2012 IEEE 26th International Parallel and Distributed Processing Symposium

Optimal Resource Rental Planning for Elastic Applications in Cloud Market

Han Zhao*, Miao Pan†, Xinxin Liu*, Xiaolin Li† and Yuguang Fang†
*Department of Computer & Information Science & Engineering
University of Florida, Gainesville, FL 32611
†Department of Electrical and Computer Engineering
University of Florida, Gainesville, FL 32611

***Abstract*—This paper studies the optimization problem of minimizing resource rental cost for running elastic applications in cloud while meeting application service requirements. Such a problem arises when excessive generated data incurs significant monetary cost on transfer and inventory in cloud. The goal of planning is to make resource rental decisions in response to varying application progress in the most cost-effective way. To address this problem, we first develop a Deterministic Resource Rental Planning (DRRP) model, using a mixed integer linear program, to generate optimal rental decisions given fixed cost parameters. Next, we systematically analyze the predictability of the time-varying spot instance prices in Amazon EC2 and find that the best achievable prediction is insufficient to provide a close approximation to the actual prices. This fact motivates us to propose a Stochastic Resource Rental Planning (SRRP) model that explicitly considers the price uncertainty in rental decision making. Using empirical spot price data sets and realistic cost parameters, we conduct simulations over a wide range of experimental scenarios. Results show that DRRP achieves as much as 50% cost reduction compared to the no-planning scheme. Moreover, SRRP consistently outperforms its DRRP counterpart in terms of cost saving, which demonstrates that SRRP is highly adaptive to the unpredictable nature of spot price in cloud resource market.**

***Index Terms*—Cloud Computing; Amazon EC2; Spot Instance; Resource Rental Planning; Stochastic Optimization**

I. Introduction

With the rapid progress of computing, storage and network technologies, distributed computing paradigms have undergone profound changes in the past decade. Such a revolution enables application service providers (ASPs) to deploy terascale or even petascale applications for production purpose. For example, utilizing cluster or grid computing facilities, scientists are able to run large-scale weather forecast models with a large amount of data generated from the most advanced scientific instruments, and routinely publish the forecast results to the general public [1], [2]. However, due to the huge computational complexity and high volume of data, such applications require demanding resource usage and put a heavy economic burden on the organization who setups, maintains and operates these resources.

In the realm of large-scale distributed computing, the emerging cloud computing concept, with its virtually infinite resources and elasticity, is being adopted with the promise to liberate ASPs from the up-front setup and maintenance cost of the infrastructure. Under such a "pay-as-you-go" cost model, the major issue faced by the ASP is how to minimize the resource rental cost while meeting service demand. In particular, the following aspect of the cost optimization problem draws significant attention from researchers, leveraging resource elasticity, how to optimally provision cloud resources to meet service requirements (avoid the cost due to over-provisioning and the penalty due to under-provisioning) [3]–[6]. However, little work has been devoted to studying how to leverage application elasticity to optimally provision cloud resources from the application's perspective, for example, how to control the data generation progress of a running application to meet the service demand and minimize resource rental cost, given resource pricing and valuation to the application over time. Complementary to autonomic resource scaling, this work opens tremendous new research opportunities, coined as autonomic application scaling, and aimed to minimize resource cost without compromising service-level agreements from the application's perspective. We believe that a complete solution of resource rental planning should leverage both resource elasticity and application elasticity to achieve optimal resource provisioning in cloud market.

In addition to cost tradeoff, another obstacle lies in the uncertainty of computational resource cost. Such a challenge is encountered in the spot resource market [7], [8] emerged in recent years. In spot market, the cost of the resource is fluctuating all the time depending on the resource supply and demand levels. For example, Amazon implements an auction mechanism to determine instance pricing in its spot market. Since resources in spot market are typically idle resources on the on-demand market, they are auctioned off in a price much lower than the regular on-demand price. As a result, the market for Amazon's spot instance grows rapidly. Yet very few research studies have examined how to model the randomness of the spot instance price, and more importantly, how to improve resource usage for achieving cost-effective cloud services under this price uncertainty.

In this paper, we first develop an optimal resource rental planning model for elastic applications in a cloud environment. In particular, given known demand patterns over a specific time period, the ASP needs to periodically review the application

The work presented in this paper is partially supported by National Science Foundation (grants CCF-0953371, CCF-1128805, OCI-0904938, CNS-0709329, and CNS-0916391).

1530-2075/12 $26.00 © 2012 IEEE
DOI 10.1109/IPDPS.2012.77

progress so that no cost is wasted on excessive computation, data transfer and storage. Our first major contribution is the formulation of a deterministic optimization model that effectively solves the resource rental planning problem. We show that our proposed planning model is especially useful for high-cost Virtual Machine (VM) class. This is because cost saving from the proposed planning model primarily comes from eliminating unnecessary data generation cost by decreasing VM rental frequency over the planning horizon. From this perspective, our formulation of the deterministic planning problem is consistent with the dynamic lot-sizing problem commonly met in the field of production planning. The solution to this deterministic optimization model is useful for autonomic resource rental control that balances resource tradeoffs in meeting application goals.

The second major contribution of this paper is that we further address the challenge of price uncertainty in our optimization model. With respect to resource rental planning, the price uncertainty leads to a variety of new research possibilities. In particular, two possible solutions are jointly explored in this paper. We first systematically analyze the predictability of spot price in Amazon EC2 and show that the prediction results cannot provide adequate approximation to be used in the deterministic optimization model. Having shown the limitation of prediction, we propose a multistage recourse model for stochastic optimization. The model decomposes the stochastic process into sequential decision processes with learning of the random data at various stages. As such, the stochastic optimization problem is transformed into a large-scale linear programming problem solvable by commercial optimization packages. The solution to this multistage reformulation takes full advantage of the specific structure of the stochastic planning problem, and thus provides a near-optimal cost for resource rental under stochastic cost parameters.

In summary, the key contributions of this paper are listed as follows:

- A new planning perspective on achieving cost-effective resource rental in cloud market.
- A MILP formulation for determining the optimal resource rental over a fixed planning horizon.
- A systematic analysis of the predictability of spot prices using real price traces of Amazon's spot instances.
- A stochastic optimization solution to deal with spot price uncertainty in resource planning.

The rest of the paper is organized as follows. Section II surveys the related work. In Section III, we present the system model, clarify key assumptions and formulate the deterministic resource rental planning problem. In Section IV, we analyze the predictability of the price of Amazon's spot instance, and propose a stochastic optimization model to solve the problem. The results for performance evaluation are presented in Section V. Finally, we conclude this paper in Section VI.

II. Related Work

Nowadays, a wide variety of computational and data intensive applications utilize cloud to their benefit. For example, enterprise users deploy their applications to provide on-demand customer service, and the scientific community uses cloud resources to solve advanced computational problems. As a result, it becomes imperative to understand the cost-benefit of running applications in cloud in order to make cost-efficient resource rental decisions. Compared with running applications on conventional platforms such as grid, cloud 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 [9], and efforts have been made to mitigate such cost in cloud [10], [11]. In this paper, we present a planning model that optimizes resource usage for elastic applications with comprehensive cost considerations.

Finding an optimal resource utilization strategy is challenging for both resource providers and users. From the perspective of the resource provider, the target is to reduce the operational cost and maximize leasing revenue. Most existing research has focused on this aspect. The general problem on minimizing resource while meeting job demand is NP-hard [12], and various optimization models based on (non-)linear programming were proposed in the current literature. For example, Goudarzi and Pedram [13] formulated the multi-dimensional SLA-based resource allocation problem as a mixed integer non-linear programming problem, and provided a heuristic solution based on force-directed scheduling. Qian and Medhi [14] presented an optimization model for minimizing server operational cost in data center. Resource scheduling for the emerging spot market was proposed in [15]. 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. Different from these works, our paper minimizes the rental cost for resource users. The most relevant work to our research can be found in [16], where the authors presented an optimal VM placement algorithm that minimizes the cost of resource provisioning in a multiple cloud providers environment. Similar to our work, a stochastic optimization model was provided to handle parameter uncertainties. Our work differs from [16] in both application scenario and model formulation. Moreover, our models are formulated and evaluated in a more realistic setting that takes Amazon's cost model and spot price history into account.

The stochastic optimization model proposed in this paper is used to deal with the price uncertainty in the spot resource market. Such a spot market is either formed by multiple resource providers [17] or a single resource provider. The most successful example for the latter is the Amazon's EC2 spot market. Due to its low cost, researchers have proposed to utilize spot instances to temporarily add capacity to dedicated clusters during peak periods [18]. The biggest concern for utilizing spot instances is that it is hard to guarantee resource availability. Recent publications [19], [20] addressed this problem using statistical analysis. However, the effectiveness of such approaches is still in question because the future spot price is jointly affected by the aggregate bidding behaviors and Amazon's resource supply level. This paper simply assumes that users will bid according to their true valuations on the spot instances.

III. Deterministic Rental Planning for On-Demand Resource Market

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. In this section, we present the system model and formulate the deterministic rental planning problem in the context of this model.

A. System Model

Consider an ASP provides data services to customers over a network. Instead of using ASP's local resources, the tasks of computing and intermediate data storage are completely outsourced to a shared resource pool operated by some Infrastructure-as-a-Service (IaaS) provider(s), as shown in Figure 1. Such a system model covers a wide range of practical examples in today's IT environment. For example, the proposed system model could be mapped to a Software-as-a-Service (SaaS) provider providing on-demand data analysis to its customer firms, or an academic institution offering scientific research data processing and accessing to the general public. All resources used to provide the service are rented from a cloud market formed by a single IaaS provider, e.g., Amazon, or a coalition of multiple IaaS providers, e.g., a federation of private clouds resided in distributed data centers belonging to different administrative domains.

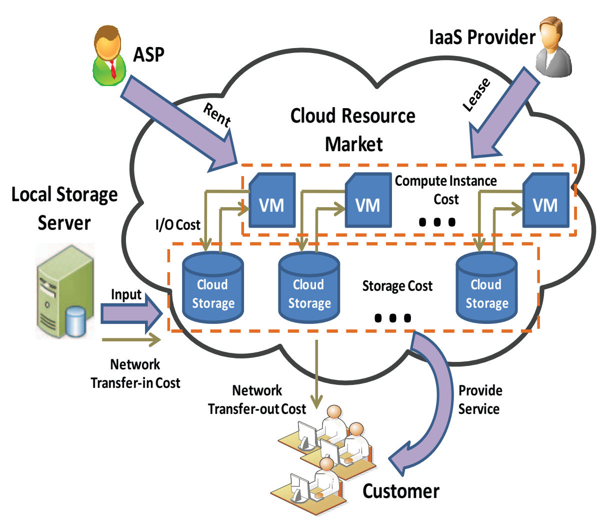


Fig. 1. System model for the problem of resource rental planning

The use of the resources rented from the cloud market incurs certain cost to the ASP. As illustrated in Figure 1, the resource rental cost presents in various forms, depending on the specific resource involved in the rental process. The complete rental cost model is illustrated in the following use case. First, input data is injected into the cloud from local storage media, introducing network transfer-in cost. Next, ASP needs to pay for the launched VM instances to complete the computing tasks. After these tasks are finished, the output data including results and logs is stored in the storage space rented from the cloud market, incurring storage cost for the data accommodation. The storage cost also applies to the input data fetched into the cloud. In addition to the storage cost, reading from and writing to the cloud storage also bring in I/O cost. Finally, transferring data outside of the cloud to meet the demand from the customers results in network transfer-out cost billed to the ASP. In our system model, one needs to consider all costs listed above in order to perform a complete cost/benefit analysis.

B. Resource Rental Planning for Elastic Applications

In general, resource rental planning is conducted over a specified time horizon T. We first target at an on-demand cloud market to present our rental planning strategy. An on-demand cloud market consists of compute instances that can be accessed instantly. Each instance is associated with a fixed rental cost depending on the specific VM class. As such, all cost parameters are deterministic throughout the planning horizon. We assume that the application deployed on the cloud platform is elastic [21], i.e., it can opportunistically adjust its resource usage during the planning horizon T, based on the principle of balancing cost/benefit tradeoffs in meeting application goals. The planning horizon T is divided into fixed time slots $t = 1, ..., T$. We refer the start of each time slot as a *decision point*. At each decision point, the application needs to use the most cost-effective resources available to meet the demand emerged in the current slot.

Considering an ASP rents n compute instances of the same VM class from the cloud market, each serving $1/n$ of the total demand. Since this paper does not consider resource rental under demand uncertainty, we simply assume that the arrival of the total demand at each time slot is known to the ASP, and these n instances are sufficient to satisfy the total demand throughout the planning horizon. Otherwise, the ASP has to decide when to scale up resources to cope with spiky workload, which leads to another set of problems that are out of the scope of this paper. Based on these assumptions, the overall resource cost is calculated as n times the rental cost associated with a single compute instance rented from the cloud market, a.k.a. per-instance cost. The cost per instance includes computing, storage, and data transfer cost spent on dealing with the assigned workload on a single VM instance. Since n for each instance class is fixed, our proposed resource rental planning scheme is conducted on a **per-instance** basis. The design of the planning strategy focuses on changing the work an application plans to do, based on the most cost-effective resources that are available to them. As such, the problem is inherently an optimization problem. We introduce its formulation in the next subsection.

TABLE I
NOTATION BOX

Variables	
$\alpha_{i,t}$	data generated by one class-i instance in time slot t
$\beta_{i,t}$	space for storing data generated by one class-i instance at the end of slot t
$\chi_{i,t}$	binary decision variable representing rental decision of one class-i instance in time slot t
Parameters	
$\mathcal{T}$	Set of time slots
$\mathcal{I}$	Set of VM classes
$C_p(i,t)$	Instance rental cost (per class-i instance $\cdot$ slot length)
$C_s(t)$	Data storage cost (per data unit $\cdot$ slot length)
$C_{io}(t)$	Data I/O cost (per data unit $\cdot$ slot length)
$C_f^+(t)$	Network transfer-in cost (per data unit $\cdot$ slot length)
$C_f^-(t)$	Network transfer-out cost (per data unit $\cdot$ slot length)
$\mathrm{D}(i,t)$	Request volume for one class-i instance in time slot t
$\mathrm{P}(i)$	Average bottleneck resource consumption rate (per data unit generated) for one class-i instance (application specific)
$\mathrm{Q}(i,t)$	Bottleneck resource available for one class-i instance in time slot t
Φ_i	Average input-output ratio for one class-i instance (application specific)

C. Problem Formulation

Let $\mathcal{I}$ denote the set of VM classes, and let $\mathcal{T}$ be the set of all time slots. The goal of the resource rental optimization is to minimize total rental cost for $|\mathcal{I}|$ VM classes over the planning horizon. In order to accomplish this goal, three sets of variables are devised to identify the rental decisions to be made for each VM class at every time slot. Recall that our target application is elastic, the first set of variables, $\alpha_{i,t}$, denotes the amount of data units to be generated by the application on a single class-i instance during time slot t. In practice, the implementation of $\alpha_{i,t}$ can be achieved by controlling the number of input tasks to class-i instance in t (assuming BoT applications). Next, at the end of slot t, we use $\beta_{i,t}$ to represent the request cloud storage space to hold the generated data. Finally, let binary decision variables $\chi_{i,t}$ denote if class-i instance 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 $\beta_{i,t}$ defines the rental decisions for the storage resources in cloud market. If all these variables are determined, a control policy is formed to guide the rental decisions in the cloud market for optimal resource usage.

A number of cost parameters are associated with our resource rental optimization. Specifically, the compute instance rental cost (processing cost) for class-i instance 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 Section III-A, data transfer incurs nontrivial cost in a cloud environment. 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 requested data volume for class-i instance during slot t. For readers' reference, we summarize the notation used throughout the paper in Table I.

We also observe that almost all IaaS providers adopt a linear cost model to implement the "pay-as-you-go" scheme, i.e., the cost is linearly proportional to the amount of the requested resources as well as the length of the rental period. The total cost per instance is defined as the summation of costs spent on all possible consumed resources associated with the instance. In particular, it consists of three parts, each following the linear cost model, as illustrated in Figure 2:

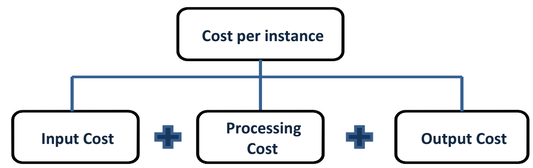


Fig. 2. Cost decomposition associated with a single instance unit

Naturally, our objective function aims at minimizing the total cost of $|\mathcal{I}|$ VM classes over the entire planning horizon. At each decision point, a fixed rental cost of $C_p(i,t)$ is incurred if the ASP decides to rent one class-i instance ($\chi_{i,t}=1$). Now, given the presence of this fixed rental cost, the ASP may want to make full use of the compute instance to run applications so as to meet the demand over a number of future slots. However, doing so will increase the storage and I/O cost as more demand is satisfied earlier in time. As such, the planning problem arises as the ASP needs to carefully control the application progress in order to balance computing, storage, and data migration cost. This problem is well recognized as the dynamic lot-sizing problem in production planning. The model used to solve the dynamic lot-sizing problem determines the optimal frequency of setups so as to minimize the total cost within the resource and service constraints. In the context of cloud computing, we present the complete formulation for the **D**eterministic **R**esource **R**ental **P**lanning (DRRP) as follows:

$$\min \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}(C_f^+(t)\cdot\Phi_i\cdot\alpha_{i,t}+(C_s(t)+C_{io}(t)) \cdot\beta_{i,t}+C_f^-(t)\cdot\mathrm{D}(i,t)+C_p(i,t)\cdot\chi_{i,t}) \quad (1)$$

s.t.

$$\beta_{i,t-1}+\alpha_{i,t}-\beta_{i,t}=\mathrm{D}(i,t), \quad i\in\mathcal{I},t\in\mathcal{T} \quad (2)$$

$$\mathrm{P}(i)\cdot\alpha_{i,t}\le\mathrm{Q}(i,t), \quad i\in\mathcal{I},t\in\mathcal{T} \quad (3)$$

$$\alpha_{i,t}\le B\cdot\chi_{i,t}, \quad i\in\mathcal{I},t\in\mathcal{T} \quad (4)$$

$$\beta_{i,0}=\varepsilon, \quad i\in\mathcal{I} \quad (5)$$

$$\alpha_{i,t},\beta_{i,t}\in\mathbb{R}_+, \quad i\in\mathcal{I},t\in\mathcal{T} \quad (6)$$

$$\chi_{i,t}\in\{0,1\}, \quad i\in\mathcal{I},t\in\mathcal{T} \quad (7)$$

Note that we have not considered I/O and storage cost for input data in the objective function (1). This is because we choose to fetch input data into the cloud on the fly to complete the application. Another option is to transfer all input data once and store them in cloud throughout the planning horizon. The choice on which option is better depends on the specific data access pattern and the duration of planning. This transfer/storage tradeoff problem was noticed by Juve et al. [22] in their investigation on running scientific workflow applications on Amazon EC2. In this paper, we assume that input data is not reused often. Hence, adopting the "transfer-on-demand" approach is more cost-effective.

Constraint (2) is called the inventory balance constraint. It simply specifies that customer demand should be satisfied 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 customer demand emerged in the current time slot, i.e., $\beta_{i,t-1} + \alpha_{i,t} \geq \mathrm{D}(i,t)$. The over provisioning amount becomes the storage amount $\beta_{i,t}$ at the end of t to serve future demand. The initial storage amount is set to be some constant ε in constraint (5), depending on the specific planning scenario. Next, let $\mathrm{P}(i)$ be the average bottleneck resource consumption rate for one class-i instance, and let $\mathrm{Q}(i,t)$ denote the bottleneck resource available for one class-i instance in slot t, constraint (3) ensures that the data generation does not exceed the available bottleneck resource. For example, for data-intensive applications, I/O bandwidth becomes the bottleneck resource. During slot t, the total output data at class-i instance cannot exceed the aggregate throughput volume on that instance.

Constraint (4) is often referred to as the forcing constraint. It states that there will be no data generated in slot 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 (6) and (7) specify domains of the variables.

The formulation of DRRP is a mixed integer linear programming (MILP) problem that is NP-complete in nature. With reasonable input size, this problem can be solved using the branch-and-bound (B&B) method in most optimization software packages.

IV. Dealing with Cost Uncertainty in Spot Resource Market

In this section, we extend the resource rental planning model by including cost uncertainty. Such uncertainty is introduced by the spot resource market created by many IaaS providers willing to sell their idle resources at discounted prices. Examples are Amazon's spot instance [7] and SpotCloud [8]. The price fluctuation of spot instances over time forms a time series data. We first show that the standard time series forecasting does not produce satisfying results for deterministic resource rental planning. Based on this observation, 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.

Before we proceed, a few assumptions need to be clarified. First, we assume that ASPs will bid truthfully in the spot instance acquisition process. This assumption is in line with the assumption made in [20]. Under this assumption, an ASP will bid according to their true valuation rather than bid strategically. In addition, whether a strategic bid approach, such as intentional overbidding, i.e., to bid more than one can expect to win, achieves a desired level of availability is still in question. It only works when no strategic changes are performed by other bidders participating in the auction. From a game theoretic perspective, intentionally overbidding (or underbidding) is not dominant (if every bidder overbids, the spot price is pushed up and only benefits the IaaS provider). Second, an *out-of-bid* event occurs when an ASP's bid price is lower than the spot price. If such an event happens, the ASP needs to rent the desired number of instances from the regular on-demand instance market in order to meet the demand requirement.

A. Price Prediction in Spot Market

1) Background: In this paper, we use Amazon's spot instance market as the target market for price prediction and cost optimization. Launched at December 2009, spot market offers a new way to purchase Amazon EC2 instances. It allows cloud customers to bid on unused Amazon EC2 capacity and use them as long as their bid exceeds the current spot price, which is updated periodically by Amazon based on supply and demand. Payment in spot instance auction is uniform, meaning that all winners in the auction will pay a per unit price equal to the lowest winning bid (a.k.a the spot price), regardless of their actual bids. While running spot instances saves huge cost (typically over 60% according to [23]), it also presents significant uncertainty to cloud customers because of the risk of losing the spot instances unexpectedly if their bid prices are lower than the current spot price. As a result, previous formulation of DRRP is insufficient since the actual cost of resource rental relies on the realization of the random spot price.

If one is able to predict spot prices with relatively high accuracy, then these prediction values could be fed into the DRRP optimization model presented in Section III-C to obtain an approximated optimal solution. However, such a prediction is challenging to cloud customers because there is little, if any, information on the underlying operating mechanism of the spot market. In [15], the authors attempted to predict demand for spot market 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 data. The only work we found for predicting Amazon's spot 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 the subsequent subsections, we will assess the predictability of spot instance price. Our predictability analysis is based on an auto-regressive moving average model, and explores the minimum prediction error over a specified prediction time interval using empirical data set of Amazon EC2.

2) Methodology: We have collected the historical data (published on [24]) 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 first step in our investigation is to identify the outliers in the original data set. Figure 3 plots the box-and-whisker diagram for the spot price data sets corresponding to 4 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 ($< 3\%$).

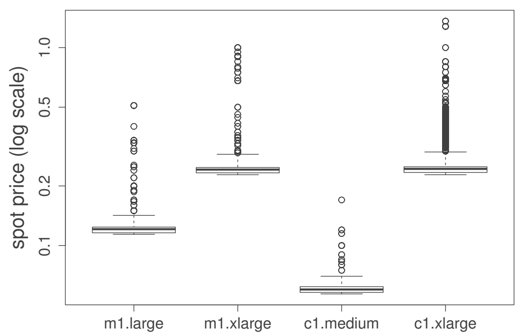


Fig. 3. Box-and-Whisker diagram for spot price data sets

Having trimmed out the outliers, we still cannot apply standard time series analysis techniques because the derived data set is unequally spaced with inconsistent sampling interval, as shown in Figure 4. It plots the daily spot price update frequency for instance 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 considered unchanged.

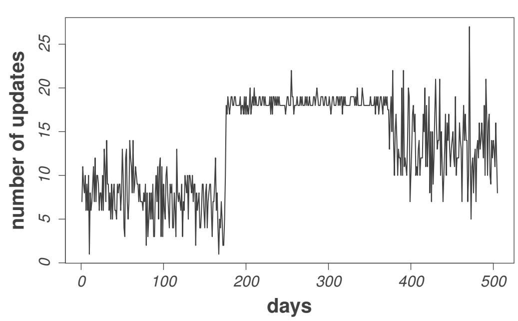


Fig. 4. Variation of daily spot price update frequency for VM class linux-c1-medium

We have performed various experiments on this converted data set, each with different time scale of prediction (both short-term and long-term). Due to the space limit, we only show a representative prediction result for instance of class *linux-c1-medium* over a period of two 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 two-month historical records is used to provide the next-day price forecasting. The daily update frequency throughout this period is relatively stable. In Figure 5, we plot the histogram and density of the selected data set. We also randomly generate the same number of data points from a normal distribution with the same mean and variance, and plot the curve in Figure 5. Apparently, normal distribution is inadequate to approximate the selected data set. This conclusion is further supported by the Shapiro-Wilk test for normality (results are omitted).

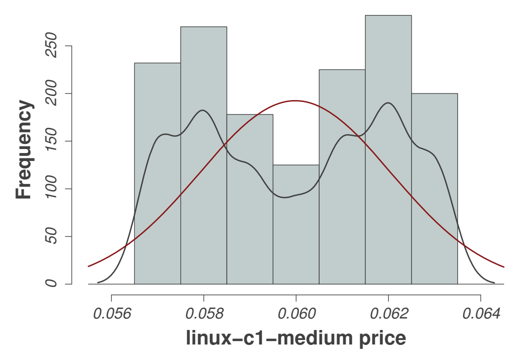


Fig. 5. Histogram plot for the selected spot price history (linux-c1-medium). Black line depicts the density, and the red line depicts the approximate normal distribution sampled from the same mean and variance of the series.

In order to identify patterns in the selected series and forecast, we use the ARIMA approach developed by Box and Jenkins [25], which retains great flexibility in recognizing data patterns and is relatively lightweight compared to machine learning techniques. Two common processes are used in ARIMA to identify time series patterns. The first process is the Auto-Regressive (AR) process that decomposes an observation 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:

$$(1 - \sum_{i=1}^{p} \phi_i \mathrm{L}^i)(1 - \mathrm{L})^d \mathrm{X}_t = (1 + \sum_{i=1}^{q} \theta_i \mathrm{L}^i)\varepsilon_t, \qquad (8)$$

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 ARIMA modeling 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 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 Figure 6, 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 employ a Seasonal ARIMA (SARIMA) model which takes the seasonal component into account. It can be expressed as SARIMA$(p, d, q) \times (P, D, Q)_{24}$ further including the seasonal parameters for price prediction.

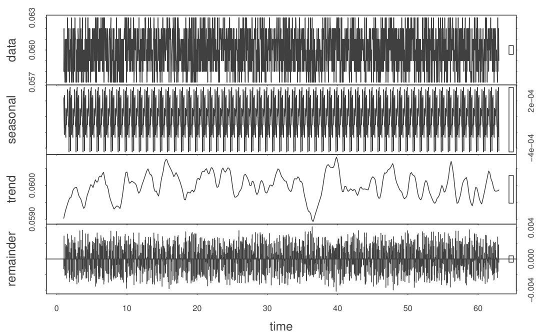


Fig. 6. Data decomposition for the selected data series

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 Figure 7. 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 can 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 because its value is greatly deviated from 1 (1 indicates perfect correlation).

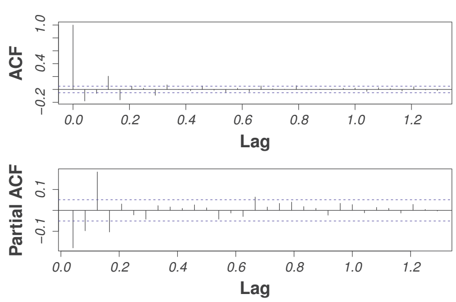


Fig. 7. ACF and PACF for the selected series

Finally, the identification of the most appropriate model parameters is achieved by the forecast package in R [26]. 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 conducts 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 Figure 8. The blue solid points and the red hollow points represent the predicted and the actual prices on February 1st, 2011. The black lines represent spot price variation in the past 48 hours. 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. Therefore, it does not yield satisfactory accuracy and is hardly useful in parameterizing DRRP in practice.

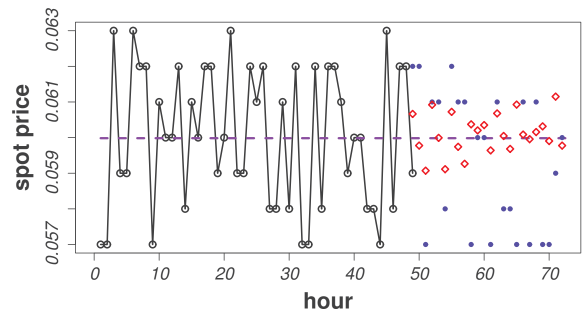


Fig. 8. Day-ahead prediction for the selected series: black line shows the past 48-hour price variation, red solid and blue hollow points represent predict and real prices respectively, purple dashed line denotes the average price in the selected data series.

3) Discussion: Due to the insufficiency of price approximation achieved by the prediction method, we turn our attention to an alternative approach that takes the stochastic nature of the spot price into account. In order to model the price uncertainty, it is useful to set up the probability distribution of the actual prices. Compared to the fixed-price approximation, such a distribution considers all possible scenarios in the actual realization, including the risk of losing the auction (an out-of-bid event occurs). In addition, planning using price distributions is more adaptive to the uncertain availability of the spot instance than deterministic planning. This is because the approximation errors introduced by the bid prices are "diluted" by scenario division at each decision point. This fact motivates us to propose a stochastic optimization model utilizing the bid-based probability distribution. The details of the model are presented in the following subsections.

B. Solution Overview

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 $\mathcal{T}$, i.e., C_p for class-i instance is a collection: $\{C_p(i, t) : t \in \mathcal{T}\}$. The true valuation of the spot prices over the planning horizon is represented by: $\{\widehat{C_p}(i, t) : t \in \mathcal{T}\}$. 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 (1) in DRRP can be reformulated as follows:

$$\delta_{exp} = \mathbf{E}_{C_p}\{\sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}(C_f^+(t)\cdot\Phi_i\cdot\alpha_{i,t} + (C_s(t) + C_{io}(t)) \cdot\beta_{i,t} + C_f^-(t)\cdot \mathrm{D}(i,t) + C_p(i,t)\cdot\chi_{i,t}\}, \tag{9}$$

where δ_{exp} is the expected total cost for $|\mathcal{I}|$ types of instances. The optimization model for SRRP now becomes to minimize (9), subject to constraints (2), (3), (4), (5), (6), and (7). We summarize our solution to the SRRP problem as follows:

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

C. Bid-Dependent Dynamic Sampling

Let $\mathbb{S}_i$ be the finite state space for the spot price of class-i instance. 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 the out-of-bid event. 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 part of the distributions is substituted by the following probability representing the likelihood of the out-of-bid event:

$$Pr(C_p(i,t) = \lambda_i) = 1 - \sum_{s_i \leq \widehat{C_p}(i,t)} Pr(C_p(i,t) = s_i) \tag{10}$$

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. Therefore, the dynamically generated distribution based on the ASP's bid price is merely 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 SRRP in Section V.

D. A Multistage Recourse Model for SRRP

This subsection presents a multistage resource model to solve SRRP. Such a 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, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, presented in Figure 9. A scenario tree has $T+1$ stages (or levels). 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 $t \in \mathcal{T}$ stands for the state of the system that can be distinguished by information available up to stage t. Each vertex $v \in \mathcal{V}$, 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: $\mathcal{G}(v) = (\mathcal{V}' \in \mathcal{V}, \mathcal{E}' \in \mathcal{E})$ containing all descendants of vertex v. The complete tree is represented by $\mathcal{G} = \mathcal{G}(0)$.

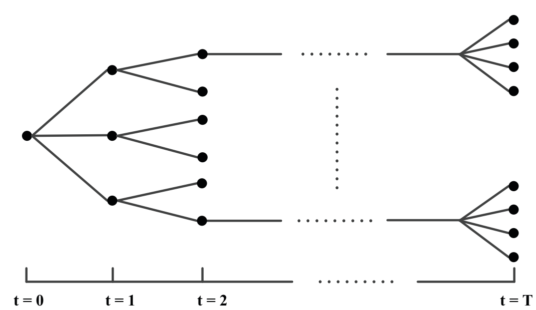


Fig. 9. An example of multistage scenario tree: each leaf vertex represents a scenario, and each non-leaf vertex represents an intermediate state within the planning horizon. A probability is associated with each branch representing the likelihood of state transition.

Let the set of leaf vertices of $\mathcal{G}(0)$ be $\mathcal{L}$, and let the set of vertices on the path from the root to vertex v be $\mathcal{P}(v)$. If $v \in \mathcal{L}$, then $\mathcal{P}(v)$ represents a scenario of the problem describing a joint realization of the stochastic parameters over all stages. Otherwise, $\mathcal{P}(v)$ 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:

$$X_{i,t} \Rightarrow \{X_{i,v} | \tau(v) = t\}, t \in \mathcal{T} \tag{11}$$

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 Section IV-C. Given a scenario tree with a set S scenarios, 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:

$$\{\alpha_{i,v}, \beta_{i,v}, \chi_{i,v}\}, v \in S_j \quad (12)$$

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

E. 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 $\mathcal{G}(0)$. 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 for all nodes of the scenario tree. The complete deterministic equivalent formulation of SRRP is given below:

$$\min \sum_{i\in\mathcal{I}} \sum_{v\in\mathcal{V}} p_v \cdot (C_f^+(\tau(v)) \cdot \Phi_i \cdot \alpha_{i,v} + (C_s(\tau(v)) + C_{io}(\tau(v))) \cdot \beta_{i,v} + C_f^-(\tau(v)) \cdot \mathrm{D}(i,\tau(v)) + C_p(i,\tau(v)) \cdot \chi_{i,v}) \quad (13)$$

s.t.

$$\beta_{i,\pi(v)} + \alpha_{i,v} - \beta_{i,v} = \mathrm{D}(i,\tau(v)), \quad i\in\mathcal{I}, v\in\mathcal{V} \quad (14)$$

$$\mathrm{P}(i)\cdot\alpha_{i,v} \le \mathrm{Q}(i,v), \quad i\in\mathcal{I}, v\in\mathcal{V} \quad (15)$$

$$\alpha_{i,v} \le B\cdot\chi_{i,v}, \quad i\in\mathcal{I}, v\in\mathcal{V} \quad (16)$$

$$\beta_{i,0} = \varepsilon, \quad i\in\mathcal{I} \quad (17)$$

$$\alpha_{i,v}, \beta_{i,v} \in \mathbb{R}_+, \quad i\in\mathcal{I}, v\in\mathcal{V} \quad (18)$$

$$\chi_{i,v} \in \{0,1\}, \quad i\in\mathcal{I}, v\in\mathcal{V} \quad (19)$$

Solving SRRP is equivalent to solving a large-scale MILP. There exists a number of standard techniques to solve this problem, for example, branch-and-cut [27] and Benders decomposition [28]. Compared to the fixed-parameter deterministic planning model, the stochastic model is more accurate in decision making because it adapts to the price uncertainty better. Its advantage on rental cost reduction is presented in the next section.

V. Performance Evaluation

A. Parameter Setting

We consider three VM classes $\mathcal{I} = \{\text{c1.medium}, \text{m1.large}, \text{m1.xlarge}\}$ and conduct two sets of simulations to evaluate the performances of DRRP and SRRP, respectively. The rental planning decisions for both models are made in an hourly basis, spanning over short-term planning horizons. In particular, the decision model is solved for a planning horizon of 24 hours for DRRP, and 6 hours for SRRP. Using this setting, the computational complexity of solving rental planning does not become exorbitant for the ASP. The MILP formulations in both problems are solved using the CPLEX$^{\text{TM}}$ [29] solver integrated in AIMMS 3.11 [30]. We sample the hourly data service demand from a normal distribution $\mathcal{N}(0.4, 0.2)$ (in the unit of GB and is always positive). It is assumed that the software required by the application services has been configured on VMs rented from the cloud market. Therefore, we do not consider the initial VM-image setup process.

The cost parameters used in model formulations are set according to Amazon's EC2 pricing policy. Specifically, the hourly on-demand compute instance rental cost is $\{\$0.2, \$0.4, \$0.8\}$ for the three VM 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 by Berriman et al. [31] 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 [32] for details) hosted on EC2, we normalize the I/O cost to $\$0.2$ per GB, and set the average input-output ratio Φ_i to 0.5 for all $i \in \mathcal{I}$. According to the data provided in [31] (runtime, input and output volume, etc.), the VMs are able to offer sufficient resources for serving the randomly generated demand. Therefore, constraint (3) in DRRP and constraint (15) in SRRP are omitted.

B. Results for Deterministic Rental Planning

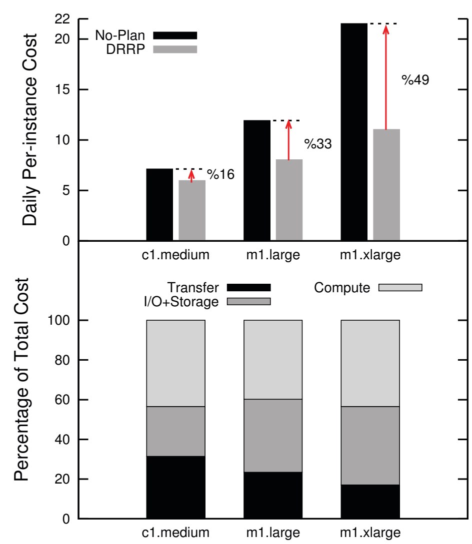


Fig. 10. Cost comparison for DRRP and resource rental without planning

We first show the cost advantage of the DRRP model over resource rental without planning. The results are shown at the upper side of Figure 10. In our simulation, per-instance costs over an 24-hour planning horizon for both schemes are compared. From the results, we can observe that cost derived by DRRP is significantly lower than that of the no-planning solution. As the compute instance becomes more powerful, the cost reduction becomes more significant. Especially, the cost reduction for VM of class m1.xlarge achieves nearly fifty percent drop-off. This is because compared to the no-planning solution, the cost reduction primarily comes from the saving of computing cost (compute instances are turned off in cloud when the current storage meets demand). Therefore, more saving is expected for high-cost VM classes. For DRRP, the cost structure for each VM class is presented in the lower side of Figure 10. The proportion of computing cost is relatively stable in all three classes. However, we observe that more money is spent on I/O and storage as VM instance becomes more powerful. This is because more powerful VM class incurs higher instance rental cost each time the rental decision is made. As a result, an ASP tends to utilize inventory data more often to serve the customer demand and uses compute instance less frequently.

Next, we conduct a sensitivity analysis to the proposed DRRP model and plot the results in Figure 11. We define cost ratio as the cost of DRRP to the cost of resource rental without planning. The base ratio (67%) is set to the cost ratio of VM class m1.large derived in the last simulation. From this base ratio, we first vary the weights of I/O and computing cost gradually. In one direction, we keep the I/O cost fixed and increase the computing cost with a fixed step of 0.1, and we increase the I/O cost in the other direction similarly. The result showed in the left part of Figure 11 clearly demonstrate that the cost reduction achieved by 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 Figure 11. In particular, we alter the mean of the distribution for demand from 0.2 to 1.6 GB/hour. As more demand is generated, the processors 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.

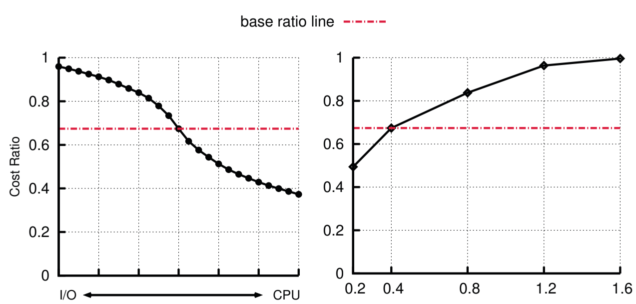


Fig. 11. Sensitivity analysis for DRRP

C. Results for Stochastic Rental Planning

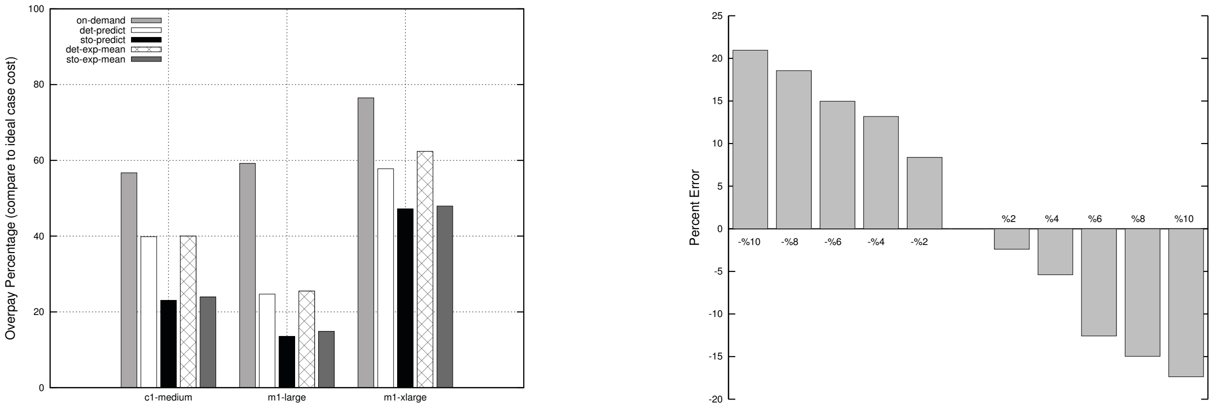
The second set of simulations evaluates the performance of SRRP model. Imagine an oracle who knows all the future realization of spot instance price in advance, and takes them as input to the DRRP model. We denote the cost generated by this method as the ideal case cost for resource rental planning in spot market, and compute the overpay percentages of all other evaluated approaches. The price distribution is drawn from the same representative data set described in Section IV-A2, paragraph 3. The results are plotted in Figure 12(a). Here, we use the prediction values obtained from the approach described in Section IV-A as the bid prices, because they are the best approximation values we can get using statistical analysis of past price history. The cost of SRRP using predicted values is labeled as "sto-predict", and the cost of its DRRP counterpart and the cost of using on-demand instances are labeled as "det-predict" and "on-demand", respectively. It is not surprising to see that the on-demand scheme yields the most overpay. In addition, SRRP model is more cost efficient than its DRRP counterpart for all three VM classes. This is because SRRP model better adapts to the spot price dynamics and thus generates more accurate plans for resource rental. By considering the price distributions at every decision point, SRRP better hedges against the risk of the unexpected out-of-bid event compared to rental planning based on fixed bid in DRRP. 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 DRRP and SRRP. The results shown on Figure 12(a) yield the same conclusion that SRRP outperforms its DRRP counterpart.

Finally, we investigate the impact of bid price approximation precision to the SRRP model with regard to cost reduction for VM class c1.medium. This evaluation is necessary because according to Section IV-B, the solution quality of the SRRP model 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%[1] 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 to the baseline cost are plotted in Figure 12(b). 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 Section IV-A 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. This is probably because 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.

D. Further Discussion

In the simulations, we demonstrate the performance advantage of our proposed planning models. In practice, the resource rental planning is often conducted in a *rolling horizon* fashion,

[1]prices that are more than $+/-10\%$ from the actual prices are out of the price range



(a) SRRP performance: cost comparison

(b) Impact of approximation precision

Fig. 12. Performance evaluation for stochastic resource rental planning

i.e., a revised plan is issued periodically (after a few slots of the whole planning horizon) to include the new information. This is reasonable especially for the SRRP model. Due to the high complexity of computation associated with the large problem size, SRRP is more suitable for short-term rather than long-term planning. Another concern is that SRRP performs significantly better than DRRP only when the chance of losing the spot instance auction is nontrivial. This is because SRRP explicitly includes this risk into its model evaluation while DRRP does not. If a bidder is capable of winning the auction all the time, the performances of both models are expected to be close. In this paper, we simply assume ASPs will bid based on their true valuations, and develop the SRRP model independent of the specific bidding strategy.

VI. Conclusion

In this paper, we investigated the problem of optimizing resource rental planning in a cloud environment. Our optimization model is based on a thorough rental cost analysis of running elastic applications in cloud. Considering the cost tradeoff between data generation and storage, we developed a deterministic optimization model that minimizes the unit rental cost of covering customer demand over a planning horizon. This model works well with deterministic cost parameters but not suitable for the emerging spot instance market in cloud computing. By analyzing the predictability of spot price in Amazon EC2, we show that the spot instance price cannot be well approximated to be used in the deterministic model. Based on this observation, we further designed a stochastic optimization model that seeks to minimize the expected resource rental cost given the presence of spot price uncertainty. Simulations based on realistic settings clearly demonstrate the advantages of our proposed optimization solutions in rental cost reduction. Moreover, we investigated the impact of various parameter settings on the performance of both models. We believe that the proposed optimization models and solutions offer effective means for resource rental planning in practice. Our future work will investigate stochastic optimization solutions for cloud resource provisioning with time-varying workloads.

References

[1] M. Xue, D. Wang, J. Gao, K. Brewster, and K. Droegemeier, "The advanced regional prediction system (arps), storm-scale numerical weather prediction and data assimilation," *Meteorology and Atmospheric Physics*, vol. 82, no. 1, pp. 139–170, 2003.

[2] H. Zhao, X. Yang, X. Li, H. Huang, and X. Ming, "HiCloud: Taming clouds on the clouds," in *2nd IEEE International Scalable Computing Challenge (IEEE SCALE '09)*, 2009.

[3] B. Urgaonkar and A. Chandra, "Dynamic provisioning of multi-tier internet applications," in *Proceedings of the Second International Conference on Automatic Computing (ICAC '05)*, 2005, pp. 217–228.

[4] J. Zhang, J. Kim, M. Yousif, R. Carpenter, and R. J. Figueiredo, "System-level performance phase characterization for on-demand resource provisioning," in *Proceedings of the 2007 IEEE International Conference on Cluster Computing (CLUSTER '07)*, 2007, pp. 434–439.

[5] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem, "Adaptive control of virtualized resources in utility computing environments," *SIGOPS Oper. Syst. Rev.*, vol. 41, pp. 289–302, March 2007.

[6] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *2010 International Conference on Network and Service Management (CNSM '10)*, 2010, pp. 9–16.

[7] "EC2 Spot Instance," http://aws.amazon.com/ec2/spot-instances/.

[8] "SpotCloud," http://www.spotcloud.com/.

[9] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good, "The cost of doing science on the cloud: the montage example," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing (SC '08)*, 2008.

[10] D. Yuan, Y. Yang, X. Liu, and J. Chen, "A cost-effective strategy for intermediate data storage in scientific cloud workflow systems," in *2010 IEEE International Symposium on Parallel Distributed Processing (IPDPS '10)*, 2010, pp. 1–12.

[11] H. M. Monti, A. R. Butt, and S. S. Vazhkudai, "Catch: A cloud-based adaptive data transfer service for hpc," in *2011 IEEE International Parallel Distributed Processing Symposium (IPDPS '11)*, 2011, pp. 1242–1253.

[12] V. T. Chakaravarthy, G. R. Parija, S. Roy, Y. Sabharwal, and A. Kumar, "Minimum cost resource allocation for meeting job requirements," in *2011 IEEE International Parallel Distributed Processing Symposium (IPDPS '11)*, 2011, pp. 14–23.

[13] H. Goudarzi and M. Pedram, "Multi-dimensional sla-based resource allocation for multi-tier cloud computing systems," in *IEEE Cloud 2011*, 2011.

[14] H. Qian and D. Medhi, "Server operational cost optimization for cloud computing service providers over a time horizon," in *Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services (Hot-ICE'11)*, 2011.

[15] Q. Zhang, E. Gürses, R. Boutaba, and J. Xiao, "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.

[16] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *IEEE Asia-Pacific Services Computing Conference (APSCC '09)*, 2009, pp. 103–110.

[17] N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. Tantawi, and C. Krintz, "See spot run: using spot instances for mapreduce workflows," in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing (HotCloud'10)*, 2010, pp. 7–7.

[18] M. Mattess, C. Vecchiola, and R. Buyya, "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)*, 2010, pp. 180–188.

[19] A. Andrzejak, D. Kondo, and S. Yi, "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)*, 2010, pp. 257–266.

[20] M. Mazzucco and M. Dumas, "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] A. Demberel, J. Chase, and S. Babu, "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.

[22] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, and P. Maechling, "Data sharing options for scientific workflows on amazon ec2," in *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC '10)*, 2010, pp. 1–9.

[23] "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/.

[24] "Cloud Exchange," http://www.cloudexchange.org/.

[25] G. E. P. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990.

[26] "forecast package for R [online]," Available: http://robjhyndman.com/software/forecast/.

[27] Y. Guan, S. Ahmed, G. L. Nemhauser, and A. J. Miller, "A branch-and-cut algorithm for the stochastic uncapacitated lot-sizing problem," *Mathematical Programming*, vol. 105, pp. 55–84, 2006.

[28] J. R. Birge, "Decomposition and partitioning methods for multistage stochastic linear programs," *Operations Research*, vol. 33, no. 5, pp. 989–1007, 1985.

[29] "IBM ILOG CPLEX optimizer [online]," Available: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[30] "AIMMS Optimization Software," Available: http://www.aimms.com/.

[31] G. B. Berriman, E. Deelman, G. Juve, M. Regelson, and P. Plavchan, "The application of cloud computing to astronomy: A study of cost and performance," *CoRR*, 2010.

[32] J. C. Jacob, D. S. Katz, G. B. Berriman, J. C. Good, A. C. Laity, E. Deelman, C. Kesselman, G. Singh, M. Su, T. A. Prince, and R. Williams, "Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking," *Int. J. Comput. Sci. Eng.*, vol. 4, pp. 73–87, July 2009.