-d], it can be seen that in each budget interval,

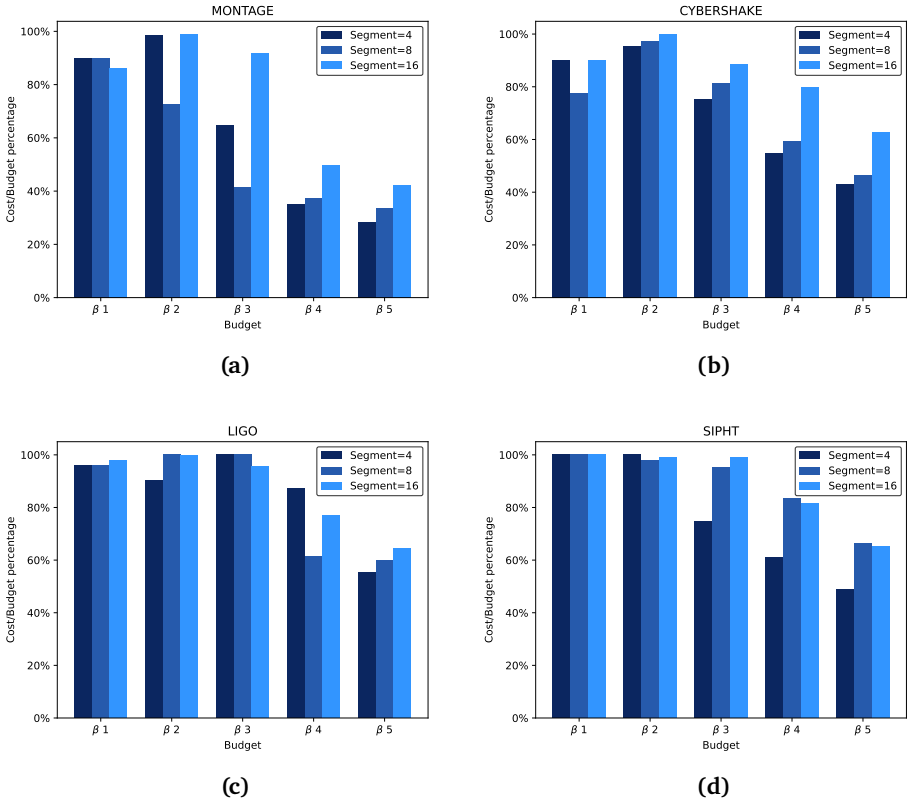


Figure 6.9: Optimal cost to budget ratios obtained for each of the workflows with 1000 tasks

by increasing segment size, the optimal cost per budget rate also increases. The reason for this is that by increasing the size of the segment, our proposed model tries to select VMs with more vCPUs. Because the best performance of a VM will be when the size of the segment is equal to or less than the number of vCPU. Therefore, our proposed model selects VMs with more vCPU to minimize the makespan. For instance, for segment size 16, the proposed model looks for VMs with 16 or 8 vCPU, which, in addition to higher performance, usually cost more than VMs with 2 or 4 vCPU.

6.2.4 Optimal cost per most relaxed budget with respect to variations in Tasks number

This subsection explains the evaluation of the impact of increasing the number of workflow tasks on optimal cost. This evaluation is performed for four workflows, and each workflow has several tasks between 50 and 1000. In addition, each workflow is evaluated with different budget sizes (β_1, β_3, and β_5, respec-

tively). All results for this subsection are obtained and evaluated for segment size 8 ($\mu = 8$). The results of this evaluation are illustrated in Figure 6.10 [a-d]. In this figure, the x-axis represents the number of workflow tasks, and the y-axis represents the optimal cost to the most relaxed budget ratio (β_5).

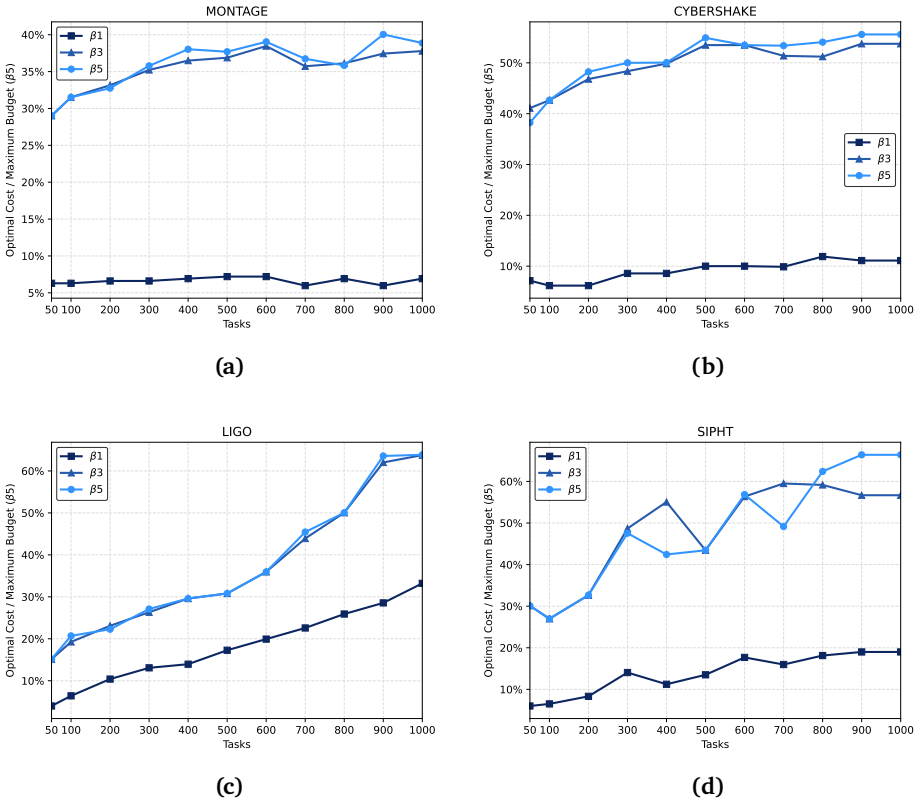


Figure 6.10: Optimal cost to most relaxed budget ratios obtained for each of the workflows with segment size 8

In Figure 6.10 [a-d], It is observed that with the increase in the number of workflow tasks, the optimal cost also increases. This is quite clear: as the number of tasks to be processed increases, so does the duration of use of VMs. Similarly, long-term use of VMs increases the cost of using VMs and ultimately increases the optimal cost. The only point is that in CPU-intensive workflows (6.10c and 6.10d), this slope increases relatively more. The reason, as mentioned earlier, is related to the nature of this type of workflow.

On the other hand, by looking closely at Figure 6.10, it can be concluded that the optimal cost increases with the increasing budget. The growing gap between the optimal cost for higher budgets is narrowing, and they are almost equal. Nevertheless, for smaller budgets, this gap is quite apparent. One of the reasons that can be said is that when the budget exceeds a certain threshold,

our proposed model will have more maneuverability and choice. Therefore, powerful and expensive virtual machines are always selected so that they can complete the relevant workflow in the shortest possible time. For this reason, for higher budgets, above the threshold, the optimal costs are almost close to each other.

7

Future work

This section will describe future work that could prove interesting in the context of this thesis. The future work will be described in as much detail as convenient and should not be considered the detailed specification for solutions. Instead, the work presented here is intended to build the foundations laid by this thesis's contributions.

7.1 Scheduling model improvement

Given that our proposed model utilizes a multi-cloud environment, data transfer from one cloud provider to another cloud provider will be inevitable. On the other hand, data transfer will be time-consuming with the analysis conducted, especially for I/O-Intensive workflows. This time-consuming will increase the usage time of virtual machines, and eventually, the scheduling cost will also increase. For this purpose, in future work, we will do data transfer simultaneously as task processing. This means that each task in a bag, after completing its processing and generating output data, that data is immediately transferred to the next cloud provider. This will reduce the usage time of VMs. Also, instead of transferring data directly from one cloud provider to another, a shared data source (e.g., a key-value store) can be used. In that case, adding parameters such as the read and write speed of that shared source somewhat improved the proposed model in terms of scheduling makespan and cost of using VMs. Using a shared datastore also increases the overall fault-tolerant system. So

that in case of problems in the data transferring, the data is not lost and can be reaccessed from the shared source.

7.2 Security considerations

Security has always been considered an important issue and an open challenge. In our proposed model, since a multi-cloud environment is used and the exchange and transfer of data between different bags placed on the Internet, there are usually several security issues and challenges. Challenges include secure data transfer between different cloud providers, user authentication and authorization, access control, etc. Therefore, security issues are considered a potential improvement in future work.

7.3 Energy consumption minimization

Another interesting topic that can be considered in future work is the discussion of optimizing and reducing energy consumption; since the challenge of global warming and greenhouse gas emissions, the importance of reducing energy consumption doubles. For this purpose, by reviewing the proposed model and calculating the energy consumption of each of the available resources and criteria (such as VMs, communications and data transfer, etc.) and adding these constraints, a mathematical model can be developed to select the most optimal resources (in terms of energy) for scheduling and resource provisioning.

7.4 Real-world Implementation

Currently, the system has not been tested in a "Real-World" environment. We intend to implement the real-world system and then commercialize it as a software system and deploy it as a service to serve different users. The microservice model can be used for implementation, so the system has different services for different tasks. For instance, a service to receive workflow from the user and perform preliminary processing (e.g., pipelining and segmenting workflow's tasks), another service to schedule and optimize segments, another service to produce results and prepare relevant reports, and another service to check the security and etc. Figure 7.1 is a general overview of the microservice architecture that will be used as the primary plan to implement the proposed model in the future as a real-world software system. Furthermore, access to this service through various platforms and devices (mobile, web, desktop) is another

feature considered in implementing the real-world software system.

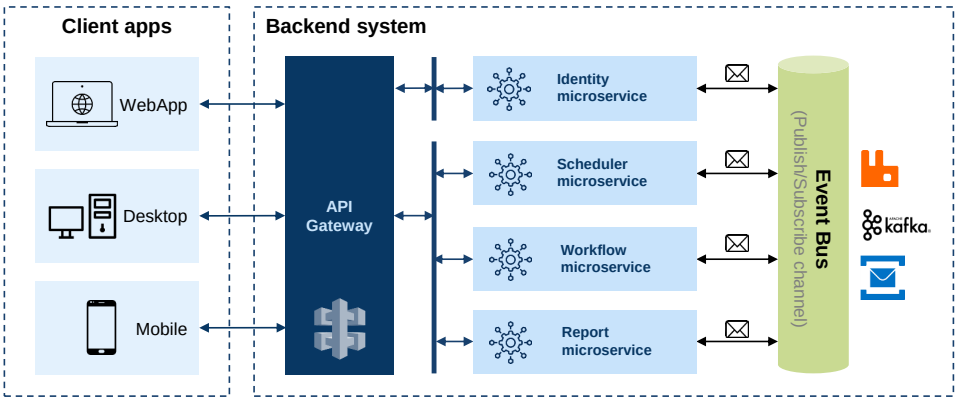


Figure 7.1: General overview of Microservice architecture for future work

8

Conclusion

This thesis proposed a fundamental mathematical model for scheduling scientific workflow in multi-cloud environments. The model aims to minimize the overall workflow makespan while meeting a user-defined budget constraint. The scheduling problem was formulated as a Mixed-Integer Linear Programming (MILP) model and solved with the GPLK solver in reasonable time and cost. The proposed model was developed for using multi-vCPU instance-type VMs. In addition, the proposed model divides a workflow into different Bag of Tasks (Bags) and segments and calculates the power of processing of each VM based on its vCPU and the segments of the workflow. For the proposed scheduling approach, the communication cost ad time (in transferring data) were considered in the analysis, and the cost of leasing VMs was calculated on an hourly basis.

We analyzed the effects of changes in budget, segment size, and workflow size on optimal makespan and cost per budget rate. For this purpose, four well-known and real-world workflows (e.g., Montage, CyberShake, LIGO, and SIPHT) were used to evaluate the proposed model. The following insights were derived from the experiments:

1. The effects of changes in the budget on optimal makespan are considerably more remarkable in the CPU-intensive workflows than in the I/O-intensive workflows.

2. As the size of the segments increases, the workflow's makespan decreases,

but on the other hand, the optimal cost also increases relatively. Therefore, a trade-off between makespan and optimal cost must be selected to select the right segment size.

3. As the number of workflow tasks increases, so the makespan also increases. This increase is more significant for lower budgets. That is, as the budget increases, the makespan decreases. On the other hand, in any given budget, the makespan decreases as the size of the segment increases.

4. As the number of workflow tasks increases, so does the optimal cost. Nevertheless, for higher budgets, they are almost equal. The slope of optimal cost changes is also higher for CPU-intensive workflows.

Bibliography

[1] Maria Alejandra Rodriguez and Rajkumar Buyya. A taxonomy and survey on scheduling algorithms for scientific workflows in iaas cloud computing environments. *Concurrency and Computation: Practice and Experience*, 29(8):e4041, 2017. e4041 cpe.4041.

[2] Yolanda Gil, Ewa Deelman, Mark Ellisman, Thomas Fahringer, Geoffrey Fox, Dennis Gannon, Carole Goble, Miron Livny, Luc Moreau, and Jim Myers. Examining the challenges of scientific workflows. *Computer*, 40(12):24–32, 2007.

[3] J.D. Ullman. Np-complete scheduling problems. *Journal of Computer and System Sciences*, 10(3):384–393, 1975.

[4] T.L. Casavant and J.G. Kuhl. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Transactions on Software Engineering*, 14(2):141–154, 1988.

[5] Kaiyue Wu, Ping Lu, and Zuqing Zhu. Distributed online scheduling and routing of multicast-oriented tasks for profit-driven cloud computing. *IEEE Communications Letters*, 20(4):684–687, 2016.

[6] Ali Sunyaev. *Cloud Computing*, pages 195–236. Springer International Publishing, Cham, 2020.

[7] Xiaomin Zhu, Chao Chen, Laurence T. Yang, and Yang Xiang. Angel: Agent-based scheduling for real-time tasks in virtualized clouds. *IEEE Transactions on Computers*, 64(12):3389–3403, 2015.

[8] Peter Mell and Timothy Grance. The nist definition of cloud computing, 2011-09-28 2011.

[9] Zhifeng Zhong, Kun Chen, Xiaojun Zhai, and Shuange Zhou. Virtual machine-based task scheduling algorithm in a cloud computing environment. *Tsinghua Science and Technology*, 21(6):660–667, 2016.

[10] Shishir Bharathi, Ann Chervenak, Ewa Deelman, Gaurang Mehta, Mei-Hui Su, and Karan Vahi. Characterization of scientific workflows. In *2008 Third Workshop on Workflows in Support of Large-Scale Science*, pages 1–10, 2008.

[11] Mainak Adhikari, Tarachand Amgoth, and Satish Narayana Srirama. A survey on scheduling strategies for workflows in cloud environment and emerging trends. *ACM Comput. Surv.*, 52(4), aug 2019.

[12] Fuhui Wu, Qingbo Wu, and Yusong Tan. Workflow scheduling in cloud: A survey. *J. Supercomput.*, 71(9):3373–3418, sep 2015.

[13] Igor Griva, S Nash, and Ariela Sofer. *Linear and Nonlinear Optimization*. SIAM publishing, 2 edition, 2009.

[14] David G. Luenberger and Yinyu Ye. *Linear and Nonlinear Programming*. Springer Publishing Company, Incorporated, 4 edition, 2015.

[15] Somayeh Abdi, Latif Pourkarimi, Mahmood Ahmadi, and Farzad Zargari. Cost minimization for bag-of-tasks workflows in a federation of clouds. *J. Supercomput.*, 74(6):2801–2822, jun 2018.

[16] Saeid Abrishami, Mahmoud Naghibzadeh, and Dick H.J. Epema. Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. *Future Generation Computer Systems*, 29(1):158–169, 2013. Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.

[17] Maciej Malawski, Kamil Figiela, Marian Bubak, Ewa Deelman, and Jarek Nabrzyski. Scheduling multilevel deadline-constrained scientific workflows on clouds based on cost optimization. *Scientific Programming*, 2015, 03 2015.

[18] Arash Deldari, Mahmoud Naghibzadeh, and Saeid Abrishami. Cca: A deadline-constrained workflow scheduling algorithm for multicore resources on the cloud. *J. Supercomput.*, 73(2):756–781, feb 2017.

[19] Maria Alejandra Rodriguez and Rajkumar Buyya. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. *IEEE Transactions on Cloud Computing*, 2(2):222–235, 2014.

[20] Saurabh Bilgaiyan, Santwana Sagnika, and Madhabananda Das. Workflow scheduling in cloud computing environment using cat swarm optimization. In *2014 IEEE International Advance Computing Conference (IACC)*,

pages 680–685, 2014.

[21] Amelie Chi Zhou, Bingsheng He, and Cheng Liu. Monetary cost optimizations for hosting workflow-as-a-service in iaas clouds. *IEEE Transactions on Cloud Computing*, 4(1):34–48, 2016.

[22] Wei Zheng and Rizos Sakellariou. Budget-deadline constrained workflow planning for admission control. *Journal of Grid Computing*, 11:633–651, 2013.

[23] Lingfang Zeng, Bharadwaj Veeravalli, and Albert Y. Zomaya. An integrated task computation and data management scheduling strategy for workflow applications in cloud environments. *Journal of Network and Computer Applications*, 50:39–48, 2015.

[24] Lingfang Zeng, Bharadwaj Veeravalli, and Xiaorong Li. Scalestar: Budget conscious scheduling precedence-constrained many-task workflow applications in cloud. In *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, pages 534–541, 2012.

[25] Xiangyu Lin and Chase Qishi Wu. On scientific workflow scheduling in clouds under budget constraint. In *2013 42nd International Conference on Parallel Processing*, pages 90–99, 2013.

[26] Hamid Arabnejad and Jorge Barbosa. A budget constrained scheduling algorithm for workflow applications. *Journal of Grid Computing*, page 15, 03 2014.

[27] Deepak Poola, Saurabh Kumar Garg, Rajkumar Buyya, Yun Yang, and Kotagiri Ramamohanarao. Robust scheduling of scientific workflows with deadline and budget constraints in clouds. In *2014 IEEE 28th International Conference on Advanced Information Networking and Applications*, pages 858–865, 2014.

[28] Juan J. Durillo, Radu Prodan, and Jorge G. Barbosa. Pareto tradeoff scheduling of workflows on federated commercial clouds. *Simulation Modelling Practice and Theory*, 58:95–111, 2015. Special Issue on TECHNIQUES AND APPLICATIONS FOR SUSTAINABLE ULTRASCALE COMPUTING SYSTEMS.

[29] Rafaelli de C. Coutinho, Lúcia M.A. Drummond, Yuri Frota, and Daniel de Oliveira. Optimizing virtual machine allocation for parallel scientific workflows in federated clouds. *Future Generation Computer Systems*, 46:51–68, 2015.

[30] Juan J. Durillo, Hamid Mohammadi Fard, and Radu Prodan. Moheft: A multi-objective list-based method for workflow scheduling. In *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings*, pages 185–192, 2012.

[31] Hamid Mohammadi Fard, Radu Prodan, Juan Jose Durillo Barrionuevo, and Thomas Fahringer. A multi-objective approach for workflow scheduling in heterogeneous environments. In *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pages 300–309, 2012.

[32] Sonia Yassa, Rachid Chelouah, Kadima Hubert, and Bertrand Granado. Multi-objective approach for energy-aware workflow scheduling in cloud computing environments. *TheScientificWorldJournal*, 2013:350934, 01 2013.

[33] Kahina Bessai, Samir Youcef, Ammar Oulamara, Claude Godart, and Selmin Nurcan. Bi-criteria workflow tasks allocation and scheduling in cloud computing environments. In *2012 IEEE Fifth International Conference on Cloud Computing*, pages 638–645, 2012.

[34] Lingfang Zeng, Bharadwaj Veeravalli, and Xiaorong Li. Saba: A security-aware and budget-aware workflow scheduling strategy in clouds. *Journal of Parallel and Distributed Computing*, 75:141–151, 2015.

[35] Zhaomeng Zhu, Gongxuan Zhang, Miqing Li, and Xiaohui Liu. Evolutionary multi-objective workflow scheduling in cloud. *IEEE Transactions on Parallel and Distributed Systems*, 27(5):1344–1357, 2016.

[36] R.K. Jena. Multi objective task scheduling in cloud environment using nested pso framework. *Procedia Computer Science*, 57:1219–1227, 2015. 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015).

[37] A. K. M. Khaled Ahsan Talukder, Michael Kirley, and Rajkumar Buyya. Multiobjective differential evolution for scheduling workflow applications on global grids. *Concurrency and Computation: Practice and Experience*, 21(13):1742–1756, sep 2009.

[38] Phyo Thandar Thant, Courtney Powell, Martin Schlueter, Masaharu Munetomo, and Emiliano Tramontana. Multiobjective level-wise scientific workflow optimization in iaas public cloud environment. *Sci. Program.*, 2017, jan 2017.

[39] Somayeh Abdi, Latif PourKarimi, Mahmood Ahmadi, and Farzad Zargari. Cost minimization for deadline-constrained bag-of-tasks applications in federated hybrid clouds. *Future Generation Computer Systems*, 71:113–128, 2017.

[40] Thiago A. L. Genez, Luiz F. Bittencourt, and Edmundo R. M. Madeira. Using time discretization to schedule scientific workflows in multiple cloud providers. In *2013 IEEE Sixth International Conference on Cloud Computing*, pages 123–130, 2013.

[41] Bing Lin, Wenzhong Guo, Naixue Xiong, Guolong Chen, Athanasios V. Vasilakos, and Hong Zhang. A pretreatment workflow scheduling approach for big data applications in multicloud environments. *IEEE Transactions on Network and Service Management*, 13(3):581–594, 2016.

[42] Hamid Mohammadi Fard, Radu Prodan, and Thomas Fahringer. A truthful dynamic workflow scheduling mechanism for commercial multicloud environments. *IEEE Transactions on Parallel and Distributed Systems*, 24(6):1203–1212, 2013.

[43] Radu Durillo, Juan J.and Prodan. Workflow scheduling on federated clouds. In *Euro-Par 2014 Parallel Processing*, pages 318–329, Cham, 2014. Springer International Publishing.

[44] Leonard Heilig, Eduardo Lalla-Ruiz, and Stefan Voß. A cloud brokerage approach for solving the resource management problem in multi-cloud environments. *Computers and Industrial Engineering*, 95:16–26, 2016.

[45] Rubing Duan, Radu Prodan, and Xiaorong Li. Multi-objective game theoretic schedulingof bag-of-tasks workflows on hybrid clouds. *IEEE Transactions on Cloud Computing*, 2(1):29–42, 2014.

[46] Maciej Malawski, Kamil Figiela, and Jarek Nabrzyski. Cost minimization for computational applications on hybrid cloud infrastructures. *Future Generation Computer Systems*, 29(7):1786–1794, 2013. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services & Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond.

[47] Ana-Maria Oprescu and Thilo Kielmann. Bag-of-tasks scheduling under budget constraints. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, pages 351–359, 2010.

[48] Marco A. S. Netto and Rajkumar Buyya. Offer-based scheduling of

deadline-constrained bag-of-tasks applications for utility computing systems. In *2009 IEEE International Symposium on Parallel Distributed Processing*, pages 1–11, 2009.

[49] Ioannis A. Moschakis and Helen D. Karatza. Multi-criteria scheduling of bag-of-tasks applications on heterogeneous interlinked clouds with simulated annealing. *Journal of Systems and Software*, 101:1–14, 2015.

[50] Amandeep Verma and Sakshi Kaushal. A hybrid multi-objective particle swarm optimization for scientific workflow scheduling. *Parallel Computing*, 62:1–19, 2017.

[51] Wei Zheng and Rizos Sakellariou. Budget-deadline constrained workflow planning for admission control. *Journal of Grid Computing*, 11(4):633–651, dec 2013.

[52] Maciej Malawski, Gideon Juve, Ewa Deelman, and Jarek Nabrzyski. Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in iaas clouds. *Future Generation Computer Systems*, 48:1–18, 2015. Special Section: Business and Industry Specific Cloud.

[53] Thiago A. L. Genez, Luiz F. Bittencourt, and Edmundo R. M. Madeira. Workflow scheduling for saas / paas cloud providers considering two sla levels. In *2012 IEEE Network Operations and Management Symposium*, pages 906–912, 2012.

[54] Maria A. Rodriguez and Rajkumar Buyya. Budget-driven scheduling of scientific workflows in iaas clouds with fine-grained billing periods. *ACM Transactions on Autonomous and Adaptive Systems*, 12(2), may 2017.

[55] Amol Jaikar and Seo-Young Noh. Cost and performance effective data center selection system for scientific federated cloud. *Peer-to-Peer Networking and Applications*, 8(5):896–902, sep 2015.

[56] Andrew Wylie, Wei Shi, Jean-Pierre Corriveau, and Yang Wang. A scheduling algorithm for hadoop mapreduce workflows with budget constraints in the heterogeneous cloud. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1433–1442, 2016.

[57] Adel Nadjaran Toosi, Rodrigo N. Calheiros, and Rajkumar Buyya. Interconnected cloud computing environments: Challenges, taxonomy, and survey. *ACM Comput. Surv.*, 47(1), may 2014.

[58] iPerf - The ultimate speed test tool for TCP, UDP and SCTP. `https://iperf.fr/`, 2022. [Online; accessed 31-March-2022].

[59] Hao Wu, Xin Chen, Xiaoyu Song, Chi Zhang, and He Guo. Scheduling large-scale scientific workflow on virtual machines with different numbers of vcpus. *The Journal of Supercomputing*, 77:679–710, 2021.