Chapter 2
Optimal Resource Rental Management

Abstract Application services using cloud computing infrastructure are proliferating over the Internet. In this chapter, we study the problem of how to minimize resource rental cost associated with hosting such cloud-based application services, while meeting the projected service demand. This problem arises when applications incur significant storage and network transfer cost for data. Therefore, an Application Service Provider (ASP) needs to carefully evaluate various resource rental options before finalizing the application deployment. We choose Amazon® EC2 marketplace as a case of study, and analyze the optimal strategy that exploits the tradeoff of data caching versus computing on demand for resource rental planning in cloud. Given fixed resource pricing, we first develop a deterministic model, using a mixed integer linear program, to facilitate resource rental decision making. Next, we investigate planning solutions to a resource market featuring time-varying pricing. We conduct time-series analysis over the spot price trace and examine its predictability using Auto-Regressive Integrated Moving-Average (ARIMA). We also develop a stochastic planning model based on multistage recourse. By comparing these two approaches, we discover that spot price forecasting does not provide our planning model with a crystal ball due to the weak correlation of past and future price, and the stochastic planning model better hedges against resource pricing uncertainty than resource rental planning using forecast prices.

This chapter is organized as follows. Section 2.1 provides an overview of the problem and summarizes our proposed optimal planning methods. Section 2.2 surveys the related work. In Sect. 2.3, we formulate the system model, provide a deterministic planning model for the resource rental problem, and evaluate the performance of the deterministic pricing resource planning approach. Finally, in Sect. 2.4, we analyze the predictability of Amazon® EC2 spot pricing using time-series analysis techniques, propose a stochastic optimization model to solve the rental planning problem, and perform simulations to compare the two approaches.

H. Zhao and X. Li, *Resource Management in Utility and Cloud Computing*,
SpringerBriefs in Computer Science, DOI 10.1007/978-1-4614-8970-2_2,

2.1 Overview

The emerging cloud computing model, with its virtually infinite resources and elasticity, liberates organizations from the expensive infrastructure investment. As a result, more and more Application Service Providers (ASPs) recognize the separation between the actual application and the infrastructure necessary to run it, and begin to deploy applications on resources rented from infrastructure providers. According to a recent forecast by Gartner® [16], Software-as-a-Service and Cloud-based business application services will grow from $13.4 billion in 2011 to $32.2 billion in 2016.

In cloud computing, a major issue faced by the ASPs is how to minimize the resource rental cost while meeting their application service demand. Significant research efforts have been directed toward developing optimal resource provisioning schemes to meet service requirements (avoid the cost due to over-provisioning and the penalty due to under-provisioning) [7, 17, 23, 28, 30]. These works, although offer effective resource provisioning controls in response with varying workload, are still coarse-grained in terms of exploring application elasticity with regard to different resource pricing options. We believe that resource rental planning should be conducted in a cost-aware manner to reduce ASPs' operational cost. Specifically, we propose a fine-grained planning scheme to regulate the rental activities on a time-slotted basis, exploring hourly charging rate of various types of resources, in order to meet the projected service demand and minimize resource rental cost at the same time. Complementary to prior resource scaling solutions, our approach focuses on application scaling that optimizes resource rental plan in cloud without compromising the service-level agreement.

In addition to the planning optimization complexity, another obstacle lies in the uncertainty of computational resource pricing. This challenge is encountered in the spot resource market emerged in recent years. In a spot resource market, depending on the resource supply and demand level, the unit price of a computational instance is fluctuating all the time. For example, at the time Amazon® first launched its spot instance service in December 2009, an auction mechanism was employed to determine instance pricing. Since spot instances leverage idle cycles from the regular on-demand server pool, they are auctioned off at a price much lower than that of the regular on-demand instances most of the time. As a result, this real-time bidding market has attracted many ASPs who wish to increase server capacity at low cost. There is a growing research interest in utilizing spot instance service. However, modeling and analyzing spot instance pricing is largely neglected due to the lack of demand and resource provision information. We believe that our study is helpful to understand spot pricing, and more importantly, to improve resource utilization under spot pricing.

The research presented in this chapter represents our initial design for cost-effective resource utilization and management in utility and cloud computing. In particular, we develop optimal resource rental planning strategies for fixed pricing and stochastic pricing resource markets, respectively. The first part of this chapter presents our approach for a fixed pricing resource market. Given a forecast demand schedule, the ASP needs to periodically review the running progress of the deployed service and make optimal job allocation as well as resource rental decisions so as not to waste money on excessive computation, storage or data transfer. We formulate a deterministic planning model for resource rental decision making over a specific planning horizon. The solution to this model serves as a guide to make cost-effective resource rental decisions in real time. We show that our planning model is especially useful for high-cost Virtual Machine (VM) classes. This is because cost saving from our model primarily comes from eliminating unnecessary job running by decreasing VM rental frequency. From this perspective, our model formulation is aligned with the dynamic lot-sizing model commonly encountered in the field of production planning.

Next, we analyze and solve the fine-grained cloud resource rental planning problem under the pricing uncertainty challenge. In particular, two possible solutions are jointly explored. We systematically analyze the predictability of Amazon® EC2 spot pricing and use the predictive prices to perform planning. Furthermore, we propose a multistage resource model for stochastic resource rental planning. This model decomposes the stochastic process of decision making under varying price into sequential decision making processes with the aid of price distribution at various stages. As such, the stochastic optimization problem is transformed into a large-scale deterministic optimization problem. Through simulations, we demonstrate that the stochastic planning approach is more cost-effective than predictive planning.

2.2 Related Work

Nowadays, a wide variety of computational and data intensive applications utilize cloud to their benefit. Therefore, it becomes imperative to understand the cost-benefit of running resource-demanding applications in cloud in order to make cost-effective resource rental decisions. Cloud computing eliminates up-front setup and operational cost for distributed resources. However, moving and storing large data set in cloud incur significant cost comparable to the computing cost [13]. Efforts have been made to mitigate such cost in cloud [22, 29]. In this chapter, we present a planning model that optimizes resource usages for elastic applications with comprehensive cost considerations.

Finding an optimal resource utilization strategy is challenging for both cloud infrastructure providers and application service providers who rely on rented infrastructure. From the perspective of the cloud infrastructure provider, the challenge is how to reduce the operational cost and maximize leasing revenue. Many existing research has focused on this aspect. The general problem of minimizing resource allocation cost while meeting job demand is NP-hard [9]. Resource scheduling for the emerging spot market was proposed in [31]. The proposed framework includes: (1) a market analyzer periodically forecasting supply and demand, (2) a capacity planner determining the spot price based on the forecast results, and (3) a VM scheduler maximizing the revenue by solving a NLIP model for the scheduling problem. From the perspective of the application service providers, the challenge becomes how to minimize resource rental cost while meeting service demand from customers. Many resource planning schemes rely on predictive workload assessment [17, 21]. Our work takes one step further that presents an application scaling control model based on the forecast demand. Our model takes full consideration of various resource types and their associated costs within a cloud resource market, and strives to find the optimal tradeoff among various resource usage in resource rental allocation.

The stochastic planning model proposed in this chapter deals with the price uncertainty in the spot resource market. Such a spot market is either formed by multiple resource providers [10] or by a single resource provider. Amazon® EC2 spot market is the most representative example that attracts significant research attentions. Researchers are interested in utilizing spot instances to temporarily add capacity to dedicated clusters during peak periods [19]. The biggest concern for utilizing spot instances is that it is hard to guarantee resource availability. Recent works [3, 20] addressed this problem using statistical analysis. Notably, Ben-Yehuda et al. [1] reversely engineered spot prices by constructing a spare capacity pricing model based on existing price traces. However, the effectiveness of these approaches is still unclear due to unsubstantiated assumptions on Amazon® EC2 spot service. In this chapter, we take EC2 as a case study and targets at a general spot resource market where prices are market-driven and users bid according to their true valuations (simple-minded assumption). The most relevant works to this study are presented in [8, 25]. In [8], the authors presented an optimal VM placement algorithm that minimizes the cost of resource provisioning in a multiple cloud providers environment, and in [25], the authors proposed a profit-aware dynamic bidding algorithm to optimize ASP's profits in EC2 spot market. Our work's application scenario is different from [8], and we develop our model based on realistic application and price traces. Comparing with [25], our approach proposes a different model that takes storage and network transfer cost into account in addition to computational instance bidding.

2.3 Deterministic Resource Rental Planning

Resource rental planning entails the acquisition and allocation of computational and storage resources to applications so as to satisfy demand over a specified time horizon. An application scaling control scheme is proposed to optimize rental decision on a time slotted basis. In this section, we target at a fixed pricing cloud resource market. After describing the system model, we model the rental planning problem using a mixed integer linear program.

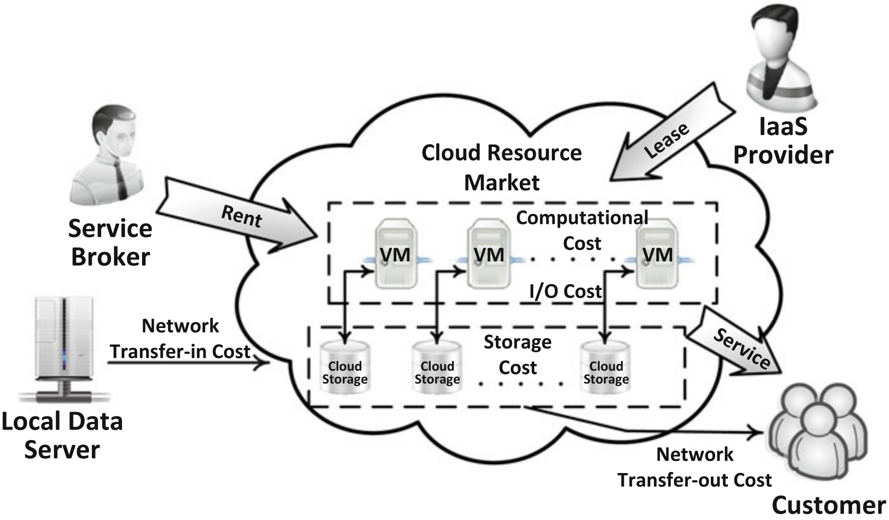

Fig. 2.1 System model for the resource rental planning problem

2.3.1 System Model

We present a scenario where an ASP offers some computational and data intensive application services (example services are data visualization, data analytics, data indexing, etc.) to customers over a network. Instead of using local resources, the tasks of computation and data storage are completely outsourced to a shared resource pool operated by some Infrastructure-as-a-Service (IaaS) provider(s), shown in Fig. 2.1. The depicted system model resembles a broad range of practical examples in today's cloud-based service market. For instance, the ASP could be mapped to some Software-as-a-Service provider who offers routine data analytics to its customer firms, or some academic institution that provides scientific data visualization services to the general public.

As illustrated in Fig. 2.1, resource usage incurs monetary cost to the ASP in various forms. Rental activities are charged throughout the life cycle of the deployed service as follows. First, input data is imported into the cloud from the local storage media, introducing network transfer-in cost. Next, a number of VM instances (hereby referred as Virtual Servers, or VS for short) are launched to perform data processing tasks. Each of them costs certain amount of money depending on both VS unit price and rental duration. After the computational jobs are completed, results and logs are saved to cloud storage, and may later be dumped into local persistent storage. Many often the data size is large (e.g., images or videos) and incurs significant storage and network transfer-out cost for the ASP. The storage cost may also apply to input data already fetched into the cloud but not processed yet. Finally, high performance applications often feature tremendous I/O requirements and some resource provider will charge for I/O activities. When performing resource rental planning, an ASP needs to consider all costs described above in order to understand the cost-benefit ratio of possible choices.

Now, considering an ASP rents a number of VSs from the cloud resource market for the purpose of data processing and presentation, in order to achieve resource auto-scaling for efficient resource utilization, the first step is to identify the client workload pattern and build a forecast demand schedule for each VM. Once the forecast demand pattern is built up, the ASP is able to schedule resource rental through job addition, replication, migration and removal.

2.3.2 *Optimizing Planning for Deterministic Pricing Market*

The first resource rental planning model targets at an on-demand resource market where each VS costs a fixed amount of money. Each VS belongs to a specific VS type specifying the hardware configuration. We assume the applications to be elastic and composed of jobs easy to scale gracefully and automatically. For example, applications processing Bags-of-Tasks (no job dependencies). Similar to [14], we are interested in self-aware solutions that can plan resource usage of cloud applications under various pricing. The planning horizon T is divided into fixed time slots $t = 1, \ldots, T$. We refer the start of each time slot as a *decision point*. At each decision point, a rental operation is performed to access the most cost-effective resource available for the application.

Let $\mathscr{T}$ be the set of decision points. The goal of resource rental planning is to minimize the total rental cost associated with processing the forecast workload over the planning horizon T. In order to accomplish this goal, three sets of variables are introduced to identify the rental decisions to be made at each decision point. The first set of variables, $\alpha_{i,t}$, denotes the amount of data to be processed by the application during time slot t on a type-i VS. Next, at the end of slot t, we use $\beta_{i,t}$ to represent the desired storage space for holding the data. Finally, let binary decision variables χ_t denote if powering on a type-i VS is needed at time slot t. $\alpha_{i,t}$ and $\chi_{i,t}$ specify how to make use of the computational resources to control the application progress, while $\beta_{i,t}$ determines the amount of storage resources to reserve in a cloud market. If all these variables are determined, an application scaling control policy is formed to guide the rental activities in the cloud market for optimal resource utilization.

A number of cost parameters are associated with our resource rental optimization problem. Specifically, the rental cost (processing cost) for type-i VS in time slot t is $C_p(i,t)$, and the storage rental cost per data unit for slot t is $C_s(t)$. As presented earlier in Sect. 2.3.1, many IaaS providers charge nontrivial cost for data transfer across the cloud boundary. For each time slot t, let $C_{io}(t)$ be the I/O cost for data transfer from and to the cloud storage, and let $C_f^+(t)$ and C_f^- be the cost for transferring into and out of the cloud, respectively. In addition to the cost parameters, we assume the customer's demand function is $D(\cdot)$, where $D(i,t)$ denotes the forecast workload demand profile for a type-i VS in slot t. We summarize the notation used throughout the chapter in Table 2.1.

Table 2.1 Summary of notations

Variables	
$\alpha_{i,t}$	Output data size generated by one type-i VS in time slot t
$\beta_{i,t}$	Storage space for data produced by one type-i VS at the end of slot t
$\chi_{i,t}$	Binary decision variable representing rental decision of one type-i VS in time slot t
Parameters	
$\mathscr{T}$	Set of time slots
$\mathscr{I}$	Set of VS types
$C_p(i,t)$	VS rental cost (per type-i VS · slot duration)
$C_s(t)$	Storage cost (per data unit · slot duration)
$C_{io}(t)$	I/O operation cost (per data unit · slot duration)
$C_f^+(t)$	Network transfer-in cost (per data unit · slot duration)
$C_f^-(t)$	Network transfer-out cost (per data unit · slot duration)
$\mathrm{D}(i,t)$	Demand to be satisfied for one type-i VS at the end of slot t
$\mathrm{P}(i)$	Average bottleneck resource consumption rate (per data unit generated) for one type-i VS
$\mathrm{Q}(i,t)$	Bottleneck resource available for one type-i VS in time slot t
Φ_i	Average output-to-input ratio for one type-i VS (application specific)

With all the prerequisites, we formulate the rental payment function following a linear cost model. More specifically, the rental cost is linearly proportional to the consumed resource amount as well as to the duration of the rental period. Naturally, our objective function aims at minimizing the rental cost for each type-i VS over the entire planning horizon T. At each decision point, a fixed rental cost $C_p(i,t)$ is charged if the ASP decides to rent one type-i VS ($\chi_{i,t} = 1$). Now, given the presence of this computational resource cost, the ASP may choose to make full use of the VS capacity so as to meet the forecast workload demand over a number of future time slots. However, doing so will increase the storage and I/O cost as more workload is processed earlier in time. As such, the planning problem emerges as the ASP needs to carefully trade off the computational rental cost versus storage and data migration costs. In production planning, similar problems are recognized as the dynamic lot-sizing problem. The solution to the dynamic lot-sizing problem determines the optimal frequency of setups so as to minimize the total cost within the resource and demand constraints. In the context of cloud computing, we formulate the planning problem under fixed resource pricing as the **D**eterministic **R**esource **R**ental **P**lanning (DRRP) problem. DRRP models cloud resource rental on a per-VS basis, forming a fine-grained control policy for rental planning. The complete model formulation is given as follows.

$$\min \sum_{t \in \mathscr{T}} (C_f^+(t) \cdot \Phi_i \cdot \alpha_{i,t} + (C_s(t) + C_{io}(t)) \cdot \beta_{i,t} + C_f^-(t) \cdot \mathrm{D}(i,t) + C_p(i,t) \cdot \chi_{i,t}) \tag{2.1}$$

s.t.

$$\beta_{i,t-1} + \alpha_{i,t} - \beta_{i,t} = \mathrm{D}(i,t), \qquad i \in \mathscr{I}, t \in \mathscr{T} \tag{2.2}$$

$$\mathrm{P}(i) \cdot \alpha_{i,t} \le \mathrm{Q}(i,t), \qquad i \in \mathscr{I}, t \in \mathscr{T} \tag{2.3}$$

$$\alpha_{i,t} \le B \cdot \chi_{i,t}, \qquad i \in \mathscr{I}, t \in \mathscr{T} \tag{2.4}$$

$$\beta_{i,0} = \varepsilon, \qquad i \in \mathscr{I} \tag{2.5}$$

$$\alpha_{i,t}, \beta_{i,t} \in \mathbb{R}_+, \qquad i \in \mathscr{I}, t \in \mathscr{T} \tag{2.6}$$

$$\chi_{i,t} \in \{0,1\}, \qquad i \in \mathscr{I}, t \in \mathscr{T} \tag{2.7}$$

Note that the objective function does not take I/O and storage cost for input data into account. This is because we assume that input data is brought into cloud on the fly to complete the computational jobs. Another option is to copy all input data once and store them in cloud throughout the entire planning horizon. The decision on which option is better depends on the data access pattern and the duration of planning horizon. Here, we assume that input data is "transfer-on-demand" to simplify the presentation.

Constraint (2.2) is analogous to the inventory balance constraint in the dynamic lot-sizing problem. It specifies that workload demand should be met at any time slot. At slot t, the data stored at the previous time slot $\beta_{i,t-1}$, and the data generated in the current slot $\alpha_{i,t}$, are combined together to serve the forecast demand profile emerged in the current time slot, i.e., $\beta_{i,t-1} + \alpha_{i,t} \ge \mathrm{D}(i,t)$. The overprovisioning amount becomes the storage amount $\beta_{i,t}$ at the end of t. The initial storage space is set to be some constant ε in constraint (2.5), depending on the specific planning scenario. Next, let $\mathrm{P}(i)$ be the average bottleneck resource consumption rate for one type-i VS, and let $\mathrm{Q}(i,t)$ denote the bottleneck resource available for one type-i VS in t, constraint (2.3) ensures that the workload processing rate does not saturate the available bottleneck resource.

Constraint (2.4) is often referred to as the forcing constraint. It states that there will be no data generated in t if no rental decision is made ($\chi_{i,t} = 0$). B is set to be a very large constant that exceeds the maximum possible value of $\alpha_{i,t}$. Finally, constraints (2.6) and (2.7) specify domains of the variables.

The formulation of DRRP is a mixed integer linear program (MILP) that is NP-complete in nature. With reasonable input size, this problem can be solved using standard techniques such as the branch-and-bound (B&B) method. These algorithms are implemented in many optimization software packages. For more details with regard to the algorithms, we refer readers to [32].

2.3.3 Evaluation of DRRP

We consider three VS classes $\mathscr{I}$= {c1.medium, m1. large, m1.xlarge}, and perform simulations to evaluate the solution to DRRP based on realistic pricing and application-usage scenarios. The rental planning decisions are calculated in an hourly basis, spanning over daily planning horizon (24 h). The MILP formulation is solved by the CPLEX$^{\text{TM}}$ [12] solver integrated in AIMMS 3.11 [2]. We sample the hourly data processing demand from a normal distribution $\mathscr{N}(0.4, 0.2)$ (expressed in the unit of Gigabyte). It is assumed that the software required by the application services has been configured on virtual servers rented from the cloud market. Therefore, we do not take the initial environment preparation into account.

The cost parameters used in model formulations are set according to Amazon®'s EC2 on-demand pricing policy.[1] Specifically, the hourly on-demand VS rental costs are $\{\$0.2, \$0.4, \$0.8\}$ for the three VS classes. Using Elastic Block Store (EBS), the storage cost is \$0.1 per GB/month, and 0.1 per million I/O operations. The inbound and outbound transfer cost is \$0.1 and \$0.17 per GB. In order to provide realistic parameter estimates in our proposed models, we refer to a recent paper [4] studying the cost and performance of running scientific workflow applications on Amazon® EC2. Based on the 3-year cost of a mosaic service (generated by an astronomical application Montage, see [18] for details) hosted on EC2, we normalize the I/O cost to \$0.2 per GB, and set Φ_i to 0.5 for all $i \in \mathscr{I}$. According to the data provided in [4] (runtime, input and output volume, etc.), the virtual servers are able to offer sufficient resources for serving the randomly generated demand. Therefore, constraint (2.3) in DRRP is omitted.

We first show the cost-saving advantage of our proposed solution over resource rental without planning. The results are shown at the upper side of Fig. 2.2. In our simulation, per-VS costs over daily planning horizon for both schemes are compared. From the results, we observe that cost derived from solving DRRP is significantly lower than that of the no-planning solution. As VS becomes more powerful, the cost reduction becomes more significant. Especially, the cost reduction for VS of class m1.xlarge achieves nearly 50% drop off. This is because compared to the no-planning solution, the cost reduction primarily comes from the saving of computational cost (virtual servers are turned off in cloud when demand is satisfied by cached data in cloud storage). Therefore, more saving is expected for high-cost VS classes. The cost structure for each VS class is presented in the lower side of Fig. 2.2. The proportion of computational cost is relatively stable in all three classes. However, we observe that more money is spent on I/O and storage as VS becomes more powerful. This is because more powerful VS incurs higher VS rental cost each time the rental decision is made. As a result, an ASP tends to utilize caching more often to serve the customer demand and rents VS less frequently.

[1] Amazon® has declared lower pricing for EC2 when we prepared this manuscript. Since our simulation is based on [4], the study presented here is by no means up-to-date, but serves as a representative case of study.

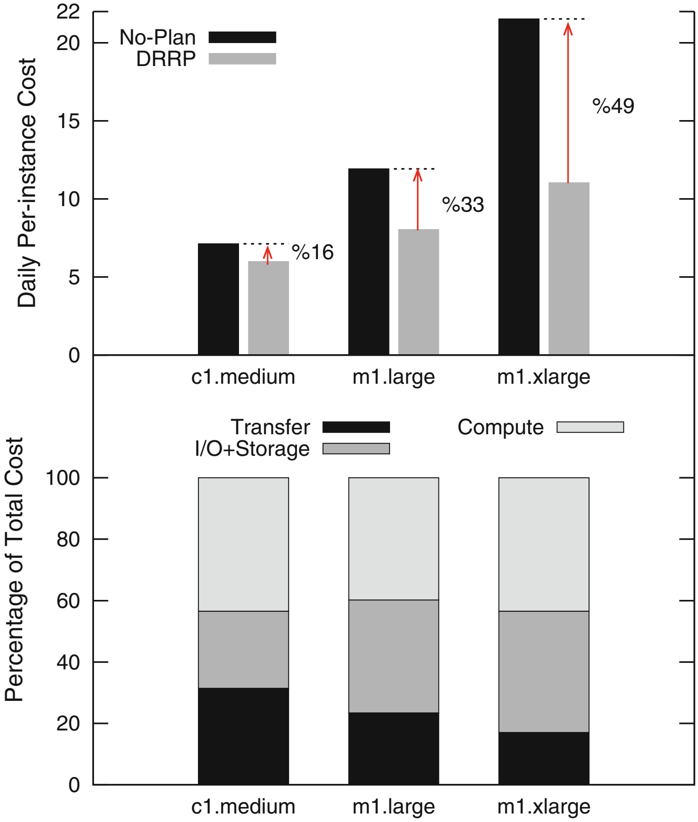

Fig. 2.2 Cost analysis of DRRP

Next, we conduct a sensitivity analysis to the solution for DRRP and plot the results in Fig. 2.3. We define cost ratio as the cost of rental planning based on DRRP to the cost of resource rental without planning. The base ratio (67%) is set to the cost ratio of VS class m1.large calculated in the last simulation. From this base ratio, we first vary the weights of I/O and computational cost gradually. In one direction, we keep the I/O cost fixed and increase the computational cost with a fixed step of 0.1, and then we increase the I/O cost in the other direction similarly. The result showed in the left part of Fig. 2.3 clearly demonstrate that the cost reduction achieved by solving DRRP becomes more salient for expensive computational resources. This conclusion confirms the analysis we previously provided. The impact of demand is investigated in the right part of Fig. 2.3. In particular, we alter the mean of the demand distribution from 0.2 to 1.6 GB/h. As more demand is generated for services, the computational resources tend to be kept busy all the time because the current storage cannot meet the demand. As a result, cost reduction is not noticeable for heavy service demand.

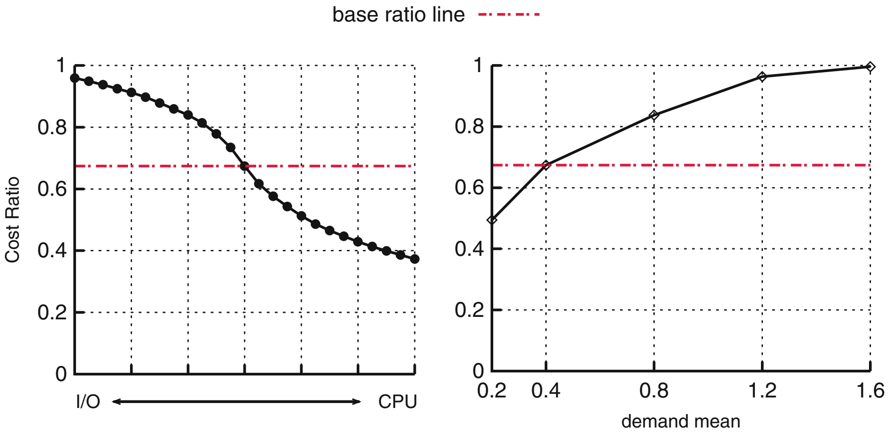

Fig. 2.3 Sensitivity analysis for DRRP

2.4 Dealing with Spot Pricing Uncertainty in Cloud

In this section, we extend the resource rental planning model by including cost uncertainty. Such uncertainty is introduced by many IaaS providers who offer a spot pricing option for idle computational resources. Example markets can be found in [15, 26]. The price fluctuation of spot resources over time creates time series data for analysis. Using Amazon®'s spot market as a case of study, we take two routes to attack the resource rental planning problem with spot pricing uncertainty. First, we apply time series forecasting to spot price history crawled from [11]. The prediction results are then fed into our deterministic planning model (hereafter labeled as **predictive planning**). Next, we propose an alternative approach that leverages the price distribution information (hereafter labeled as **stochastic planning**). A dynamic programming algorithm is also presented to solve the stochastic optimization problem. We compare the two approaches in the end of this section.

Before we proceed, a few assumptions need to be clarified. First, we assume that ASPs will bid truthfully in the spot resource acquisition process. This assumption is in line with the assumption made in [20]. With this assumption, an ASP will not bid strategically. In fact, whether strategic bidding is helpful to achieve some desired level of resource availability is controversial. On the one hand, by exploiting prior price history, it is viable to optimize bidding using probabilistic models for a single bidder [3]. On the other hand, one should also consider bidding strategies of other bidders before making decisions. From a game theoretic perspective, intentionally overbidding or underbidding is not a dominant strategy (e.g., if every bidder overbids, the spot price increases, only benefiting the IaaS provider). Second, an *out-of-bid* event occurs when an ASP's bid price is lower than the spot price.

If an out-of-bid event happens, the ASP needs to rent the desired number of virtual servers from the regular on-demand resource market in order to meet the demand requirement.

2.4.1 Predictive Planning in Amazon® Spot Market

2.4.1.1 Introduction

In this study, we use Amazon®'s spot instance market as a case of study for price prediction and cost optimization. Launched on December 2009, Amazon®'s spot instance market offers a new way to purchase EC2 instances in a discount rate. It allows cloud customers to bid on unused server capacity and use them as long as the bid exceeds the current spot price, which is updated periodically based on supply and demand. Payment in spot instance auction is uniform, i.e., all winners in the auction will pay a per-unit price equal to the lowest winning bid (a.k.a the spot price). While running spot instances saves huge cost (typically over 60 % according to [27]), it also introduces significant uncertainty for resource availability. As a result, previous resource rental planning model based on deterministic resource pricing does not apply.

If one is able to forecast spot prices with relatively high accuracy, then these predictions can be used to instantiate the DRRP model presented in Sect. 2.3.2 to obtain a near-optimal solution. However, performing forecasting is challenging for customers because they do not possess the global information of supply and demand as Amazon® does. In [31], the authors attempted to predict customer demand from the view of an IaaS provider. They proposed a simple auto-regression model for prediction but no prediction results were reported due to the lack of realistic demand information. Another study on the predictability of Amazon®'s spot instance price was presented in [20]. Their work focused on achieving availability guarantee with spot instances, and used a quantile function of the approximate normal distribution to predict when the autocorrelation of current and past price is weak. When the autocorrelation is strong, a simple linear regression prediction model was adopted. However, we found that such an approximation is inaccurate in some test cases that cannot be taken as a generic approach. In this section, we will assess the predictability of spot instance price based on a statistical approach (ARIMA), and estimate the prediction errors using empirical data set.

2.4.1.2 Methodology

We collected the historical data (published in [11]) for spot price variation from February 1, 2010 to June 22, 2011. The data source represents spot price variations for Linux instances in *us-east-1* region. The data size is approximately 100K records. We employ a statistical approach to analyze the predictability of Amazon® EC2 spot pricing, and plot the results in Fig. 2.4.

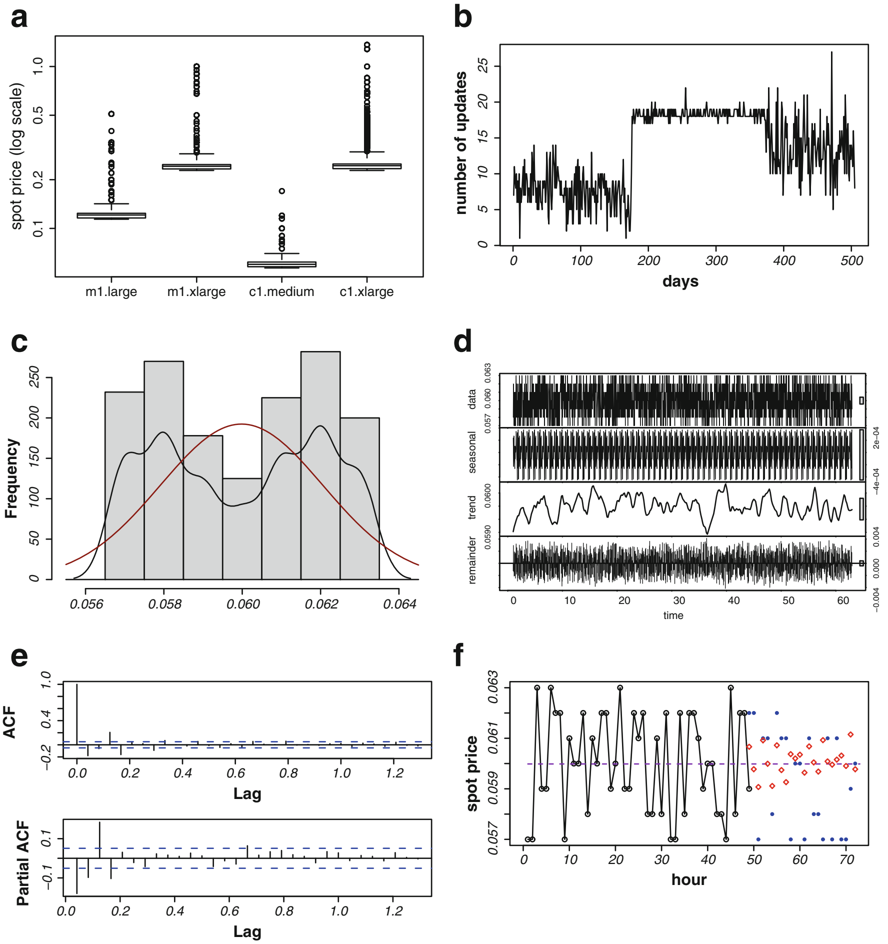

Fig. 2.4 Analyzing the predictability of Amazon® EC2 spot price. (**a**) Box-and-Whisker diagram; (**b**) frequency analysis; (**c**) histogram plot; (**d**) price decomposition; (**e**) correlation analysis; (**f**) 24-h prediction

The first step in our investigation is to identify the outliers in the original data set. Figure 2.4a plots the box-and-whisker diagram for the spot price data set corresponding to four different Linux VM classes. The outliers are identified as those points beyond the whiskers (1.5 IQR (interquartile range) of the upper quartile). We can see that more outliers present in more powerful VM class, indicating increasing price dynamics in more powerful types. However, even for the most powerful instance (c1.xlarge), the number of outliers still contributes a trivial amount to the overall data set (<3 %).

Having trimmed out the outliers, we still cannot apply standard time series analysis because the derived data set is unequally spaced with inconsistent sampling intervals, as shown in Fig. 2.4b. It plots the daily price update frequency for VS of class *linux-c1-medium*. For that reason, we further convert the data into equally spaced time series data with a regular update frequency of 24 times per day. At the start of each hour, the spot price is set to be the most recent updated price in the last hour. If no update appears in the last hour, the spot price is unchanged.

We have performed various experiments on this converted data set, each with different time scale of prediction (both short-term and long-term). Here we show a representative prediction result for instance of class *linux-c1-medium* over a period of 2 months. Specifically, we use the data ranging in [12/1/2010, 1/31/2011] as the estimation data set, and data in 2/1/2011 as the validation data set. In other words, the data collected from the 2-month historical records is used to provide the next-day price forecasting. In Fig. 2.4c, we plot the histogram and density of the selected data. We also randomly generate the same number of points from a normal distribution characterized by the three main measures in quantitative statistics (mean, variance and standard deviation), and plot the curve in Fig. 2.4c for comparison. Examination of the Shapiro-Wilk test result (omitted here) verifies that the pricing data does not fit the normal distribution.

In order to identify patterns in the selected series and perform prediction, we use the ARIMA approach developed by Box and Jenkins [6], which retains great flexibility in recognizing data patterns and is relatively lightweight compared to machine learning techniques such as artificial neural networks or support vector machines. Two common processes are used in ARIMA to identify the correct time series pattern. The first process is the Auto-Regressive (AR) process that decomposes observations into a random error component and a linear combination of prior observations. The second process is called the Moving Average (MA) process. In MA, each observation is made up of a random error component, and a linear combination of prior random errors. Given a time series of data X_t, the general form of an ARIMA process is given as follows:

$$\left(1 - \sum_{i=1}^{p} \phi_i \mathrm{L}^i\right)(1 - \mathrm{L})^d \mathrm{X}_t = \left(1 + \sum_{i=1}^{q} \theta_i \mathrm{L}^i\right)\varepsilon_t, \tag{2.8}$$

where L is the lag operator, ϕ_i and θ_i are the parameters for AR and MA process, respectively, and ε_t are error terms. The key to the ARIMA model is to identify parameters p (AR parameter), d (differencing pass), and q (MA parameter) correctly. This is achieved through a series of steps. First, we verify that our test data series is statistically stationary (statistical properties such as mean and variance are constant over time), and does not require further differencing. The decomposition of the selected series is presented in Fig. 2.4d, where the original time series is decomposed into three parts: trend, seasonal, and random noise. We can see that the target series does not exhibit clear trend, but advertises certain cyclic pattern as shown in the seasonal decomposition. For that reason, we revise our prediction approach by employing a Seasonal ARIMA (SARIMA) model, which takes the seasonal component into account. It can be expressed as SARIMA, $(p, d, q)\times(P, D, Q)_{24}$, which includes the seasonal parameters for price prediction.

The next step for identifying the SARIMA model parameters is to plot the correlograms for autocorrelation function (ACF) and partial autocorrelation function (PACF), as displayed in Fig. 2.4e. These two functions help to detect trend and seasonality of the selected series. Note that the x-axis is normalized by frequency so that 1.0 corresponds to lag $=$ 24. From the graphs we observe that, the selected series has certain degree of correlation with its past at certain lag value, e.g., lag $=$ 3, because these values exceed the 95 % confidence limit. However, such a correlation is not strong enough since its value is still far from 1.0 (which indicates perfect correlation).

Finally, the identification of the most appropriate model parameters is achieved by the forecast package developed in R [24]. In the forecast package, the calling of *auto.arima* function will return the best model according to Akaike information criterion (AIC) or Bayesian information criterion (BIC) values. The function performs a search over possible models within the order constraints provided. Through extensive trials, we found that most test series fit SARIMA $(2, 0, 1 \text{ or } 2)\times(2, 0, 0)_{24}$ best. The prediction result for the selected series is shown in Fig. 2.4f. The solid points and the hollow points represent the predicted and the actual prices on February 1st, 2011, respectively. The horizontal dashed line represents the average price in the selected data series, while the fluctuating solid lines represent spot price variation in the past 48 h. We observe that the predicted prices are mostly hanging over the average price line. While this model returns the least prediction error compared to other models, its mean squared prediction error (MSPE) is only slightly better than the simple prediction using the expected mean value.

2.4.2 *Stochastic Planning for Spot Pricing Market*

2.4.2.1 Solution Overview

In addition to the predictive planning approach, we propose an alternative approach that takes the stochastic nature of the spot pricing into account. We model the fluctuation of the spot instance rental cost $C_p(i,t)$ as a stochastic process C_p with state space $\mathbb{S}$. C_p is a collection of $\mathbb{S}$-valued random variables on a probability space Ω indexed by the time slot set $\mathscr{T}$, i.e., C_p for class-i instance is a collection: $\{C_p(i,t) : t \in \mathscr{T}\}$. The true valuations of the spot prices over the planning horizon are represented by set: $\{\widehat{C_p}(i,t) : t \in \mathscr{T}\}$. The goal of the stochastic resource rental planning is to optimize the expected overall cost over the complete state and probability space. In particular, the objective function (2.1) in DRRP can be reformulated as follows:

$$\delta_{exp} = \mathbf{E}_{C_p}\left\{\sum_{t\in\mathscr{T}}(C_f^+(t)\cdot\Phi_i\cdot\alpha_{i,t} + (C_s(t) + C_{io}(t))\right.$$
$$\left.\cdot\beta_{i,t} + C_f^-(t)\cdot \mathrm{D}(i,t) + C_p(i,t)\cdot\chi_{i,t})\right\}, \tag{2.9}$$

where δ_{exp} is the expected total cost. The optimization model now becomes to minimize (2.9), subject to constraints (2.2)–(2.7). We summarize our solution to stochastic resource planning as follows.

1. Generate bid prices $\widehat{C_p}(i,t)$ for the class-i VS at every $t \in \mathscr{T}$, based on the true valuations.
2. Calculate the base probability distribution according to the pricing history.
3. Derive new probability distributions at all $t \in \mathscr{T}$ according to the base distribution and the bid price.
4. Reformulate using a multistage recourse approach, based on the newly generated distributions.
5. Solve the deterministic equivalent reformulation.

Due to the possibility of losing the auction, the actual realizations of spot prices are possibly different at multiple decision points. Steps (1)–(3) summarize our solution to this challenge. We call our proposed approach bid-dependent dynamic sampling. After calculating the distributions, a multistage resource model is used to optimize the expected total cost.

2.4.2.2 Bid-Dependent Dynamic Sampling

Let $\mathbb{S}_i$ be the finite state space for the spot price of a class-i VS. A base probability distribution is the summarized discrete probability distribution over a selected historical price series: $Pr(C_p(i,t) = s_i), s_i \in \mathbb{S}_i$. This distribution cannot be used in our stochastic optimization model because it does not include the risk of out-of-bid. Therefore, we propose to use the following approach to dynamically generate the probability distribution at every decision point t. The values in the finite state space $\mathbb{S}_i$ is sorted in the ascending order (no equivalent values are present in $\mathbb{S}_i$). Suppose the fixed on-demand cost is λ_i. At each decision point, we keep all the probability distributions for those prices in the base distribution whose values are less than the bid prices, i.e., $s_i \leq \widehat{C_p}(i,t)$. The rest of the distributions are substituted by the following probability representing the likelihood of the out-of-bid event.

$$Pr(C_p(i,t) = \lambda_i) = 1 - \sum_{s_i\leq\widehat{C_p}(i,t)} Pr(C_p(i,t) = s_i) \tag{2.10}$$

Note that it is impossible to generate the precise distribution at each decision point because we do not know the actual realization of the spot price in advance.

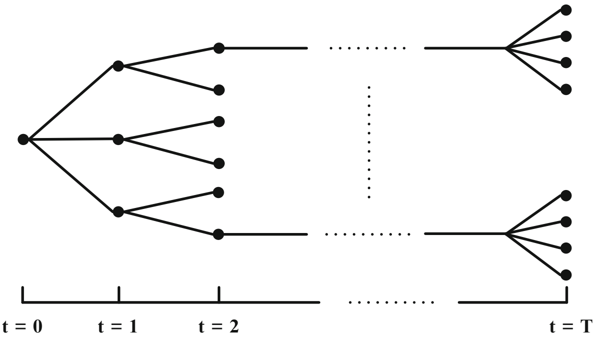

Fig. 2.5 An example of multistage scenario tree: each leaf vertex represents a scenario, and each non-leaf vertex represents an intermediate state within the planning horizon. A probability is associated with each branch representing the likelihood of state transition

Therefore, the dynamically generated distribution based on the ASP's bid price is an **approximation** to the actual spot price distribution. However, stochastic planning using this approximated distribution outperforms deterministic planning using fixed cost parameters. We will illustrate this point as well as the impact of approximation precision to stochastic planning in the later part of this section.

2.4.2.3 Transforming Using Multistage Recourse

We formulate the problem of **S**tochastic **R**esource **R**ental **P**lanning (SRRP) as a stochastic optimization problem, and build a multistage recourse model to solve this problem. The multistage recourse model allows the application planner to adopt a decision policy that can respond to random events as they unfold. Initially, decisions are made given present resources. As time evolves, possible adjustments (recourse actions) become available to the application planner. As to SRRP, rental planning decisions at various decision points are recourse variables.

The dynamic stochastic spot prices are represented in a multistage scenario tree, $\mathscr{G} = (\mathscr{V}, \mathscr{E})$, presented in Fig. 2.5. A scenario tree has $T + 1$ stages. The first stage represents the current state of the world, and all subsequent stages correspond to the future time slots when new information is available to the application planner. A vertex v in stage $t \in \mathscr{T}$ stands for the state of the system that can be distinguished by information available up to stage t. Each vertex $v \in \mathscr{V}$, except the root vertex (indexed as $v = 0$), has a unique parent vertex $\pi(v)$. The probability associated with the state represented by vertex v is p_v. Let $\tau(v)$ denote the time stage of vertex v in the tree, we have: $\sum_{\tau(v)=t} p_v = 1$. Each non-leaf vertex v is the root of the subtree: $\mathscr{G}(v) = (\mathscr{V}' \subseteq \mathscr{V}, \mathscr{E}' \subseteq \mathscr{E})$ containing all descendants of vertex v. The complete tree is represented by $\mathscr{G} = \mathscr{G}(0)$.

Let the set of leaf vertices of $\mathscr{G}(0)$ be $\mathscr{L}$, and let the set of vertices on the path from the root to vertex v be $\mathscr{P}(v)$. If $v \in \mathscr{L}$, then $\mathscr{P}(v)$ represents a scenario of the problem describing a joint realization of the stochastic parameters over all stages. Otherwise, $\mathscr{P}(v)$ denotes a partial realization of the problem up to the stage $\tau(v)$. With the notations defined above, a decision variable $X_{i,t}$ defined in the deterministic problem is replaced by a set of scenario-dependent decision variables (recourse variables) presented below.

$$X_{i,t} \Rightarrow \{X_{i,v} | \tau(v) = t\}, t \in \mathscr{T} \tag{2.11}$$

The multistage scenario tree is perfectly balanced because each path from root to leaf vertex has the same length T. However, the numbers of possible states appeared in each stage are not necessarily equal because of the bid-based dynamic sampling process presented in Sect. 2.4.2.2. Given a scenario tree with a scenario set S, the ASP wishes to set a policy that makes different resource rental decisions under different scenarios. For a scenario $S_j \in S$, decisions made at stage t if encountered by scenario S_j is a vector:

$$\{\alpha_{i,v}, \beta_{i,v}, \chi_{i,v}\}, v \in S_j \tag{2.12}$$

The solution must conform to the flow of available information (non-anticipativity). It guarantees that decisions do not rely on information that is not yet available.

2.4.2.4 Deterministic Reformulation of SRRP

Having built the multistage recourse model, we derive a deterministic equivalent formulation of SRRP. In the reformulation, the time-dependent decision variables are eliminated. The new formulation introduces a set of new variables that are indexed by the vertices presented in $\mathscr{G}(0)$. Each variable indexed by vertex v is associated with a probability p_v. As such, the goal of resource rental planning is to solve MILP with regard to the scenario tree. The complete deterministic equivalent formulation of SRRP is given below:

$$\begin{aligned}\min \sum_{v \in \mathscr{V}} p_v \cdot (C_f^+(\tau(v)) \cdot \Phi_i \cdot \alpha_{i,v} + (C_s(\tau(v)) + \\ C_{io}(\tau(v))) \cdot \beta_{i,v} + C_f^-(\tau(v)) \cdot \mathrm{D}(i, \tau(v)) + \\ C_p(i, \tau(v)) \cdot \chi_{i,v})\end{aligned} \tag{2.13}$$

s.t.

$$\beta_{i,\pi(v)} + \alpha_{i,v} - \beta_{i,v} = \mathrm{D}(i, \tau(v)), \qquad i \in \mathscr{I}, v \in \mathscr{V} \tag{2.14}$$

$$\mathrm{P}(i) \cdot \alpha_{i,v} \leq \mathrm{Q}(i, v), \qquad i \in \mathscr{I}, v \in \mathscr{V} \tag{2.15}$$

$$\alpha_{i,v} \leq B \cdot \chi_{i,v}, \qquad i \in \mathscr{I}, v \in \mathscr{V} \tag{2.16}$$

$$\beta_{i,0} = \varepsilon, \qquad i \in \mathscr{I} \tag{2.17}$$

$$\alpha_{i,v}, \beta_{i,v} \in \mathbb{R}_+, \qquad i \in \mathscr{I}, v \in \mathscr{V} \tag{2.18}$$

$$\chi_{i,v} \in \{0, 1\}, \qquad i \in \mathscr{I}, v \in \mathscr{V} \tag{2.19}$$

Since variables at each $t \in \mathscr{T}$ are associated with a number of possible realizations, solving SRRP is equivalent to solving a large-scale MILP. There exist a number of standard techniques to solve this problem, for example, using Benders decomposition [5]. However, due to the huge search space for optimization, they are only suitable for performing short-term resource rental decisions. Fortunately, efficient algorithms are developed that approximate the objective with runtime proportional to the number of nodes on the multistage scenario tree. Readers can refer to Sect. 2.4.2.6 in [32] for more detailed discussions.

2.4.2.5 Evaluation of Stochastic Rental Planning Model

In this section, we perform simulations to evaluate the solution to SRRP model. The simulation setting is based on realistic spot pricing history and application-usage scenario presented in Sect. 2.3.3. First, imagine an oracle who knows all the future realizations of spot prices in advance, and takes them as inputs to the DRRP model. We denote the cost generated by this method as the ideal case cost for fine-grained resource rental planning. We then compute the overpay percentages against the ideal case cost for all other approaches. The price distribution is drawn from the same representative data set described in Sect. 2.4.1.2, paragraph 3. The results are plotted in Fig. 2.6. Here, we use the prediction values obtained from the approach described in Sect. 2.4.1 as the bid prices, because they are the best approximation values we can obtain using statistical analysis of past price history. The cost derived by solving SRRP using forecast prices is labeled as "stochastic planning", and the cost of solving its DRRP counterpart and the cost of using on-demand virtual servers are labeled as "predictive planning" and "on-demand-deterministic", respectively. It is not surprising to see that the deterministic planning scheme using on-demand virtual instances yields the most overpay. In addition, stochastic planning is more cost efficient than predictive planning for all three VS types. This is because planning using price distributions is more adaptive to the uncertain availability of spot resources than deterministic planning, and the approximation errors introduced by bidding are "diluted" by fine-grained scenario division at each decision point. When considering the price distribution at every decision point, stochastic planning better hedges against the risk of the unexpected out-of-bid event compared to rental planning based on forecasting values in predictive planning. We also mimic a common bid strategy that ASPs bid a fixed price equal to the expected mean price of the historical data, and compare its cost derived by stochastic and predict planning. The results shown on Fig. 2.6 draw the same conclusion that stochastic planning has better cost advantage.

Next, we investigate the impact of bid price approximation precision to the stochastic planning approach with regard to cost reduction for VS type c1.medium.

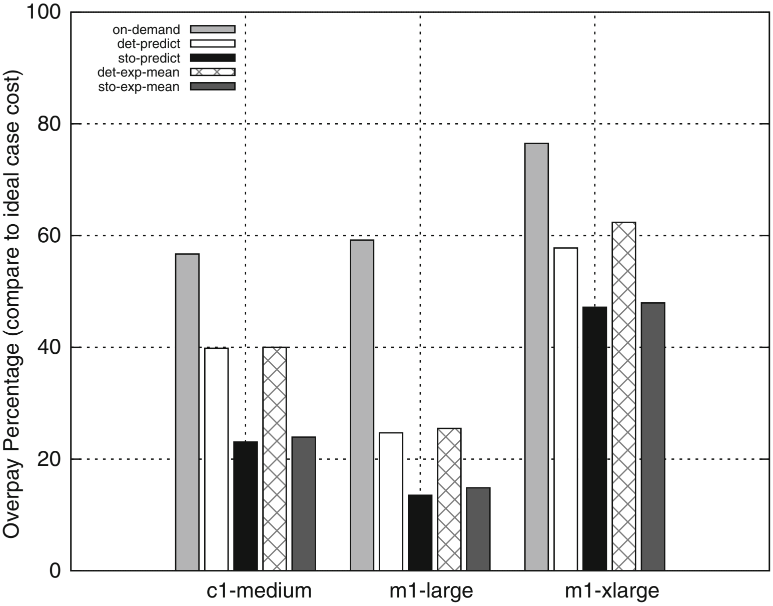

Fig. 2.6 Comparing predictive and stochastic planning

This evaluation is necessary because according to Sect. 2.4.2.1, the solution quality of stochastic planning is closely related to the true valuation $\widehat{C_p}(i,t)$, which is inaccurate in nature with respect to the actual spot price. Taking the cost derived by actual realization of spot price as the baseline cost, we create artificial bid prices that are ±2 % to 10 %[2] deviated from the actual price realizations, and measure the cost deviation from the baseline cost introduced by the approximation errors. The results converted to percent errors against the baseline cost are plotted in Fig. 2.7. Clearly, the errors increase as approximation becomes less accurate. We use the mean squared prediction error (MSPE) to measure the approximation errors. The MSPE of our best approximation achieved based on the method presented in Sect. 2.4.1 falls between that of 2 and 4 % deviation of the model. However, the actual percent error using our approximation is −12 % from the baseline cost. A possible explanation is that our approximations present a mixture of over- and under-estimations of the actual price realizations, thus are different from the pattern of the artificial approximated bid prices we created in the simulation. In conclusion, if one bids according to the best approximation result in practice, the percentage error introduced by approximation is generally acceptable.

[2] Prices that are more than ±10 % from the actual prices are out of the actual price range.

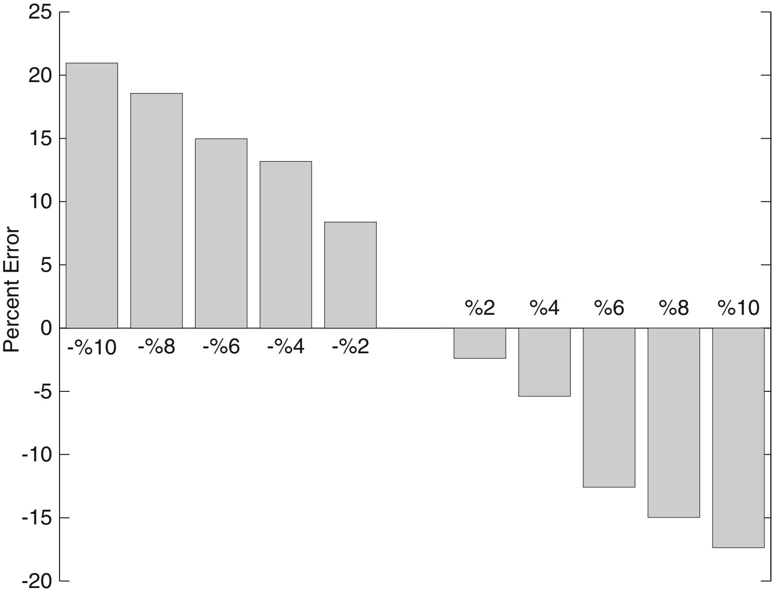

Fig. 2.7 Impact of approximation precision

References

1. Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., Tsafrir, D.: Deconstructing amazon ec2 spot instance pricing. In: IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom 2011), pp. 304–311 (2011)
2. AIMMS Optimization Software. Available: http://www.aimms.com/
3. Andrzejak, A., Kondo, D., Yi, S.: Decision model for cloud computing under sla constraints. In: Proceedings of the 2010 IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS '10), pp. 257–266 (2010)
4. Berriman, G.B., Deelman, E., Juve, G., Regelson, M., Plavchan, P.: The application of cloud computing to astronomy: A study of cost and performance. CoRR (2010)
5. Birge, J.R.: Decomposition and partitioning methods for multistage stochastic linear programs. Operations Research **33**(5), 989–1007 (1985)
6. Box, G.E.P., Jenkins, G.: Time Series Analysis, Forecasting and Control. Holden-Day, Incorporated (1990)
7. Buyya, R., Ranjan, R., Calheiros, R.: Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In: Algorithms and Architectures for Parallel Processing, *Lecture Notes in Computer Science*, vol. 6081, pp. 13–31 (2010)
8. Chaisiri, S., Lee, B.S., Niyato, D.: Optimal virtual machine placement across multiple cloud providers. In: IEEE Asia-Pacific Services Computing Conference (APSCC '09), pp. 103–110 (2009)
9. Chakaravarthy, V.T., Parija, G.R., Roy, S., Sabharwal, Y., Kumar, A.: Minimum cost resource allocation for meeting job requirements. In: 2011 IEEE International Parallel Distributed Processing Symposium (IPDPS '11), pp. 14–23 (2011)
10. Chohan, N., Castillo, C., Spreitzer, M., Steinder, M., Tantawi, A., Krintz, C.: See spot run: using spot instances for mapreduce workflows. In: Proceedings of the 2nd USENIX conference on Hot topics in cloud computing (HotCloud'10), pp. 7–7 (2010)

11. Cloud Exchange. Http://www.cloudexchange.org/
12. IBM ILOG CPLEX optimizer [online]. Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/
13. Deelman, E., Singh, G., Livny, M., Berriman, B., Good, J.: The cost of doing science on the cloud: the montage example. In: Proceedings of the 2008 ACM/IEEE conference on Supercomputing (SC '08) (2008)
14. Demberel, A., Chase, J., Babu, S.: Reflective control for an elastic cloud application: an automated experiment workbench. In: Proceedings of the 2009 conference on Hot topics in cloud computing (HotCloud'09) (2009)
15. EC2 Spot Instance. http://aws.amazon.com/ec2/spot-instances/
16. Market Trends: Platform as a Service, Worldwide, 2012–2016, 2H12 Update. ID: G00239236 (5 October 2012)
17. Gong, Z., Gu, X., Wilkes, J.: Press: Predictive elastic resource scaling for cloud systems. In: 2010 International Conference on Network and Service Management (CNSM '10), pp. 9–16 (2010)
18. Jacob, J.C., Katz, D.S., Berriman, G.B., Good, J.C., Laity, A.C., Deelman, E., Kesselman, C., Singh, G., Su, M., Prince, T.A., Williams, R.: Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking. Int. J. Comput. Sci. Eng. **4**, 73–87 (2009)
19. Mattess, M., Vecchiola, C., Buyya, R.: Managing peak loads by leasing cloud infrastructure services from a spot market. In: Proceedings of the 2010 IEEE 12th International Conference on High Performance Computing and Communications (HPCC '10), pp. 180–188 (2010)
20. Mazzucco, M., Dumas, M.: Achieving performance and availability guarantees with spot instances. In: Proceedings of the 13th International Conference on High Performance Computing and Communications (HPCC'11) (2011)
21. Mishra, A.K., Hellerstein, J.L., Cirne, W., Das, C.R.: Towards characterizing cloud backend workloads: insights from google compute clusters. SIGMETRICS Perform. Eval. Rev. **37**(4), 34–41 (2010)
22. Monti, H.M., Butt, A.R., Vazhkudai, S.S.: Catch: A cloud-based adaptive data transfer service for hpc. In: 2011 IEEE International Parallel Distributed Processing Symposium (IPDPS '11), pp. 1242–1253 (2011)
23. Padala, P., Shin, K.G., Zhu, X., Uysal, M., Wang, Z., Singhal, S., Merchant, A., Salem, K.: Adaptive control of virtualized resources in utility computing environments. SIGOPS Oper. Syst. Rev. **41**, 289–302 (2007)
24. forecast package for R [online]. Available: http://robjhyndman.com/software/forecast/
25. Song, Y., Zafer, M., Lee, K.W.: Optimal bidding in spot instance market. In: IEEE INFOCOM 2012, pp. 190–198 (2012)
26. SpotCloud. http://www.spotcloud.com/
27. How to run MapReduce in Amazon EC2 spot market. Available: http://huanliu.wordpress.com/2011/06/22/how-to-run-mapreduce-in-amazon-ec2-spot-market/
28. Urgaonkar, B., Chandra, A.: Dynamic provisioning of multi-tier internet applications. In: Proceedings of the Second International Conference on Automatic Computing (ICAC '05), pp. 217–228 (2005)
29. Yuan, D., Yang, Y., Liu, X., Chen, J.: A cost-effective strategy for intermediate data storage in scientific cloud workflow systems. In: 2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS '10), pp. 1–12 (2010)
30. Zhang, J., Kim, J., Yousif, M., Carpenter, R., Figueiredo, R.J.: System-level performance phase characterization for on-demand resource provisioning. In: Proceedings of the 2007 IEEE International Conference on Cluster Computing (CLUSTER '07), pp. 434–439 (2007)
31. Zhang, Q., Gürses, E., Boutaba, R., Xiao, J.: Dynamic resource allocation for spot markets in clouds. In: Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services (Hot-ICE'11) (2011)
32. Zhao, H.: Exploring Cost-Effective Resource Management Strategies in the Age of Utility Computing. Ph.D. thesis, University of Florida, Gainesville, FL, USA (2013)