Aalto University
School of Science
Master's Programme in ICT Innovation

Andrea Nodari

Cost Optimization in Cloud Computing

Master's Thesis
Espoo, June 22, 2015

Supervisor: Professor Jukka K. Nurminen, Aalto University

Advisor: Christian Frühwirth, M.Sc.

Aalto University
School of Science

Aalto University School of Science Degree Programme in Computer Science and Engineering Master's Programme in ICT Innovation	ABSTRACT OF MASTER'S THESIS

Author: Andrea Nodari

Title: Cost Optimization in Cloud Computing

Number of Pages: 84 | **Date:** June 22, 2015 | **Language:** English

Professorship: Data Communication Software | **Code:** T-110

Supervisor: Professor Jukka K. Nurminen, Aalto University

Advisor: Christian Frühwirth, M.Sc.

In recent years, cloud computing has increased in popularity from both industry and academic perspectives. One of the key features of the success of cloud computing is the low initial capital expenditure needed compared to the cost of planning and purchasing physical machines. However, owners of large and complex cloud infrastructures may incur high operating costs.

In order to reduce operating costs and allow elasticity, cloud providers offer two types of computing resources: *on-demand instances* and *reserved instances*. On-demand instances are paid only when utilized and they are useful to satisfy a fluctuating demand. Conversely, reserved instances are paid for a certain time period and are independent of usage. Since reserved instances require more commitment from users, they are cheaper than on-demand instances. However, in order to be cost-effective compared to on-demand instances, they have to be extensively utilized.

This thesis focuses on cost optimization of cloud resources by balancing on-demand and reserved instances. The challenge is to find an optimal resource allocation under uncertainty. In order to solve the problem, this study introduces a theoretical model based on Inventory Theory and a heuristic-based implementation for reserved instances optimization.

The inventory theory model provides a theoretical framework for cost optimization. In addition, the model describes a mathematical method to solve the optimization problem. The heuristic-based implementation analyzes the cloud infrastructure of a company and proposes a purchase plan of reserved instances. The implemented system validates the theoretical finding.

In order to evaluate the proposed approaches, this work describes a set of experiments, using simulations and data from an industry case. The experiments demonstrate the effectiveness of the reserved instances optimizer and the validity of the theoretical model.

Keywords: Cloud Computing, Cost Optimization, Inventory Theory

Acknowledgments

Writing this thesis would have not been possible without the help, the support, and the encouragement of family, friends, and mentors.

My sincere gratitude goes to my supervisor Prof. Jukka Nurminen, for the many discussions, ideas, and encouragement. Thanks also to Prof. Ritabrata Dutta, for the valuable feedback.

I want to thank all my colleagues at Nordcloud. Especially, I want to express my gratitude to Christian Frühwirth, for being my instructor during this period and Ilja Summala for the guidance during these months. A big thank goes to Piotr Sokołowski, for his daily help, support, and patience.

As my Master's degree allowed me to travel in many countries, my friends are spread across Europe. Thanks to Matteo and Federico, for the priceless time we spent together; thanks Nicola and Federica for always being close, no matter how far. Finally, thanks to all the friends I have met here in Finland.

My deepest gratitude goes to my family, a constant source of energy and love.

Last, but certainly not least, thanks to Andrea for her patience and support in these months.

Espoo, June 22, 2015

Andrea Nodari

Abbreviations and Acronyms

Acronyms

AWS	Amazon Web Services
EC2	Elastic Compute Cloud
IaaS	Infrastructure as a Service
IRR	Internal Rate of Return
NVP	Net Present Value
OD	On-Demand
PaaS	Platform as a Service
RI	Reserved Instance
RIO	Reserved Instances Optimizer
ROI	Return on investment
ROV	Real option valuation
SaaS	Software as a Service
SLA	Service Level agreement

Nomenclature

c_{od}	Hourly cost for an on-demand instance
c_{ri}	Effective hourly cost for a reserved instance
y	Number of reserved instances
D	Random variable representing the demand of instances
d_i	i^{th} observation of demand D
$C(D, y)$	Total cost to satisfy demand D with y reserved instances
$E[C(D, y)]$	Expected value of the total cost
φ_D	Probability distribution function of the demand (approximated as a continuous random variable)

Φ_D	Cumulative distribution function of demand (approximated as a continuous random variable)
F_D	Cumulative distribution function of demand
$L(\theta\|X)$	Likelihood of parameters θ on the sample X
Φ_D^{-1}	Quantile function of the demand D
q_p	p^{th} quantile

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, cloud computing has transformed large part of the Information Technology industry. Cloud computing refers to a new paradigm of providing computational assets. While computing resources are provided as a service, users are not required to own physical machines.

The key characteristics of cloud computing are the ability of scaling resources practically infinitely, the capability to pay only when a resource is actually needed, and the elimination of large upfront costs for users [5, 6]. In addition, low prices and ease of use encourage enterprises to utilize cloud computing to host their IT infrastructure.

Cloud computing is offered by cloud providers, among which the most prominent examples are Amazon Web Services (AWS)[1], Google Cloud[2], and Microsoft Azure[3]. Every cloud provider has different pricing strategies; however, for computing resources they offer two categories of products: *on-demand instances* and *reserved instances*. On-demand instances are virtual machines created and paid for only when utilized. A cloud user adds and removes an on-demand instance with maximum flexibility. Conversely, reserved instances are computational resources reserved and paid for a certain period, with an upfront fee. The latter category requires a higher level of commitment for the user; therefore, if extensively utilized, they result to be cheaper during a long-term utilization.

In order avoid unnecessary expenses, users of cloud computing need careful planning. On one hand reserved instances are useful for cost savings. On the other hand, if reserved instances are underutilized, they generate unnecessary costs. In particular, it is crucial to analyze the usage of computational resources and allocate reserved instances as well as on-demand instances ac-

[1] `https://aws.amazon.com`
[2] `https://cloud.google.com`
[3] `https://azure.microsoft.com`

cordingly. A correct balance of these two types of resources allows cost savings and flexibility.

Currently, researchers have extensively studied the field of cost optimization in cloud computing. One of the most promising methods is to utilize Integer Programming to model the optimization problem [22, 23]. Other authors exploit a two-step approach: first, they propose a demand forecaster and then, they aim to find an optimal solution with evolutionary algorithms [44, 56].

Nonetheless, the current state-of-the-art approaches have some limitations. First, they do not take risks into account as researchers obtain the optimal planning strategy regardless of the investment of a client company. Second, authors calculate the optimal number of reserved instances assuming the demand distribution as given, which may lead to incorrect estimation and ultimately losses.

Certain industrial solutions are becoming increasingly popular (Table 2.4). However, some drawbacks are caused when adopting these tools as they need access to sensitive information such as billing data. In addition, users are confronted with vendor lock-in risk and cannot access the algorithms used for optimization. Therefore, it is difficult to verify the effectiveness of the plan suggested by these tools.

This thesis aims to overcome some of the limitations of the state-of the-art approaches. This study proposes a theoretical model based on Inventory Theory to describe the cost optimization problem. In addition, a heuristic-based cost optimizer has been developed in a case company.

The proposed theoretical model utilizes Inventory Theory, which aims to scientifically describe the behavior of an inventory system. The model describes the cost optimization problem as an inventory-keeping problem. After the formulation, it is possible to derive the optimal policy which minimizes the costs. To the best of our knowledge, this is the first study that applies Inventory Theory to cost optimization in cloud computing.

The heuristic-based cost optimizer outputs a purchase plan that correctly balances on-demand instances and reserved instances in order to minimize the costs. The proposed solution analyzes the hourly usage of the resources, selects the best options in terms of opportunity size, and suggests a purchase plan to decision-makers. The developed algorithm aims to be fast, extensible, and close to the optimal solution. In addition, it takes into account risks involved in cloud infrastructure planning. The proposed strategy considers the past usage of cloud resources and estimates the risk by looking at the release date of each instance type.

In order to validate the performance of the algorithm, this thesis analyzes real data from a company active in cloud computing. The evaluation shows

that theoretical results from the model are validated by the heuristic-based optimization algorithm.

In conclusion, this study proves that the proposed approach will benefit cloud users willing to optimize the cost of cloud computing assets.

The contributions of this thesis are the following.

- A survey of cost optimization techniques related to cloud computing, considering recent literature studies and current tools from applied in the industry.
- A theoretical model based on Inventory Theory, which describes the cost optimization problem.
- A heuristic-based cloud optimizer which helps cloud users to plan their infrastructure with a focus on cost savings. The proposed solution has been implemented and tested with real data from an industry case.

1.1 Research Questions

This thesis aims to answer the following research questions.

- How cost optimization in cloud computing can be modeled with Inventory Theory?
- How an heuristic-based reserved instance planner perform?
- Does the implemented system verify the theoretical results of the inventory theory model?

1.2 Structure of the Thesis

This thesis is structured as follows. Chapter 1 introduces the topic of the thesis, underlying the research questions and contributions.

Chapter 2 describes the body of knowledge related to cloud computing and cost optimization. The survey, provides detailed information regarding the state-of-the-art approaches on the topic from both academia and industry.

Chapter 3 describes Inventory Theory and how it can be applied to cost optimization in cloud computing. Furthermore, it shows the analytical derivation of the optimal solution for reserved instance planning.

Chapter 4 proposes the details of the implemented solution, explaining the algorithms and the heuristics utilized in the system.

Chapter 5 evaluates the theoretical model comparing it with the implemented algorithms. Furthermore, this chapter analyzes the performance and efficiency of the heuristic-based implementation.

Chapter 6 analyses the results of the experiments. In addition, it states some limitations and some possible improvement of the system.

Finally, Chapter 7 summarizes the work, proposes future directions, and concludes the thesis.

Chapter 2

Background

2.1 Cloud Computing

According to the National Institute of Standards and Technology (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." [46].

The document lists five the main characteristics of Cloud Computing model:

- *On-demand, self-service.* Cloud users add resources at any time without system administration intervention.
- *Broad Network access.* Cloud users access resources via network.
- *Resource pooling.* Cloud providers heavily utilize virtualization and resource pooling to exploit an economy of scale.
- *Rapid elasticity.* Cloud users have the capability to scale resources to satisfy a fluctuating demand.
- *Measured service.* Resources are monitored to allow cost optimization and performance analysis.

NIST also describes three different service models:

- *Infrastructure as a Service.* This model allows the user controlling a full stack of software from the hardware to the applications. It gives the highest flexibility, but it requires more expertise from the user. An example of IaaS provider is Amazon Web Services [8].

- *Platform as a Service.* In this model users obtain a platform on top of which they build software. Providers take care about low-level details such as availability, security patches, and scalability of resources. Some examples of PaaS providers include Heroku[1] or Google App Engine [2].

- *Software as a Service.* This model includes all the software provided to the users as a service. This broad category ranges from customer relationship management[3] to chat services[4].

Researchers argues that to differentiate service models is misleading [6]. Since all of the three models refers to computing provided *as a service*, the authors prefer to use the more generic term *utility computing*. Utility Computing is characterized by properties such as flexibility, portability, and ease of use.

Authors describe two main actors involved in cloud computing: cloud users and cloud providers [5]. Cloud users are those enterprises which rely on the cloud computing for their business. Cloud providers are companies that provide cloud resources. A remarkable category of cloud providers is Infrastructure as a Service providers, which are the companies that own physical data centers and provide computational resources as a service.

IaaS providers benefit from an economy of scale to provide resources such as disk, storage, and computing capacity. One of the largest IaaS providers is Amazon Web Services (AWS). Amazon Web Services offers a variety of cloud services that relates different types of resources such as disk (e.g. Amazon Simple Storage Service), network (e. g. Amazon Cloudfront), and computing capacity (e.g. Amazon EC2) [45].

The next paragraphs extensively describe the offer of computing capacity in cloud computing. Amazon Web Service is taken as example; however, some of the concepts are relevant for other cloud providers such as Google Cloud or Microsoft Azure.

Amazon Elastic Cloud Compute (EC2) is the most popular AWS product for computing capacity [35]. AWS EC2 allows users renting virtual server accessible via API [10]. This service well summarizes all the characteristics of cloud computing as it is self-service, elastic, inexpensive, and accessible through network.

There are different categories of access computing capacity with Amazon Web Services.

[1] `http://heroku.com`
[2] `https://cloud.google.com/appengine`
[3] `https://pipedrive.com`
[4] `https://slack.com`

- *On-demand Instances.* In the most common version of EC2, cloud users activate or deactivate a machine at will. The pricing is calculated on a pay-per-use basis, enabling elasticity.

- *Dedicated Instances* [9]. They are similar to on-demand instances, but the cloud provider guarantees to the user a dedicated machine. This solution solves multi-tenancy problems, in which tenants sharing the same physical machine may influence each other, deteriorating the performance.

- *Reserved Instances* [13]. This type of instances requires an initial commitment from users. Typically, they have to pay an upfront fee and reserve an instance from a time slot (e.g. Amazon AWS has one and three years plans). Buying this type of instances requires more commitment, therefore the hourly prices are lower. In addition, reserved instances allow reserving capacity in advance, so that when needed, instances are immediately available.

- *Spot Instances* [14]. In this type of instances a user bid a price. The price of the instances fluctuates with demands and as soon as the price is lower than the bid, the machine is assigned to the user. Using this type of machines usually leads to lower prices; however, the execution might be interrupted in any moment. This type of instances is particularly useful in executing long running jobs which can be paused and resumed.

- *Lambda* [11]. This type of service allows a user running code in response to an event. The cost of the computation depends on the time needed to execute the code. It is not part of AWS EC2 offering; however, *Lambda* pushes even further the concepts of pay-as-you-go computing.

The next sections introduce the challenges of cost optimization in cloud computing.

2.2 Cost Optimization in Cloud Computing

As in cloud computing there are two main actors involved, there are two sides of cost optimization: cost optimization performed by providers and cost optimization performed by users.

Cost optimization performed by cloud providers mainly focuses on minimizing the cost to maintain a physical data center. The cost minimization is typically achieved by reducing electricity consumption. A proposed approach

involves dynamically halting network devices [38]. Another study proposes architectural principles, algorithms, and resource allocation policies for energy savings [21].

Conversely, one of the most popular techniques for cost optimization executed by cloud users is to choose the correct balance the types of instances, i.e. cloud infrastructure planning.

This thesis concentrates on cost optimization performed by users. In particular, this study focuses on finding the correct balance between on-demand instances and reserved instances. The choice is made for two reasons. First, while spot instances and Lambda are specific to Amazon Web Services, on-demand and reserved instances might be relevant for different IaaS providers. Therefore, a larger part of cloud users may benefit from the results of this thesis. Second, researchers and practitioners studied the effectiveness of cost optimization using reserved instances; hence, contributions in this field might be more significant.

2.2.1 Reserved Instances

Companies usually encounter two phases. At first the demand of computing resources is highly uncertain and cost optimization is not a priority. Afterwards, companies confront with two phenomena. First, the increase of available capital. Second, the growth of operating costs due to a large IT infrastructure. These trends allow a company exploiting an effective model for cost optimization: reserved instances (RIs) [54].

The reserved instance model is simple. Users commit to pay an instance for a determined time period and providers offer them a discount. After the purchase, a reserved instance is similar owning a physical machine. If the instance is running most of the time, the providers guarantee cost savings compared to the same on-demand instance at the end of the commitment time. Table 2.1 shows a qualitative comparison between on-demand and reserved instances.

On-demand Instances	**Reserved Instances**
Pay as-you-go (hourly fee)	Upfront fee and hourly fee
For flexibility	For cost savings
No need for planning	Need careful planning

Table 2.1: Qualitative comparison between on-demand and reserved instances.

The two described phenomena particularly fit the reserved instances model. First of all, as reserved instances are similar to an investment, money for an initial commitment has to be available. In addition, low operating costs imply no need for cost optimization with reserved instances.

This study focuses on the purchase options of reserved instances for Amazon Web Services. In this case, users buy RIs for a term of one or three years. There are three purchase options, i.e. methods to pay a reserved instance. Each of them is characterized by a different commitment level and therefore, a different savings rate compared to on-demand instances.

- *All Upfront.* Users pay the reserved instances with a single upfront payment, without an hourly fee. This option guarantees the highest cost savings compared to an equivalent on-demand plan.
- *Partial Upfront.* Users pay the reserved instances with an upfront payment and a discounted hourly fee.
- *No Upfront.* Users pay the reserved instances with a discounted hourly fee, without paying any upfront fees. This option has a limited cost savings.

It is important to underline that in the *partial upfront* and *no upfront* cases, users pay the hourly fee every hour, even if the instance is not running.

When buying a reserved instance, users specify four properties: term length, operating system, availability zone, and size. Table 2.2 shows an example of pricing for a reserved instance. Table 2.3 explains the parameters providing examples of Amazon Web Services.

Payment Option	**Upfront fee**	**Effective hourly**	**On-demand hourly**
No Upfront	$0.00	$0.0580	
Partial Upfront	$222	$0.0533	$0.077 per Hour
All Upfront	$457	$0.0522	

Table 2.2: An example of pricing for a reserved instance, the characteristics are: 1 year term, Linux, Europe Ireland (eu-west), m3.medium. "The effective hourly price shows the amortized hourly cost of the instance"[13]. The price refers to the public pricing available at 2015-05-01.

Term length	It refers to the period of time when the RIs are assigned to a user. In AWS is one or three years.
Operative System	It concerns the operative system of the instance. Available operating systems in AWS are the following. • Linux/UNIX • SUSE Linux Enterprise Server • Red Hat Enterprise Linux • Windows • Windows with SQL Server
Availability Zone	It specifies where to create the instance. Amazon EC2 instances are hosted in different locations called region [12]. Within a region, AWS provides the concept of availability zones, logically isolated, but connected with low-latency links. The availability zones are developed for high reliability applications. Examples of regions are the following. • Europe Ireland (eu-west-1) • Europe Frankfurt (eu-central-1) • US California (us-west-1) • US Virginia (us-east-1) Each region usually has from one to four availability zones, usually identified with a letter. For instances, a valid availability zone is *eu-west-1a*.

Size	It refers to the characteristics of the machine in terms of CPU, memory, and disk. AWS provides a long list of sizes organized in families [15]. Each family groups machines with certain properties. The list includes families for general purpose computing (t2 and m3 families), compute optimized (c3 and c3 families), and memory optimized (r3 family). Within a certain family, users choose the characteristics that best fit their need.

Table 2.3: The parameters that have to be chosen when buying a reserved instance.

An interesting problem is to minimize costs, finding the right balance between on-demand instances and reserved instances with the correct characteristics.

The problem is challenging due to usage uncertainty. Assuming a perfect knowledge of the future, cost optimization is a trivial problem. The solution is to buy the highest possible number of reserved instances and to compensate the usage peaks with on-demand instances. This scenario is infeasible since usage of a certain instance might decrease, instances might become obsolete, and infrastructure might change. In contrast, using only on-demand instances guarantees maximum elasticity, but with higher costs. Therefore, uncertainty is a crucial parameter to take into account in the optimization.

Recently, researchers have identified two major sources of inefficiency in buying reserved instances: under-provisioning costs and over-provisioning costs [24]. The first refers to the problem of buying a limited number of RIs. In other words, it is a missed opportunity, since a larger investment would have been more beneficial. The second refers to a waste of money for buying an excessive number of reserved instances and pay them without a cost-effective utilization.

From a company perspective under-provisioning costs are less harmful than over-provisioning. The cloud flexibility allows a company buying instances in small batches. The remaining need of cloud resources is compensated with on-demand instances, which does not need planning. Conversely, having underutilized instances is a bigger problem. Cloud providers usually offers tools to mitigate this risk. For example, it is possible to modify a reserved instances or to sell them if not needed. However, these operations are complex and necessitate of time and expertise [31].

Another characteristic of reserved instances is an increased service guar-

antees. Amazon Web Services, for example, guarantees availability of on-demand instance type when a RIs contract exists. Without a reserved instance machine provisioning might fail due to high load. Despite these remarks might be important in high availability systems, this thesis studies the reserved instances model only as mean for cost savings.

The next sections review the body of knowledge related to cost optimization using reserved instances.

2.2.2 Optimization Using Integer Programming

Integer Programming is one of the most important strategies of theory of optimization [20]. An integer programming problem is generically described as the maximization (or minimization) of a function under determined constraints. Among them, there must be an integrality constraint (Equations 2.2 and 2.3). Without the integrality constraint this method is called *linear programming*.

$$Maximize \sum_{j=1}^{n} c_j x_j, \tag{2.1}$$

Subject to:

$$\sum_{j=1}^{n} a_{i,j} x_j = b_i \qquad (i = 1, 2, ..., m) \tag{2.2}$$

$$x_j > 0, x_j \text{ integers} \tag{2.3}$$

Integer Programming has been extensively studied in different scenarios. A remarkable example regards capital budgeting. In this case, the problem is to choose whether to invest or not in projects estimating their profitability. The goal is to optimize the profit, selecting the investments with a limited amount of money.

A subcategory of integer programming is binary integer programming, also known as knapsack problem [49]. In this case the variables assume only binary values. This approach is particularly useful in modeling decisions that might or might not be engaged.

Regarding the field of cost optimization in cloud computing, a remarkable work formulates the optimization problem and solve it using stochastic integer programming [23]. The objective is to minimize the overall cost of the infrastructure. The optimization problem fits in a system with four components: cloud consumers, who demands to run jobs; cloud providers, which

are responsible to meet users demands; virtual machine, which group the resources needed to run a job; and cloud broker, that optimally provisions resources for the customer.

Resources are provisioned in determined points in time called *provisioning phases*. In these moments, the cloud broker solves the optimization problem to allocate resources. The possible actions include provisioning of an on-demand instance or perform a new reservation.

The described model is powerful; however, the approach has some limitations. The goal of the proposed optimization strategy focuses on finding the optimal combinations of reserved instances and on-demand instances, so that user demands is satisfied with the minimum cost. Nonetheless, the authors do not take into account risks and budget limitation. A crucial point to underline is that on-demand instances are always available and they can satisfy user demand. Using the proposed approach might result in large and risky investments. An additional remark is that the model assumes a limitation in the resources is not realistic in case of public cloud. Finally, the authors takes uncertainty of future demand into account, by assuming a known distribution.

The authors improve their work focusing of Amazon Web Service offering [22]. They develop an algorithm for cost optimization by provisioning of on-demand instances, reserved instances, and spot instances. The model is based on long-term and short-term provisioning algorithms. The fluctuating demand is taken into account as a known probability distribution.

The proposed solution reveals interesting results in the minimization of the total provisioning cost, in particular, considering spot instances. However, risks and budget constraints are not studied.

Despite the previous articles focus on Amazon AWS as IaaS provider, a similar approach has been applied using Microsoft Azure as cloud provider [25]. The model of Microsoft Azure is similar to AWS. There are two options: the first is pay-as-you-go and the second is a six months subscription with a discounted hourly rate and an upfront fee. The study focuses on maximizing the profit of value-added services implemented using the cloud resources. The profit is assumed to be dependent by a known probability distribution.

Additional improvements are also proposed to efficiently solve the optimization problem, which might be hard it the size of the problem increases [24].

Another study exploit linear programming to optimize the costs of replacing an in-house data center with a cloud computing. In modeling the replacement, the authors consider the possibility to use reserved instances in order to minimize the operating costs. The method is based on two steps. First, the input data is obtained by monitoring the existing system. Sec-

ond, a linear programming problem is formulated and solved. An important difference from other studies is the multi-resource strategy. The authors, in fact, study the load curve of RAM, CPU, and storage [51].

Researchers propose a linear programming model, which include on-demand instances, reserved instances, and spot instances [53]. An interesting remark relates on how the authors force the model to diversify the investment in the three types of models.

To summarize, Integer Programming fits particularly well the problem of cost optimization in cloud computing. Authors provide sophisticated and effective models for decision-making. However, a major weakness of these methods relate to the fact that they consider equally important under-provisioning and over-provisioning. Furthermore, most of the studies do not take into account risks and budget constraints. Finally, some studies assume a known usage demand distribution which might leads to wrong result and it can be infeasible to estimate.

2.2.3 Optimization Using Evolutionary Algorithms

Evolutionary algorithms mimic biological processes to solve optimization problem. Given an initial population of individuals, a function defines the *fitness* of a certain environment. As in biology, the fittest survives and generates offspring, progressively approach a optimal solution.

Exploiting this phenomenon, researchers create optimization algorithms. The search for optimal solutions "are based on the collective learning process within a population of individuals, each of which represents a search point in the space of potential solutions to a given problem" [16]. After a population is created, the entities progressively evolve towards a better solution, until convergence. A remarkable category of evolutionary algorithms is genetic algorithms. Figure 2.1 shows the pseudo-code of a generic algorithm of this category.

This section presents studies which apply evolutionary algorithms to the problem of cost optimization in cloud computing. Authors propose evolutionary algorithms as a more efficient alternative to Integer Programming. Despite Integer Programming offers are well-grounded method to approach optimization problem, it might become computationally infeasible with increasing input size [44].

Researchers propose a two-step approach [44]. At first, the authors design a demand forecaster. The experiments show accurate results of three algorithms: simple Kalman filter, Double Exponential Smoothing, and Markov Chain. The next step utilizes a hybridized algorithm consisting of different evolutionary algorithms for optimization: standard Genetic Algorithm,

```
1: Generate initial population
2: Compute fitness of each individual
3: while !Finished do
4:     for Population Size / 2 do
5:         Select two individuals from old generation for mating
6:         Recombine the two individuals to give offspring
7:         Compute fitness of offspring
8:         Insert offspring in the new generation
9:     end for
10:    if Population has converged then
11:        finished ← true
12:    end if
13: end while
```

Figure 2.1: An example of genetic algorithm [17].

Particle Swarm, and Ant Colony. The researchers compare the results with Stochastic Integer Programming solution, showing similar conclusions. Given a demand curve, the proposed model takes into account reserved instances as a possibility for cost saving.

Other work follows the two-step method [56]. In this study, firstly, future usage demand is predicted using a Neural Network. Secondly, an Imperialist Competitive Algorithm is used to yield the optimal plan. The work shows results of a simulation that demonstrate the effectiveness of the methods.

Another work studies how genetic algorithms are used to predict future demands [52]. The authors argue that a crucial point for an effective planning is to understand future demand. The study shows the result of the experiments conducted using the load curve of four popular websites. The evaluation demonstrate that Genetic Algorithms generate accurate results.

These studies propose evolutionary algorithms as a effective alternative to Integer Programming. In particular, they might be more effective in case where the input size is larger. An important remark is to notice the importance to an accurate forecasting of future usage in order to build a precise plan. The effectiveness of reserved instance planning, in fact, highly depends on accurate demand prediction.

2.2.4 Optimization Using Machine Learning

Some of the previously presented work have already discussed machine learning techniques as effective forecasting methods [44, 56]. However, this section describes studies that utilize machine learning as main technique.

Authors study a novel way to pro-actively provision virtual machines necessary to satisfy load peak [26]. The proposed solution utilizes K-Nearest-Neighbors classifier to decide whether to provision new virtual machines or not. As the study focuses on database load, the classifier utilizes database parameters as features including average query throughput, number of active connections, read write I/O, lock ratio, CPU/Memory/Disk usage.

Another contribution to the topic of pro-actively infrastructure provisioning, is achieved using Principal Component Analysis (PCA) [42]. The goal of the authors is to provide a method to reduce management costs in scientific computing using the cloud. The proposed solution analyzes the past workload to predict the infrastructure needed of a given experiment.

Others effective techniques are Neural Networks and Linear Regression [41]. These techniques are proved to be effective in predicting peaks and automatically provision virtual machines. Simulations using load from e-commerce application show that Neural Networks perform better than linear regression.

Different works provide a good explanation about how forecasting techniques are used to design optimal plan [19]. At first, future demand for each instance is calculated. The authors compare two approaches: *Previous Period*, that is to predict the future demand as equal to a previous period, and *Double-Seasonal Holt-Winters time series analysis*. The experiments show that both approaches generate accurate result, even if the latter is more effective than the former.

In order to output the plan, the authors propose PMA: Purchase Management Algorithm. At each time a new reservation could be performed the following operations are executed. First, the system runs the forecasting algorithm and output a histogram of the number of machines. Second, the maximum profit is calculated given a possible reservation contract.

This section presents studies which utilize machine learning algorithms for cost optimization in cloud computing. The goal of the articles is to predict future usage in order to provision virtual machines and satisfy user demand. Nonetheless, these approaches can be combined with optimization techniques for an effective planning.

2.2.5 Miscellaneous Methods

This section describes miscellaneous methods. Despite the fact that these articles do not fit in the previous categories they remarkably contribute to body of knowledge of cost optimization.

One of the first studies in cost optimization in cloud computing utilizes a utility function to assess the costs and the revenues of a company [47].

The utility function is then maximized under service level agreement (SLA) constraints such as availability, average response time, and throughput. The study does not consider reserved instances as an option, but it discusses the number of virtual machine to be provisioned.

Another work thoroughly describes the problem of unnecessary costs in cloud computing [40]. The authors propose two different situations. The first problem relates to cloud users having a fixed margin of unused virtual machines ready in case of load peaks. Provisioning of virtual machines, in fact, it is not immediate, and a margin is needed to manage a fluctuating demand. The researchers propose a dynamic margin allocation with SLA guarantees to diminish the number of virtual machines and, hence, the cost. The second problem is about reserved instance planning. The proposed algorithm looks at the utilization ratio of existing machines and evaluates whether purchasing a reserved instance will be cost-effective.

Recent advances in forecasting usage demand utilizes a fractal model [36]. The authors argue that the previous models are unable to predict the complex dynamics of cloud computing. Users and providers might use the proposed solution for a more efficient planning.

2.3 A Financial Perspective

A reserved instances purchase can be seen as a financial investment. It involves committing money upfront to obtain a discounted price and save money in the future. This section aims to describe methods used in finance to evaluate an investment. It can be interesting to use some of the following methods in evaluating or comparing reserved instances purchases.

In order to assess an investment, different strategies exist, including: Return on Investment, Profit margin, Net Present Value, Cash Flow, Internal Rate of Return, and Payback period [34, 50, 55].

Return on Investment (ROI) It measures the performance of an investment by calculating the ratio between the inflows subtracted by the cost and the cost of the investment.

$$ROI = \frac{\text{Gain from investment} - \text{Cost of investment}}{\text{Cost of the investment}} \tag{2.4}$$

Profit Margin It refers to the ratio of net profit and total revenue.

$$PM = \frac{\text{Revenue} - \text{Expenses}}{\text{Revenue}} \tag{2.5}$$

Cash flow Cash flow is a movement of money in or out a business. Cash inflow refers to money coming in a business, as opposite as outflow.

Net Present Value (NVP) This method allows obtaining the value of an investment in today's money. Equation 2.6 calculates the Net Present Value of an investment. The cash inflow is reduced by a discount rate in order to correctly evaluate an investment.

$$NVP = \sum_{t=1}^{T} \frac{C_t}{(1+r)^t} - C_0 \tag{2.6}$$

where:

$$C_t = \text{Net cash inflow during the analyzed period} \tag{2.7}$$
$$C_0 = \text{Initial Investment} \tag{2.8}$$
$$r = \text{Discount rate} \tag{2.9}$$
$$T = \text{The total number of periods} \tag{2.10}$$

Internal Rate of Return (IRR) IRR refers to the discount rate such that the Net Present Value is zero. This means that the negative cash flow and the negative cash flow are set to equal. The higher the IRR the better is the investment.

Payback Period The time needed to compensate the initial money needed for an investment.

In addition another model available in finance to evaluate an investment is *real option valuation (ROV)* [4, 18]. The term *real option* refers to the possibility to change strategy if certain conditions arise. The term *real* relates to the fact that the decision is about tangible resources.

Examples of options include termination, initiation, temporary stop, increasing, decreasing budget for a project. The crucial point in this analysis is to take into account options in an investment evaluation.

Since purchasing reserved instances can be seen as an investment, it might be interesting to apply financial methods to evaluate a purchase plan. In particular, ROV seems to better fit the problem as it respects the high flexibility of cloud computing. To the best of our knowledge, literature on cost optimization in cloud computing do not take these techniques into account.

Feature	Cloudability	Cloudhealth	Cloudyn	Orbitera
Multiple cloud providers support	No	No	Yes	No
Optimization method	Analysis based on variable time range	Analysis based on fixed time range	Analysis based on fixed time range	Analysis based on fixed time range
Risk analysis	None	None	None	None
Custom reports capabilities	Advanced	Medium	Advanced	Low

Table 2.4: Summary of the features of the tools that provide cost optimization.

2.4 Tools for Cost Optimization

This section proposes an overview of the tools available in industry for cost optimization. Table 2.4 summarizes the features of the tools.

The analyzed tools are the following.

Cloudability [27]. Cloudabilty is a company that offers three solutions. Cost monitoring, cost optimization, and cost management. Cost optimization is achieved looking at the break-even point utilization on the hourly spending [28, 29].

Cloudhealth [30] Cloudhealth proposes good analytics and customizable reports. The algorithm for cost optimization utilized a fixed time range of past data.

Cloudyn [32] Cloudyn offers a detailed and fine service for cost optimization. The strategy is similar to other companies: looking at past data and provide suggestions based on a fixed time range.

Orbitera [48] Orbitera is a company that offers analytics services. The algorithm for cost optimization utilized a fixed time range of past data.

Industry solutions provide detailed analytics regarding optimization using reserved instances. All the evaluated tools excel in user experience, giving to the users good insights into unnecessary costs and planning suggestions.

Nonetheless, they have some limitations. First of all, they do not take risks into account and they only show maximum savings opportunities. In addition, they need access to sensitive information such as billing data. Furthermore, users are confronted with high vendor lock-in risk, since they have to be well integrated with the company that adopt them. Finally, they do not provide details of the algorithms to compute the suggestions, therefore it is hard to compare them and improve the proposed plan.

Chapter 3

The Inventory Theory Model

3.1 Background

Inventory Theory is a branch of operations research focused on scientific inventory management. It aims to mathematically describe an inventory system and to determine the optimal policies for minimizing the costs while satisfying demand. Other goals of Inventory Theory include finding how and when to replenish an inventory, how to manage extra stock of a product, how to keep control of an inventory [39].

The components of an inventory theory model are the following.

Cost of ordering or manufacturing It is the cost of the product needed to satisfy the demand.

Holding cost It is the unit cost paid to keep a product in stock. Holding Cost is also referred as *inventory cost* or *storage cost*.

Shortage cost It is the unit cost paid in case the firm is unable to satisfy the demand of a product.

Revenue It represents the revenue for selling a product, it can be excluded if the *loss revenue* is already modeled by the other types of cost.

Salvage value It is a value of unsold product. It can be also a negative value, representing disposal costs.

Discount rate A discount rate can be applied to the model to consider the time value of money.

There are two ways to classify inventory theory models. The first category divides the models according to the way of dealing with demand of goods.

There are two approaches: *deterministic models* and *stochastic models*. The former assumes the usage to be known, the latter model the demand as a random variable.

The second category arranges the model depending on the way the inventory is reviewed. In *continuous review* models, the inventory is replenished when the stock level is below a threshold. In *periodic review* models, the inventory is checked with fixed intervals.

Inventory Theory has been extensively studied and applied in supply chain management. Researchers illustrate a review of inventory theory models and methodologies [1]. The study provides further details about the different classification, as well as limitation of the models.

Recent studies have utilized Inventory Theory to electric-power plant planning [57]. The crucial components of the proposed model are the stock of energy sources (e.g. coal) and the demand of electricity. Inventory Theory is demonstrated to work in this context, ensuring a stable electric power production and minimizing the costs for inventory management.

3.2 Inventory Theory Applied to Cloud Computing

This section proposes a mathematical model of the cost optimization in cloud computing based on Inventory Theory. The development of a theoretical framework to solve a problem is particularly useful. As underlined by Hillier and Lieberman, a mathematical model abstracts the details, unveiling the core aspects, the structure, and causality relationship of the problem [39].

The problem of minimizing the cost by using reserved instances has remarkable similarities with Inventory Theory. Inventory Theory helps companies to deal with stock of goods in order to minimize the costs. The goal of the model is to describe the optimal purchase in order to satisfy the demand of a product. Similarly, in the context of cloud computing, the decision-makers manage a reserved instances portfolio and optimize purchases to satisfy the demand with minimum cost.

This work proposes a stochastic model which formulates the total cost involved to satisfy the demand of computing capacity as a random variable. Afterwards, the analytical solution to minimize the costs is found. The proposed approach follows Hillier et al., which discuss the applications of operations research in greater details [39].

Inventory theory models focus on a product to store and manage. In this context the product is a reserved instance. The parameters of the proposed

model can be directly mapped to Inventory Theory literature (Table 3.1).

An important remark is that the proposed model considers only one reserved instance type. In other words, the goal is to find the optimal purchase given *operative system*, *availability zone*, *size*, and *purchase option* (Table 2.3). These four parameters completely characterize a reserved instance.

Inventory Theory nomenclature	**Symbol**	**Description**
Cost of ordering	c_{ri}	It is the effectively hourly cost of buying a reserved instance. In other words, an instance is paid for a term using upfront fees and hourly fees. The effective hourly fee amortizes the hourly fees on the length of the term (Section 2.2.1).
Holding Cost	$h = 0$	It is the cost of storing a reserved instance. Since there are no costs involved in stocking reserved instances this parameters is set to zero.
Shortage Cost	c_{od}	It is the cost incurred if there are not enough reserved instances compared with the demand. In the context of Inventory Theory this value refers to customers dissatisfaction or disposal costs. In cloud computing context, when there are not enough reserved instances it is necessary to use on-demand instances. Therefore the shortage cost per unit is the on-demand hourly price.

Table 3.1: The parameters of the inventory theory model applied to cost optimization in cloud computing.

3.2.1 Description of the Model

The model describes the total cost as a random variable. Afterwards, the expected value is calculated and it is analytically minimized.

Let:

$$y = \text{Number of purchased reserved instances} \tag{3.1}$$

$$D = \text{Random variable representing the hourly demand of instances} \tag{3.2}$$

$$d_i = i^{th} \text{ observation of the demand } D \tag{3.3}$$

$$c_{ri} < c_{od} \tag{3.4}$$

Assuming no initial inventory (i.e. no initial reserved instances) the total cost of a period of N hours can be express as:

$$\text{Total Cost} = \sum_{i=1}^{N} \left(c_{ri}\, y + c_{od}\, \max\{0, d_i - y\} \right) \tag{3.5}$$

This model abstracts the problem of cost optimization and describes it in clearer terms. For example, a company which does not buy any reserved instances (i.e. $y = 0$) has to relies completely on on-demand instances to satisfy the demand. The total cost will be:

$$\text{Total Cost} = \sum_{i=1}^{N} \left(c_{ri}\, 0 + c_{od}\, \max\{0, d_i - 0\} \right) \tag{3.6}$$

$$= \sum_{i=1}^{N} c_{od}\, d_i \tag{3.7}$$

Conversely, a company that exceeds in buying reserved instances would incur in a large ordering cost. Assuming that $y > D$ the total cost will be:

$$\text{Total Cost} = \sum_{i=1}^{N} c_{ri}\, y \tag{3.8}$$

Now, let us assume to obtain information about the distribution of D for all values of d.

$$P_D(d) = P\{D = d\} \tag{3.9}$$

In addition, let us consider the cost express in terms of random variable D. In order to minimize the total cost, the goal of the next step is to find the optimal value for y, the number of reserved instance to purchase. The cost with demand D and y RIs is expressed as follows.

$$C(D, y) = c_{ri}\, y + c_{od}\, \max\{0, D - y\} \tag{3.10}$$

Since $C(D, y)$ is a random variable it is possible to calculate the expected cost:

$$C(y) = E[C(D, y)] = \sum_{d=0}^{\infty} (c_{ri}\, y + c_{od}\, \max\{0, d - y\})\, P_D(d) \tag{3.11}$$

$$= c_{ri}\, y + \sum_{d=y}^{\infty} c_{od}\, (d - y)\, P_D(d) \tag{3.12}$$

In order to find an analytical solution, let us approximate the discrete random variable D with a continuous random variable.

$$\varphi_D(\xi) = \text{Probability density function of } D \tag{3.13}$$

$$\Phi_D(a) = \text{Cumulative distribution function } D \tag{3.14}$$

$$= \int_0^a \varphi_D(\xi)\, d\xi \tag{3.15}$$

The total cost $C(y)$ can now be expressed as:

$$C(y) = E[C(D, y)] = \int_0^{\infty} C(\xi, y)\, \varphi_D(\xi)\, d\xi \tag{3.16}$$

$$= \int_0^{\infty} (c_{ri}\, y + c_{od}\, \max\{0, d - y\})\, \varphi_D(\xi)\, d\xi \tag{3.17}$$

$$= c_{ri} y + \int_y^{\infty} c_{od}\, (\xi - y)\, \varphi_D(\xi)\, d\xi \tag{3.18}$$

At this point it is necessary to minimize the expected total cost by taking the derivative and set it to zero. Assuming that the cost function has one point of minimum the expression would be:

$$C(y) = c_{ri} y + c_{od} \int_0^{\infty} (\xi - y)\, \varphi_D(\xi)\, d\xi - c_{od} \int_0^{y} (\xi - y)\, \varphi_D(\xi)\, d\xi \tag{3.19}$$

$$\frac{dC(y)}{dy} = c_{ri} - c_{od} + c_{od} \int_0^{y} \varphi_D(\xi)\, d\xi = 0 \tag{3.20}$$

$$= c_{ri} - c_{od}\, [1 - \int_0^{y} \varphi_D(\xi)\, d\xi] = 0 \tag{3.21}$$

$$= c_{ri} - c_{od}\, [1 - \Phi_D(y)] = 0 \tag{3.22}$$

Solving this expression results in:

$$\Phi_D(y) = \frac{c_{od} - c_{ri}}{c_{od}} \tag{3.23}$$

Therefore, the value y such that the condition above holds, minimize the cost.

In a discrete scenario, the cumulative distribution function of D is:

$$F_D(a) = \sum_{d=0}^{a} P_D(d) \tag{3.24}$$

The optimal solution is the smallest integer y such that:

$$F_D(y) \geq \frac{c_{od} - c_{ri}}{c_{od}} \tag{3.25}$$

To demonstrate that the solution minimizes C(y), let us calculate the second order derivative and verify that it is ≥ 0 for every value of y.

$$\frac{d^2C(y)}{dy} = c_{od}\,\varphi_D(y) \geq 0 \tag{3.26}$$

In the derivation the property of probability density is applied.

$$\int_0^{\infty} \varphi(\xi)\,d\xi = 1 \tag{3.27}$$

In addition, to find the derivative of a definite integral, the following formula has been used.

$$\frac{d}{dy}\int_{g(y)}^{h(y)} f(x,y)dx = \int_{g(y)}^{h(y)} \frac{\partial f(x,y)}{\partial y}dx + f(h(y),y)\frac{dh(y)}{dy} - f(g(y),y)\frac{dg(y)}{dy} \tag{3.28}$$

In the particular case of the derivation above:

$$\frac{d}{dy}\Big[-c_{od}\int_0^y (\xi - y)\varphi_D(\xi)d\xi\Big] = c_{od}\int_0^y \varphi_D(\xi)dx + (y-y)\varphi_D(y)\,1 - 0 \tag{3.29}$$

$$= c_{od}\int_0^y \varphi_D(\xi)dx \tag{3.30}$$

Similar approach is applied in the second order derivative.

$$\frac{d^2}{dy}\Big[-c_{od}\int_0^y (\xi - y)\varphi_D(\xi)d\xi\Big] = \frac{d}{dy}c_{od}\int_0^y \varphi_D(\xi)dx \tag{3.31}$$

$$= c_{od}\Big[\int_0^y 0dx + \varphi_D(y) - 0\Big] \tag{3.32}$$

$$= c_{od}\,\varphi_D(y) \tag{3.33}$$

3.2.2 Model with Initial Stock

Some additional remarks are needed in order to correctly model the scenario where a company already bought some reserved instances, i.e. model a initial stock level.

Let us assume to have purchased x reserved instances. The amount of reserved instances to be ordered in order to have y after the purchase is $y-x$. Therefore, it is necessary to change the cost equation of the expected cost:

The objective function can be expressed as follows.

$$\min_{y \geq x} \left[c_{ri}(y-x) + \int_{y}^{\infty} c_{od}(\xi - y)\varphi_D(\xi)d\xi \right] \tag{3.34}$$

The constraint $y \geq x$ ensures new reserved instances are purchased only if the current amount it is not sufficient. In particular, let y be the optimal number of reserved instances. If $x \geq y$ then there so need for a new purchase, because there are already enough reserved instances. Conversely, if $x < y$ it is necessary to order $y - x$ reserved instances to reach the optimum number y.

3.2.3 The Model and Cost Optimization in Cloud Computing

The main finding of the proposed model is expressed as following.

$$\Phi_D(y) = \frac{c_{od} - c_{ri}}{c_{od}} \tag{3.35}$$

This formula explains the relation between the demand of instances D, the optimal number of reserved instances y and the prices of reserved instances and on-demand instances.

Intuitively, the higher the cost-savings of reserved instances over on-demand instances the higher should be the optimal value of reserved instances. This fact is visible from the model, as $\frac{c_{od}-c_{ri}}{c_{od}}$ represents the saving rate using RIs. In addition, the models shows how the demand is a crucial component in reserved instances optimization.

The main objective of the model is to output the optimal number of reserved instances given observations of demand observations. However, it assumes the probability distribution of the demand to be known. In order to utilize the formula this work proposes two strategies. First, to fit a probability distribution to the demand and use the fitted cumulative distribution function. Second, it is possible to calculate the optimal number of instances

using the empirical distribution function of the observations. Section 5.1 extensively describes the former approach, while Section 6.3 explain the latter.

In order to validate the proposed model, the next chapter describes the implemented reserved instances optimizer. Furthermore, Chapter 6 discusses how the implemented system is improved after the theoretical findings.

Chapter 4

RIO: Reserved Instances Optimizer

As presented in Chapter 2, researchers and practitioners extensively studied the problem of cost optimization in cloud computing. In particular, researchers studied the positive effect of reserved instances planning by applying different methods such as Integer Programming, Evolutionary Algorithms, and Machine Learning.

However, the proposed approaches might be difficult to apply for the following reasons. First, all the methods aim to minimize both under-provisioning and over-provisioning costs, which is unrealistic in an industry case. Second, they are hard to implement and to maintain, hampering customization. Third, they do not take risks into account.

Industrial companies offer cost optimization tools which allow cloud users an easier and effective solution, providing useful data to support decision-makers. Table 2.4 summarizes the features of the tools available in industry. Nonetheless, the use of these tools has privacy problems and vendor lock-in risks.

This thesis proposes RIO, Reserved Instances Optimizer. RIO is a simple, effective, and extensible tool for cost optimization in cloud computing. The implementation is based on state-of-the-art techniques from industry and literature. It operates in four steps: calculation of opportunity size, reserved instances planning, visualization, and risk analysis.

The implemented system utilizes an heuristic approach to find the optimal number of reserved instances. The results obtained using RIO are then compared to the theoretical findings of Chapter 3.

Figure 4.1 shows an overview of Reserved Instances Optimizer. The system utilizes a bottom-up approach: the first step calculates the opportunity size for each type of reserved instances, selecting the best options to pur-

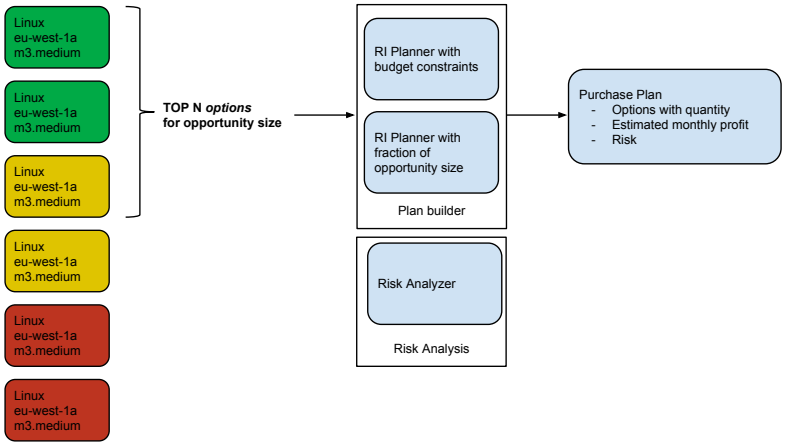


Figure 4.1: High level overview of the system.

chase. Afterwards, different types of reserved instances are bundle together according to the requirements of the decision-makers. Finally, the result is visualized in a report, showing the risks involved in the purchase.

4.1 Calculation of Opportunity Size

4.1.1 The Parameters to Choose

The calculation of opportunity size aims to assess which types of reserved instances are more beneficial to purchase. This section refers to different reserved instances types as *options*. A plan is composed of a set of *options*, and an *option* is composed of the parameters needed to define a reserve instance. In other words, building a purchase plan means to purchase a certain amount of *options*.

An option has five elements: *operating system*, *availability zone*, *size*, *term length*, and *purchase option*. A more formal definition follows.

$$O = \{linux, windows\} \tag{4.1}$$
$$Z = \{eu\text{-}central\text{-}1a, eu\text{-}central\text{-}1b, eu\text{-}central\text{-}1c...\} \tag{4.2}$$
$$S = \{m3.medium, m3.large, t2.small...\} \tag{4.3}$$
$$T = \{1\ year, 3\ years\} \tag{4.4}$$
$$P = \{no\ upfront, partial\ upfront, all\ upfront\} \tag{4.5}$$
$$option_i = (O_{os}, Z_{zone}, S_{size}, T_{term}, P_{purchase\_option}) \tag{4.6}$$

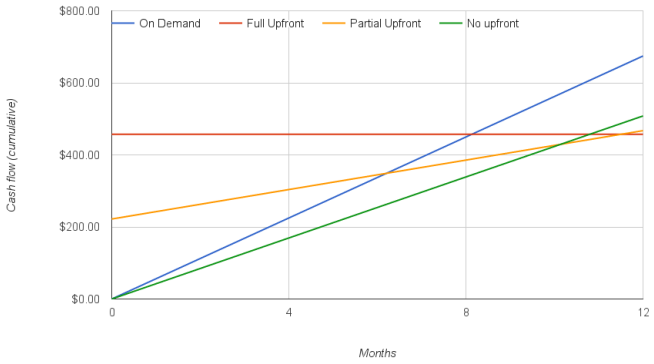


Figure 4.2: Comparison of the RI purchase options available at AWS. The chart refers to an option using *linux*, *eu-west-1a*, *m3.medium*. The on-demand cost is calculated using 100% of utilization.

This study focuses only on one-year term options. Three years term are excluded because they require more data about demand and infrastructure planning.

Moreover, despite the presence of multiple purchase options, this work utilizes only *partial upfront*. *Partial upfront* provides the right balance between initial investment and cost savings in the long term. *All upfront* has about 1-3% more savings after one year; however, the initial upfront fee is remarkably higher. This fact increases the risk; therefore, *all upfront* is excluded from the analysis. A possibility would be to include *no upfront*; nonetheless, the savings are significantly lower and it is avoided.

Figure 4.2 compares the three purchase options in terms of cost during a year. The chart refers to an option using *linux*, *eu-west-1a*, *m3.medium*. The on-demand cost is calculated using 100% of utilization of the instances.

Full upfront is a horizontal line, because after the upfront fee no more payments are needed. *On-demand* and *no upfront* linearly increases with time without upfront; however, the on-demand hourly price is higher. Finally, *partial upfront* has a lower upfront fee compared to *all upfront*, but costs occur during the year.

At month 12, the chart shows the different costs. *Full Upfront* and *partial upfront* are similar, while the others are significantly higher.

It is important to remark that the cost of on-demand instances depends on the extent to which the instances is utilized. For example, if an instance is utilized for one hour, the cost will be minimum. Conversely, reserved

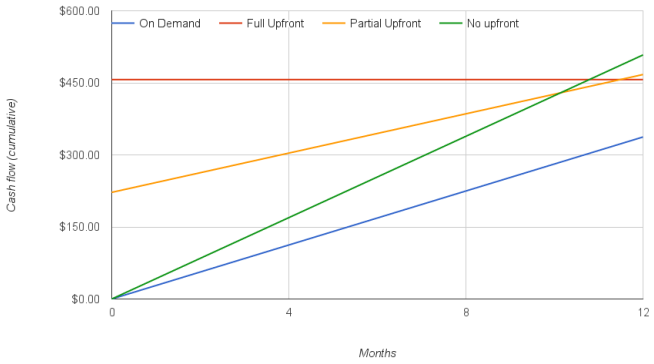


Figure 4.3: Comparison of the RI purchase options available at AWS. The chart refers to an option using *linux*, *eu-west-1a*, *m3.medium*. The on-demand cost is calculated using 50%. After 12 months the break-even point is not reached; therefore, it is cheaper to utilize only on-demand instances rather than reserved instances.

instances are paid every hour, independently of the demand. Therefore, in some cases is more convenient to use only on-demand instances (Figure 4.3). The scope of the analysis is to purchase reserved instances without incurring in this scenario.

To summarize, the calculation of opportunity size focuses on finding the types of reserved instances that yield the highest cost-savings. A type of reserved instances is identified by *operating system*, *size*, and *zone*.

4.1.2 Analysis of the Hourly Demand

In order to find the most profitable purchase, RIO processes the *hourly demand* of all instances of a company. The *hourly demand* refers to the number of instances per hour in a given time range.

As reported by the literature of cost optimization, forecasting future demand might have a strong impact on the performance of the optimizer [19]. However, forecasting demand requires a large amount of data and can be imprecise in case of limited data, leading to incorrect planning. As an alternative, studies show that an effective approach to deal with uncertainty is to analyze past data, for example the past thirty days [19]. The implemented system follows this approach to deal with uncertainty.

The analysis involves finding two values for each option: *maximum profit*

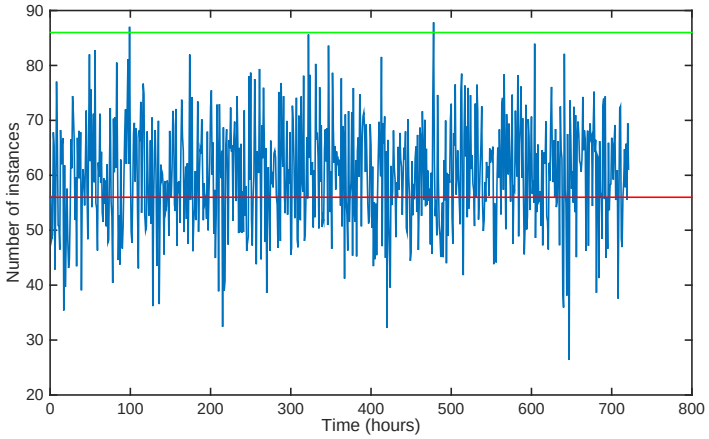


Figure 4.4: The figure illustrates a simulated demand during 30 days, the red line represents the maximum profit threshold, the green line represents the loss threshold.

threshold and the *loss threshold*. These metrics evaluate the effectiveness of an *option*.

The *maximum profit threshold* is the cost-saving associated with the optimal number of reserved instances, defined as *opportunity size*. The *opportunity size* is the number of reserved instances which maximizes the savings compared to an equivalent on-demand plan. In other words, the opportunity size is the number of reserved instances such that, buying one less results in under-provisioning and buying one more result in over-provisioning.

The *loss threshold* is the number of reserved instances, such that the over-provisioning costs exceed the cost-saving. In this case, the demand does not compensate the costs of a reserved instances investment. This threshold allows assessing the robustness of the investment. The higher the loss threshold, the more robust is the purchase plan. The reason is due to the fact that in case of harmful event, such as a decrease in demand, the reserved instances will still generate cost-saving compared to on-demand instances.

Both metrics highly depend on a time range. Figure 4.4 illustrates the two thresholds, thirty days of hourly demand is simulated with a Gaussian distribution with $\mu = 60.0$ and $\sigma = 10.0$.

4.1.3 Profit Function

The *profit function* represents the cost-saving achieved utilizing a certain amount of reserved instances. In other words, it is the margin between the

costs using only on-demand instances and the costs using a certain amount of reserved instances. It is calculated using a time range and it is formally defined as following.

$$c_{od} = \text{Hourly cost for an on-demand instance} \tag{4.7}$$
$$c_{ri} = \text{Effective hourly cost for an reserved instance} \tag{4.8}$$
$$d_i = \text{Number of on-demand instance running in a given hour} \tag{4.9}$$
$$y = \text{Number of reserved instances} \tag{4.10}$$
$$H = \text{Number of hours in the analyzed time frame} \tag{4.11}$$
$$\tag{4.12}$$
$$profit(d_1 ... d_H, y) = \sum_{d=1}^{H} c_{od}\, d_i - \left[y\, c_{ri} + c_{od}\, \max\{0, d_i - y\}\right] \tag{4.13}$$

where:

$$\tag{4.14}$$
$$d_i\, c_{od} = \text{Cost for using only on-demand instances} \tag{4.15}$$
$$y\, c_{ri} = \text{Cost for buying } y \text{ reserved instances (effective hourly cost)} \tag{4.16}$$
$$c_{od} \max\{d_i - y, 0\} = \text{ Cost of remaining instances after purchasing } y \text{ RIs} \tag{4.17}$$

The profit is calculated using the *effective hourly cost*. This value amortizes the hourly cost of a reserved instance over the length of the term, including upfront payments. The effective hourly cost simplify the calculation and allow flexibility in case of price change.

Figure 4.5 illustrates the $profit$ function, simulating the demand as random variable with Gaussian distribution with $\mu = 60.0$ and $\sigma = 10.0$.

The profit function proposed in this section is equivalent to the cost function formulated in the Inventory Theory model (Section 3.2.1). While the goal of the theoretical model is to minimize the cost, the implemented solution maximizes the profit. In the implemented system the demand is obtained directly from past data.

The goal of the proposed system is to maximize the profit function, and find automatically and effectively the thresholds. Reserved Instance Optimizer utilizes *hill-climbing*.

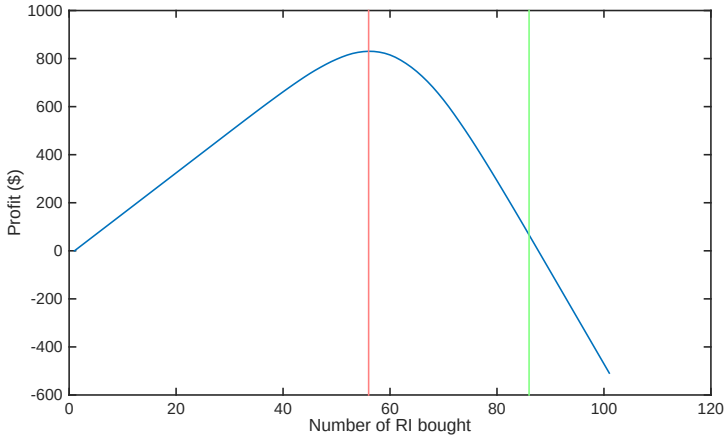


Figure 4.5: Profit function simulating demand as a Gaussian function with $\mu = 60.0$ and $\sigma = 10.0$. The red line represents the maximum profit threshold; the green line shows the loss threshold.

4.1.4 Hill-climbing

Hill-climbing refers to a local search heuristic [43]. Given a target, function $f(X)$, where X is a vector, the algorithm changes a single element in X at the time and adjust the value until no changes can be found to improve the result. Hill-climbing works effectively because the *profit function* has a global optimum.

Figures 4.6 and 4.7 show the pseudocode used to find the *maximum profit threshold* and the *loss threshold*.

4.2 Reserved Instances Planning

After analyzing single *options*, it is necessary to bundle different options in a plan. The goal of reserved instances planning step is to suggest a purchase plan, respecting some constraints.

RIO offers two types of constraints: finding the best plan given budget constraints and finding the best plan exploiting a fraction of the full opportunity size.

Both strategies rely on a pre-processing step which selects the top N options in terms of *maximum profit threshold*. An important decision is to decide how many options to take into account, i.e. how to decide the value of N. Since this is also a business decision, the system does not choose the parameter automatically, but it assumes it as given.

```
1: function findMaximumProfitThreshold(t_start, t_end)
2:     hourlyDemand ← getHourlyDemand(t_start, t_end)
3:     RICount ← min(hourlyDemand)
4:     bestRICount ← 0.0
5:     lastProfit ← 0.0
6:     finished ← false
7:     while !finished do
8:         profit ← calculateProfit(hourlyDemand, RICount)
9:         if profit ≥ lastProfit then
10:            bestRICount ← RICount
11:            lastProfit ← profit
12:            RICount ← RICount + 1
13:        else
14:            finished ← true
15:        end if
16:    end while
17: return (lastProfit, bestRICount)
18: end function
```

Figure 4.6: Hill-climbing approach to find the RI count the returns the best profit.

```
1: function findLossThreshold(t_start, t_end)
2:     hourlyDemand ← getHourlyDemand(t_start, t_end)
3:     RICount ← min(hourlyDemand)
4:     lossThreshold ← 0.0
5:     lastProfit ← 0.0
6:     finished ← false
7:     while !finished do
8:         profit ← calculateProfit(hourlyDemand, RICount)
9:         if profit ≥ 0.0 then
10:            lossThreshold ← RICount
11:            lastProfit ← profit
12:            RICount ← RICount + 1
13:        else
14:            finished ← true
15:        end if
16:    end while
17: return (lastProfit, lossThreshold)
18: end function
```

Figure 4.7: Hill-climbing approach to find the RI count the returns the loss threshold.

In order to assess a purchase plan, it is necessary to calculate the overall profit. Since a plan is a set of *options*, the profit of a plan is simply defined as the sum of the profit of the elements.

$$profit(plan) = \sum_i profit(option_i) \tag{4.18}$$

where:

$$option_i \text{ belongs to the plan} \tag{4.19}$$

The next sections present the details of the algorithms for creating a plan.

4.2.1 Budget Constraints

There are two types of costs related to a plan: upfront fees and monthly fees. The upfront fees are the sum of the upfront fees of all the *options*. The monthly fees are the sum of the hourly costs of the *options* in a month.

The algorithm assumes a budget for each type of cost, namely a budget for upfront fee and a budget allocated each month for the monthly fees. The objective is to find the plan that maximizes the profit margin without exceeding the budget constraints.

The formulation of the problem is the following.

$$Maximize\, profit(plan) \tag{4.20}$$

Subject to:

$$Upfront\ Cost \leq Upfront\ Budget \tag{4.21}$$
$$Monthly\ Cost \leq Monthly\ Budget \tag{4.22}$$

This problem is an instance of the *M-constraints Unbounded Knapsack problem* [49].

The knapsack problems are a well-known combinatorial optimization family of problems. In the general form, a knapsack problem is stated as follows. There are a set of items, each one with a volume and a value, and a knapsack with a fixed capacity. The goal is to find the collection with maximum value that fits in the knapsack.

The family of knapsack problems is divided in three categories, depending on the characteristics of the items.

0/1 knapsack problem Each item can be chosen only once.

Bounded knapsack problem Each item can be chosen a limited amount of times.

Unbounded knapsack problem There are no constraints on the number of items to be chosen.

A *M-constraints* variant has multiple constraints. For instance, having weight and volume constraints for each item.

The problem of building a plan can be modeled as an instance of knapsack problems in the following way. Every item is an *option* that can be purchase indefinitely. The limits are given by the budget, and the profit function, which takes into account the hourly demand. Every option has a value, the *profit*, and two characteristics linked with constraints *upfront fee* and *monthly fee*.

The optimization problem is proven to be $\mathcal{NP}$-hard [49]. Techniques to approach this family of problems include dynamic programming, branch and bound, and evolutionary algorithms. Researchers also underline the similarities between knapsack problems and integer programming [49]. Therefore, state of the art methods for cost optimization in cloud computing might be extended to solve this problem.

This work proposes a heuristic-based approach to find an approximate solution to this problem (Figure 4.8). The implemented solution iterates over the best N options and checks if adding a new reservation fits in the budget. If the new instance fits in the budget, it is added to the plan, and the available money decreases. The loop continues until there are no more instances available, or the budget is totally utilized.

The proposed solution is not guarantee to be optimal. For example, the system can choose to purchase the best option and completely fill the budget, while buying two options which are less profitable singularly might lead to a higher overall profit.

4.2.2 Exploiting a Fraction of Opportunity Size

In case of absence of budget constraints, a possible strategy is to purchase a fraction of the full opportunity size. The opportunity size is the number of reserved instances that generate the maximum profit. A use case, for instance, is a company want to buy 50% of the opportunity size, regardless the budget.

The proposed heuristics is simple and effective. Given the top N options, a fraction of the opportunity size is allocated. Figure 4.9 illustrates the algorithm: it iterates through the possible options and allocates a fraction of then for the purchase plan.

```
1: function createPlanWithBudget(N, t_start, t_end, upfront_budget,
   monthly_budget)
2:     topNcombinations ← getTopCombinations(N, t_start, t_end)
3:     B_upfront ← upfront_budget
4:     B_monthly ← monthly_budget
5:     plan ← new Plan()
6:     while enough_money && enough_instances do
7:         for c ∈ topNcombinations do
8:             if c fits in the budget then
9:                 plan.add((c, 1))
10:                B_upfront ← B_upfront - upfront_fee(c)
11:                B_monthly ← B_monthly - monthly_fee(c)
12:            end if
13:        end for
14:    end while
15: return plan
16: end function
```

Figure 4.8: Pseudocode of the proposed solution to build a plan using constraints on upfront fees and monthly fees.

```
1: function createPlanWithRate(N, t_start, t_end, rate)
2:     topNcombinations ← getTopCombinations(N, t_start, t_end)
3:     plan ← new Plan()
4:     for c ∈ topNcombinations do
5:         RICount ← findBestProfit(c)
6:         suggestedCount ← RICount * rate
7:         plan.add((c, suggestedCount))
8:     end for
9: return plan
10: end function
```

Figure 4.9: Algorithm for plan builder exploiting a fraction of opportunity size

Suggested plan (TOP 3 / 20% of opportunity)

Operating System	Zone	Size	Purchase option	How many
Linux	eu-west-1a	m3.medium	partialUpfront	5
Linux	eu-west-1a	m3.large	partialUpfront	5
Linux	eu-west-1a	t2.small	partialUpfront	5

Upfront Fee	$3,895.00
Monthly Fee	$332.15
Monthly Ammortize margin	$228.62

Figure 4.10: Summary of the proposed suggested plan with additional parameters.

4.3 Visualization

The visualization of the result is a crucial step. The goal is to provide an effective summary of relevant data to decision-makers. The implemented system generates a report showing the proposed plan. In addition, the report shows for each option belonging to the plan a detailed analysis. The detailed analysis illustrates the opportunity size in different months, the loss threshold, the hourly demand, and the utilization of previously purchased reserved instances. Figures 4.10, 4.11, and 4.12 show parts of the report generated with a simulation.

4.4 Risk Analysis

The risk analysis step analyzes the risks involved in purchasing a set of reserved instances. The identified risks are the following.

- Demand of an instance decreases. In other words, a possible risk is that the utilization of an option lowers until the reserved instance are not cost-effective.
- A new instance type is released. Cloud providers release new instances regularly. Since reserved instances require a commitment for a long term, a risk is to be forced to utilize old instances. This could happen if a company execute the purchase before the release of new instances.

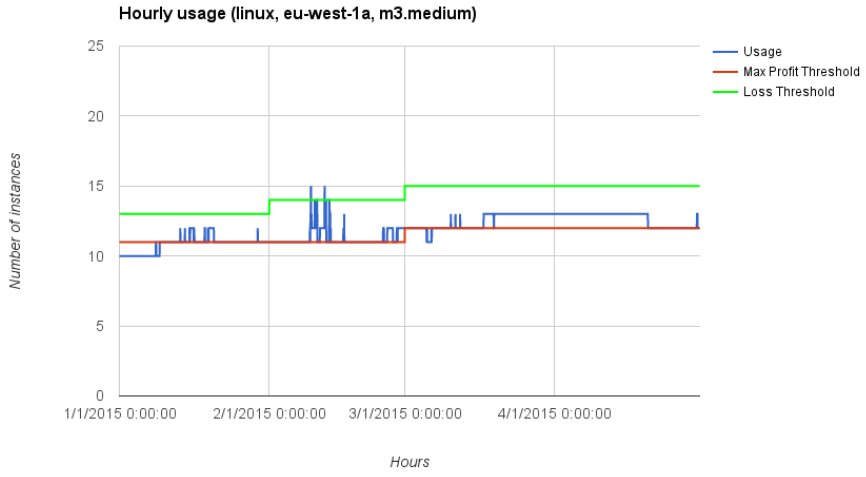


Figure 4.11: Detailed analysis of a single option belonging to the proposed plan in different months.

Operating System	Zone	Size	Purchase Option	Quantity	Active from	Utilization ratio
linux	eu-west-1a	m3.large	Partial upfront	3	1/4/2015	100.00%
linux	eu-west-1a	m3.medium	Partial upfront	3	1/4/2015	100.00%
linux	eu-west-1a	t2.small	Partial upfront	3	1/4/2015	100.00%

Figure 4.12: Monitoring of existing reserved instances.

- Infrastructure changes. Another possible risk is to be unable to change infrastructure because of a high number of reserved instances.
- The cloud provider lower the prices. Periodically cloud providers decrease the upfront fee and hourly fee of instances. A potential risk is to perform a large investment before price changes.

Cloud providers usually offer tools to mitigate these risks. For example, it is possible to modify reserved instances or to sell them [7, 31]. However, these operations can be complex to execute since they need time and expertise. This step aims to pro-actively deal with the risk involved in a reserved instances purchase.

4.4.1 Risk Mitigation Strategies

This section describes some risk-mitigation strategies crucial part of the reserved instances purchase process.

An important strategy is to iterate the process of buying reserved instances regularly and in short time frames. This approach mitigates the risk of large investments before the release of a new instance type. In addition, short iteration diminish the harmful effect of price decrease. Furthermore, the past data analyzed for the optimization is constantly updated and the opportunity size increases or decreases with the actual demand, leading to more accurate estimations.

Decision-makers are also encouraged to evaluate the risks from results of the analysis. *Loss threshold* illustrates the maximum number of reserved instances until the underutilization costs will hamper their profitability. Another recommendation for practitioners is to purchase a fraction of the opportunity size, so that if the demand decreases the reserved instances will remain fully utilized.

An additional approach is to choose the IT infrastructure of a company, so that the instances belong to the same family and same availability zone. Amazon Web Services allows changes to reserved instances if they belong to the same region and the same family [31]. In particular, small instances can be combined to form a bigger one (e.g. two m3.medium are merged in m3.large). Conversely, it is possible to split large sizes to smaller ones (e.g. one m3.large is split in two m3.medium). Buying reserved instances in the same family allows more flexibility in changing the infrastructure.

4.4.2 Risk Calculation

The implemented system automatically takes into account some of the risk parameters involved in a reserved instances purchase.

First, the system excludes from the purchase plan any *option* which is announced to be retired from the market. Amazon Web Services, for example, regularly retires old instances and releases new ones. When a new plan is created the old instances are excluded in favor of new ones.

Second, RIO provides to decision-makers a degree of risk based on the age of the instances. The thresholds of risks are empirically decided: if an instance type has been released for more than two years than the risk is *high*. After two years, in fact, instance types are usually substituted or prices decrease. If the release date is between one year and two years the risk is *medium*. If it is less than one year the risk level is *low*. This categorization aims to direct decision-makers purchase newer instances, which are more efficient and less susceptible to price change. Table A.1 provides information about the release dates from Amazon Web Services.

Chapter 5

Evaluation

This chapter describes the experiments designed to evaluate the theoretical model and the implemented system. The conducted experiments utilize real data from an industry case in order to demonstrate the effectiveness of the implemented system and the theoretical model. However, data regarding demand of cloud resources is confidential. In order to describe the experiments and illustrate the results, the traces are obfuscated and some information is omitted. In addition to real data, this work proposes experiments in an ideal environment using simulations.

This thesis has been developed at the R&D team at Nordcloud[1]. Nordcloud is a European cloud infrastructure consulting and managed service company, with headquarters in Helsinki, Finland. As cost-optimization problem arises in the company, the proposed solution has been developed and evaluated. In particular, Nordcloud provides the real data for the evaluation.

5.1 Evaluation of the Model

Chapter 3 presents a model based on Inventory Theory. The optimal number of reserved instances y has proved to respect the following condition.

$$\Phi_D(y) = \frac{c_{od} - c_{ri}}{c_{od}} \tag{5.1}$$

[1] `http://www.nordcloud.com`

Where Φ_D is the cumulative distribution function of the demand, approximated with a continuous random variable. Modeling the demand with a discrete random variable results in the following formula. Given F_D, the cumulative distribution function of the discrete demand D, the optimal number of instances is the smallest integer y such that:

$$F_D(y) \geq \frac{c_{od} - c_{ri}}{c_{od}} \tag{5.2}$$

In the evaluation of the model it is crucial to understand the performance of the model in the ideal case and using real data. This work proposes two experiments. The first experiment test ideal cases by simulating the demand and by using different probability distributions. The second experiment utilizes demand data from a industry case.

5.1.1 Simulation

Experiment Setup

This set of experiments generate hourly demand of instances using known distributions. The generated traces simulate a month of data.

The simulation assumes that the demand is a set of observations independent identically distributed. The evaluation proposes four probability distributions: Uniform, Exponential, Poisson, and Gaussian. Uniform and Exponential demonstrate an analytical approach. Poisson is an example of discrete probability distribution. Finally, Gaussian distribution is also proposed. The distribution are chosen empirically investigating real data and they serve to verify the correctness of the model in a ideal case.

After generating the demand, each experiment computes the optimal number of reserved instances using the inventory theory model and the hill-climbing approach. Then, the two results are then compared.

All the calculations assume that the demand refer to an *option* with parameters *linux*, *eu-west-1a*, *m3.medium*, *partial upfront*. Table 2.2 illustrates the pricing details of this instance type[2].

Uniform Distribution

A uniform distribution is a distribution with constant probability. The probability density function and cumulative distribution function in the interval $[a, b]$ are the following.

[2]The price refers to the public pricing available at 2015-05-01.

$$\varphi_D(\xi) = \begin{cases} 0 & \text{if } \xi < a. \\ \frac{1}{b-a} & \text{if } a \leq \xi \leq b \\ 0 & \text{if } \xi > b. \end{cases} \tag{5.3}$$

$$\Phi_D(y) = \begin{cases} 0 & \text{if } y < a. \\ \frac{y-a}{b-a} & \text{if } a \leq y \leq b \\ 1 & \text{if } y > b. \end{cases} \tag{5.4}$$

Figure 5.1a shows an example of the simulated demand using a uniform distribution in the interval $[10, 15]$. The simulation uses a continuous uniform distribution to approximate the demand. Assuming the optimal number of reserved instance is $10 \leq y \leq 15$, we obtain the following.

$$\Phi_D(y) = \frac{c_{od} - c_{ri}}{c_{od}} \tag{5.5}$$

$$\frac{y - 10}{15 - 10} = \frac{0.077 - 0.0533}{0.077} \tag{5.6}$$

$$y = 11.54 \tag{5.7}$$

Rounded at:

$$y = 12 \tag{5.8}$$

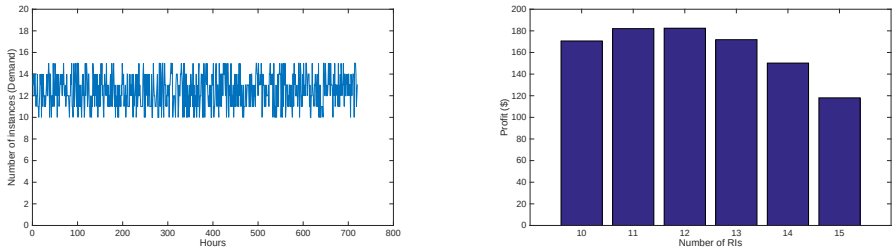
The hill-climbing approach finds the same results. It iterates from the minimum value in the hourly distribution and it increases the optimal number until the profit starts to decrease. Figure 5.1b shows the profit function and the optimal value found with hill-climbing.

Exponential Distribution

This section utilizes an exponential distribution to simulate the demand. The characteristic of probability distribution function and cumulative distribution function are the following.

$$\varphi_D(\xi) = \begin{cases} 0 & \text{if } \xi < 0. \\ \lambda e^{-\lambda \xi} & \text{if } \xi \geq 0. \end{cases} \tag{5.9}$$

$$\Phi_D(y) = \begin{cases} 0 & \text{if } y < 0. \\ 1 - e^{-\lambda y} & \text{if } y \geq 0. \end{cases} \tag{5.10}$$



(a) Hourly demand trace. (b) Profit function.

Figure 5.1: Simulation of demand using a continuous uniform distribution in $[10, 15]$

Assuming the demand to follow an exponential distribution with $\lambda = 0.25$, The optimal solution is obtained with the following steps.

$$\Phi_D(y) = \frac{c_{od} - c_{ri}}{c_{od}} \tag{5.11}$$

$$1 - e^{-0.25y} = \frac{0.077 - 0.0533}{0.077} \tag{5.12}$$

$$e^{-0.25y} = -\frac{0.077 - 0.0533}{0.077} + 1 \tag{5.13}$$

$$-0.3y = ln\Big(1 - \frac{0.077 - 0.0533}{0.077}\Big) \tag{5.14}$$

$$y = 1.47 \tag{5.15}$$

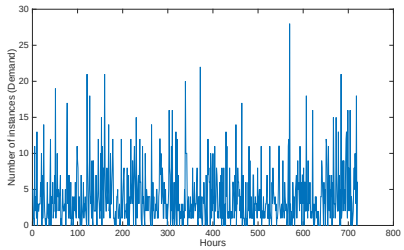
Rounded at:

$$y = 2 \tag{5.16}$$

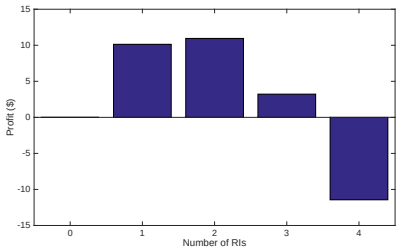
Figure 5.2a shows the simulation of the demand using an exponential distribution with $\lambda = 0.25$. Figure 5.2b shows the maximization of the profit function. Hill-climbing outputs the same optimal value as the inventory theory model.

Poisson Distribution

This section utilizes a Poisson distribution to simulate the demand. The Poisson distribution is a discrete probability distribution and the probability mass function and cumulative distribution function are the following.

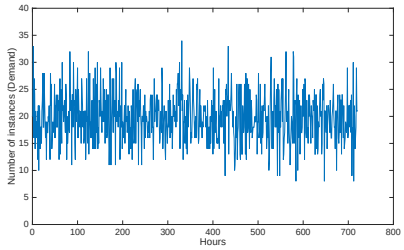


(a) Hourly demand trace.

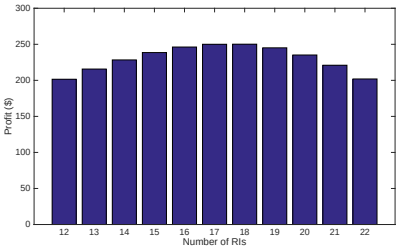


(b) Profit function.

Figure 5.2: Simulation of demand using a exponential distribution with $\lambda = 0.25$



(a) Hourly demand trace.



(b) Profit function.

Figure 5.3: Simulation of demand using a Poisson distribution with $\lambda = 20$

$$p(x;\lambda) = \frac{\lambda^x}{x!}e^{-\lambda} \qquad \text{for } x = 0, 1, 2, ... \tag{5.17}$$

$$F(x;\lambda) = \sum_{i=0}^{x} p(x;\lambda) = \sum_{i=0}^{x} \frac{\lambda^x}{x!}e^{-\lambda} \tag{5.18}$$

Figure 5.3a shows the simulated demand and figure 5.3b illustrates the profit function used in the implemented system.

The theoretical optimum is calculated numerically. The same result is obtained running the hill-climbing approach.

$$F(x;\lambda) \geq \frac{c_{od} - c_{ri}}{c_{od}} \tag{5.19}$$

$$x = 18 \tag{5.20}$$

Gaussian Distribution

Finally, this work proposes a simulation of the hourly demand using a Gaussian distribution with $\mu = 12$ and $\sigma = 2$. The Gaussian distribution is characterized by the following probability density function and cumulative distribution function.

$$\varphi_D(\xi) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2} \tag{5.21}$$

$$\Phi_D(y) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{y} e^{-(t-\mu)^2/2\sigma^2} dt \tag{5.22}$$

The theoretical result is expressed as $\Phi_D(y) = \frac{c_{od}-c_{ri}}{c_{od}}$. This experiment obtains the value of y numerically using the tool *Matlab* [3].

$$\Phi_D(y) = \frac{c_{od} - c_{ri}}{c_{od}} \tag{5.23}$$

when:

$$y = 10.99 \tag{5.24}$$

Rounded at:

$$y = 11 \tag{5.25}$$

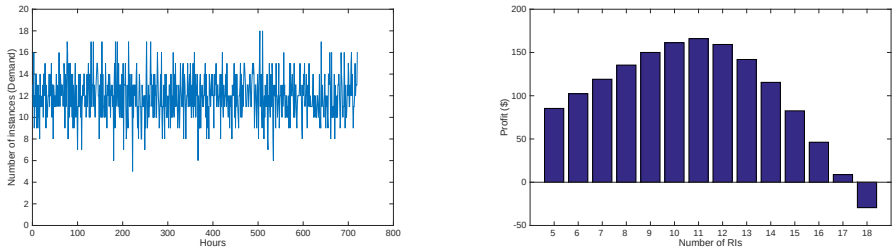
Figure 5.4a shows the simulated trace, while Figure 5.4b shows the profit function varying the number of reserved instance. As in the previous cases, the theoretical results and the empirical results confirm the same optimal solution.

Final remarks

This set of experiments demonstrate that the theoretical findings and the implemented system are equivalent in an ideal case. The ideal case is represented by a known probability distribution of the demand. Despite some errors might occur if modeling demand with a continuous distribution, the experiments show accurate results.

The next section describes the performance of the model and the reserved instance optimizer using real data.

[3] `http://se.mathworks.com/products/matlab/`



(a) Hourly demand trace. (b) Profit function.

Figure 5.4: Simulation of demand using a Gaussian distribution with $\mu = 12$, $\sigma = 2$.

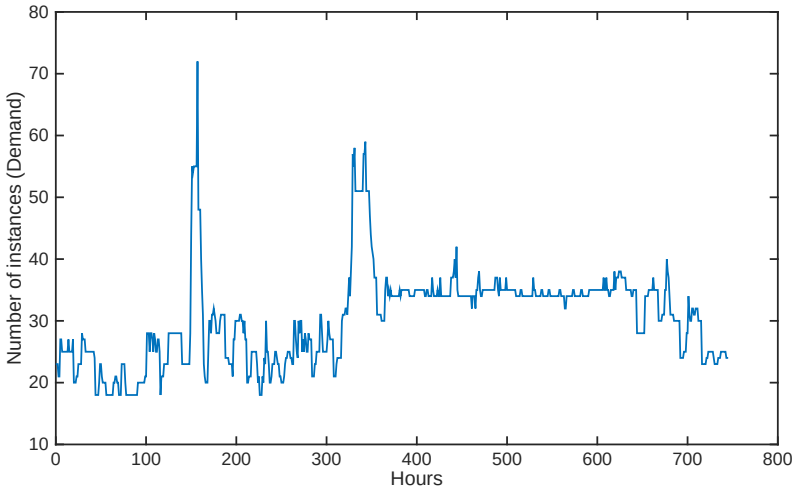


Figure 5.5: Obfuscated hourly distribution from the industry case.

5.1.2 Industry Case

Experiment Setup

In order to measure the performance of the model using real data, the experiment utilizes the hourly distribution from the industry case. For confidentiality, this study report an obfuscated trace. The parameters of the studied option are *linux*, *eu-west-1a*, *m3.medium*, *partial upfront*. Figure 5.5 shows the trace of hourly demand for the industry case.

The experiment consists in two steps. The first step finds the optimal number of reserved instances using hill-climbing. The second step calculate the result using the model and compares the result. In order to utilize the model, it is necessary to find a probability distribution which fits the real

demand trace. The estimation is achieved using maximum likelihood estimation.

Hill-climbing

The obfuscated demand is given as input to the hill-climbing heuristic. The optimal number of reserved instances results to be 25. Figure 5.6 reports the profit with different value of reserved instances.

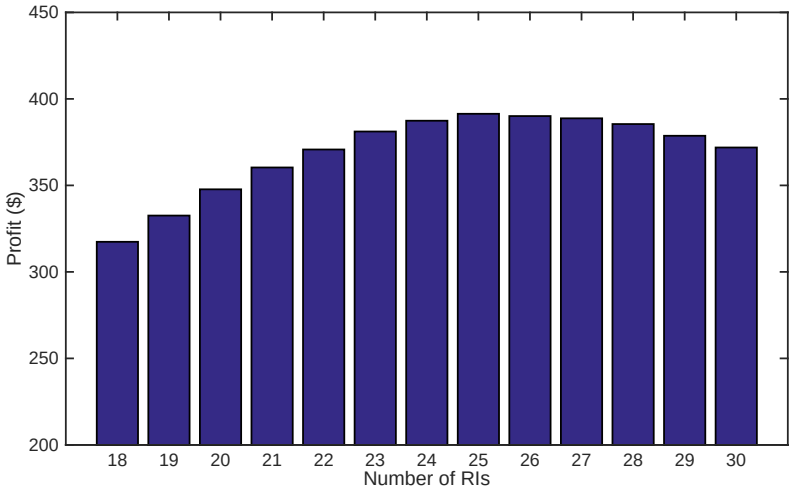


Figure 5.6: Profit calculated varying the number of reserved instances, using the obfuscated demand.

Parameter Estimation

The performance of the proposed model highly depends on the utilized probability distribution. In order to find the parameters of a probability distribution, this work utilizes maximum likelihood estimation.

Maximum likelihood estimation is a well-known method in inferential statistics. Let us assume to have N independent identically distributed observations $x_i \sim p(x|\theta)$. The goal is to find the parameters θ of the distribution.

The maximum likelihood estimation finds the parameter θ, such that $p(\theta|X)$ is maximum. Intuitively, $p(\theta|X)$ is exactly the probability of having a parameter θ observing the data X [2].

Assuming that the demand trace is composed of observations independent identically distributed, it is possible to apply the maximum likelihood approach to estimate the parameters of the distribution. In particular, the

total likelihood of the observations results to be the product of the likelihood of the single ones. Besides, it is possible to calculate the *log-likelihood* to simplify calculations [2].

$$L(\theta|X) \equiv p(X|\theta) = \prod_{i=1}^{N} L_i = \prod_{i=1}^{N} p(x_i|\hat{\theta}) \tag{5.26}$$

This work proposes three estimations. The first and the second use a Uniform and Poisson distribution respectively. The maximum likelihood estimation is calculated analytically. The third uses a Gaussian distribution and the estimator is derived numerically.

Uniform Distribution

Assuming that the demand data is drawn from a uniform distribution, it is necessary to estimate the parameters θ_1 and θ_2: the extreme of the interval of the uniform distribution [58].

The probability distribution function follows.

$$p(X|\theta) = \begin{cases} \frac{1}{\theta_1 - \theta_2} & \text{if } \theta_1 \leq X \leq \theta_2. \\ 0 & otherwise \end{cases} \tag{5.27}$$

Let us now calculate the likelihood:

$$L = \prod_{i=1}^{n} p(x_i|\theta) = \prod_{i=1}^{n} \frac{1}{\theta_2 - \theta_1} \tag{5.28}$$

Applying the natural logarithm the expression would be:

$$ln\, L = \sum_{i=1}^{n} ln \frac{1}{\theta_2 - \theta_1} \tag{5.29}$$

$$= \sum_{i=1}^{n} -ln(\theta_2 - \theta_1) = -n\, ln(\theta_2 - \theta_1) \tag{5.30}$$

We need to maximize the likelihood, which is achieved by minimizing $ln\,(\theta_2 - \theta_1)$. The logarithm is minimum when the argument is minimum; therefore, the goal is to minimize $\theta_2 - \theta_1$. For construction of the probability density function, $\theta_1 \leq x_i$ for every i, and $\theta_2 \geq x_i$ for every i. Therefore, the maximum likelihood estimators are:

$$\hat{\theta_2} = \max(x_1 ... x_n) \tag{5.31}$$

$$\hat{\theta_1} = \min(x_1 ... x_n) \tag{5.32}$$

In the industrial case, the estimators are the maximum and the minimum of the demand trace.

$$\hat{\theta_2} = 72 \tag{5.33}$$

$$\hat{\theta_1} = 18 \tag{5.34}$$

Therefore, applying the inventory theory model, we calculate the optimal number of reserved instances y:

$$y = 35 \tag{5.35}$$

Poisson Distribution

Assuming the demand follows a Poisson distribution, we need to estimate the parameter λ using maximum likelihood estimation [33].

The probability distribution function follows.

$$p(x;\lambda) = \frac{\lambda^x}{x!}e^{-\lambda} \qquad \text{for } x = 0, 1, 2, ... \tag{5.36}$$

The likelihood is:

$$L = \prod_{i=1}^{n} \frac{\lambda^{x_i}}{x_i!}e^{-\lambda} = \frac{e^{-\lambda\, n}\, \lambda^{\sum_{i=1}^{n} x_i}}{\prod_{i=1}^{n} x_i!} \tag{5.37}$$

The log-likelihood is:

$$ln\, L = -\lambda\, n + \sum_{i=1}^{n} x_i\, ln(\lambda) - \sum_{i=1}^{n} ln(x_i!) \tag{5.38}$$

Let us compute the derivative with respect to λ in order to maximize the likelihood and set it to zero.

$$\frac{\partial ln\, L}{\partial \lambda} = -n + \frac{\sum_{i=1}^{n} x_i}{\lambda} \tag{5.39}$$

Let us set the derivative to zero.

$$-n + \frac{\sum_{i=1}^{n} x_i}{\lambda} = 0 \tag{5.40}$$

And find the maximum likelihood estimator:

$$\hat{\lambda} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{5.41}$$

Let us take the second order derivative of the likelihood to check that the likelihood is maximized.

$$\frac{\partial ln\, L}{\partial \lambda} = -n + \frac{\sum_{i=1}^{n} x_i}{\lambda} \tag{5.42}$$

$$= -\frac{\sum_{i=1}^{n} x_i}{\lambda^2} < 0 \qquad \text{for every } \lambda \neq 0 \tag{5.43}$$

Therefore, the maximum likelihood is maximized. Applying the maximum likelihood estimation in the real demand case we have:

$$\hat{\lambda} = 30.3858 \tag{5.44}$$

We can use a Poission distribution with $\lambda = \hat{\lambda}$ in the model and obtain the following optimal number of RIs:

$$y = 27 \tag{5.45}$$

Gaussian Distribution

This experiment utilizes the maximum likelihood estimation for the real demand observation, assuming the observations belongs to a Gaussian distribution.

The estimated parameters are calculated numerically.

$$\mu = 30.3858 \tag{5.46}$$

$$\sigma = 7.48802 \tag{5.47}$$

At this point, it is possible to utilize the finding of the model plugging the estimated parameters.

$$\Phi_D(y) = \frac{c_{od} - c_{ri}}{c_{od}} \tag{5.48}$$

when:

$$y = 26.62 \tag{5.49}$$

Rounded at:

$$y = 27 \tag{5.50}$$

Final Remarks

The experiments conducted using real data provide good results. The theoretical model and the hill-climbing approach are similar using Poisson and Gaussian distribution. Uniform distribution might not be a good choice in this context, because it tends to overestimate peaks of usage, which are common in real traces.

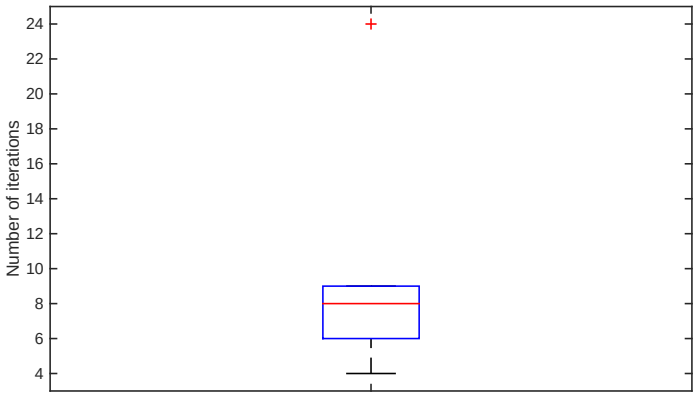


Figure 5.7: Number of iterations of the hill-climbing approach.

5.2 Evaluation of Reserved Instances Optimizer

This section proposes a set of experiments to evaluate the implemented solution. In particular, the first experiment analyzes the performance of the hill-climbing approach. The second experiment focuses on assessing the performance of the heuristic for creating a plan with budget constraints.

5.2.1 Hill-climbing Heuristic

In the implemented system the hill-climbing heuristic is crucial to obtain the optimal number of reserved instances. The proposed approach calculates the profit using a certain number of RIs and stops at the maximum value. It is interesting to measure the number of iterations using different *options*. Figure 5.7 depicts the results.

5.2.2 Reserved Instance Planning with Budget Constraints

The proposed solution is based on two types of instances planner: reserved instances planner with budget constraints and reserved instances planner exploiting a fraction of the opportunity size. This experiment focuses on analyzing the performance of the first type.

The proposed heuristic is not optimal; however, the system outputs a plan that efficiently utilizes the budget. In order to demonstrate this fact,

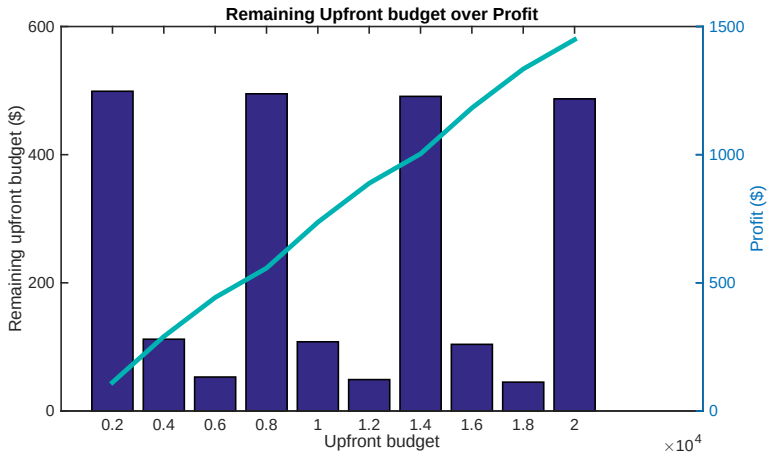


Figure 5.8: Remaining upfront budget and profit. The calculation are done using a 30 days period, using the top 5 options.

this work proposes two experiments.

In both experiments the algorithm analyzes the best five options in terms of opportunity size. The first experiment runs the algorithm with an increasing upfront budget, the monthly budget is set large enough to be negligible. Figure 5.8 shows the budget not allocated, together with the profit. A remarkable result is that the profit increases linearly with the upfront fee; therefore, the resources are efficiently allocated. A similar experiment is performed setting a large upfront fee and using an increasing monthly budget. Figure 5.9 shows the result. The experiments prove that with an increasing investment the resources are allocated accordingly and it efficiently utilizes the budget.

5.3 Pilot Purchase

This thesis has been developed at the R&D team at Nordcloud. From Nordcloud website it is possible to read: "we are experts in the automation and management of modern Cloud infrastructure. We offer Cloud service consultancy, migration and capacity planning, optimization, monitoring, and infrastructure management, and we assist enterprises to gain from lean business processes and maximum cost benefit from their Cloud investments, while delivering world-class security, reliability and service quality".

In other words, the value proposition of Nordcloud focuses on supporting companies in benefit from cloud computing. In particular, the offer of Nordcloud concentrates on Infrastructure as a Service providers (IaaS). IaaS gives

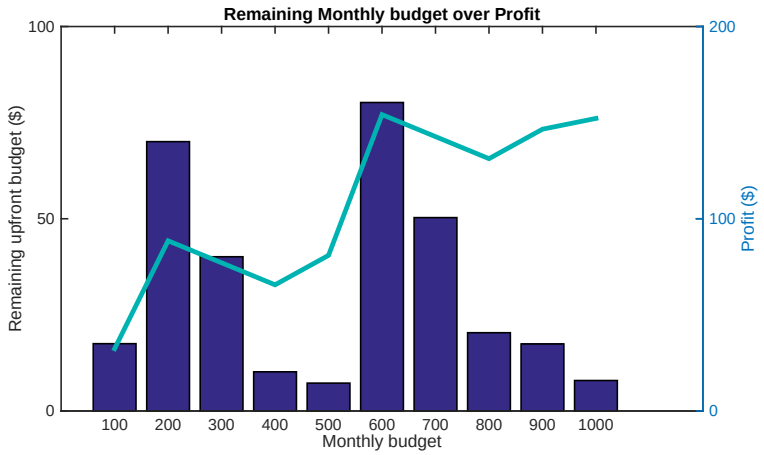


Figure 5.9: Remaining monthly budget and profit. The calculation are done using a 30 days period, using the top 5 options.

to the user full control of the cloud resources and it allows to build complex infrastructure. However, it is harder to manage and utilize.

An important component of the business of Nordcloud relates to cost optimization. Nordcloud operates complex cloud infrastructures on behalf of its customers, which incur significant cloud capacity costs. The Reserved Instance Optimizer has been developed to be part of the software utilized by Nordcloud to ensure high level services, both internally and for the customers.

Nordcloud management team executed a pilot purchase of reserved instances using the suggested plan of the implemented Reserved Instances Optimizer. The details of the purchase plan and the performance of the instances are confidential. However, the reserved instances purchased following the suggestion of the system are allocated 100% of the time. This scenario produces the highest cost-saving.

Chapter 6

Discussion

6.1 Maximum Likelihood Estimation of the Demand

Section 5.1 evaluates the inventory theory model in two scenarios: an ideal case where the distribution of demand is known and in a real industry case. In both scenarios the theoretical findings are compared with the hill-climbing approach, which outputs the ground-truth value.

In the ideal case the model and the empirical approach match in all the cases. Small errors might occur when approximating the demand with a continuous random variable.

In the real demand case the main challenge is to estimate the parameters of the distribution to use in the model. The proposed approach is to use maximum likelihood estimation. However, this approach has some limitations. In particular, it assumes that the observations, i.e. the values of demand, are independent identically distributed. This assumption is not valid in the case of the demand, since it is a time series and the previous values influence the future ones.

Nonetheless, the results in the industry case are similar to the hill-climbing results, using a Poisson or Gaussian distribution. Uniform distribution, instead, does not perform well. The Uniform distribution tends to overestimate outliers. Since in cloud computing peaks in demand occur frequently, this distribution might be avoided in this context.

The differences between hill-climbing and the model are due to the fact that the estimated distribution does not exactly fit the real one. Figure 6.1 shows the real demand and the trace generated with the Poisson distribution estimated in Section 5.1.2.

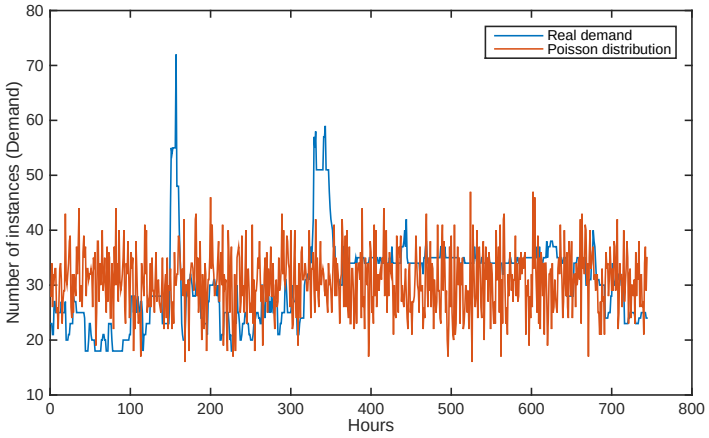


Figure 6.1: Obfuscated demand from the industry case and a simulation with the estimated Poisson distribution.

6.2 Granularity of the Analysis

The implemented system and the experiments propose analysis based on hourly demand. Hourly demand is the most precise value of demand that cloud providers offer. Furthermore, it offers detailed information about changes during the day, such as differences during working hours and night time.

However, less granularity in the analysis might decrease the noise of peaks of demand. In addition, aggregated values decrease the number of data to analyze, improving the performance of the algorithm.

Figure 6.2 shows a comparison between the hourly and the daily demand. The aggregation is performed taking the average demand for each day. Despite the possible advantages, this strategy might lead to wrong results. Further investigations are needed to better understand this approach.

6.3 Improving the Implemented System

As discussed in Section 5.1, the inventory theory model predicts the optimal number of reserved instances. This section discusses possible improvements of the implemented system after the theoretical findings.

The main theoretical finding from the model states that the optimal num-

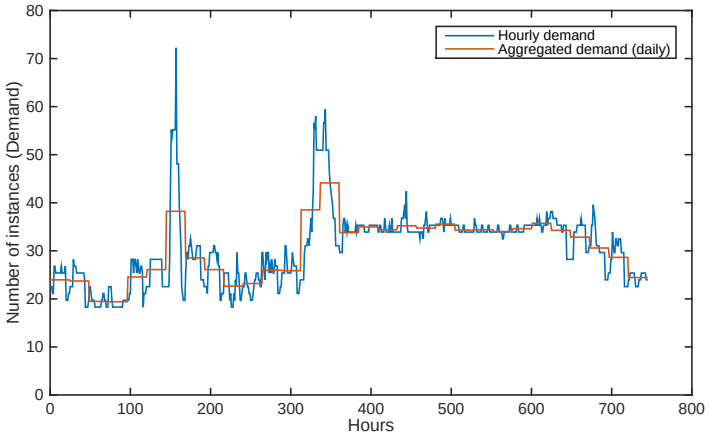


Figure 6.2: Comparison between the hourly demand and the daily demand.

ber of reserved instances y is such that.

$$\Phi_D(y) = \frac{c_{od} - c_{ri}}{c_{od}} \tag{6.1}$$

Let us apply to both members of the equation Φ_D^{-1}, the inverse of the cumulative distribution function. At this point we obtain the following.

$$y = \Phi_D^{-1}\left(\frac{c_{od} - c_{ri}}{c_{od}}\right) \tag{6.2}$$

The quantile function Φ_D^{-1} is a function such that $\Phi_D^{-1}(n)$ is the p^{th} quantile of the distribution. The p^{th} quantile is defined as:

$$q_p = \min\{q : \Phi_D(q) \geq p\} \tag{6.3}$$

In order to find the p^{th} quantile, this study proposes the quantile estimation of a set of observation [37]. As discussed in Section 5.1.2, we assume every observation of the hourly demand to be independent identically distributed.

Given $d_1 ... d_N$, N observations of the demand D, we calculate the i^{th} order statistic of the sample. In other words, we re-order the observations such that $d_{(1)} \leq d_{(2)} \leq ... \leq d_{(N)}$. Then we calculate:

$$q_n = D_{(\lceil Np \rceil)} \tag{6.4}$$

Where $D_{(\lceil Np \rceil)}$ is the smallest $\lceil Np \rceil^{th}$ value of demand.

The finding of the model states that we should find the p^{th} quantile where:

$$p = \frac{c_{od} - c_{ri}}{c_{od}} \tag{6.5}$$

The algorithm to find the p^{th} quantile is simple. Given N observations of the hourly demand, the steps are the following.

1. Sort $d_1 ... d_N$ in $d_{(1)} ... d_{(N)}$
2. Calculate $p = \frac{c_{od} - c_{ri}}{c_{od}}$
3. Output $D_{(\lceil Np \rceil)}$

Let us explain the concept with an example from the industry case utilized in Section 5.1.2[1].

Let d_1 be the demand at hour 1. In general, d_i is the demand at the i^{th} hour. Since the analyzed option is *linux*, *eu-west-1a*, *m3.medium* we have:

$$p = \frac{0.077 - 0.055}{0.077} = 0.3 \tag{6.6}$$

$$N = 744 \qquad \text{Number of hours in a month} \tag{6.7}$$

$$\lceil Np \rceil = \lceil 744 * 0.3 \rceil = 224 \tag{6.8}$$

$$D_{(224)} = 25 \qquad \text{Same result as with hill-climbing (Section 5.1.2).} \tag{6.9}$$

Extensive empirical tests have been performed using all possible combinations of the parameters. The estimator of quantile outputs accurate results, always matching with the hill-climbing approach.

Calculating the *opportunity size* utilizing quantile results to be also more efficient. While the hill-climbing approach iterates until convergence, the quantile estimator approach sorts the data and selects the corresponding element. Figure 6.3 compares the performance of the hill-climbing approach and the quantile estimation approach. The outcomes shows that quantile estimation is a more efficient approach.

[1]The price refers to the public pricing available at 2015-05-01.

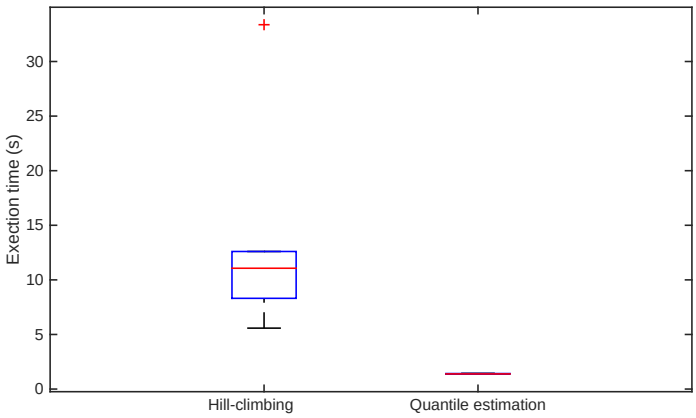


Figure 6.3: Efficiency of the two proposed approaches used to calculate the opportunity size. The performance measurements are repeated multiple times, the picture illustrates the execution time of the evaluations.

Chapter 7

Conclusion

This thesis analyzes the problem of cost optimization in cloud computing. Cloud computing refers to a paradigm for accessing computing resources which is becoming increasingly popular. Despite the fact that having a cloud infrastructure is usually cheaper than maintaining a physical data center, owners of large and complex IT infrastructure might incur large costs. Therefore, the problem of cost optimization in cloud computing is becoming increasingly important.

One of the most common and effective techniques for cost savings is to utilize reserved instances. Reserved instances are computing resources reserved for a fixed term. Since they require an higher commitment from the users, they result to be cheaper than on-demand instances. However, reserved instances need careful planning in order to be cost-effective and to maximize the savings.

As discussed in Chapter 2, researchers extensively studied the topic of cost optimization in cloud computing, proposing interesting approaches such as Integer Programming, Evolutionary Algorithms, and Machine Learning techniques. Industry tools also offer analysis of a cloud infrastructure and perform cost optimization (Table 2.4).

Despite researchers and practitioners studied this topic, still work has to be done. This thesis aims to answer the following research questions.

- How cost optimization in cloud computing can be modeled with Inventory Theory?
- How an heuristic-based reserved instance planner perform?
- Does the implemented system verify the theoretical results of the inventory theory model?

Chapter 3 answers the first question by proposing a theoretical model based on Inventory Theory. Inventory Theory aims to describe the behavior of an inventory system. The problem of planning reserved instances has remarkable similarities with inventory-keeping challenges and the model results to be accurate.

The second question is answered by Chapter 4. This work proposes RIO, a Reserved Instances Optimizer. The goal of the implemented system is to output a purchase plan which maximizes the cost-saving. Currently, the system is undergoing trial use at Nordcloud, a cloud infrastructure consulting and managed service company. Nordcloud performed a pilot purchase following the purchase plan suggested by the implemented system with good results.

Finally, Chapter 5 proposes an extensive evaluation of the model and compares the theoretical results with the implemented solution. The experiments evaluate the model in an ideal case and using real demand data from an industry case. In both cases, the model accurately predicts the optimal number of reserved instances. As a consequence of the theoretical findings, the implemented system has been further improved (Section 6.3). Therefore, we can positively answer to the third research question.

In conclusion, the main contributions of this work are the following. A survey of the body of knowledge of cost optimization in cloud computing, analyzing literature and industry tools. In addition, this thesis proposes a novel application of Inventory Theory to cost optimization in cloud computing. Furthermore, this work introduces the implementation of an heuristic-based reserved instances planner.

7.1 Future Directions

This section describes future research directions and possible improvements related to the survey of the body of knowledge, the inventory theory model, and the implemented system.

The survey includes literature of cost optimization in cloud computing. In addition, Section 2.3 proposes financial perspective of the problem of cost optimization. A reserved instances purchase could be modeled and analyzed as an investment. Therefore, financial methods such as real option valuation (ROV) might be further investigated.

The theoretical model proposed in Chapter 3 is a first application Inventory Theory in the context of cloud computing. Despite the model outcomes are accurate, additional improvements are possible. A potential improvement is to include discount rate to take into account the time value of money. More-

over, a research direction would be to model an entire purchase plan utilizing a multi-product inventory system [3]. Finally, a further improvement would be to formulate theoretically a risk analysis.

Risk analysis could be modeled mathematically by estimating a probability distribution of events that change the demand or the prices. Assuming probability distributions of harmful events, it would be possible to develop a more sophisticated model of the cost. However, this approach might be infeasible due to the difficulties in estimating probability distributions of negative events.

Future work about heuristic-based reserved instances planner involves risk analysis. The current method empirically works; however, more complex analysis could be performed. Potential improvements include forecasting of demand and infrastructure changes simulation. Both methods aim to take into account the impact of harmful events in cloud infrastructure planning.

Bibliography

[1] Aggarwal, S. C. A review of current inventory theory and its applications. *International Journal of Production Research* (1974).

[2] Alpaydin, E. *Introduction to Machine Learning*. The MIT Press, 2010.

[3] Altiok, T., and Shiue, G. A. Single-stage, multi-product production/inventory systems. In *American Control Conference, 1991* (1991), IEEE, pp. 443–448.

[4] Amram, M., and Howe, K. M. Real options valuations: Taking out the rocket science. *Strategic Finance 83*, 8 (2003).

[5] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. A view of cloud computing. *Communications of the ACM 53*, 4 (2010), 50–58.

[6] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. Above the clouds: A berkeley view of cloud computing. Tech. rep., Electrical Engineering and Computer Sciences University of California at Berkeley, 2009.

[7] (AWS), A. W. S. Amazon web services marketplace. `https://aws.amazon.com/marketplace`.

[8] (AWS), A. W. S. Amazon web services website. `http://aws.amazon.com/`.

[9] (AWS), A. W. S. Documentation of aws dedicated instances. `http://aws.amazon.com/ec2/purchasing-options/dedicated-instances/`.

[10] (AWS), A. W. S. Documentation of aws ec2. `http://aws.amazon.com/ec2/`.

[11] (AWS), A. W. S. Documentation of aws lambda. `http://aws.amazon.com/lambda/`.

[12] (AWS), A. W. S. Documentation of aws region. `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html`.

[13] (AWS), A. W. S. Documentation of aws reserved instances. `http://aws.amazon.com/ec2/purchasing-options/reserved-instances/`.

[14] (AWS), A. W. S. Documentation of aws spot instances. `http://aws.amazon.com/ec2/purchasing-options/spot-instances/`.

[15] (AWS), A. W. S. List of aws ec2 instance types. `http://aws.amazon.com/ec2/instance-types/`.

[16] Bäck, T., and Schwefel, H.-P. An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation 1* (1993).

[17] Beasley, D., Martin, R., and Bull, D. An overview of genetic algorithms: Part 1. fundamentals. *University computing 15* (1993).

[18] Borison, A. Real options analysis: where are the emperor's clothes? *Journal of applied corporate finance 17*, 2 (2005), 17–31.

[19] Bossche, R. V. D., Vanmechelen, K., and Broeckhove, J. Optimizing iaas reserved contract procurement using load prediction. In *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on* (2014), IEEE, pp. 88–95.

[20] Bradley, S. P., Hax, A. C., and Magnanti, T. L. *Applied mathematical programming*. Addison-Wesley Reading, MA, 1977.

[21] Buyya, R., Beloglazov, A., and Abawajy, J. Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. *CoRR* (2010).

[22] Chaisiri, S., Kaewpuang, R., Lee, B.-S., and Niyato, D. Cost minimization for provisioning virtual servers in amazon elastic compute cloud. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on* (2011), IEEE, pp. 85–95.

[23] Chaisiri, S., Lee, B.-S., and Niyato, D. Robust cloud resource provisioning for cloud computing environments. In *Service-Oriented Computing and Applications (SOCA), 2010 IEEE International Conference on* (2010), IEEE, pp. 1–8.

[24] Chaisiri, S., Lee, B.-S., and Niyato, D. Optimization of resource provisioning cost in cloud computing. *Services Computing, IEEE Transactions on 5*, 2 (2012), 164–177.

[25] Chaisiri, S., Lee, B.-S., and Niyato, D. Profit maximization model for cloud provider based on windows azure platform. In *Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2012 9th International Conference on* (2012), IEEE, pp. 1–4.

[26] Chen, J., Soundararajan, G., and Amza, C. Autonomic provisioning of backend databases in dynamic content web servers. In *Autonomic Computing, 2006. ICAC'06. IEEE International Conference on* (2006), IEEE, pp. 231–242.

[27] Cloudability. Cloudability website. `https://cloudability.com`.

[28] Cloudability. The new aws ri model. `http://blog.cloudability.com/understanding-the-new-aws-reserved-instance-model/`.

[29] Cloudability. The science behind choosing ri. `http://www.slideshare.net/cloudability/ri-webinar-december2/`.

[30] Cloudhealth. Cloudhealth website. `http://www.cloudhealthtech.com`.

[31] Cloudyn. Amazon aws documentation - modifying reserved instances. `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ri-modifying.html`.

[32] Cloudyn. Cloudyn website. `https://www.cloudyn.com`.

[33] Dutta, P. R. T-61.3050 machine learning: Basic principles, parametric methods. Lecture at Aalto University.

[34] Farris, P. W., Bendle, N. T., Pfeifer, P. E., and Reibstein, D. J. *Marketing metrics: The definitive guide to measuring marketing performance*. Pearson Education, 2010.

[35] Gartner. Magic quadrant for cloud infrastructure as a service, worldwide. `http://www.gartner.com/technology/reprints.do?id=1-2G2O5FC&ct=150519`.

[36] Ghorbani, M., Wang, Y., Xue, Y., Pedram, M., and Bogdan, P. Prediction and control of bursty cloud workloads: a fractal framework. In *Proceedings of the 2014 International Conference on Hardware/Software Codesign and System Synthesis* (2014), ACM, p. 12.

[37] Haas, P. J. Quantile estimation. Management Science & Engineering lectures notes at Stanford University.

[38] Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., and McKeown, N. Elastictree: Saving energy in data center networks. In *NSDI* (2010).

[39] Hillier, F. S., and Lieberman, G. J. *Introduction to stochastic models in operations research*. McGraw-Hill Higher Education, 1995.

[40] Hong, Y.-J., Xue, J., and Thottethodi, M. Selective commitment and selective margin: Techniques to minimize cost in an iaas cloud. In *Performance Analysis of Systems and Software (ISPASS), 2012 IEEE International Symposium on* (2012), IEEE, pp. 99–109.

[41] Islam, S., Keung, J., Lee, K., and Liu, A. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems 28*, 1 (2012), 155–162.

[42] Kim, S., Koh, J.-i., Kim, Y., and Kim, C. A science cloud resource provisioning model using statistical analysis of job history. In *Dependable, Autonomic and Secure Computing (DASC), 2011 IEEE Ninth International Conference on* (2011), IEEE, pp. 792–793.

[43] Maringer, D. *Portfolio management with heuristic optimization*, vol. 8. Springer Science & Business Media, 2005.

[44] Mark, C. C. T., Niyato, D., and Chen-Khong, T. Evolutionary optimal virtual machine placement and demand forecaster for cloud computing. In *Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on* (2011), IEEE, pp. 348–355.

[45] Mathew, S., and Varia, J. Overview of amazon web services. Tech. rep., Amazon Whitepapers, 2014.

[46] Mell, P., and Grance, T. The nist definition of cloud computing. *Computer Security Division, Information Technology Laboratory, National Institute of Standards and Technology* (2011).

[47] Menascé, D. A., and Ngo, P. Understanding cloud computing: Experimentation and capacity planning. In *Computer Measurement Group Conference* (2009).

[48] Orbitera. Orbitera website. `http://www.orbitera.com`.

[49] Pisinger, D. *Algorithms for knapsack problems*. PhD thesis, University of Copenhagen, 1995.

[50] Reichelstein, S. Investment decisions and managerial performance evaluation. *Review of Accounting Studies* (1997), 157–180.

[51] Singer, G., Livenson, I., Dumas, M., Srirama, S. N., and Norbisrath, U. Towards a model for cloud computing cost estimation with reserved instances. In *Proc. of 2nd Int. ICST Conf. on Cloud Computing, CloudComp 2010* (2010).

[52] Stijven, S., Van den Bossche, R., Vladislavleva, E., Vanmechelen, K., Broeckhove, J., and Kotanchek, M. Optimizing a cloud contract portfolio using genetic programming-based load models. In *Genetic Programming Theory and Practice XI*. Springer, 2014, pp. 47–63.

[53] Tian, C., Wang, Y., Qi, F., and Yin, B. Decision model for provisioning virtual resources in amazon ec2. In *Proceedings of the 8th International Conference on Network and Service Management* (2012), International Federation for Information Processing, pp. 159–163.

[54] Wagner, S. The great hope of cloud economics and the overprovisioning epidemic. Tech. rep., Cloudyn whitepapers, 2013.

[55] Ye, S., and Tiong, R. L. Npv-at-risk method in infrastructure project investment evaluation. *Journal of construction engineering and management 126*, 3 (2000), 227–233.

[56] Yousefyan, S., Dastjerdi, A. V., and Salehnamadi, M. R. Cost effective cloud resource provisioning with imperialist competitive algorithm optimization. In *Information and Knowledge Technology (IKT), 2013 5th Conference on* (2013), IEEE, pp. 55–60.

[57] Zhang, Z., Li, Y., and Huang, G. An inventory-theory-based interval stochastic programming method and its application to beijing's electric-power system planning. *International Journal of Electrical Power & Energy Systems 62* (2014), 429–440.

[58] Zheng, P. S. Math 541: Statistical theory ii, maximum likelihood estimation. Lecture at Missouri State University.

Appendix A

Release dates of Amazon Web Services Instances

Table A.1 reports the release date of instances from AWS. The information has been retrieved using multiple sources: AWS blog, Twitter and the way back machine [1] [2] [3]. The release dates are updated till the release of m4 family (2015-06-11).

Instance	**Release Date**
t2.micro	2014-07-01
t2.small	2014-07-01
t2.medium	2014-07-01
m3.medium	2014-01-21
m3.large	2014-01-21
m3.xlarge	2013-01-21
m3.2xlarge	2013-01-21
m4.large	2015-06-11
m4.xlarge	2015-06-11
m4.2xlarge	2015-06-11
m4.4xlarge	2015-06-11

[1] `https://aws.amazon.com/blogs/aws/ec2-instance-history/`
[2] `http://archive.org/web/`
[3] `https://aws.amazon.com/blogs/aws/`

m4.10xlarge	2015-06-11
c4.large	2014-11-13
c4.xlarge	2014-11-13
c4.2xlarge	2014-11-13
c4.4xlarge	2014-11-13
c4.8xlarge	2014-11-13
c3.large	2013-11-14
c3.xlarge	2013-11-14
c3.2xlarge	2013-11-14
c3.4xlarge	2013-11-14
c3.8xlarge	2013-11-14
r3.large	2014-04-10
r3.xlarge	2014-04-10
r3.2xlarge	2014-04-10
r3.4xlarge	2014-04-10
r3.8xlarge	2014-04-10
g2.2xlarge	2013-11-04
g2.8xlarge	2015-04-16
i2.xlarge	2014-11-14
i2.2xlarge	2014-11-14
i2.4xlarge	2014-11-14
i2.8xlarge	2014-11-14
d2.xlarge	2015-03-31
d2.2xlarge	2015-03-31
d2.4xlarge	2015-03-31
d2.8xlarge	2015-03-31

Table A.1: Release dates of Amazon Web Services Instances.