Universiteit
Antwerpen

Literature Review on Cost-Efficient Scheduling

Research Internship II
Department of Mathematics and Computer Science
of the Faculty of Sciences, University of Antwerp.
Submitted on January 2015

Promotoren:
Prof. dr. Jan Broeckhove
dr. Kurt Vanmechelen

Hoang Trung Hieu

Research Group Computational
Modelling and Programming

Contents

List of Figures

List of Tables

Abstract

IaaS cloud computing infrastructure has become an attractive platform which offers on-demand services to run a wide range of applications, including "online" or "batch-type" workloads. IaaS allows consumers to allocate virtual machines in providers' datacenters, that run a user-defined operating system and application software stack in an on-demand fashion. That is, a user has the ability to allocate or deallocate virtual machine instances at any time according to the its workload demand.

Executing a workload tends to have minimum requirements on the resources it consumes (number of cores, speed of CPU or available memory etc). On the other hand, the owner will also require its workload execution to achieve a certain QoS level such as deadline or budget constraints. With the maturation and expansion of IaaS market, consumers have to face an increasing comlexity when acquire cloud resources in a cost-efficient manner while still respecting the QoS constraints.

In this internship report, we will provide an overview of the available literature on cost-efficient scheduling applications in a cloud setting and evaluate such studies based on a set of proposed criteria.

CHAPTER 1

INTRODUCTION

1.1 Cloud Computing

Cloud computing is emerging as an interesting topic in both research and business areas. Cloud computing refers to applications and services that run on a distributed network using virtualized resources and can be accessed using networking standards. It is realized by the notion that resources are virtual and limitless and that details of the physical systems are abstracted from the user.

In fact, cloud computing has been around for a long time which is evolved from grid computing since 1980s. However, until recent years, with the maturity of virtualization technology, cloud computing has been widely accepted and deployed from a small enterprise to multinational corporations.

Generally, cloud computing provides computation, software, data access, and storage services that do not require end-user knowledge of the physical location and configuration of the system that delivers the services. The service providers deliver applications via the internet, which are in turn accessed from a Web browser, while the business software and data are stored on servers at a remote location.

With the cloud market expected to grow quickly in the coming years, it is becoming a fierce competitive market with diverse service providers such as Amazon Web Service, Windows Azure, GoGrid, Rackspace etc. In this chapter, we will give brief introduction on the definition of cloud computing and its characteristics, service and deployment models.

1.1.1 Cloud computing definition

Even though the term cloud computing has been used for a while, its definition is still an issue that has not received wide consencus yet. However, among various

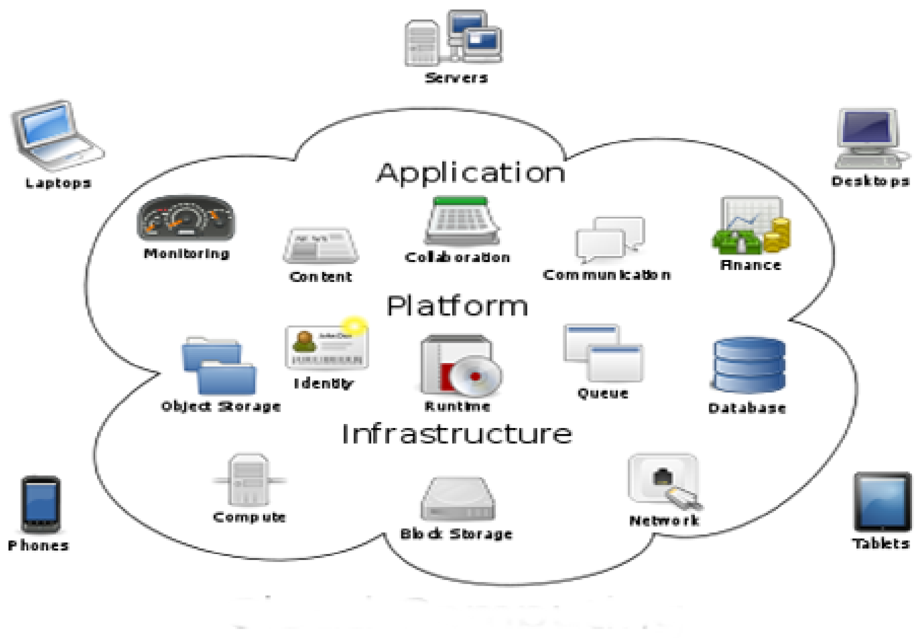

Figure 1.1: Logical Diagram of Cloud Computing [20]

definitions of cloud computing today, the definition from U.S. National Institute of Standards and Technology (NIST) has been recognized and cited in many studies.

According to 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."*[26]

1.1.2 Essential characteristics

The cloud model has described with 5 essential characteristics [26]

On-demand self-service

Cloud service providers can provision resources to its users as needed automatically without requiring human interaction. The users have access to the services and permission to change service configuration following their needs via a web-based service portal (management console)

Broadband network access

With the cloud technologies, users are able to access their resources from anywhere over the network, using a variety of devices such as PCs, laptops, smartphones, and PDAs etc.

Resource polling
Cloud service providers take full control of their physical resources, create a pool of virtual resources and allocate them to serve multiple users. The user generally has no control or little knowledge over the exact location of the provided resources but is able to specify location at a higher level of abstraction.

Elasticity
Resources can be provisioned based on customers' requests. Self-service and resource pooling make rapid elasticity possible. From the user perspective, the capabilities available at cloud providers appear to be unlimited and can be easily requested or released. This will make sure an application will have exactly the capacity it needs at any point of time.

Measured service
Resources usage is metered and reported transparently back to users. The users only pay for what they have used.

1.1.3 Service models

Various cloud services can be categorized into three service models which are widely accepted. These 3 models are together referred as SPI (Software as a Service, Platform as a Service, and Infrastructure as a Service) model of cloud computing [32]. Other services such as Storage as a Service (StaaS), Identity as a Service (IdaaS) can be covered by one of these SPI models.

Software as a Service (SaaS)
In SaaS model, the application is accessible to customer via thin-client interface (mostly web-browser).Customers are only responsible for managing their data and user interaction while operation environment with application, management and user interface are the responsibility of service providers. Examples of SaaS are Goolge Docs, Salesforce CRM, Office 365

Platform as a Service (PaaS)
PaaS provides virtual machines, operating systems, application services, development frameworks and control structures. Users can deploy any of their applications on the cloud infrastructure but languages and tools used must be supported by PaaS service providers. Examples: Force.com, Google App Engine, Windows Azure (Platform)

Infrastructure as a Service (IaaS)
In this model, an IaaS service provider manages the infrastructure including virtual machines, virtual storages, virtual infrastructure and other hardware assets while users are responsible for the whole deployment process such as operating system, application framework and interaction with the systems. Example: Amazon Web Services, Windows Azure.

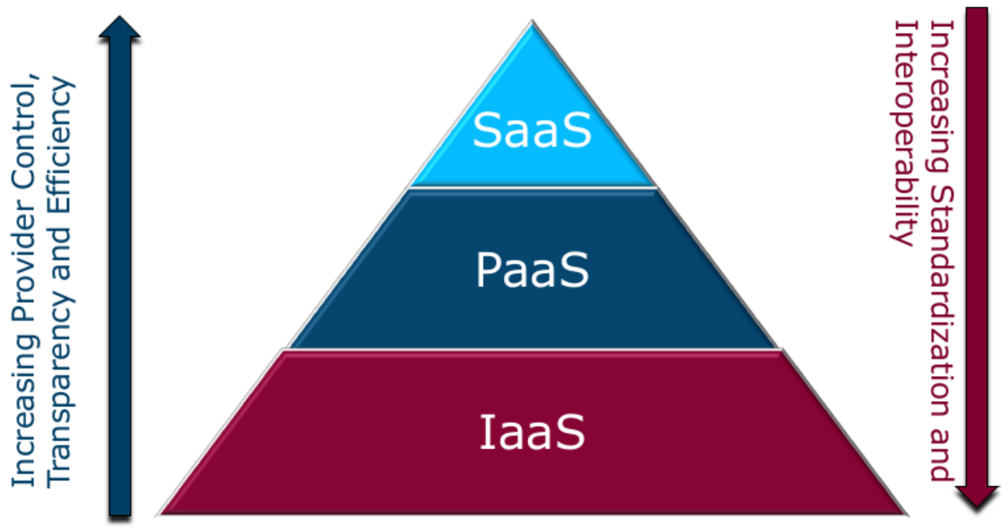
Increasing Provider Control, Transparency and Efficiency
SaaS
PaaS
IaaS
Increasing Standardization and Interoperability

Figure 1.2: Cloud Computing Service Models [19]

1.1.4 Deployment models

Deployment models identify the purpose of the cloud and the nature of how the cloud is deployed and located. There are four deployment models defined by NIST as follows [26]:

Public cloud

Resources are dynamically provisioned to users over Internet using web applications or web services. A public cloud is owned by third-party selling cloud services. Applications and data from different users are mixed and shared the same infrastructure such as servers, storage systems and network from the cloud provider. his is the most popular type of cloud system and is considered as a main stream of cloud systems [20]. Example: Amazon Web Service, Windows Azure, GoGrid.

Private cloud

A private cloud infrastructure is operated for use of an organization only. The private cloud may be either on or off-premise, meaning its deployment is in organization's IT private infrastructure or in a cloud provider infrastructure.

Hybrid cloud

A hybrid cloud is the combination of multiple public and private clouds where those clouds retain their unique identity but bound together as a unit. Hybrid cloud makes determining the distribution and scheduling the execution of applications across public and private cloud challenging issue.

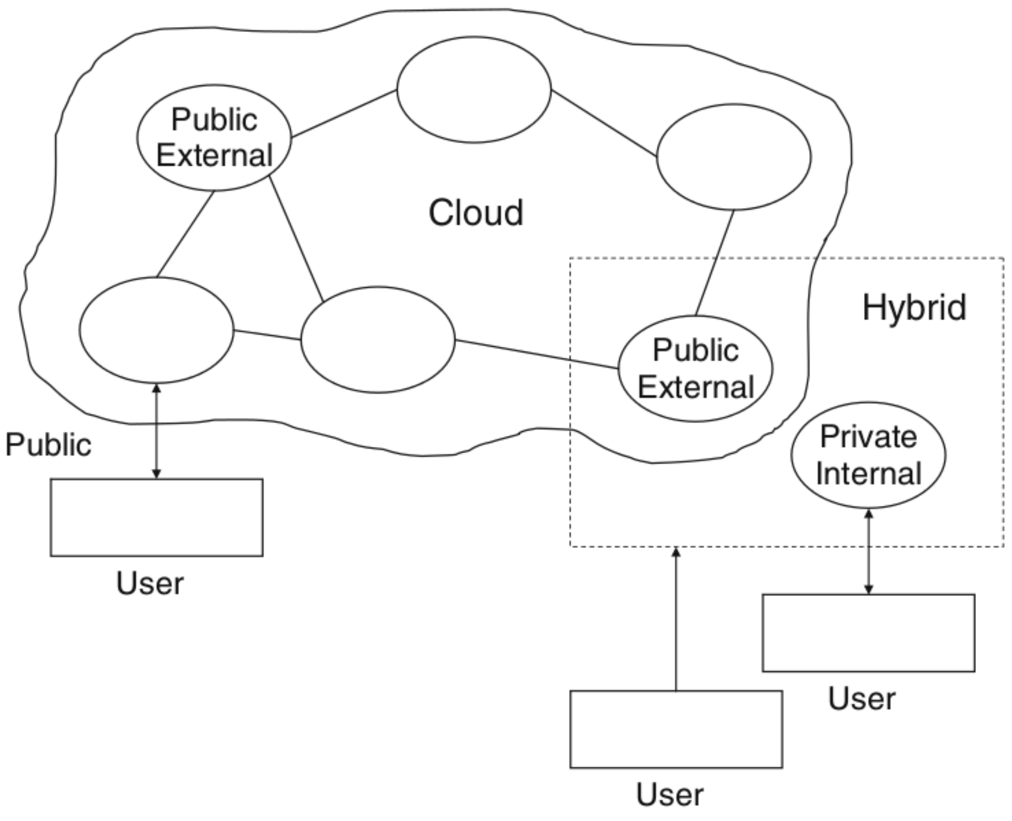

Figure 1.3: Deployment Models [3]

CHAPTER 2

PROBLEM STATEMENT

2.1 Introduction

Cloud computing introduces an attractive way to alleviate private IT infrastructure construction for companies and organizations regardless of whether they are small independent software vendors (ISVs), startups, or large corporations. Nowadays IaaS infrastructure can be used to run a wide range of applications, including both "online" applications such as web-, database- or application servers and "batch-type" workloads such as payroll processing, scientific calculations or simulations.

When users want to execute their application workload in the cloud environment, many concerns should be taken into account. The application workloads normally have minimum requirements on the resources they consume (e.g. number or speed of CPU, available memory, disk space, network or disk I/O speed etc.). It is obvious that the more resources can be acquired, the better the performance of workload execution is. In addition, the users always require their applications to achieve a certain level of quality of service corresponding to their purposes and business core.

Cloud computing with pay-as-you-go pricing model gives an alternative to users' private IT investment in which users only pay for what they use. Although cloud resouces are elastic, elasticity and performance still come at a supplemental cost. Trade-off is introduced between QoS and execution cost. The problem stems from the fact that different cloud providers offer services with different price models depending on service configuration and providers' business strategy. Even a cloud provider can deliver one service with different rates and pricing plan, i.e on-demand rate, reserved rates in 1 year, 3 year etc. Therefore, the question is how to minimize the cost spending on cloud resources to complete a workload in a cloud environment with provided quality of service level.

The goal of this research internship is to provide an overview of the available literature on cost-efficient scheduling solutions in a cloud setting. The rest of the report is organized as follows: following this introduction we present the criteria used to evaluate the contribution of reviewed papers in Chapter 3. Chapter 4 presents the summary and analysis of several papers that we believe they are relevant to the topic of the research internship. Conclusion will be given in Chapter 5.

2.2 Problem statement

A user has a workload which is considered to be executed in cloud environment. The workload can be of any type such as online, scientific or even combined workload. It is also possible to describe workloads in terms of computing power or required number of VMs.

The users require that the execution of the workload in cloud should meet their certain level of QoS. Depending on the users and the characteristics of the workload, the QoS criteria can vary. Workload types and QoS criteria will be discussed in more detail in Chapter 3.

With an elastic paradigm and pay-as-you-go price model, it seems that computation of execution cost and minimize it is a trivial task. We can pick up the cheapest virtual machines from cloud providers which are capable of running the workload. However, selecting the right services and acquiring sufficient resources is a challenging problem because of following reasons:

- Multiple cloud providers offer their services in various types of configuration, non-standardized notions and different pricing schemes.
- Even a cloud provider may have different pricing schemes for one service such as on-demand, prepaid or spot price (Spot Price is specifically offered by AWS).
- There are many factors which have significant impact on execution time and cost such as data saving/transferring delay, cloud resource activation and termination delays. In addition, the values of such factors are usually not fixed and aware upfront.
- The selected services are not only able to complete the workload but also need to satisfy a given QoS from the user. The QoS requirements are varied depending on workload and users' objectives.
- The monetary cost of workload execution should be minimized.
- Users may have invested on private infrastructure which they want to utilize before oursourcing workload to public clouds.

In fact, which expenses taken into cost computation may increase the complexity of the task. While running an application in a cloud, many types of expense may be involved, including but not limited to:

- Computing cost
- Data storage cost
- Data transfer cost

Our problem is to find out efficient schedule for workload execution in cloud environment which can achieve a certain level of QoS with optimal execution cost. A schedule should give answers to 2 questions:

- How much resource we need to acquire from cloud providers to complete a workload?
- How to allocate acquired resources to every task in a workload?

CHAPTER 3

EVALUATION CRITERIA

This section presents literature selection process and a set of criteria used to evaluate papers that are relevant to our research topic.

3.1 Literature Selection

Before going further into the review work, we should ensure that we identify and evaluate relevant literatures which may contribute to the solution of our problem. We have searched papers in the following databases:

- IEEEXplore (http://ieeexplore.ieee.org)
- ACM (http://portal.acm.org)
- Springer (http://www.springerlink.com)
- GoogleScholar (http://scholar.google.com)

It is obvious that good keywords would give relevant results. Therefore, the following keywords have been used:

- "Cost" AND "Optimization" AND "Cloud Computing"
- "Optimization" AND "Workload" AND "Cloud Computing"
- ("QoS" OR "Quality Of Service") AND "Workload" AND "Cloud Computing"
- "Scheduling" AND "Workload" AND "Cloud Computing"
- "Cost-Efficient" AND "Scheduling" AND "Cloud Computing"

Even the keywords point directly to our problem, the search results are usually quite huge. It is not feasible to take into consideration all the papers retrieved in the search results. We have performed additional steps to filter out:

- **Step 1:** Remove the papers which are not directly related to our problem (can be realized by reading the abstract or introduction section)
- **Step 2:** The priority of a paper is considered to be proportional to its number of citations
- **Step 3:** If two papers have the same number of citations then the newer one will be considered to be more valuable

After these steps, we have short-listed a certain number of papers with highest priorities which we believe we could review within the internship timeframe.

3.2 Evaluation criteria

3.2.1 Type of Workload

As mentioned earlier, cloud computing infrastructure can be used to run a wide range of applications, including both ”online” applications (i.e web-, database- or application servers) and ”batch-type” workloads (i.e payroll processing, scientific calculations or simulations). The type of workload has significant influence on the optimization algorithm. In fact, there is no generic optimization solution which can handle all type of workloads.

Although, ”online” workload has been studied in few papers, the majority of studies on optimization problem tackle only ”batch-type” workloads. It is realized during our search for relevant literatures. Papers reviewed in this report have been selected based on the process mentioned in previous section. There is no rule giving favor to a particular type of workload. However, it is shown that 8/10 papers selected (80%) concentrate on batch-type workload.

To our knowledge, one main reason is that many scientific analyses are expressed as batch-type workloads. They can be used to solve problems in many disciplines such as video coding/encoding, DNA sequence analysis and particle physics simulations etc. In addition, many studies in cloud computing learn and enhance the research results in grid computing, which mainly focus for scientific applications.

With batch-type workload, the QoS requirements are mainly defined by a specific deadline in which the workload must be completed and/or a maximum budget that a user can spend on the workload execution. Basically, a batch-type workload consists of multiple *”atomic”* tasks. The workload execution time is measured as the time duration from the first job in workload submitted until the last job completely executed. The optimization algorithm is highly dependent on the characteristics of the workload, including but not limited to:

- Tasks can be classified into groups which are optimized to fit different use cases

- Whether tasks are divisible
- Whether tasks in workload are independent or dependent (there is precedence constraint among jobs). As can be realized from [25][21] and other studies, dependent tasks are often represented as Directed Acyclic Graph (DAG) in which the vertices are the tasks themselves and the edges are the constraints(i.e data dependancies or bandwidth etc.)
- Whether tasks are executable by multiple instance types
- Whether tasks have uniform or individual deadlines/budget constraints

However, it is not a rare case that a system consists of both types of workload running at the same time. For example, a software system from a bank which receives request from users, performs end-of-quarter processing and then export continuously the report back to the users. The system can serve a vast number of users and interact with the users via Web interface.

In such system, due to different characteristics, each type of workload would have different optimization solutions. Therefore, it will introduce more complexity in order to combine the results from different approaches.

Several papers do not clearly describe the type of workload handled in their solutions. Instead, they assume to be able to convert workload into computing power (i.e a certain number of VM-hour in cloud) or a number of required VMs. However, the conversion algorithm is often not specified.

3.2.2 Quality of Service

The term QoS originated in the field of telecommunications and computer networks but it is gradually expanded to use in other areas as well. ITU-T Rec. E.800 defines QoS as *"the collective effect of service performance which determine the degree of satisfaction of a user of the service."*

ITU-T Rec. G. 1000 goes further in the definition of QoS: *"from different perspectives: Customers QoS requirements; Service providers offerings of QoS (or planned/-targeted QoS); QoS achieved or delivered; Customer survey ratings of QoS"*

QoS requirements are driven by bussiness demands and normally specified in Service Level Agreements (SLA). A SLA is an important document describing what the definition of 'satisfactory' is. A SLA can either be an informal contract between parties or a legally binding contract.

Regarding software systems, several Quality models exist. Among all, one of the most relevant was established in ISO9126 which classified the software quality in a set of characteristics as follows: [12]

- **Functionality:** Suitability, Accuracy, Interoperability, Compliance, Security
- **Reliability:** Maturity, Recoverability, Fault Tolerance
- **Usability:** Learnability, Understandability, Operability
- **Efficiency:** Time Behaviour, Resource Behaviour

- **Maintainability:** Stability, Analyzability, Changeability, Testability
- **Portability:** Installability, Replaceability, Adaptability, Conformance

We have to mention here that when considering a workload execution on a software service system, even though all that QoS parameters should be taken into account to some extent, depending on the types of concerned workload, one parameter can have higher priority to the others. For example, with Web or database applications, QoS is often measured by response time. It means the time between a request is sent (or received) and when its response is received (or sent). On the other hand, with service that performs a number of independent tasks (compute- or data-intensive), completion time is considered to be more important.

To our knowledge, among all QoS metrics mentioned earlier, deadline and budget constraints are of the most motivated to many studies relating to optimize workload execution on cloud environment.

3.2.3 Pricing Model

In theory, as the main characteristic of cloud computing is Pay as You Go, cloud computing is supposed to offer a straightforward model for consuming resources. Unfortunately, cloud pricing models are complicated which makes purchasing decisions for the users and the comparison across multiple providers a challenge. The pricing model selected will have big impact on the execution cost of workload. The two most common pricing models offered by all cloud providers are

- **On-demand:** users pay a fixed rate for an hour of usage without upfront payment or long-term commitment
- **Reservation (prepaid):** allows users to make a low, one-time payment in order to reserve instances that they plan to use in the future. As a trade-off, the users benefit from significant discount on instance-our rate for such reserved instances.

Amazon Web Services delivers a special type of virtual machine, called Spot Instance. The rates of Spot Instances fluctuate depending on the market supply and demand. [1] takes advantage of Spot Instance features to propose an optimization solution using a probabilistic model.

The optimization models are different depending on the instance selection strategy. On-demand is used in almost all scheduling approaches while reservation is taken into consideration in few studies (i.e [5][4]). In all papers reviewed, only [4] takes all pricing plans in its algorithm, including Spot Instance.

3.2.4 Deployment Model

Deployment model identifies the purpose of the cloud and the nature of how the cloud is deployed. As introduced in Section 1.1.4, there are 4 deployment models defined but only 2 of them are often taken for optimization problem:

- Public cloud

- Hybrid cloud

Challenging problems faced when moving an application to public cloud has been mentioned throughout previous sections of the report. We will discuss more about hybrid deployment in this section.

Hybrid cloud has been predicted to be the dominant type for most organization [2]. In Hybrid deployment, cost-efficient scheduling tends to maximize the utilization of the internal infrastructure and minimize the cost of running outsourced tasks to the public cloud [7].

According to [7], the private cloud infrastructure is assumed to cost its owner nothing and the optimization algorithm takes into consideration only outsourced tasks. However, in practice, the size and cost of private infrastructure cannot be ignored. They should be addressed in the algorithm as well. To my knowledge, we have not found any study concerning these parameters in the optimization approach.

3.2.5 Optimization approach

In this section, we will give brief introduction to some main techniques which are used to solve the cost-efficient task scheduling problem.

(Non-)Linear programming

Most of the studies suggest building up a system model which can be eventually formulated to an (non-)linear programming problem. Depending on the objectives, assumptions and approaches of the studies, the constraints and objective functions might be established differently to have acceptable solution in terms of time and accuration.

In [22], Ming Mao et al. proposed a model to automatically scale computing instances based on workload information and performance desire. The constraint equations take into consideration the delay on instance startup and shut-down activities. [22] also discussed the problem of distributing existing resources to tasks in the workload. The auto-scaling mechanism is then reduced to solving several integer programming problems. [21] extends the mechanism to cloud workflows composing of tasks with precedence constraints.

As can be shown in Chapter 4, (non-)linear programming is applied to solve the problem in many other studies, but with different optimization models.

In [7], a binary integer program is used to tackle the problem of scheduling deadline constrainted workloads in a cost optimal way on hybrid cloud. Therefore, several constraint functions applicable to private clouds are included. The algorithm does not evaluate on multiple cloud providers because of increasing complexity.

When uncertainty parameters (i.e resource demand or prices) are taken into consideration, some papers (i.e [5][4]) use stochastic programing (SP) model to obtain the optimal solutions. Algorithms like deterministic equivalent formulation (DEF), sample-average approximation (SAA) or Benders decomposition then can be applied to solve SP problem with reasonable time and computation complexity.

Particle swarnm optimization

In PSO, each particle is the candidate solution of the underlying problem and has n dimensions which are decided by special problem. Particles position and velocity will be initialized with random values. Each particle has a fitness value which will be evaluated by a fitness function to be optimization in each generation [11]. When the algorithm executes over a certain number of interation, the idea results can be retrieved.

For example, in [11], the authors develop a PSO algorithm which is based on small position value rule to solve the task scheduling optimization problem. Each task schedule is represented by a particle in the PSO. The proposed PSO algorithm is proved to be suitable to cloud computing, converges and runs faster than other two PSO algorithms (PSO in crossover and mutation, PSO in the local research)

Probabilistic model

This technique is to carry out a number of experiments on exemplary workload (real or simulation) and virtual machines acquired from cloud providers. Based on the experimental results, a probabilistic model is achieved which shows distribution of concerned QoS variables. Such distribution can be used to identify the optimal schedule with a given confidence and evaluate on how the cost level is expected to change with provided QoS.

Monitor-control loop

The monitor-control loop technique allows the algorithm to adapt to dynamic changes in workload or VMs configuration. Through a number of monitor-control iterations, the algorithm attempts to reduce resources provisioning cost to reach the optimal value based on the uncertain information (i.e resources demand and price etc.) observed at each iteration.

3.2.6 Scalability

Scalability is another criterion that is used to evaluate the papers. In general, scalability is ability of the algorithm to handle a growing value of input. Regarding cost-efficient scheduling algorithm, by learning from the reviewd papers, we think scalability can be expressed in several metrics including but not limited to:

- The size of the workload that the algorithm can handle
- Execution time of the algorithm with reasonably-sized workload
- Number of cloud providers supported
- Number of VMs and VMs classes supported

It is noted that even the algorithm is scalable, only a small-scaled experiment is carried out and reported in reviewed papers. In fact, not so many papers clearly mention the scalability of the algorithm. Therefore, in the scope of this report, we will evaluate the scalability of the algorithm based on the experimental environment used in the papers. However, the experiments do not show all the above metrics.

CHAPTER 4

BIBLIOGRAPHICAL REVIEW

4.1 Summary

In this section, we discuss in detail some of the work related to our problem. Table 4.1 and 4.2 summarize the reviewed papers based on criteria in Chapter 3 for a quick reference.

Table 4.1: Bibliographical Summary

		WORKLOAD			**PRICING MODEL**		
No	**Paper**	**Type**	**Characteristics**	**Optimization goals**	**OD**	**Reserved**	**Spot**
1	[7]	Batch-type	Independent	Execution cost	Yes	No	No
2	[22]	Batch-type	Independent	Execution cost	Yes	No	No
3	[28]	Batch-type	Independent Data-intensive	Execution cost	Yes	No	No
4	[5]	Computing power	N/A	Total cost of resource provisioning	Yes	Yes	No
5	[4]	Computing power (server-hours)	Divisible	Total cost of resource provisioning	Yes	Yes	Yes
6	[25]	Batch-type	Dependent Data-intensive	Execution cost	Yes	No	No
7	[29]	Batch-type	Independent	Execution cost	Yes	No	No
8	[21]	Batch-type	Dependent	Execution cost	Yes	No	No
9	[11]	Batch-type	Independent Data- and compute-intensive	Processing time and execution cost	Yes	No	No
10	[1]	Batch-type	Independent	Execution cost	No	No	Yes
11	[6]	Batch-type	Independent	Execution cost	Yes	No	No

Table 4.2: Bibliographical Summary (continue)

			SCALABILITY			
No	**Paper**	**Deployment**	**Optimization approach**	**Scalability**	**Multi Clouds**	**Experimental workload**
1	[7]	Hybrid	Binary integer programing	N/A	Yes	Simulation
2	[22]	Public	Integer programming	90 instances 45 concurrent tasks	Yes	MODIS and simulation
3	[28]	Public	Integer programming	4 cloud providers 6 data centers 144 storage systems	Yes	MODIS and BLAST
4	[5]	Public	Stochastic programming	4 providers 2 VMs classes	Yes	Simulation
5	[4]	AWS	Stochastic programming	20 instances	No (AWS only)	Simulation
6	[25]	Public	Non-linear programing	less than 2s for 2000 iterations	N/A	KDD'99
7	[29]	Hybrid	Monitor-control loop	N/A	N/A	PSA
8	[21]	Public	Monitor-control loop	N/A	Yes	Simulation
9	[11]	Public	Particle swarm optimization	250 processors 10,000 tasks	Yes	Simulation
10	[1]	Public	Probabilistic model	20,000 tasks	No (AWS only)	BOINC Catalog and Grid Workload Archive
11	[6]	Hybrid	Heuristics	1200 app./week Max. 100 tasks per app.	Yes	Simulation

4.2 Bibliographical review

4.2.1 Cost-optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads [7]

General description

In this paper, the authors develop a linear programming-based approach to formulate the scheduling problem faced in Cloud Procurement Endpoint (CPE) in hybrid deployment model. The objective of proposed algorithm is to maximize the utilization of the internal infrastructure and to minimize the cost of outsourced tasks execution in the public cloud.

Although the proposed program takes into account data transfer costs, data transfer times are assumed to be negligible.

Applications' information including tasks' runtime are assumed to be known in advance or can be easily estimated. The proposed binary integer program is evaluated with regard to its computational cost for scheduling applications in both a public and a hybrid cloud setting.

Workload model

The applications handled by algorithm in this paper is of batch-type workload with following characteristics:

- Application consists of trivial parallel tasks
- Tasks in application are independent without precedence constraints
- Each application has strict completion deadline. It also means that all tasks in an application should be finished before its deadline.
- Tasks runtime on each type of instances are known in advance or can be estimated

Optimization approach

An binary integer programming (BIP) is developed based on the workload model. The goal of the program is to deploy a number of applications while minimizing the total cost of their execution on the consumer. Objective function includes both computational cost and data traffic cost over all time slots within the schedule.

Running an application on the private cloud is assumed to be free of cost, and the applications data sets are available free of charge and without a transfer delay. In fact, this assumption is not practically true because the owner still bears some expenses to maintain internal infrastructure (i.e power supply or human resource expenses etc.)

The limited capacity of internal infrastructure (number of CPU and amount of memory used) is described as constraints in the BIP. In addition, other important constraints are introduced, such as

- Each task can be executed only in one instance type from one cloud provider

- All tasks in a application must be finished before the deadline of that application.

The authors use IBM's linear programming solver CPLEX to solve the problem with multiple inputs

Evaluate

In order to evaluate the proposed scheduling solution, the experiment is carried out with 3 cases including public, private and hybrid settings. The scenarios used in the experiments below are assembled synthetically in order to provide an insight in the correct functioning and performance of the approach.

Experimental results show that the scheduling approach seems to perform well both in terms of cost minimization, feasibility and scalability.

4.2.2 Cloud Auto-scaling with Deadline and Budget Constraints[22]

General description

The paper presents an algorithm to scale computing instances based on workload information and desired performance with lowest monetary cost. The performance is expressed as deadline and budget constraint respectively. As an improvement compared to other studies, the algorithm takes into account the start-up and shutdown activities (e.g acquisition and termination delay from cloud providers). In the algorithm, a mathematical model is built up to cover the computing requirement of workload, the computing power of instance types. Optimization problem is led to the solution of an integer programming problem. The result of the paper proves that an schedule using different instance types can be selected which is proved to save the cost as compared to schedule using a single instance type.

Additionally, the paper proposes a solution to allocate computing power of existing instances to jobs in the workload. It means on-demand and prepaid (reserved) instances can be used in the schedule.

Workload model

- Workload consists of multiple non-dependent jobs
- An instance can process a single job at one time
- All jobs have the same deadline constraint
- Jobs are grouped in different classes, such as compute-intensive and I/O intensive classes

Optimization approach

In order to complete the submitted jobs before the deadline, the idea is to ensure that the computing power of available instances is large enough to handle the workload. A computing power unit is defined and computed for all available instance types. The computing power of an instance is represented by a 2 dimensions vector with one dimension is the job class and the other is the number of jobs can be finished for that job class before the deadline on such instance type. With this definition, the workload can be similarly represented in the same way.

As mentioned earlier, the currently running instances are taken into consideration when the optimization algorithm is designed. The distribution algorithm is rather simple. The computing power of such instances will be allocated to submitted jobs proportionally to their execution time.

There are 2 cases examined

- Computing power of existing instances is greater than computing power required by workload. In this case, a number of instances should be considered to shut down in order to save the cost.
- Computing power of existing instances is not sufficient to execute the workload.

The instances which are in start-up delay would not contribute any computing power to execute the workload.

In the first case, an instance is considered to shut down only when it is reaching its full hour operation. The computing power of available instances without the concerned one will be recalculated. If it is still capable of completing the workload within deadline then that instance will be terminated. The paper does not suggest the solution when multiple instances may reach the end of full hour operation at the same time.

In the second case, more instances need to be acquired in a way that minimizing the cost and maximizing the computing power. By representing workload and computing power of instance as vectors in addition to the fact that submitted jobs can be executed by multiple instance types, the model bring the optimization problem to several integer programming problems. However, the approach in this paper is flexible enough to not depend on hard deadline as only performance metric. A user can have different requirements based on their specific workload such as minimum number of running instances. These performance metrics are expressed as different objective functions.

The proposed mechanism is realized to an implementation in Windows Azure. The evaluation has been done using both simulations and a real scientific application (MODIS). Experiment shows that the mechanism does allow submitted jobs meeting the provided deadline and give the cost saving of 20%-40% as compared to fixed instance type and only 15% more compared to optimal option.

The paper discusses the auto scaling mechanism which at first sight seems to be different to our problem. However, in case the workload information is aware in advance, auto scaling mechanism can be used to figure out the optimal option (in terms of monetary cost) to execute such workload with a given deadline constraint.

4.2.3 Optimization of Resource Provisioning Cost in Cloud Computing[5]

General description

With reserved pricing model, the cloud consumers are allow to reserve the resources in advance based on their future demand. The authors introduce two cases, *"under-provisioning"* when the reserved resources are unable to fully meet the demand and *"over-provisioning"* when the reserved resources are more than the actual demand. In the former case, on-demand resources should be acquired to handle extra demand. It is important for the cloud consumers to minimize the total cost of resource provisioning by reducing the on-demand cost and over-subscribed cost of under-provisioning and over-provisioning. To achieve this goal, the optimal computing resource management is the critical issue.

The paper introduces an Optimal Cloud Resource Provisioning (OCRP) algorithm to minimize the total cost for provisioning resources in a certain time period. In order to make an optimal decision, the uncertainty of consumers' future demand and providers' resource prices are taken into account to adjust the trade-off between on-demand and over-subscribed cost.

Workload model

Workload is not clearly described in this paper. Instead, the workload is expressed as a number of required VMs.

Optimization approach

The provision resources schedule is divided into a number of provisioning stages. A provisioning stage is the time epoch when the cloud broker makes a decision to provision resources by purchasing reserved and/or on-demand plans. Each provisioning stage consists of multiple provisioning phases. There are three provisioning phases including reservation, expanding and on-demand. During these provisioning phases, the reserved resources could be observed to decide whether additional on-demand resources is needed.

The uncertainty of future demand and providers' prices are taken as scenarios and described by probability distributions in stochastic integer programming.

With above system model and assumption, the stochastic programming with multistage recourse is presented as the core formulation of the OCRP algorithm. After the original form of stochastic integer programming formulation is derived, the paper propose three approaches which can be used to obtain the solution of the OCRP algorithm.

In first approach, the formulation is transformed into the deterministic equivalent formulation (DEF) which can be solved by traditional optimization solver software.

These other two approaches include sample-average approximation (SAA) and Benders decomposition. Benders decomposition breaks down the optimization problem into multiple smaller problem which can be solved independently and parallelly. On the other hand, SSA solves a set of selected scenarios by DEF and obtain optimal solution using numerical method. The SAA is believed to be more efficient in case the number of scenarios is numerous.

Evaluation

Experiment shows that the algorithm can optimally adjust the trade-off between reservation of resources and allocation of on-demand resources. The OCRP algorithm can be used as a resource provisioning tool for the emerging cloud computing market in which the tool can effectively save the total cost

4.2.4 Cost Minimization for Provisioning Virtual Servers in Amazon Elastic Compute Cloud[4]

General description

The paper addresses optimal virtual machines provisioning (in terms of cost) under different pricing models (including on-demand, reservation and spot options) and demand uncertainty. Two algorithms are developed for long- and short-term planning respectively. The cost optimization of resource provision under price and demand uncertainty in cloud computing and other similar models (e.g., grid computing) that considers both long- and short-term plans is first studied in this paper.

In order to deal with uncertainty of price and demand, stochastic programming (SP) robust optimization and sample-average approximation (SAA) are used to obtain the optimal solutions of the algorithms.

Stochastic programming is commonly used to solve the optimization problem in other papers. In this paper, not only resource demand uncertainty is taken into the stochastic model but also fluctuated spot prices. With the fact that only Amazon Web Services offers spot price so far, it seems that the algorithms are not applicable to other providers. However, it is stated that by adapting the characteristics of other providers to the models, users can benefit from the algorithms as well.

Workload model

An application in [4] is described as the certain amount of server-hours to finish its workload processing, regardless of the type of workload. The applications can be partitioned and assigned to the instances acquired from cloud providers. The amount of server-hours is referred to as resource demand.

Optimization approach

[4] proposes 2 algorithms to optimally purchase the provisioning options for long- and short-term. *"The on-demand option is considered by both algorithms. The long-term provisioning algorithm provisions instances for being utilized in a long time period, i.e., one or three year. Thus, the reservation option is exclusively considered by the long-term provisioning algorithm to obtain the optimal number of reserved instances. On the other hand, the short-term provisioning algorithm provisions instances for being used in short time periods, e.g., several minutes to few hours. Since the spot price could reduce the total provisioning cost in short periods, the spot option is exclusively considered by the short-term provisioning algorithm to obtain the optimal amount of bid server-hours.*

Both algorithms obtain the optimal options by formulating and solving stochastic programming models. The models are divided into 2 stages. The first stage is *here-and-now* stage in which the decision is made before the values of uncertainty variables are observed, followed by *wait-and-see* stage when such values are aware.

- In long-term algorithm, mainly uses the reservation option for the here-and-now decision, since this option has the fixed discounted price and guarantees the availability of reserved instances for customers.
- In short-term algorithm, the purpose is to yields the optimal number of bid spot instances through the here-and-now decision. The short-term algorithm is

repeatedly execute with a significant number of scenarios (values of uncertainty parameters) so the authors propose using SAA approach to solve SP problem.

Evaluation

The experimental shows that that the algorithms can significantly reduce the total provisioning cost incurred to customers.

4.2.5 Minimizing Execution Costs when using Globally Distributed Cloud Services[25]

General description

In this paper, the authors formulate a non-linear programming model to minimize the data retrieval and execution cost of data-intensive workflows in clouds. The model model retrieves data from Cloud storage resources such that the amount of data transferred is inversely proportional to the communication cost.

Different from majority of other studies where they concentrate on compute-intensive application and computing cost, in this paper, data cost (including storage and transfer cost) is the deciding factors on how to optimally deploy the application in cloud servers .

The ”intrusion detection” is used throughout the paper as application workflow because it has all the features of an data-intensive application. One of the main feature is that the application's data logs are globally distributed in cloud storage servers.

It is noticed that the proposed algorithm does optimize the cost but does not take execution time into account. Therefore, in some cases, in order to have optimal cost, the application may have to wait unreasonable time to finish.

Workload model

The author take an example of detecting the spread of malicious worm over Internet as workload. This workload has following characteristics:

- Consists of multiple tasks
- Tasks have precedence constraints
- Tasks are data-intensive

Optimization approach

The application workflow is denoted as a Directed Acyclic Graph (DAG), where the vertices is the set of tasks, and the edges represents the data dependencies between these tasks. Several assumptions are made including

- The cost of computation of a task on a compute host is inversely proportional to the time it takes for computation on that resource.
- The transfer cost is fixed by the service provider or can be calculated according to the bandwidth between the sites
- The cost of unit data access from a storage resource to a compute resource is known

With such model, the cost-optimization problem: *"Find a feasible set of partial datasets that must be transferred from one storage host to one compute host for each task such that the total retrieval cost and computation cost of the task is minimal, for all the tasks in the workflow"*. Obviously, the algorithm should not violate the dependencies among tasks.

To obtain the minimum cost, the authors formulate a non-linear program for the cost-optimization problem. In constraint functions, the tasks will be mapped onto resources based only on cost optimization objective (not time), hence time dependencies between tasks are eliminated. In fact, the solution proposes only schedule to minimize the execution cost but not the makespan. A task has to wait before its parents finish execution.

Evaluate

The authors compare the cost of multiple executions of the workflow given by a solution of our non-linear program against that given by Amazon CloudFronts nearest single data source seection. The results show a savings of three-quarters of total cost using this model.

4.2.6 Adapting Market-Oriented Scheduling Policies for Cloud Computing [29]

General description

When the local resources are not enough to meet the workload execution deadline, the computational power of IaaS providers can be used to compensate for such limited capacity. In fact, there is a trade-off between investing on local resources or spending budget to get resources from IaaS.

The paper proposes 2 market-oriented policies that aim at satisfying the application deadline by extending the computational capacity of local resources via hiring resource from Cloud providers. These policies include:

- **Time Optimization:** minimizing time, within time and budget constraints
- **Cost Optimization:** minimizing cost, within time and budget constraints

Workload model

The authors do not clearly specify which type of workload is handled by the proposed policies. However, based on the algorithm description and the experiment, it can be seen that workload consists of multiple independent tasks. Workload prediction issues are out of scope of the paper so it is assumed that all tasks of the application have the same execution time.

Optimization approach

Time Optimization Policy

The aim of this policy is to complete the application as quickly as possible, within the budget. Therefore, the policy is quite simple. The scheduler spends the whole available budget to acquire as much resources as possible from IaaS provider and assign the tasks to those resources. Obviously, the capacity of acquired resources should not exceed the remaining tasks.

Cost Optimization Policy

In this policy, the cost is optimized by executing repeatedly an monitor-control process. At each iteration, the algorithm computes demanded resources for remaining tasks. If there is available budget and existing resources is not sufficient to complete remaining tasks within deadline, more resources would be acquired. On the other hand, unneeded resources would be terminated to save the cost. However, the authors do not indicate whether the algorithm can handle the case when the existing resource and available budget are not sufficient. They simple state that *If there is not enough budget, then [...] terminates each hired resource before it starts a new charging cycle."*

Experiment

Parameter Sweep Application (PSA) is used in the experiment, together with Gridbus as user-level broker and virtual machine Amazon EC2. The experiment shows that different workload types can get completed before the deadline and within the budget using the proposed policies.

4.2.7 Auto-Scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows[21]

General description

The paper is in fact an enhancement to the auto-scaling mechanism proposed in [22]. In [22], the solution is only applicable to a small set of workload, neither dependent tasks nor tasks with individual performance requirements were supported. In this study, the authors tackle the auto-scaling mechanism for a more general application where job are expressed as workflow and allow non-uniform job deadlines. The goal of the solution is similar to that of [22], is to ensure that all tasks can be completed before their respective deadline with a minimum monetary cost. However, it is worth noting that the deadline constraint in this paper is not extremely strict, meaning missing a deadline is not treated as a complete failure of the system. Deadline constraint is considered as a metric of performance requirement specified and expected by the users.

Workload model

- A workload consists of multiple job instances.
- Each job, in turn, is composed of multiple tasks with precedence constraints
- All jobs are submitted into a single entry unit. Different jobs contain different tasks
- Jobs can be categorized into different classes based on their processing flows. A job class is represented by a directed acyclic graph (DAG) with a deadline.
- Different deadlines can be assigned to different job classes (not tasks).
- Depending on the running VM, a task can achieve different performance.

Optimization approach

With the assumption that the workload is continuously changing and there could be many job instances submitted, the auto-scaling is a repeated process. Periodically, the auto-scaling process would be initiated and run to calculate the execution schedule with updated information.

Two types of plan is introduced with different objectives:

- **Scheduling Plan:** is to determine the instance type for each running tasks at a point of time.
- **Scaling Plan:** is to determine the number of instances for each instance type at a point of time.

The authors propose the algorithm going through 2 decision stages, respectively.

Scheduling Stage

As mentioned above, a job class is expressed as a DAG. In order to break the task precedence constraints and be able to treat each task independently, a deadline assignment process is proposed. The approach is to assign individual deadlines to tasks proportional to the execution time on their most cost-efficient machines. However, the authors have not given argument on why they chose such approach and explain if it has any advantage as compared others such as allocate deadlines proportional to the tasks fastest execution time.

If with such deadline assignment, the jobs cannot be completed within the deadline, task with the highest cost-efficient rank will be schedule on the faster but more expensive machine. The process is repeated until the job execution time is less than the deadline. After this stage, the optimal mappings between a task and an instance type are known.

Scaling Stage

In this stage, a load vector is calculated for each task. The load vector represents the number of required machines to finish tasks on an instance type. This instance type, which is best suited to a task, is achieved from the Scheduling stage. This is done for all tasks and resulted in a number of load vectors, one for each machine type. If it can be ensured that the number of existing machines is always greater than or equal to the load vector at any time, the tasks will finish within the assigned execution interval.

In order to make use of hour-unit pricing model of VM, the paper suggests additional instance consolidation step. This step is to consolidate partial instance hours to run tasks (even though regarding tasks, such instance is not the most cost-efficient instance) because it is believed to give saving on cloud resources.

After sclaing stage, the instance types and its corresponding number are aware. The Earliest Deadline First (EDF) is then used to schedule tasks on each VM type.

Evaluation

The approach is evaluated using 3 types of applications (Pipeline, Parallel and Hybrid) with different workload patterns and deadlines. As compared to 2 other approaches (Greedy and GAIN), it shows cost savings from 9.8% to 40.4%.

4.2.8 Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm[11]

General description

The paper introduces a PSO algorithm, based on small position value rule (PSO embedded on SPV), to schedule tasks in cloud computing which can minimize overall time of executing and transmitting. The application type considered in this paper is the hybrid of computing intensive and data intensive types. In the execution of the application, the authors do not pose any specific performance requirement such as deadline or budget constraints. However, some useful ideas from the paper can be applicable to cost optimization problem.

Workload model

The authors do not have separate section discussing detail of concerned workload-/applications. However, throughout the paper, we can summarize some main characteristics of the application as follows:

- Application consists of multiple *atomic* tasks
- Tasks are both computing and data intensive.
- Execution cost of a task is dependent on the capacity of a processor it is running on
- The communication bandwidth between 2 tasks may change over the time

Optimization Approach

In the paper, the task scheduling is modeled as follows.

- Task scheduling is described as a task interaction graph (TIG) in which vertices represent the tasks of application and edges indicate the information exchange between these tasks.
- A node is defined as a processor center where multiple tasks can be assigned to.

To have better understanding, the paper has a separate section giving a brief description of PSO algorithm. In general, *"The PSO algorithm is similar to other evolutionary algorithm. In PSO, each particle is a candidate solution of the underlying problem and has n dimensions which are decided by special problem. Particles position and velocity are initialized randomly. Each particle has a fitness value, which will be evaluated by a fitness function to be optimization in each generation. Each particle knows the pbest and gbest (personal best position and social best position)"*. In each generation the velocity and the position of particle will be updated toward the optimal solution.

The PSO embedded in SPV algorithm will be applied to above model in order to find out the optimal schedule. Each scheduling option is mapped to a particle. In fact, particles are n dimensions vector with each element delegates a task. The initial particle population is generated randomly. According to the paper, *"the algorithm begins with k random particle vector and each particle is n dimensions. Every particle vector is a candidate solution of the underlying problem. The particles are the task to be assigned and the dimensions of the particle are the number of the special tasks in a workflow. Then, each particle moves by the direction on the pbest and gbest until the maximal number of iterations. When the algorithm executes over, the gbest and fitness value are the corresponding task scheduling and the minimal cost of the optimal strategy."*

Evaluation

The experiment shows that PSO algorithm has faster converges time and faster run time as compared to the other algorithm including CM-PSO and L-PSO.

4.2.9 Decision Model for Cloud Computing under SLA Constraints[1]

General description

The paper proposes a probabilistic model for the optimization of monetary cost with given QoS requirement and reliability from the user. The model takes advantage of Spot Instance price in AWS to lower the execution cost. It is based on the fact that although the Spot Instance price fluctuates depending on the market supply and demand, in general the prices are cheaper than that of identical On-Demand Instance type.

The authors carried out a number of experiments with different sets of input parameters in order to build up the distribution of monetary cost and execution time for available instance types and different bid prices. By studying such distribution, a user can choose an option to gain optimal cost which still meets the deadline constraint with a given confidence (probability).

Workload model

The workload studied in this paper, both simulated and exemplary ones, are of batch-type workload with following characteristics

- Job is divisible to a number of atomic tasks
- Jobs are compute-intensive
- An atomic task can be executed and finished solely by one instance one time
- Multiple tasks can be run in parallel by multiple instances
- Job is completed by one instance type, meaning all tasks in the job use only one instance type.
- Regarding one instance type, atomic tasks in the job have the same task length (task execution time)
- Users' requirement on performance is expressed as deadline and budget constraint

Optimization approach

First, a number of random variables are defined including

- Execution time ET: is the total clock time needed to process part of workload that is assigned to 1 instance
- Availability time AT is the total time in bid, meaning the duration of time that the instance is really up and run
- Expected price EP is the average price for one instance per hour computed during the execution time of atomic task
- Monetary cost M is the amount to be paid by the user per instance

Checkingpoint strategy is also taken in to account because it will influence the distribution of the earlier-mentioned random variables. Two checkingpoint strategies considered in the paper are optimal (OPT) strategy and hourly (HOUR) strategy.

Secondly, the authors used Monte Carlo method with real Spot Price trace to find out the distribution of the random variables. Each random variable provides additional characteristic of the job execution and are beneficial in defining more advanced SLA condition. The distribution is obtained via 10,000 experiments (per unique set of parameters). The main input parameters include instance type, bid price, task length and checkpointing strategy. It is assumed that the task length has linear relationship with the processing speed of an instance. The objective of building such distribution is answer 2 questions:

- Whether the deadline constraint can be achieved with a given confidence
- Whether the budget constraint can be achieved with a given confidence

The distribution of random variables gives the user a kind of look-up table. Based on such look-up table, the user can have information on how likely his job can be completed by an instance type with a given requirement (constraint) and a specific bid price. Therefore, the user can choose the option which offers the optimal cost from many feasible options

The experiments are carried out with a specific set of input parameters and workload. It is not clear whether this model can be generalized to other type of application. In addition, in order to have the results with completed set of input parameter values, a huge number of experiments have to be done.

The approach does not propose a user with optimal option automatically. Instead, the user needs to manually compare on multiple options and decide on the optimal one.

4.2.10 Heuristic for Scheduling Deadline Constrained Workloads on Hybrid Clouds [6]

General description

The paper proposes a heuristic to tackle the hybrid optimization problem that are able to operate on larger-scale problems.

As compared to [7], the authors extend a cloud model with data transmission speeds in order to take data locality into account during the scheduling process, and support the online arrival of applications. The influence of the different cost factors and workload characteristics on the heuristics cost savings is evaluated, and the results sensitivity to the accuracy of task runtime estimates is discussed.

The public scheduling component thereby takes into account data transfer costs and data transfer times. A hybrid scheduling mechanism is presented to take into account the potential cost of an application when scheduled on a public cloud provider.

Workload model
Similar to [7]

Optimization approach
In order to obtain the optimal cost in hybrid deployment model, the paper proposes a solution consisting of three loosely coupled components

Public cloud scheduler

Public cloud scheduler is to decide for incoming applications of which public cloud providers to execute. The mechanism is quite simple. The scheduler calculates the cost for running the application on each available cloud provider, and schedules all tasks of an application on the cheapest cloud provider.

Private cloud scheduler

Private cloud scheduler to schedule incoming applications on internal infrastructure. Private infrastructure is modeled as a simple queuing system in which incoming requests for an applications tasks are handled in a first-come first-served manner without backfilling. However, the number of tasks running concurrently on private cloud is limited.

Hybrid cloud scheduler

Hybrid cloud scheduler to decide whether incoming applications can be deployed on internal infrastructure or outsourced to public cloud. There are 2 approaches for hybrid cloud scheduler.

- **Private First:** Basically, the scheduler adds all incoming applications to the queue of the private cloud, unless the queue wait time for that application indicates that meeting the applications deadline is impossible by the sole use of the private cloud. In that case, the application is scheduled on the cheapest public cloud provider. Because the runtimes of all tasks in the queue are known, calculating the expected wait time in the queue for a set of tasks is trivial. However, Private First may not provide optimal schedule because it is highly dependent on the arrival order of applications.

- **Cost Oriented:** In this approach, the cost of running an application on public cloud within deadline is calculated on arrival. Priority to private cloud will be given to the most expensive applications. Obviously, an applications cannot be scheduled on arrival because it needs information on the potential cost of other applications. The problem is how long the application need to wait before it can be assigned. The longer it waits the more information is retrieved but the higher the cost is because the application needs to use more expensive instance types to be completed within shorter deadline. The authors propose a strategy to decide on a suitable waiting time which is the minimum of:

- The time up to which the applications cost of execution remains constant and no instance type switch is required
- Or the time up to which adhering to the applications deadline remains feasible.

Evaluate

Experimental results show that a cost-oriented approach pays off with regard to the number of deadlines met and that there is potential for a significant cost reduction, but that the approach is sensitive to errors in the user provided runtime estimates for tasks.

4.2.11 A Model and Decision Procedure for Data Storage in Cloud Computing[28]

General description

The paper use a mathematical model which takes a set of compatible storage service to build up an integer programming problem. Eventhough computing capacity is taken into account for the optimization algorithm, in this paper the data storage cost (including storage and transfer cost) is the decision factor. The algorithm supports selecting resources from multiple cloud providers. Performance requirements are described as latency, bandwidth and job response time.

The final goal of the paper is to *"select the beast storage system which meet the user's data requirement and optimize cost and/or access latency/bandwidth"*. Algorithm also handles the allocation of application to computational sites

Workload model

The workload is expressed as a number of applications with respective computation length. The computation length is fixed and not able to change regardless of which virtual machine it is running on.

Both the capacity of computational sites and applications' length are converted to computational hour.

Optimization approach

The solution is divided into 2 stages. The first stage is a matching process in which inputs are a list of user's requirements and the storage services' capabilities. The output of this first stage is a list of compatible storage services for each dataset in the application along with with cost and performance estimates. The user can choose a data allocation from these options to meet their requirements. However when the number of matched options is high, selection task will be challenging.

The second stage facilitates that task by solving an integer linear programming formulated based on output of first stage. The general idea is to include the cost, latency, and bandwidth as parameters in the objective function that needs to be minimized. Computation cost is introduced in the model by integer variables that represent the amount of computation required per month.

Evaluation

Experimental results based on simulation show that approach is scalable as the number of cloud providers, datacenters or storage services increase. With two real BLAST and MODIS applications, by combining local and cloud resources, the authors are able to halve the cost compared to a cloud-only approach or be 52% faster than a local-only approach.

CHAPTER 5

Conclusion

Scheduling applications in a cost-effective way on one or more cloud providers is a complex task, in which the research community is making a significant contribution. In this internship report, we have provided an overview of the available literature on cost-efficient scheduling applications in cloud computing. We have also proposed a set of criteria which is used to evaluate selected papers. By doing the literature review, it could help us to indentify research opportunities in this area.

Bibliography

[1] Andrzejak, Artur and Kondo, Derrick and Yi, Sangho. Decision Model for Cloud Computing Under SLA Constraints. In *Proceedings of the 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, MASCOTS '10, pages 257–266, Washington, DC, USA, 2010. IEEE Computer Society.

[2] Luiz Fernando Bittencourt and Edmundo Roberto Mauro Madeira. HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds. *Journal of Internet Services and Applications*, 2:207–227, 2011.

[3] Armando Escalante Borko Furht. *Handbook of Cloud Computing.* Springer.

[4] Sivadon Chaisiri, Rakpong Kaewpuang, Bu-Sung Lee, and Dusit Niyato . Cost Minimization for Provisioning Virtual Servers in Amazon Elastic Compute Cloud. pages 85–95. IEEE, 2011.

[5] Sivadon Chaisiri, Bu-Sung Lee, and Dusit Niyato. Optimization of Resource Provisioning Cost in Cloud Computing. pages 164–177. IEEE, 2011.

[6] Ruben Van den Bossche, Kurt Vanmechelen, and Jan Broeckhove. Cost-Efficient Scheduling Heuristics for Deadline Constrained Workloads on Hybrid Clouds. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference*, pages 228–235. IEEE, 2010.

[7] Ruben Van den Bossche, Kurt Vanmechelen, and Jan Broeckhove. Cost-Optimal Scheduling in Hybrid IaaS Clouds for Deadline Constrained Workloads . In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference*, pages 320–327. IEEE, 2011.

[8] Ruben Van den Bossche, Kurt Vanmechelen, and Jan Broeckhove. Online Cost-efficient Scheduling of Deadline-constrained Workloads on Hybrid Clouds. *Future Generation Computer Systems*, 29:973–985, 2013.

[9] Archana Ganapathi, Yanpei Chen, Armando Fox, Randy Katz, and David Patterson. Statistics-driven Workload Modeling for the Cloud . In *Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference*, pages 87–92. IEEE, 2010.

[10] GoGrid. GoGrid Website - http://gogrid.com.

[11] Lizheng Guo, Shigen Shen Shuguang Zhao, and Changyuan Jiang. Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm. *Journal of Networks*, 7:547–553, 2012.

[12] Marc Oriol Hilari. Quality of service (qos) in soa systems. a systematic review. Master thesis, Universitat Politecnica de Catalunya.

[13] Amazon Web Services Inc. Amazon Web Services - http://aws.amazon.com.

[14] Microsoft Inc. Microsoft Azure - http://azure.microsoft.com.

[15] Rackspace US Inc. Rackspace Website - http://www.rackspace.com.

[16] B. Javadi, R.K. Thulasiram, and R. Buyya. Statistical Modeling of Spot Instance Prices in Public Cloud Environments. In *Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference*, pages 219–228. Springer-Verlag Berlin, Heidelberg, 2012.

[17] Gideon Juve, Ewa Deelman, Karan Vahi, Gaurang Mehta, Benjamin P. Berman, Bruce Berriman, and Phil Maechling. Scientific Workflow Applications on Amazon EC2. In *E-Science Workshops, 2009 5th IEEE International Conference*, pages 59–66. IEEE, 2009.

[18] Gideon Juve, Ewa Deelman, Karan Vahi, Gaurang Mehta, and Bruce Berriman. Data sharing options for scientific workflows on amazon ec2. In *SC10 Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis.*

[19] Kurt Vanmechelen. The Cloud Computing Paradigm. Topic in Distributed Computing Lecture, University of Antwerp, 2013.

[20] Sparx IT Solutions Private Limited. A Complete Reference to Cloud Computing.

[21] Ming Mao and Marty Humphrey. Auto-scaling to minimize cost and meet application deadlines in cloud workflows. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 49:1–49:12, New York, NY, USA, 2011. ACM.

[22] Ming Mao, Jie Li, and Marty Humphrey. Cloud Auto-scaling with Deadline and Budget Constraints. In *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference*, pages 41–48. IEEE, 2010.

[23] G. Mc Evoy and B. Schulze. Understanding scheduling implications for scientific applications in clouds. In *Proceedings of the 9th International Workshop on Middleware for Grids, Clouds and e-Science*, MGC '11, pages 3:1–3:6, New York, NY, USA, 2011. ACM.

[24] Rizwan Mian, Patrick Martin, Farhana Zulkernine, and Jose Luis Vazquez-Poletti. Estimating resource costs of data-intensive workloads in public clouds. In *Proceedings of the 10th International Workshop on Middleware for Grids, Clouds and e-Science*, MGC '12, pages 3:1–3:6, New York, NY, USA, 2012. ACM.

[25] Suraj Pandey, Adam Barker, Kapil Kumar Gupta, and Rajkumar Buyya. Minimizing Execution Costs when Using Globally Distributed Cloud Services. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference*, pages 85–95. IEEE, 2011.

[26] Peter Mell and Timothy Grance. The NIST Definition of Cloud Computing, 2011.

[27] Arkaitz Ruiz-Alvarez and Marty Humphrey. An automated approach to cloud storage service selection. In *Proceedings of the 2Nd International Workshop on Scientific Cloud Computing*, ScienceCloud '11, pages 39–48, New York, NY, USA, 2011. ACM.

[28] Arkaitz Ruiz-Alvarez and Marty Humphrey. A Model and Decision Procedure for Data Storage in Cloud Computing. In *Proceeding CCGRID 12 Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 572–579. IEEE Computer Society, 2012.

[29] Mohsen Amini Salehi and Rajkumar Buyya. Adapting Market-Oriented Scheduling Policies for Cloud Computing. In *Algorithms and Architectures for Parallel Processing*, pages 351–362. Springer Berlin Heidelberg, 2010.

[30] Bhanu Sharma, Ruppa K. Thulasiram, Rajkumar Buyya, Parimala Thulasiraman, and Saurabh K. Garg. Pricing Cloud Compute Commodities: A Novel Financial Economic Model. In *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium*, pages 451–457. IEEE, 2012.

[31] Barrie Sosinsky. *Cloud Computing Bible.* Wiley Publishing, Inc.

[32] Jia Yu, Rajkumar Buyya, and Chen Khong Tham. Cost-based scheduling of scientific workflow application on utility grids. In *Proceedings of the First International Conference on e-Science and Grid Computing*, E-SCIENCE '05, pages 140–147, Washington, DC, USA, 2005. IEEE Computer Society.

[33] Miranda Zhang, Rajiv Ranjan, Surya Nepal, Micheal Menzel, and Armin Haller. A Declarative Recommender System for Cloud Infrastructure Services Selection. In *Proceedings of the 9th international conference on Economics of Grids, Clouds, Systems, and Services*, pages 102–113. IEEE, 2011.