The Pennsylvania State University
The Graduate School
College of Engineering

RESOURCE PROCUREMENT IN CLOUD SYSTEMS

A Thesis in
Computer Science and Engineering
by
Sepideh Kamrava

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2016

The thesis of Sepideh Kamrava was reviewed and approved* by the following:

Bhuvan Urgaonkar
Associate Professor of Computer Science and Engineering
Thesis Advisor, Chair of Committee

George Kesidis
Professor of Computer Science and Engineering

Chita R. Das
Distinguished Professor, Interim Head, Department of Computer Science and Engineering

*Signatures are on file in the Graduate School.

Abstract

With the growth in complexity of problems needed to be solved, the old fashion systems cannot meet the user's requirements. This niche has been identified by cloud providers offering enough resources with reasonable prices to handle any requests. The number of companies providing cloud services have been increasing in recent years. In order to compete with their rivals and attract more users, they provide more flexible options and plans. This dissertation focuses on one of the biggest cloud resource providers, i.e., Amazon's EC2 and investigate its structure and introduce different resource types and plans in details. The cloud users try to make an optimum decision on how they serve their demand with the available options. This decision making can be really complex given the degrees of freedom that exist in selecting among providers and their plans. Narrowing our attention on a smaller set of plans on Amazon's EC2 provider known as on-demand instances, reserved instances and the reserved instance marketplace, our goal is to derive a few guidelines on how different methods of assigning demands to the available plans can affect the users'total cost. To decrease the service cost, some optimization techniques including integer linear programming and dynamic programming are utilized to derive an optimal or sub-optimal decision for many general workloads. The results illustrate that how adding more plans to the system can further reduce the cost when the demand is noisier while for pure periodic workloads, the optimal cost can be achieved by only using on-demand and reserved instances. Moreover, a threshold based decision method is proposed to determine when a user should sell their available reserved instances to achieve the maximum benefit. A heuristic method is further used to incorporate the migration cost as an instance of hidden costs in the formulation. These techniques are applied to some real and synthetic workloads to confirm the expectations and derive useful guidelines.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Introduction

In this chapter, the main motivations for performing research on cloud systems are described. In the next subsection, the contributions of this thesis are summarized and at the last subsection is dedicated to the outline of this thesis.

1.2 Challenges and Motivations

With the drastic increase in the amount of data processed, the importance of a system capable of processing gigantic size data in a reasonable amount of time is evident [1]. That is where the state of art cloud computing systems become so functional. The cloud computing can be found quite everywhere from popular social networking to video streaming sites [2]. It has started a revolutionary path toward how computing resources are looked at in terms of how they are produced, priced and utilized. Such systems allow users to access resources and services that are resided in a remote data center as oppose to traditional systems with local resources only. They give the impression of unlimited resources to the users; therefore, users can adjust their resource utilization according to their requirements [3]. The cloud is explained by both the hardware and software systems that govern the existence of cloud services which are known as Infrastructures as a Service (IaaS) [4]. The specific applications performed by IaaS are known as Software as a Service (SaaS)

[4]. A large number of companies including Amazon, Microsoft, Google providing cloud services [5,6,7].

The cloud providers on one side offer variety type of services suitable to different type of users. On the other side, users try to find a provider that can best fulfill their demands and cost criteria. There are lots of issues raised in the interaction between these two sides. Due to the enormous options offered by providers, it can be a complicated decision for users to choose the best option for themselves. This issue has let to the emergence of cloud brokers who are acting like a middle man between providers and users. Moreover, the providers have their own interest to maximize their revenue, so an obvious solution would be to accommodate as many virtual machines as possible on a single real machine. In contrast, the users expect a guaranteed quality of service. Thus, the providers have some restriction on how many virtual machines they can put on a real machine without reducing the quality of their services below a threshold usually known as service level agreement (SLA). However, such an architecture that multiple users are accessing same physical resources without knowing each other can increase the chance that they experience lower performance due to the lack of information of the type of tasks each user performs. Therefore, it is possible that a machine ends up performing CPU bound tasks while the other machine is piled up with memory intensive tasks. The performance would be more satisfactory if the CPU and memory intensive jobs were distributed more evenly on these two physical machines. That is the reason some service providers deliberately offer instances with no performance guarantee at a lower price than a normal instance. One example can be the Spot instances that are offered by Amazon's EC2. To obtain these types of instances, the user should assign a bid which is usually much smaller than the normal instance price. As long as the bid succeeds the amazon price of the instance, the user have access to the instance. If Amazon experiences a high demand and needs more instances for normal users, they will increase the price so some of the bids may drop below the amazon price. At this point the instance is automatically preempted from the user.

Among the many challenging issues in the cloud, a number of important ones are mentioned here. It has been observed that cloud systems suffer from scalability.[8] Even though the cloud system is defined by an unlimited resources, in reality there is a limitation. Outages can happen on peak hours. Inconsistent pricing

meaning that if a virtual machine with a size of two times bigger than another one is requested, the price may not change linearly in terms of the sizes. Network bottleneck may happen when a large chunk of data is being transferred over the internet. Therefore, the providers usually have a limitation on the size of a data chunk for transferring. Cloud systems are highly heterogeneous because of being formed of multiple generation of equipment as technology advances during the time. Thus, the cloud providers feel the urge for incorporating the new technologies to compete with their rivals in the market. All of these factors contribute to the unpredictability of cloud systems performance on the provider's side.

On the user's side, cloud providers offer different type of instances varying in CPU and memory sizes that are optimized for a specific type of jobs. Obviously, users are charged differently based on the instance type they choose. In addition to multiple instance types, each individual instance type is available through different plans (options). One can obtain a same instance type as an on-demand instance denoting a plan where the price is normally calculated based on the number of hours it has been used, or as a reserved instance which is a kind of discount plan for sustainable users who needs to access to the instance for a longer period of time. A one year contract is an example and the award for their loyalty is a lower price per hour compared to an on-demand plan. In addition, there are many hidden cost for the users such as data transfer cost between regions, migration cost and etc. We will provide more details on this later in Chapter 2.

Based on these properties of cloud, resource provisioning from the user side is a topic demanding more investigation on both cost and performance perspectives. By cost, it means that choosing a combination of plans and instance types that minimizes the cost while satisfies the user's demand, and by performance denotes that not always the most cost efficient policy have the best performance or even it may not have a reasonable performance. In other words, we are looking for a policy to procure resources with the lowest cost without sacrificing the performance too much.

1.3 Objections and Contributions

The main challenge in the resource procurement problem is the workload predictability. How well the user can forecast their future demand can highly affect

the optimization policy behavior. The following items are investigated in this dissertation.

- Addressing the predictability of workload and its impact on the decision that leads to the minimum cost.
- Proposing a threshold policy for a specific but common scenario to decide the proper time to sell reserved instances over their lifespan.
- Providing an optimizing algorithm based on Integer Linear Programming (ILP) to find the minimum cost given the providers options.
- Incorporating reserved instance marketplace in the ILP formulation which is expected to reduce cost for less predictable workloads.
- Proposing a heuristic method in the ILP to take the migration costs into account.
- Demonstrating that for periodic demands the optimal policy can be obtained without solving the ILP directly.

1.4 Outline of the Thesis

The rest of this thesis is organized as follow

In Chapter 2, information on the cloud architecture model are provided in details. Then, the Amazon's EC2 structure, options and terminologies are studied. At the end, previous works that are related to this work are identified.

In Chapter 3, the system model is developed and the assumptions that are made are explained. The theories related to problems in resource provisioning going to be used in later chapters are presented as well as a basic introduction on dynamic and linear programming.

In Chapter 4, the dynamic programming formulation for the problem is proposed then for a specific common case a threshold based algorithm is proposed to help users to determine the best time to sell their instances to

avoid extra charges. Then, a generic algorithm based on linear programming is proposed which can be used with a wide range of cloud providers' configuration and is flexible enough to later be updated with feature provider's plan

In Chapter 5, experimental results are illustrated and compared to different based lines to demonstrate the effectiveness of the proposed algorithm.

Finally, **Chapter 6** concludes this dissertation and provides the future works.

Chapter 2
Background and Related Work

2.1 Introduction

In this chapter, the cloud systems are studied more deeply focusing on the aspects that are related to the target problem, i.e. cloud resource provisioning. Furthermore, related works regarding resource procurement and workload prediction are addressed.

Cloud computing systems are designed to make computing and storage resources such as CPU and memory available to everyone without requiring to possess a real system composing of manifold resources. Thus, virtualization[9] is the main basic block for this quite new technology. In this way, the cost for acquiring and maintaining necessary infrastructures can be extremely reduced, and users can concentrate more on the main topic of their business without concerning IT barriers. Cloud as it is known today has been gone through a revolutionary path; thus, its architecture bears some resemblance to other system models and yet has a key distinction to each of them. As an illustration, we can think of it as a grid computing system which has been evolved by adding concepts such as Quality of Service (QoS) and reliability. [4]

2.2 Architecture Model

A commonly used cloud architecture is defined by National Institute of Science and Technology, NIST, consisting of three standard models as follow [10]

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

2. **Platform as a Service (PaaS)** PaaS is defined as the capability provided to the user to deploy the user-created cloud infrastructure or acquire applications created using programming languages, libraries, services and tools supported by the provider. The user does not manage or control the underlying cloud infrastructure including network, servers, operating systems or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

3. **Infrastructure as a Service (IaaS)** IaaS denotes the capability provided to the user to provision processing, storage, networks, and other fundamental computing resources where the user is able to deploy and run arbitrary software including operating systems and applications. The user does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

In this work, the cloud that we refer to is IaaS unless otherwise specified. The other two tires although equally important, they are beyond the scope of this dissertation.

2.3 Deployment Models

Clouds are deployed in one of these three methods.[11]

1. *Private cloud*: Private cloud is merely used by a single organization. Ironically, these types of cloud's users still have to purchase and maintain a physical system

2. *Public cloud*: It is employed when the services are delivered over a public network. In terms of the architecture, private and public clouds are identical. Amazon's EC2, Microsoft Azure, Racket space and many more are examples from public clouds which are accessible over the internet.

3. *Hybrid cloud*: Finally, hybrid cloud is a combination of two or more clouds.

The focus of this dissertation is on public clouds. Although the method in this dissertation can be extended to any public platform, the solutions are embodied in the Amazon's EC2 platform. Thus, it is required to be familiar with basic concepts and definitions that are widely used by Amazon's EC2. The next section is dedicated to accomplish this goal. It is worth mentioning that quite the same concepts exist in other providers, so the idea can be utilized on other platforms after a one to one conversion between corresponding terminologies used by both platforms.

2.4 Amazon's EC2 overview

First of all, it should be noted that the Amazon's EC2 offer many services. Nevertheless, only the parts that are relevant to this work are indicated.
Amazon Web Services (AWS) is one of the providers that own the hardware required for providing cloud computing services to the users and they constantly extend their facility to better serve their customers. AWS are located in multiple places. The computing section is Amazon's EC2 which stands for Elastic Compute Cloud. It provides resizable computing resources. The three main types of instances in Amazon's EC2 are explained below.

On-Demand Instances are charged hourly without long-term commitment. So, it removes the large cost for planning, buying and maintaining hardware and converts it to smaller variable costs over a time span.

Reserved Instances are designed to be used for a longer period of time. Users can reserve them for 1 or 3 years and receive a noticeable discount on hourly rate, usually up to 75%. Furthermore, these instances can be sold on Reserved Instance Marketplace if the user does not need them anymore for some reasons such that their task ends earlier than the term expires.

Spot Instances allow users to bid on the unused capacity of the Amazon's EC2 and run the instance as long as their bid is greater than the spot instance price. The Spot instance price is not fixed and can change at any time depending on the supply and demand. The spot instances prices are much lower than the correspondence on-demand instances in exchange of the unpredictability of granting an access to them and their preemptive characteristic. Users who have some jobs that are not time sensitive and are stateless resulting in recovering each time the task is preempted can save a lot by utilizing these instances.

Reserved Instance Marketplace is not quite similar to other options. This options allows user to resell their own reserved instance to a third party or buy from a third party if the user wants so. It makes it easier for users to take a risk and purchase reserved instances since they can sell them whenever their prediction on future demand does not work well. On the other hand, users can expect to find a better deal in this market. Imagine a user, who well predicts their demand, say needs an instance for the whole two months, but not reserved or on-demand option works for them. The reserve instance market makes it possible for such users to buy reserved instances consequently and pay based on reserved instance rate while they really do not have the 1-3 years commitment. They can look for an instance with only two months left on its term. The pricing in this market can be an interesting research topic; however, there are a few rules. (1)The instance must be at least one month old and has one month left on the term to be placed in the market. (2) The life of the instance will be rounded down to the nearest month.

There are many other type of resources that AWS offers such as Storage and Databases. Upon requesting any of them, the user is charged based on some policy regardless of the different functionality of each resource.. Similar to what mentioned for computing resources, there can be multiple plans to charge users.
Another aspect of resources are their types. The type of an Amazon's EC2 instance is usually defined by the number of CPUs, memory size, storage type and size, and whether it is optimized for a specific type of workloads or not. For a complete list of available types, please refer to [12]. For example, T2 sub-category refers to burstable performance instances and M4 refers to the latest generation of general

Model	vCPU	Mem(GiB)	CPU Credit-s/hour	SSD Storage (GB)	Bandwidth (Mbps)
t2.nano	1	0.5	3	EBS only	-
t2.medium	2	4	24	EBS only	-
m4.large	2	8	-	EBS only	450
m4.4xlarge	16	64	-	EBS only	2000

Table 2.1. Sample of instances' type and configuration offered by Amazon's EC2.

purpose instances. This sub-category provides a good balance of compute, memory and network resources. Moreover, C4 and X1 sub-categories are referred to compute optimized memory and optimized instances, respectively. Table 2-1 shows a few types offered by Amazon's EC2.

In addition to choose the plan, deciding on the instance type makes the problem even more complicated. There can be many scenarios that needs subtle investigation to decide what combination is the best. For example, consider a use who needs 16 CPUs and 64 gigabytes of memory, should they go for one m4.4xlarge instance or 8 instances of type m4.large? Assuming that Amazon's EC2 hourly rate for m4.large instance is $0.12 and $0.958 for m4.4xlarge. Moreover, assume that the execution time does not change, if they go for m4.large, they have to pay $0.96; on the other hand, if they go for m4.4xlarge instance, the hourly cost would be $0.958. In terms of cost, it appears that it is not important for Amazon's EC2 what users decide otherwise they would incentivize users to pick one option. However, the performance is still in question. Answering this question can be the topic of another study, but in this work it is assumed that there is only one type of instances and focus on determining the optimum plan results in a minimum cost.
According to the studies that are going on in the cloud, the problems can be mainly divided into three groups (i) provider-side problems such as meeting the quality of service, coming up with a pricing strategy and maximizing the revenue (ii) user-side problems including resource allocation and finding optimal bid for Spot Instances. (iii) broker-side problems dealing with middleman that tries to generate a more predictable workload by aggregating workloads from multiple users in order to be qualified for more discount from the providers since. Table 2-2 shows the discount

Total Reserved Instances	Upfront Discount	Hourly Discount
Less than $500,000	0%	0%
$500,000 to $4,000,000	5%	5%
$4,000,000 to $10,000,000	10%	10%
More than $10,000,000	?	?

Table 2.2. Reserved instance volume discount from Amazon's EC2

amount the user will receive based on the total reserved instances cost. As can be seen, if at least 6,667 t2.micro instances are purchased, the user is qualified for 5% discount.Therefore, users with less sustainable demands can benefit from a lower rate than what providers offer.
The focus of this work is to address some user-side problems.

2.5 Related Works

Many researches have been conducted on resource allocation in the cloud systems. With the vast options that cloud providers offer, there is always a question for users which options is the best for them. Most of the researches have oversimplified the provider's interface in a way that only includes one or two resource types. In this theme, Jain et al. [13] attempts to find the optimum plan in the realm of batch jobs with two possible options, i.e. on-demand and spot instances. The former would be more expensive while the later increases the chance of interruption due to the bidding mechanism. To address the tradeoff between cost and performance, they propose an online adaptive algorithm which is learning from the previous performance and the history of spot prices. Their algorithm shows that the average regret approaches to zero with the time. However, this work does not consider reserved instances and reserved instance marketplace. Also, in many cases, the workloads show a high level of predictability especially in batch jobs. This predictability can be incorporated in non-online algorithms.
There are other works adding more constraints on job specifications for example assuming they are time sensitive and assigning a value to each job to indicate how important finishing the job before due date is.[14] However, they skip the variety options of computing resources.
On the other hand some works focus on the bidding strategy for spot instances

without considering other resource options. [15-16]. In [17] Wang et al. propose stochastic and deterministic online algorithms to decide between on-demand and reserved instances. They show that their algorithms achieve the best competitive ratio in both cases compared to the offline deterministic case where the future demand is known; however, they did not still consider the reserved instance marketplace.

None of the above works consider the hidden cost of relocating from one virtual machine to another known as migration cost. Migration cost can happen when the reserved instance term is expired but the task is not finished, so it must be moved to another VM, or for some reasons the user decides to sell a busy instance. Although there are works address reducing migration cost, they do not study this issue in the context of resource provisioning for minimum cost. For example, in [18] they proposed a method to reduce the migration cost for spot instances.

Chapter 3
System Model

3.1 Introduction

In this Chapter, the big picture of the system is described, and the assumptions that are made will be stated. Finally, a brief overview of theory required for developing the algorithms are presented.

3.2 Motivation and Assumptions

Figure 3-1 illustrates the resource provision framework. The public cloud providers such as Amazon EC2 and Microsoft Azure have different interfaces. Among the providers, the Amazon EC2 is picked for this research which has variety of instance types as we mentioned earlier. Here, it is assumed one instance type for simplicity, but the instance type can be accessed via different options, i.e., on-demand and reserved and can be also sold on reserved instance market. Spot instances are not the focus of this work since they add another dimension of uncertainty resulting in the work diverges from its main point. We have mentioned earlier the benefits of using reserved instance market. However, it is also worth mentioning when it does not work. Figure 3-2 compares type of workload where one is in favor of reserved instance market while the other is not. It should be noted that there is not such a good or bad option, and it all depends on the application. Some types of workloads may fit better with a specific option while may not fit good enough with another. Nevertheless, having more options is beneficial since by combining them we can

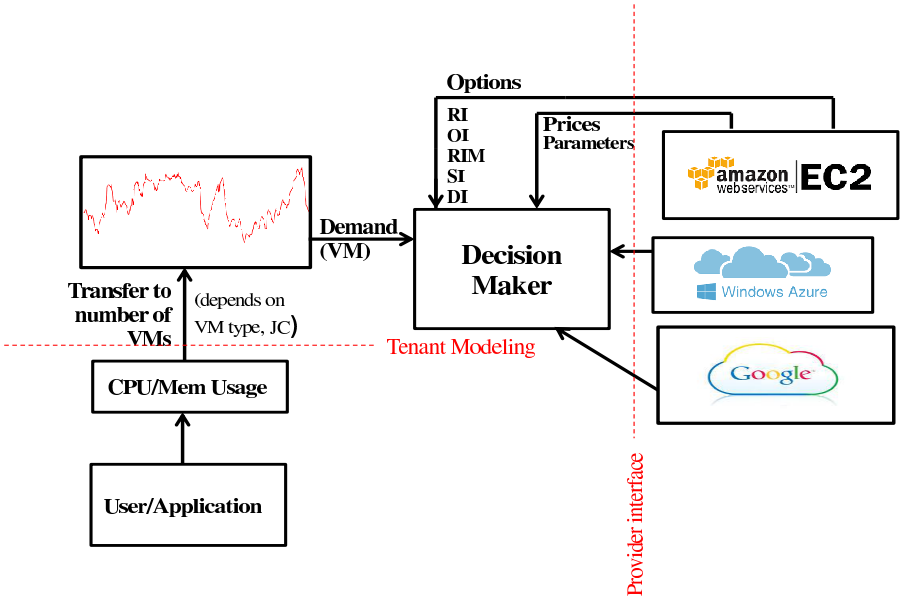

Figure 3.1. Resources provisioning framework view.

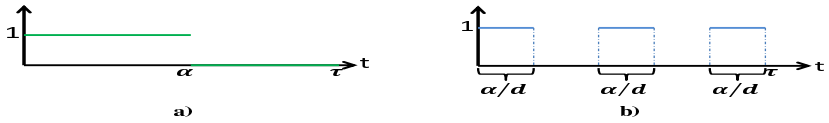

Figure 3.2. Compares simple scenarios where *RIM* can be useful vs. can't be useful .

expect to achieve a better policy. These observations lead to explore whether the combination of available options generates the most cost efficient policy.

On the other hand, the user interface is located. Some simplifications have done on this side. Modeling the user's demand has been a wide research area to decide how to translate the real resources usage such as CPU and memory to how many and what type of virtual machines are required. Note that, it is assumed earlier that one instance type is available. This assumption helps user to utilize a simpler model in order to map the CPU and memory usage to the number of virtual machines they need. Otherwise, the user must consider different types in their mapping. It is also assumed in this mapping that the job continuity is discarded meaning that after

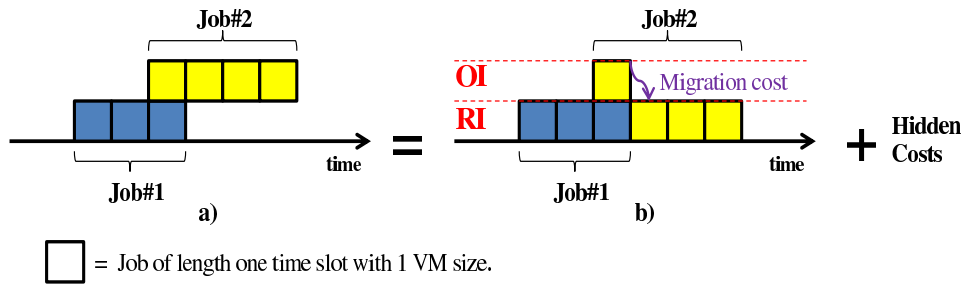

Figure 3.3. What is lost if the model does not capture the job continuity(JC).

the mapping there is no information about any relation between tasks at different time slots. In future works it can be investigated if they are sequences of a bigger task, or they are independent. Figure 3-3 depicts the effect of not considering the job continuity. In addition, it is assumed that the user's tasks show at least some level of predictability. Consequently, the raw demand which is based on user's tasks can also be predictable. Beside the raw demand generator and provider interface, there is another component in the system which is the decision maker where the proposed algorithm will reside.

3.2.1 Pricing Modeling and Assumptions

- **On-demand Instance** Without loss of generality, t2.micro instance on Linux platform in US East is selected as a reference with the rate $0.013 per hour as of time being used.

- **Reserved Instance** Amazon EC2 has a table like Table 3-1 for each instance type stating how user is charged for the reserved option. This table includes 1-year term options. The pattern in these tables can be summarized as follow,

  Regardless of the instance type, paying all upfront will save you 34% compared to using the same instance type purchased using on-demand option for 1 year, partial upfront saves 32%. and no upfront saves 31%.

  For three years plan, there is not no upfront option, so users must either pay all upfront or partially upfront and save 56% and 53%, respectively, compared to 3 years of on-demand purchase.

1-YEAR TERM					
Payment Option	**Upfront**	**Monthly**	**Effective Hourly**	**Saving over On-Demand**	**On-Demand Hourly**
No Upfront	$0	$6.57	$0.009	31%	004
Partial Upfront	$0	$6.57	$0.009	31%	004
All Upfront	$0	$6.57	$0.009	31%	004

Table 3.1. Amazon's EC2 options available for 1-year term plan.

For simplicity, in this work, the price for reserved instances is modeled according to all upfront pattern. So basically, once a reserved instance is bought, user pays the upfront amount and it is all done.

- **Reserved Instance Marketplace** is modeled by assuming reserved instances can have one of the 12 states corresponding to each month. Those that are assigned to state 1 or 12 cannot be sold for compatibility with the Amazon's rule. Although there can be some interesting pricing models, it is not the focus of this work. A simple rational pricing model that a seller would have is asking for the proportion of the term left plus the tax. Thus, for a t2.micro instance with all upfront 1-year term, the price pattern is $75 \times \frac{m}{12}$+tax, where m represents the number of months left on the term.

3.3 Theory Background

3.3.1 Dynamic Programming

Dynamic programming is a technique in optimization that to solve a complex optimization problem, breaks it down to a bunch of smaller sub-problems and uses the result of the sub-problems to achieve the solution for the main problem. This process is done by defining value function $J_t(s)$ of being at state s at time t. By moving backward the value functions for earlier points in time, $t-1$, $t-2$ and etc. can be found. All that must be done is to write a recursive function known as Bellman equation [19] that relates the current value function to the previous one. For more detailed information please refer to [20].

3.3.2 Integer Linear Programming

Let us first focus on what the linear programming is, and then we explain Integer Linear Programming.

The linear programming, also known as LP, is an optimization technique that tries to find the optimum solution for a mathematical model which can be expressed with linear terms. There are two types of terms in the LP model where both types must be linear to fit in this definition. The model has an objective function which needs to be either minimized or maximized. In this case the objective function must be a linear function. The second type is the constraints which define the relationship among variables. Put these all together, any problem that can be expressed as below is a LP.

$$\min \mathbf{c^T x}$$

$$s.t:$$

$$\mathbb{A}\mathbf{x} \leq \mathbf{b}$$

$$\mathbf{x} \geq 0$$

where $\mathbf{x}$ is the vector of variables and $\mathbf{b}$ and $\mathbf{c}$ are vectors of constant coefficients. Finally, matrix $\mathbb{A}$ is a matrix of coefficients.

If the variables are further restricted to hold integer values, then them problem becomes integer linear programming , ILP. To learn more about the linear programming please refer to [21].

There are many solvers that can be used to solve this type of problems. For example, JOptimizer, OpenOpt and Pyomo are all free and open-source.

Chapter 4
Algorithms

4.1 Introduction

In this chapter, a few techniques are used to model the problem. First, the notations used in the rest of this dissertation are defined. In the second section, the model is built based on dynamic programming and an especial case of the problem is further studied. In the third section, a more general model is proposed based on linear programming, and it is shown that for a specific type of workload, the problem is reduced to a simple binary search problem.

4.2 Notations

Before delving into the proposed algorithm, the notations that are used in this work are described as follow.
The demand at time slot t is represented by d_t, and the number of the hours in the optimization is shown by T which is also known as optimization horizon. D refers the number of the discrete levels for reserved instances. For example, as mentioned earlier, for one year term, the Amazon's EC2 has 12 levels corresponding to months. The number of the hours in each discrete length is denoted by *gamma*. p_o represents the on-demand instance hourly rate while element p_t in vector $\mathbf{p}$ refers to the reserved instance rate in the reserved instance marketplace. The first element in $\mathbf{p}$ represents the price for instance with one time of discrete length term, and the second price is for instance with two times of discrete length term and so forth. p_m denotes the migration cost. Vector $\mathbf{a}_t$ refers to the actions at time t.

Depending on the problem formulation, the vector's size can have various lengths. If only on-demand and reserved options are available, then the action vector has size of one representing the number of the purchased reserved instances. On the other hand, if the reserved instance marketplace is also included, the action vector size is equal to size D. Consequently, column t in matrix $\mathbf{A}$ represents the action vector for time t. Moreover, vector $\mathbf{s}$ refers to the state of the system with size of τ. Each element in vector $\mathbf{s}$ captures the number of reserved instances as well as the number of hours left in their term. Finally, e_t represents the numbers of unexpired reserved instance at time t regardless of their left over term.

4.3 Methods

4.3.1 Dynamic Programming Method

Using dynamic programming technique, the cost-to-go function is defined as

$$c_t(\mathbf{a_t}, \mathbf{s_t}) = \mathbf{a_t^\intercal p} + (d_t - \sum_{i=1}^{\tau} s_{it})^+ p_o \tag{4.1}$$

where the cost is a function of the current state $\mathbf{s_t}$ and the current action $\mathbf{a_t}$. The first term is the cost (income) imposed by the reserved instances that are either purchased directly from Amazon's EC2 or purchased (sold) from(in) the reserved instance marketplace. The second term represents the on-demand cost for handling demands exceeding purchased and existing reserved instances.
With each feasible action, the state space is updated. Actions is feasible only if the updated state is positive since it is not possible to sell more than available resources. Mathematically this can be written as

$$\mathbf{s_t}(i\gamma) = \mathbf{a_t}(i) + \mathbf{s_t}(i\gamma) \qquad \forall i \in 1, 2, ..., D \tag{4.2}$$

Note that only those elements of the state corresponding to storing instances with full number of discrete terms are updated . The action is a valid if after updating to the current state, the following two conditions hold.

$$\mathbf{s_t} \geq 0 \qquad \forall i \in 1, 2, ..., \tau \tag{4.3}$$

$$\mathbf{a_t}(D) \geq 0 \tag{4.4}$$

where the first condition guarantees the positivity of the state. The second condition denotes that the last element in action vector corresponding to number of full term instances required to buy must always be positive due to the assumption made earlier indicating that it is not possible to sell them in the instance reserved marketplace. Before moving to time $t+1$, the state is updated according to

$$\mathbf{s_{t+1}} = [\mathbf{s_t}(2:\tau), 0] \tag{4.5}$$

which simply shifts the state one element up and adds a zero at the end. The bellman equation is given by

$$J_t(\mathbf{s_t}) = \max_{\mathbf{a_t}} c_t(\mathbf{a_t}, \mathbf{s_t}) + J_{t-1}(\mathbf{s_{t-1}}) \tag{4.6}$$

where $J_t(\mathbf{s})$ is the minimum cost of being in state $\mathbf{s}$ at time t.
In this model, only instances with a term of multiple times of discrete length can be sold as oppose to the Amazon's EC model where the term of the instance is rounded down to the closest full number of months left on the instance. For example, if a user has a reserved instance with 2 months and 16 days left on it, they can put it for sale on Amazon's EC2 as an instance with 2 months term but in this formulation this instance cannot be sold until it has exactly 2 months left on its term.
Even though the proposed model takes into account many properties of the system, it has a big drawback. The time and space complexity for solving such problems can grow exponentially considering the size of the state and the number of the options that must be considered at each step. Therefore, despise the theoretical value, it has limitation in practical implementation. To eliminate this complexity, a very useful special case of the above formulation is investigated in the following subsection.

Special Case: Threshold Based Method

The first technique is inspired from [22]. Given that R is the number of brand new reserved instances with τ term, the goal is when and how many of reserved instances must be sold in the reserved instance market to reduce the cost assuming that the demand sequence has independent identical distribution (i.i.d.). The idea

behind this problem is that, the reserved instances are viewed as a depreciation asset. The predicament is whether to sell the instance given the offers from the buyers or keep the instance to achieve the optimum cost. Note that the user must satisfy the future demand in the optimization horizon. Thus, if they decide to sell the asset (the reserved instance), they have to buy on-demand instances in order to handle the future demand. A threshold based solution is proposed for this problem such that at each time, the user compares the offered prices, and if there is any one exceeding the threshold, they decide to sell a number of instances associated with that threshold. Otherwise, they keep them. More details on how this algorithm works are explained below.
Using dynamic programming, the cost-to-go function is defined as

$$J_t(r) = \max_{0 \le a \le R} \{ J_{t+1}(r-a) + \mathbb{E}(ap_t) - \mathbb{E}(d_t - (r-a))\, p_o \}$$

with these two base cases:

$$J_t(0) = \sum_{k=t}^{\tau} \mathbb{E}(d_t)\, p_o$$
$$J_\tau(R) = \mathbb{E}(d_\tau - R)\, p_o$$

Next, the expressions for finding the thresholds from selling no instances to selling any arbitrary amount of instances are derived.

- **No sell at all at time** t
  To not sell the reserved instances at time t given r instances, the following r inequalities must be hold simultaneously.

$$\begin{aligned}
&J_{t+1}(r) - \mathbb{E}(d_t - r)\, p_o \\
&> J_{t+1}(r-1) + \mathbb{E}(p_t) - \mathbb{E}(d_t - (r-1))\, p_o, \\
&> J_{t+1}(r-2) + 2\mathbb{E}(p_t) - \mathbb{E}(d_t - (r-2))\, p_o, \\
&> J_{t+1}(r-3) + 3\mathbb{E}(p_t) - \mathbb{E}(d_t - (r-3))\, p_o, \\
&\vdots
\end{aligned}$$

$$> J_{t+1}(r-r) + r\mathbb{E}(p_t) - \mathbb{E}(d_t - (r-r))\, p_o$$

After some mathematical manipulation, this is obtained

$$\begin{aligned}
&p_t < J_{t+1}(r) - J_{t+1}(r-1) + p_o = \alpha_{t,0,1} \\
&p_t < \frac{1}{2}\left(J_{t+1}(r) - J_{t+1}(r-2) + p_o\right) = \alpha_{t,0,2} \\
&\vdots \\
&p_t < \frac{1}{r}\left(J_{t+1}(r) - J_{t+1}(r-r) + r.p_o\right) = \alpha_{t,0,r}
\end{aligned}$$

So if

$$p_t < \min_{1\le a\le r}\{\alpha_{t,0,a}\}$$

the optimum decision is to not sell any instances at time t.

- **Selling one instance at time** t
  In this case, the following conditions must be hold simultaneously.

$$\begin{aligned}
&J_{t+1}(r-1) + \mathbb{E}(p_t) - \mathbb{E}(d_t - (r-1))\, p_o \\
&\quad > J_{t+1}(r) - E(d_t - r)\, p_o, \\
&> J_{t+1}(r-2) + 2\mathbb{E}(p_t) - \mathbb{E}(d_t - (r-2))\, p_o, \\
&> J_{t+1}(r-3) + 3\mathbb{E}(p_t) - \mathbb{E}(d_t - (r-3))\, p_o, \\
&\vdots \\
&> J_{t+1}(r-r) + r\mathbb{E}(p_t) - \mathbb{E}(d_t - (r-r))\, p_o
\end{aligned}$$

After simplifying the above conditions, it is obtained

$$\begin{aligned}
&p_t > -\left(J_{t+1}(r-1) - J_{t+1}(r) - p_o\right) = \alpha_{t,1,0} \\
&p_t < J_{t+1}(r-1) - J_{t+1}(r-2) + P = \alpha_{t,1,2} \\
&\vdots \\
&p_t < \frac{1}{r-1}\left(J_{t+1}(r-1) - J_{t+1}(r-r) + (r-1)\, p_o\right) = \alpha_{t,1,r}
\end{aligned}$$

Therefore, in this case, the threshold is given by

$$p_t < \min_{1<a'\leq r}\{\alpha_{t,1,a'}\} \bigcap p_t > \max_{0\leq a'<1}\{\alpha_{t,1,a'}\}$$

where $\alpha_{t,a,a'}$ represents the threshold at time t for choosing to sell one instance from $a' \leq r$ available instances. This value can be evaluated using the following expression.

$$\alpha_{t,a,a'} = \frac{1}{a'-a}\left(J_{t+1}\left(r-a\right) - J_{t+1}\left(r-a'\right) + \left(a'-a\right)p_o\right)$$

- **Selling a instances at time t**
  In general, at time t, selling a instances will be chosen from a' possibilities, if

$$p_t < \min_{a<a'\leq r}\{\alpha_{t,a,a'}\} \bigcap p_t > \max_{0\leq a'<a}\{\alpha_{t,a,a'}\}$$

4.3.2 Generic Method

In this part, an algorithm based on integer linear programming (ILP) is proposed for the main problem taking migration cost into account. The ILP formulation is given by

$$\min_{\mathbf{A}} \sum_{i=1}^{D}\sum_{t=1}^{T} a_{it}p_i + \sum_{t=1}^{T}\left(d_t - e_t\right)^{+}p_o + \sum_{t=1}^{D}\sum_{t=1}^{T} a_{it}^{+}p_m$$

s.t:

$$\begin{aligned}
&a_{it} \in \mathbb{Z}, \forall i,t\\
&a_{it} > -s_{it}, \forall i,t\\
&s_{it} = \sum_{j=i}^{D}\sum_{\substack{t'=\\ t-\tau(i-j+1)+1}}^{t-\tau(i-j)} a_{jt'}\\
&s_{it} \geq 0, \forall i,t\\
&\sum_{i=1}^{D} a_{it} < d_{\max} - e_t, \forall t\\
&e_t = \sum_{i=1}^{D} s_{it}, \forall t
\end{aligned}$$

In the objective function, the first term takes into account the money exchanged for the reserved instance whether those are bought or sold in the reserved instance market. Note that the negative actions correspond to sell instances. The second term is the cost imposed by on-demand requests, and the last term represents the migration cost assuming that each purchased reserved instance has a hidden cost for migrating some tasks to the new virtual machine. The goal is to find the actions such that the total cost is minimized given the constraints mentioned above. The first constraint denotes that actions must be real number, and the second one says that it is not feasible to sell an instance type more than the available instances from that type. The third constraint defines the number of instances from each type available at a given time. The fact that it is not possible to have negative instances is considered in fourth condition. The fifth condition further defines the upper bound on the number of instances of a specific type that can be purchased at a given time minus the total number of available instances regardless of their type. This last quantity, total number of available instances, is defined by the last condition.
To convert the formula to ILP format, the max operation must be removed. Thus, two slack variables are defined as

$$
\begin{aligned}
& b_t \geq 0, \forall t \\
& b_t \geq d_t - e_t, \forall t \\
& c_{it} \geq a_{it}, \forall i, t \\
& c_{it} \geq 0, \forall i, t
\end{aligned}
$$

The objective function can be then written as

$$\min_{\mathbf{A}} \sum_{i=1}^{D} \sum_{t=1}^{T} a_{it} p_i + \sum_{t=1}^{T} b_t p_o + \sum_{t=1}^{D} \sum_{t=1}^{T} c_{it} p_m$$

Special Case: Binary Search Method

As a special case, if the demand is periodic with a period much smaller than the reserved instance term and the on-demand and reserved instances are available, the optimal solution is reduced to finding one number representing the number of

reserved instances. Since the pattern repeats after some time, whatever decision made at the time is valid for later as well. The cost of increasing the number of reserved instances is a polynomial of degree two. In other words, the cost function is convex, and as the number of the reserved instances increases, the cost decreases up to some point and then increases. . The minimum point determines the number of reserved instances that are required to achieve the optimal policy. Therefore, in this case, it is not necessary to solve the integer linear programming. Instead, a binary search method can be used to find the optimal policy.

Chapter 5
Empirical Evaluation

5.1 Experimental Setup

As we use the proposed algorithm to obtain the optimal decision, some kind of demand sequence is required. The demand sequences that are deployed in this work consist of both real and synthetic traces. The real demand traces are from Facebook and Google. To generate synthetic traces, the periodic components of real traces are extracted. The residual, i.e., the difference between the real trace and its periodic component, is used to construct the noise. This noise is then multiplied by a constant factor, c to illustrate different noise levels. Thus, the real traces are considered as weak-noise, the synthetic traces generated by adding two times of the residual to the periodic component are defined as medium-noise traces, and finally, tripling the residual plus the periodic component is called the high-noise traces. Obviously, except from the low-noise traces which are directly taken from real data, the other derivations are synthetic workloads. As discussed earlier, these traces do not provide any information about the job continuity and only give us information on demands in terms of the number of the virtual machines at each time slot. Figures 5.1 and 5.2 present some samples of the real and synthetic traces from Facebook and Google, respectively.

Inputs: As it was mentioned earlier, under some circumstances, users of Amazon EC2 can benefit from 35% to 75% discount. The discount ratios for reserved instances, r_d, are assigned 35%, 55% and 75%. We use three different values to consider the effect of r_d,i.e, $r - d = 0.35,0.55,0.75$. Moreover, without loss of

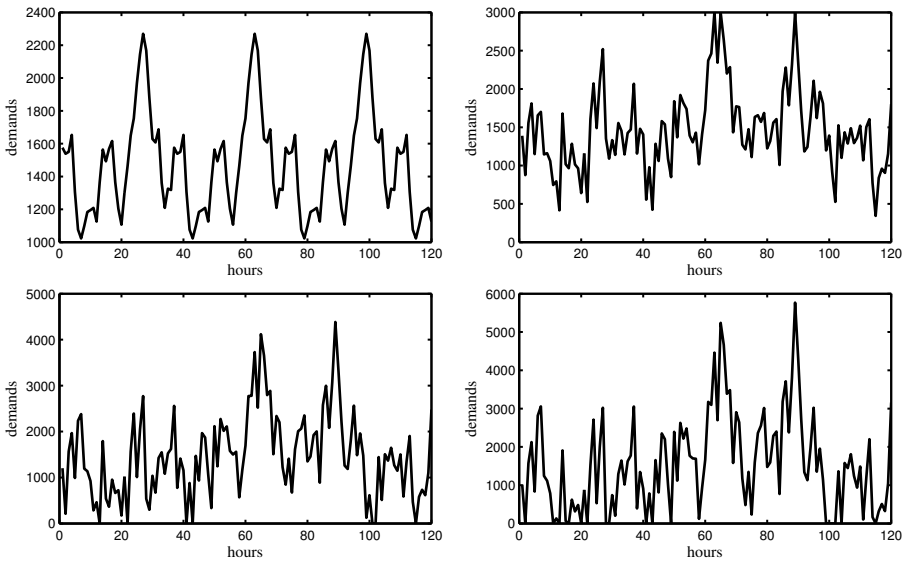

Figure 5.1. Four derivations from Google trace, i.e., periodic, weak noise (real), medium noise and strong noise.

generality, the on-demand price, p_o is assigned to one. The price for the partial reserved instances have two elements. One is the cloud provider commission plus the tax to allow you to sell your reserved instance which is 12% of the instance price. Another element is proportional to the instance term left.

Parameters: We explore many combinations of the parameters such as the noise multiplier and the discount ratio, r_d. α is a scalar between 0 and 1 representing the possibility of having switching cost when an instance is started. We pick two values for α equal to 0 and 1 corresponding respectively to very short jobs where there is no switching cost and very long jobs such that always there is a switching cost. For a given demand, we calculate the cost for three cases as below

- There are only on-demand instance which is equivalent to force the action matrix to zero, $\mathbf{A} = 0$.
- There are on-demand and full reserved instances corresponding to assign the action matrix indexes for partial buying to zero.
- We have on-demand and reserved instances as well as the reserved instance marketplace to sell partial used instances. In this case $\mathbf{A}$ can take any values to minimize the cost.

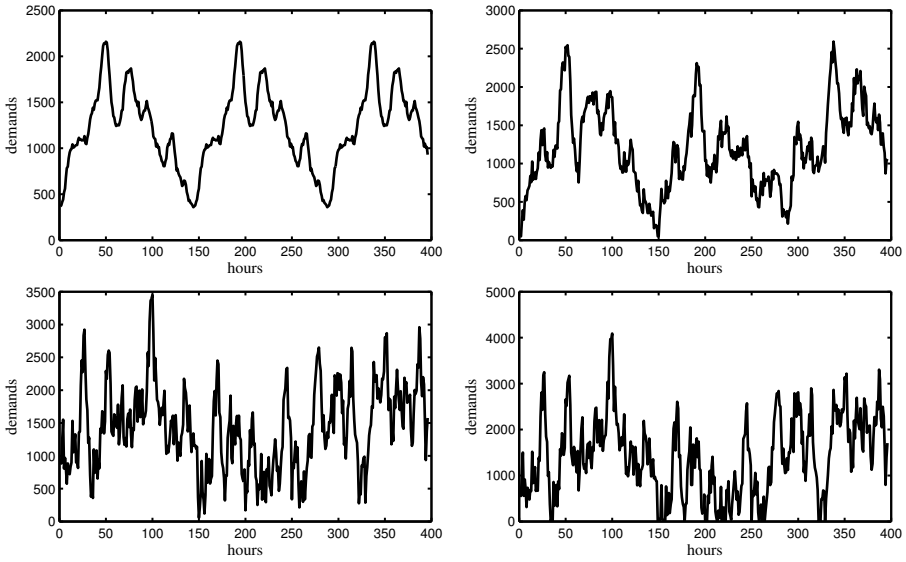

Figure 5.2. Four derivations from Facebook trace, i.e., the periodic, low noise (real), medium noise and high noise.

The optimization horizon is $T = 4$ months and a full reserve instance life cycle, τ is 2 months. We also have the option of buying an instance with one month length. Thus, $D = 2$ corresponds to a full instance or a half life time instance.

Baseline: Given the problem definition, the best strategy is to pay the reserved instance rate for each task. This will produce the lowest cost, but in reality, this may not be possible due to the conditions that must be met to have a reserved instance. However, it is clear that no algorithm can ever produce a result better than this. Therefore, the lower baseline is calculated assuming all the jobs are served with the reserved instance rate. On the other hand, the worst scenario that can happen if the demand pattern is such that only the on-demand instances can be used. In this case, as no algorithm can improve the efficiency, the upper bound is calculated assuming all the tasks are served by the on-demand rate.

5.2 The implications of the reserved instances

In this evaluation, the perfect knowledge of the demand sequence is assumed. Also, it is assumed that no switching cost and no reserved instance marketplace are incorporated. Thus, the last term in objective function is zero, and the elements in the

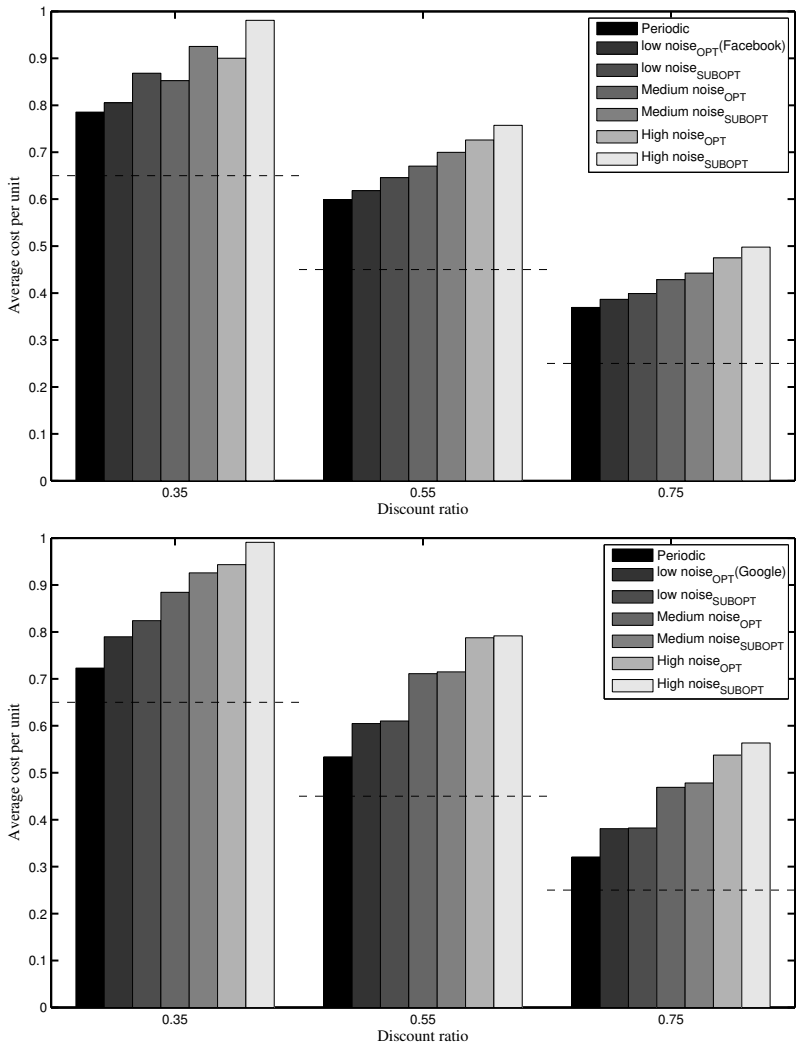

Figure 5.3. Optimal and periodic policy used in the on-demand and reserved instance setup. Top: Facebook, Bottom : Google

action matrix can only be positive while the on-demand and reserved instances are possible options. The experiment is repeated for each of three levels of the discount ratio per each demand sequence mentioned earlier. In other words, corresponding to each discount ratio, we have four results for each of the four demand cases from Facebook and Google traces. Note that there are eight results for each discount ratio rather than four in Figure 5.3. Those with SUBOPT index indicate that the optimal policy for the periodic result is used on other types of workloads. The reason for doing it is, that finding the optimal policy for periodic demands can be reduced to a simple threshold problem as mentioned earlier. Therefore, this extra

experiment demonstrates what happens if accuracy is sacrificed by simplicity, or how diverging from the periodically affects the cost while the policy has not been changed. The normalized costs have been shown in the results. Thus, the value of one represents the upper bound, and the dot lines represent the lower bound in each case.

Performance expectation

- It is expected to observe less cost efficiency when the periodic optimal policy used for non periodic demands.
- As the discount ratio increases, the total cost increases compared to the same experiment with a lower discount rate.
- With a fixed discount rate, it is presumed that the cost increases by increasing the noisiness level due to increase of unpredictability.

Key insights

1. It can be inferred from the results that as the level of noise increases while the discount ratio decreases, the reserved instances become less useful. So, the on-demand instances can work as good as the reserved instances. In other words, this can help users with a high noisy demand to not bother themselves with the reserved instances.
2. Although the cost reduction due to using the optimal algorithm is higher than using periodic optimal decision, given that the sub-optimal solution can be achieved without solving the integer linear programming, the periodic optimal decision can still be a satisfactory solution. In other words, it all depends on users how important the difference is for them.
3. In contrast to (1), as the discount ratio increases, users can save a lot if they deploy reserve instances.

5.3 The implications of the reserved instance market

In this section, the reserved instance market is added to the previous settings as well. Note that, there is no migration cost. The mathematical effect of the reserved

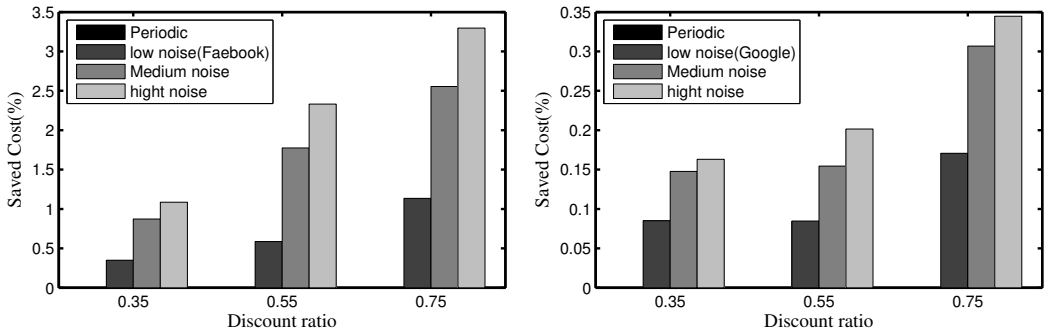

Figure 5.4. Percentage of saved cost when the reserved instance market is incorporated in addition to reserved and on-demand instances. Left : Facebook, Right : Google

instance market is that the actions can also take negative integers. Figure 5.4 the results for this setup.

Performance expectation

- Even more cost reduction is expected compared to the previous setup, i.e., no reserved instance marketplace. This can be justified by having an extra option. Therefore, the result should be at least as good as the previous setup.

Key insights

1. As the results show, the reserved instance marketplace does not have any impact when the demand is periodic, so the optimal solution for the periodic signal is the same as the previous case.

2. The reserved instance market is more appealing when demand is noisier. On the other hand, it is not useful at all when we have a poor periodic demand.

3. For the same demand, using a higher discount ratio reduces the cost more when we have the reserved instance market.

4. For some demands such as Google, there is no need to solve the ILP problem. In these cases, using the binary search method proposed for the periodic demand and the optimal policy for the real demand results in almost the same cost where the difference can be neglected. However, it is not true for Facebook trace. This can be attributed to the period of the demand sequence. Google trace has a smaller period than the Facebook trace. Since the τ is

fixed, this implies that the shorter period, the better optimal policy of the periodic demand works on noisy demands.

Chapter 6
Conclusion

6.1 Summary

In this dissertation, the resource procurement in cloud systems have been addressed in order to decide on how and when the various options of reserved instances should be used to minimize the users'cost.
The problem has been approached from different aspects. We have focused on the user's side of the problem as oppose to the provider's side. From Several different algorithms have been proposed to help users to decide on the available options. To deal with uncertainty in workload, a threshold based method has been used to avoid extra losses in case of a bad decision making in the first place.
Our solutions have been based on the abstract representation of demand given exact knowledge of demand for a long enough length. Our proposed methods can be used as building blocks for more general scenarios.

6.2 Future Work

The methods proposed for resource provisioning in this work do not incorporate the spot instances which is one of the powerful plans offered by Amazon's EC2. Adding this extra option can make our algorithm more efficient.
Moreover, if the information about the job continuity is available, it can affect the optimization by using a better prediction of migration. Another direction that can be investigated is the workload prediction and how that can be integrated with the proposed algorithms. Our proposed algorithms can handle a quite short

optimization horizon while the demand sequence can have infinite length. Since the prediction of demand in far future can be less accurate, the best approach would be to update the prediction every other days or so. Incorporating prediction along with providing an algorithm for infinite length job demands has not been well addressed and needs more investigation.

Bibliography

[1] Armbrust, Michael, et al. "A view of cloud computing." Communications of the ACM 53.4 (2010): 50-58.

[2] https://aws.amazon.com/solutions/case-studies/

[3] Vaquero, Luis M., et al. "A break in the clouds: towards a cloud definition." ACM SIGCOMM Computer Communication Review 39.1 (2008): 50-55.

[4] https://en.wikipedia.org/wiki/Cloud_computing

[5] https://aws.amazon.com/

[6] https://cloud.google.com/

[7] https://azure.microsoft.com/en-us/

[8] Furht, Borivoje, and Armando Escalante. Handbook of cloud computing. Vol. 3. New York: Springer, 2010.

[9] https://en.wikipedia.org/wiki/Virtualization

[10] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." Communications of the ACM 53.6 (2010): 50.

[11] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing." (2011).

[12] https://aws.amazon.com/ec2/instance-types/

[13] Menache, Ishai, Ohad Shamir, and Navendu Jain. "On-demand, spot, or both: Dynamic resource allocation for executing batch jobs in the cloud."11th International Conference on Autonomic Computing (ICAC 14). 2014.

[14] Jain, Navendu, et al. "Near-optimal scheduling mechanisms for deadline-sensitive jobs in large computing clusters." ACM Transactions on Parallel Computing 2.1 (2015): 3.

[15] Zafer, Murtaza, Yang Song, and Kang-Won Lee. "Optimal bids for spot vms in a cloud for deadline constrained jobs." Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on. IEEE, 2012.

[16] Song, Yang, Murtaza Zafer, and Kang-Won Lee. "Optimal bidding in spot instance market." INFOCOM, 2012 Proceedings IEEE. IEEE, 2012.

[17] Menache, Ishai, Ohad Shamir, and Navendu Jain. "On-demand, spot, or both: Dynamic resource allocation for executing batch jobs in the cloud." 11th International Conference on Autonomic Computing (ICAC 14). 2014.

[18] Yi, Sangho, Artur Andrzejak, and Derrick Kondo. "Monetary cost-aware checkpointing and migration on Amazon cloud spot instances." IEEE Transactions on Services Computing 5.4 (2012): 512-524.

[19] https://en.wikipedia.org/wiki/Bellman_equation

[20] https://www.topcoder.com/community/data-science/data-science-tutorials/dynamic-programming-from-novice-to-advanced/

[21] Garfinkel, Robert S., and George L. Nemhauser. Integer programming. Vol. 4. New York: Wiley, 1972.

[22] Bertsekas, Dimitri P., et al. Dynamic programming and optimal control. Vol. 1. No. 2. Belmont, MA: Athena Scientific, 1995.