Exploring Fine-Grained Resource Rental Planning in Cloud Computing

Han Zhao, *Member, IEEE,* Miao Pan, *Member, IEEE,* Xinxin Liu, *Member, IEEE,* Xiaolin Li, *Member, IEEE,* and Yuguang Fang, *Fellow, IEEE*

Abstract—Application services based on cloud computing infrastructure are proliferating over the Internet. In this paper, we investigate the problem of how to minimize cloud resource rental cost associated with hosting such cloud-based application services, while meeting the projected service demand. This problem arises when applications generate high volume of data that incurs significant cost on storage and transfer. As a result, 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 economical trade-off for on-demand resource rental strategies. Given fixed resource pricing, we first develop a deterministic model, using a mixed integer linear program, to facilitate resource rental decision making. Evaluation results show that our planning optimization model reduces resource rental cost by as much as 50% compared with a baseline strategy. Next, we further investigate planning solutions to resource market featuring time-varying pricing (Amazon Spot Instance Market). We perform 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.

Index Terms—Cloud Computing, Amazon EC2, Resource Rental Planning, Linear Programming, Stochastic Optimization

1 Introduction

With the rapid progress of computing, storage, and networking technologies, distributed computing paradigms have undergone profound changes in the past decade. In particular, 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. For example, Foursquare uses Amazon EC2 to perform analytics across more than 5 million daily checkins, and saves 53% in costs while maintaining scalability needs [1]. According to a recent forecast by Gartner, Software-as-a-Service and Cloud-based business application services will grow from $13.4 billion in 2011 to $32.2 billion in 2016 [2].

By adopting 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) [3]–[5]. 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 a fine-grained resource rental planning scheme is needed to further reduce ASPs' operational cost. Specifically, this paper proposes a fine-grained control 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, this work opens up tremendous new research opportunities, coined as application scaling, aimed to develop the most economic resource rental plan without compromising the service-level agreement.

Yet another obstacle lies in the uncertainty of computational resource pricing. This challenge is encountered in the spot resource market emerged in recent years. On a spot resource market, depending on the

- *Han Zhao and Xinxin Liu were with the Department of Computer & Information Science & Engineering, University of Florida, E405 CSE Building, Gainesville, FL, 32611. Email: {han,xinxin}@cise.ufl.edu*
- *Miao Pan is with the Department of Computer Science, Texas Southern University, 122 Nabrit Science Center, Houston, TX, 77004. Email: panm@tsu.edu*
- *Xiaolin Li and Yuguang Fang are with the Department of Electrical and Computer Engineering, University of Florida, 216 Larsen Hall, Gainesville, FL, 32611. Email: {andyli,fang}@ece.ufl.edu*

The work presented in this paper is supported in part by National Science Foundation (grants ACI 1245880, ACI 1229576, CCF-1128805, OCI-0904938, and CNS-0709329).
The work of Y. Fang was partially supported by National Science Foundation under grant CNS-1343356.

resource supply and demand level, the unit price of a computational instance is fluctuating all the time. For example, in December 2009, Amazon launched its spot instance service and implemented an auction mechanism to determine instance pricing, as auctions are effective mechanisms to achieve economically efficient resource allocations and the maximized revenue of the cloud provider [6]–[8]. 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 instance 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, and 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 information. We believe that our study is helpful to understand spot pricing, and more importantly, to improve resource utilization under spot pricing.

In this paper, we first develop a fine-grained optimal resource rental planning model for elastic application scaling in a cloud resource market. In particular, given a forecasted 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. Our first contribution is the formulation of a deterministic planning model that approximates resource rental decision making process over certain planning horizon. The solution to this model serves as a guide to make cost-efficient 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.

The second contribution of this paper is that 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 in this paper. We systematically analyze the predictability of Amazon EC2 spot pricing and show that prediction cannot provide adequate approximation to be used in deterministic planning model. For the purpose of comparison, 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. We further present a polynomial-time solution to the problem. Through simulations, we have shown that the stochastic planning approach is more cost-efficient than predictive planning.

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

- A fine-grained planning approach for cost-efficient resource rental in cloud resource market.
- A MILP formulation for determining the optimal resource rental over a fixed planning horizon.
- A systematic time-series analysis of the predictability of spot pricing using real price traces of Amazon's spot instance service.
- A stochastic optimization solution to deal with spot price uncertainty in resource rental planning.

The rest of the paper is organized as follows. Section 2 surveys the related work. In Section 3, we present the system model, provide a deterministic planning model for the problem, give out an optimal solution, and evaluate the performance of the deterministic pricing resource planning approach. In Section 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 evaluate the performance of our solution. We highlight some interesting issues for further discussion in Section 5 and finally, conclude this paper in Section 6.

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-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]. Tremendous research efforts have been made to mitigate such cost in cloud. 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 cloud infrastructure providers and application service providers. Many studies [10]–[12] attempted to reduce the operational cost and maximize leasing revenue from the perspective of the cloud infrastructure provider. However, the general problem of minimizing resource allocation cost while meeting job demand is NP-hard [13]. This paper approaches cost optimization from the angle of application service providers and proposes solutions to take most advantage of the infrastructure resources by planning ahead.

As computational resources become tradable in cloud, recent works [14]–[16] were geared towards developing novel trading strategies based on information acquired from the market. Amongst all the emerging cloud resource markets, Amazon EC2 is the most representative example which attracts significant research attentions.

For instance, researchers have proposed ideas to allocate resources via brokerage services [17], or take advantage of the low price offered by reserved instances to save cost [18], [19]. With regard to the spot instance market, many works [20]–[22] proposed solutions to achieve resource availability guarantee using statistical analysis. Notably, Ben-Yehuda et al. [21] reverse engineered spot prices by constructing a spare capacity pricing model based on existing price traces. Javadi et al. [22] conducted a comprehensive analysis of spot instances based on one year price history in EC2, and proposed a statistical model to capture the dynamics for price variation in hour-in-day and day-of-week. This paper focuses on analysis of on-demand and spot resource market. The proposed solution targets at the general cloud market and takes EC2 as a case of study. Our work takes one step further from the prior art by presenting a fine-grained planning control model based on the forecast demand. The optimization models take full consideration of various resource types and their associated costs within a cloud resource market, and investigate the optimal tradeoff point in resource rental allocation.

In this paper, the cost optimization model applies to some cloud market where price is market-driven and users bid according to their true valuations (simple-minded assumption). The most relevant works to this paper are presented in [23] and [24]. In [23], the authors presented an optimal VM placement algorithm that minimizes the cost of resource provisioning in a multiple cloud providers environment, and in [24], 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 [23], and we develop our model based on realistic application and price traces. Comparing with [24], our approach presents a different model that takes storage and network transfer cost into account in addition to computational instance bidding. Finally, in a recent study conducted by Xu and Li [25], the authors investigated Amazon EC2 pricing, and developed a revenue management framework to maximize revenue for the cloud provider. Compared to their work, this paper targets at maximizing cost effectiveness for the ASPs and has different problem formulation.

3 Fine-Grained Resource 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. A fine-grained control scheme is proposed to optimize rental decision on a time slotted basis. In this section, we present the system model, formulate the fine-grained rental planning problem in the context of this model, and examine solutions to solve the problem.

3.1 System Model

We consider the 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), as shown in Figure 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 who provides scientific data visualization services to the general public.

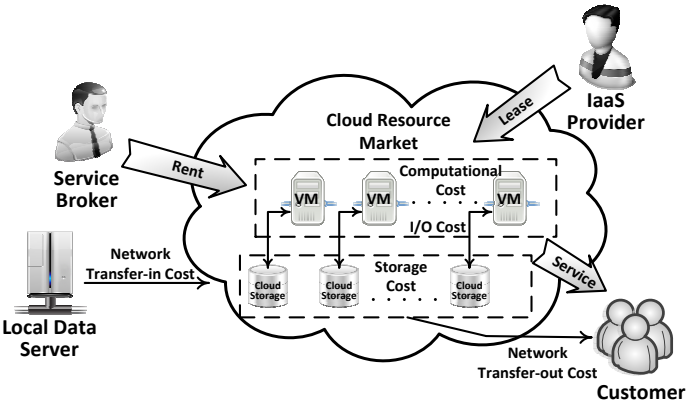


Fig. 1. System model for cloud-based resource rental planning problem

As illustrated in Figure 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 necessary to run the program is imported into the cloud from the local storage media, introducing network transfer-in cost. Next, a number of VM instances are launched to perform data processing tasks. Each of them costs certain amount of money depending on both VM unit price and rental duration. After the computational jobs are completed, results and logs are saved to cloud storage, and may be dumped into local persistent storage later. 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.

3.2 Motivation for Fine-Grained Resource Rental Planning

Resource management in cloud entails multiple levels of scheduling. At a higher level, the ASP needs to decide how many resources to allocate to meet service demand (auto scaling). Essentially, auto resource scaling approaches are grouped into two categories, proactive and reactive. Proactive scaling, due to its preventive nature, gains popularity for tight QoS control. As a predictive resource scaling schedule is established, what's often overlooked is that the ASP can reorganize the usage profile of various types of resources in the predictive horizon to save costs associated with resource rental activities. Such fine-grained resource rental planning stems from the observation that certain types of applications can leverage multiple types of cloud resources to provide services. For example, some image recognition services need to serve user requests in real time, but can leverage pre-processed image processing results fetched from cloud storage to meet current or future service requests.

In this paper, we assume the forecasted demand schedule is established and attempt to build a cost model to analyze the optimal strategy of resource rental activities in a time slotted fashion. Once the ASP creates a clear rental plan over the planning horizon at this fine-grained level, the schedule of jobs[1] is adjusted (through job addition, replication, migration or removal) to line up with the resource rental plan. Suppose we already know the number of computational instances needed at each decision point by solving the higher level scheduling problem, for each instance, we further investigate if we can leverage other types of cloud resources to bring down the rental cost. Therefore, we build our cost optimization model at per-VM level, rather than at the whole resource group level. A motivating example is demonstrated in Figure 2. We plot an example demand forecast for a single VM instance. The top dash line represents the VM's maximum load handling capacity. If the VM strictly follows a job processing schedule in accordance with the forecast workload pattern (red steps), computing cycles are wasted. For example, the ASP pays full rent for the VM between 2pm and 3pm, but does not fully utilize the server's computing capability shown in the shaded area. In this paper, we propose to let the ASP "consolidate" future computing jobs in current rental slot, given that transfer and storage cost less. Our approach introduces operational and management science techniques into cloud computing and achieves better cost-effectiveness compared to on-demand resource provisioning.

3.3 Optimization Planning Model for Deterministic Instance Pricing

The first resource rental planning model targets at an on-demand resource market where each VM costs a fixed

1. Here "job" refers to the general concept of computational tasks.

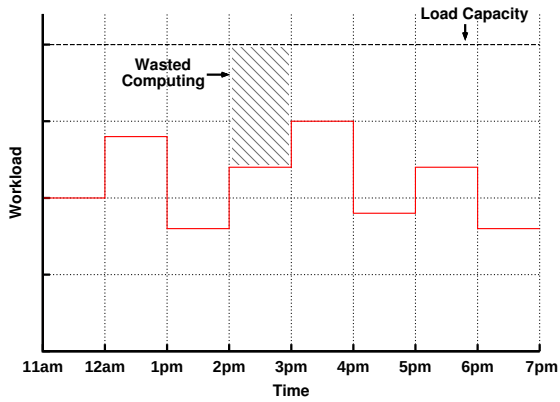


Fig. 2. Example forecasted demand schedule for a VM instance.

amount of money. Each VM belongs to a specific VM type specifying the hardware configuration. We assume the applications are elastic and composed of jobs easy to scale gracefully and automatically. For example, applications processing Bags-of-Tasks (no job dependencies). Similar to [26], 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 $t = 1, ..., T$. We refer to 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 $\mathcal{T}$ 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 represent 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 VM. 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 VM 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, a 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 VM 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 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 this paper, we assume a real-time service model that demand needs

to be satisfied before the end of each decision point. We represent the customer's demand function as $D(\cdot)$, where $D(i,t)$ denotes the forecast workload demand profile (in this paper defined as the output data amount sent to customers) for a type-i VM in slot t.

For readers' reference, we summarize the notation used throughout the paper in Table 1.

TABLE 1
Notation Box

Variables	
$\alpha_{i,t}$	output data size generated by one type-i VM in time slot t
$\beta_{i,t}$	storage space for data produced by one type-i VM at the end of slot t
$\chi_{i,t}$	binary decision variable representing rental decision of one type-i VM in time slot t
Parameters	
$\mathcal{T}$	Set of time slots
$\mathcal{I}$	Set of VM types
$C_p(i,t)$	VM rental cost (per type-i VM · 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 VM at the end of slot t
$\mathrm{P}(i)$	Average bottleneck resource consumption rate (per data unit generated) for one type-i VM
$\mathrm{Q}(i,t)$	Bottleneck resource available for one type-i VM in time slot t
Φ_i	Average output-to-input ratio for one type-i VM (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 VM over the entire planning horizon T (we minimize the cost at per-VM level, as explained in Section 3.2). At each decision point, a fixed rental cost $C_p(i,t)$ is charged if the ASP decides to rent one type-i VM ($\chi_{i,t} = 1$). Now, given the presence of this computational resource cost, the ASP may choose to make full use of the VM 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 arises 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-VM basis, forming a fine-grained control policy for rental planning. The complete model formulation is described in function (1) to (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. In this paper, we simply assume that input data is "transfer-on-demand" to simplify the presentation. In addition, the formulation roughly models the expected average performance of each VM type. In reality, the performance heterogeneity exists within the same VM type [27], [28]. However, we focus on cost analysis in this paper and will not include this factor into our formulation.

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

s.t.

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

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

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

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

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

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

Constraint (2) is analogous to the inventory balance constraint in the dynamic lot-sizing problem. It simply 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 forecasted 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 (5), depending on the specific planning scenario. Next, let $\mathrm{P}(i)$ be the average bottleneck resource consumption rate for one type-i VM, and let $\mathrm{Q}(i,t)$ denote the bottleneck resource available for one type-i VM in t, constraint (3) ensures that the workload processing rate does not saturate the available bottleneck resource.

Constraint (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 (6) and (7) specify domains of the variables. Note that each VM might not be fully utilized during a time slot, as VM boot-up time is non-trivial according to some recent study [29]. According to the DRRP formulation, this could affect the setup of $\mathrm{Q}(i,t)$. Moreover, depending on the effective length of t, the optimal solution is subject to change given

the fixed service demand constraint (Constraint (2)). For simplicity, we assume boot-up time is not counted for defining an effective charging period.

3.4 Solution to Deterministic Pricing Resource Rental Planning

In this section, we briefly demonstrate the techniques which find an optimal plan for objective function (1) subject to constraints (2) to (7). The formulation is a Mixed Integer Linear Program (MILP) that is NP-complete in general. When considering only the production variable $\alpha_{i,t}$ and the inventory variable $\beta_{i,t}$, the problem can be solved by translating the original problem to the minimum cost flow problem [30]. We show the transformation to flow network in Figure 3.

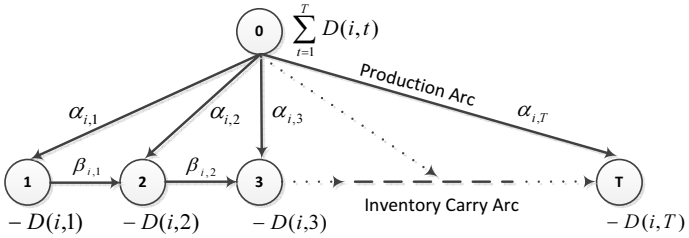


Fig. 3. Flow network of the deterministic pricing resource rental planning problem

As shown in Figure 3, the network has $T+1$ vertices. Vertex 0 is the "source" of all data generation, and is hereafter referred to as the source node. The t-th vertex ($t = 1, ..., T$) represents the t-th planning time slot. These vertices are referred to as sink nodes. Directed edges are categorized into two types. Those connecting the source node with sink nodes are labeled as the production arcs, and those connecting two successive sink nodes are labelled as the inventory carry arcs. We also assign the supply value of $\sum_{t=1}^{T} \mathrm{D}(i,t)$ to the source node, and the demand value of $-\mathrm{D}(i,t)$ to each sink node $t = 1, ..., T$. By establishing this flow network, the inventory balance constraint (2) is translated to the flow balance constraint in the minimum cost flow problem. The resource constraint (3) is enforced by imposing capacities on the production and inventory carry arcs in each period.

When the binary variable $\chi_{i,t}$ and the forcing constraint (4) are included, we can solve the proposed MILP with standard techniques such as the Branch-and-Bound algorithm. We will briefly demonstrate how to use the general Branch-and-Bound technique to solve DRRP as follows.

The first step of the algorithm is to replace the constraint that $\chi_{i,t}$ must be 0 or 1 by a weaker constraint, i.e., $0 < \chi_{i,t} < 1$ for $t = 1...T$. The formulation result is a linear relaxation of the original DRRP program, labeled as DRRP-LR. Let $Z(P)$ be the optimal value of some linear program P. Apparently, the objective value of any feasible solution found, referred to as Z', defines the upper bound of the solution. Therefore, we have,

$$Z(\text{DRRP-LR}) <= Z(\text{DRRP}) <= Z'.$$

The Branch-and-Bound algorithm uses divide-and-conquer principle and construct a search tree over the feasible solution space. Since DRRP-LR is a linear program without integer constraints, we can easily solve it with standard techniques. If it returns a feasible solution, we will use it to initialize the lower bound of $Z(\text{DRRP})$. Without loss of generality, we assume that the solution to DRRP-LR is bounded, otherwise the corresponding DRRP is either unbounded or infeasible. We also initialize the current best solution to DRRP to this value.

If $\chi_{i,t}$ for all $t = 1...T$ are integers, we obtain the best solution already. Otherwise, for $\chi_{i,t}$ has a fractional value, create two new subprograms (branching step), one in which $\chi_{i,t} = 1$ and the other in which $\chi_{i,t} = 0$. Now compute the optimal solution for each subprogram. This process is recursive, as the computation will be conducted on two disjoint sets created by new branching variables. When a new solution is calculated, we compare it with the current best solution. If it's greater, the solution is either upper bounded by the best solution so far or infeasible, therefore no further branching is needed (bounding step), otherwise the value overwrites the current best solution and remove the founded solution from the variable domain. The process terminates when the remaining variable set becomes empty.

3.5 Evaluation of Deterministic Pricing Resource Rental Planning

We consider three VM classes $\mathcal{I} = \{\text{c1.medium}, \text{m1.large}, \text{m1.xlarge}\}$, and perform simulations to evaluate the solution to DRRP based on realistic pricing and application-usage scenarios. The rental planning decisions are made hourly to align with Amazon's charging intervals, spanning over a daily planning horizon (24 hours). Each hour corresponds to one time slot in DRRP. The MILP formulation is solved using the CPLEX™ [31] solver integrated in AIMMS 3.11 [32]. We sample the hourly data processing demand from a normal distribution $\mathcal{N}(0.4, 0.2)$ (expressed in the unit of Gigabyte). It is assumed that the software required by the application services has been configured on VM instances 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[2]. Specifically, the hourly on-demand VM rental costs are $\{\$0.2, \$0.4, \$0.8\}$ for the three VM classes. Among the three VM classes, c1.medium is compute-optimized while m1.large and m1.xlarge are general-purpose instance classes. We choose them as representative VM classes to evaluate due to the following reasons: (1) to

2. Amazon has declared lower pricing for EC2 when we prepared this manuscript. Since our simulation is based on [33], the study presented here is by no means up-to-date, but serves as a representative case of study.

evaluate VM instances that differ in hardware capability; and (2) to evaluate if compute-optimized instance class has an impact to the cost structure breakdown. 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 [33] 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 [34] 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 $i \in \mathcal{I}$. According to the data provided in [33] (runtime, input and output volume, etc.), the VM instances are able to offer sufficient resources for serving the randomly generated demand. Therefore, constraint (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 Figure 4. In our simulation, per-VM costs over daily planning horizon for both schemes are compared. From the results, we can observe that cost derived from solving DRRP is significantly lower than that of the no-planning solution. As VM becomes more powerful, the cost reduction becomes more significant. Especially, the cost reduction for VM 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 (VMs are turned off in cloud when demand is forecasted and satisfied by previously rented computational cycles). Therefore, more saving is expected for high-cost VM classes. The cost structure for each VM class is presented in the lower side of Figure 4. We observe that more money is spent on I/O and storage as VM becomes more powerful. This is because more powerful VM incurs higher VM rental cost each time the rental decision is made. As a result, an ASP tends to rent VM less frequently to serve the customer demand. Another interesting observation is that c1.medium has similar compute cost percentage with m1.xlarge. Although c1.medium is cheaper and in general less powerful than m1.large, it has better computational resources (5 Compute Units VS. 4 in m1.large). It is a clear evidence that for compute-optimized VM instances, allocating more time for computation yields more cost saving in general. Therefore, the relative capability of the hardware components does have a positive impact to the optimal cost saving model.

Next, we investigate cost saving with regard to different planning horizons, and plot the results in Figure 5. In the first set of evaluations, we use the same demand data and calculate the total cost for DRRP with the step increment of four hours. The results are then compared against the cost without planning, and are summarized with labels "vm-class-24h" corresponding to the bottom x axis in Figure 5. Note that cost saving varies because it

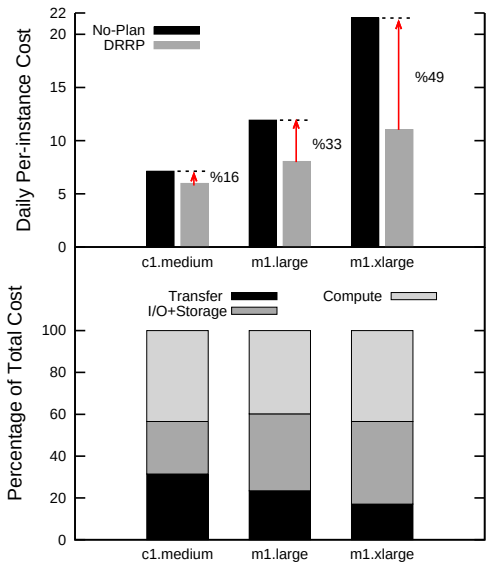


Fig. 4. Cost comparison for DRRP and resource rental without planning

is correlated with the demand input, therefore, we cannot draw the conclusion on the optimal planning horizon from this set of evaluations. However, we do know that given the same demand input across the planning horizon, the effectiveness of DRRP is independent of specific VM classes as they exhibit the same trend across time. In the second set of evaluations, we generate demand with a new distribution, and create a "cyclic" input pattern for every 12 hours, i.e., the same input is used for every 12 hours. The cost saving percentage is plotted with label "m1.xlarge-72h" corresponding to the top x axis in Figure 5. It might be tempting to increase planning horizon for better cost efficiency. However, doing so will not give ASPs a lot of gains, because we observe the cost saving is relatively stable across time. On the other side, increased planning horizon results in more inaccuracy due to more variables in larger prediction window and increased computational overhead. ASPs should select the optimal planning horizon based on the most confident prediction window within acceptable computational overhead (for solving DRRP).

Finally, we conduct a sensitivity analysis to the solution for DRRP and plot the results in Figure 6. 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 VM 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 Figure 6 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

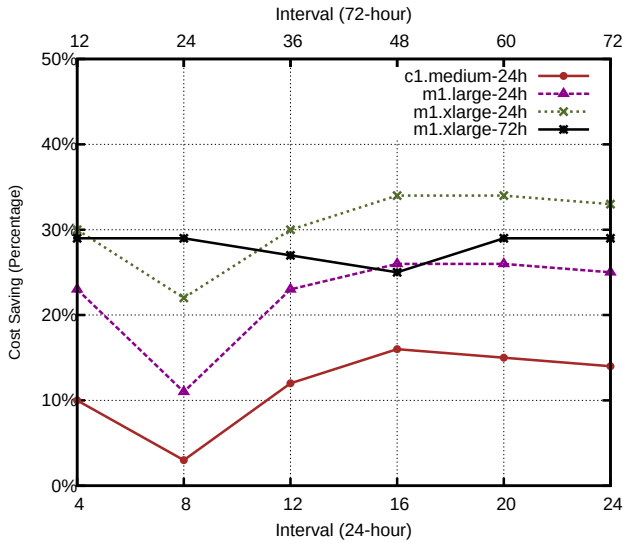


Fig. 5. DRRP: cost saving over time

of demand is investigated in the right part of Figure 6. In particular, we alter the mean of the demand distribution from 0.2 to 1.6 GB/hour. 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.

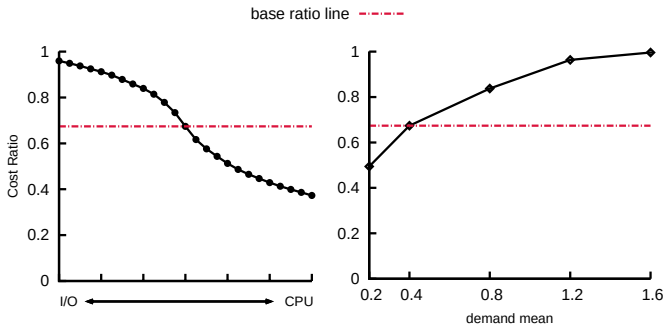


Fig. 6. Sensitivity analysis for DRRP

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 spot pricing option for idle computational resources. 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 [35]. The prediction results are fed into our deterministic planning model (hereafter labelled as **predictive planing**). Next, we propose an alternative approach that leverages the price distribution information (hereafter labelled 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, and conclude that the stochastic planning approach saves more cost than the predictive planning approach.

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 [36]. 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 such an event happens, the ASP needs to rent the desired number of VMs from the regular on-demand resource market in order to meet the demand requirement.

4.1 Predictive Planning in Amazon Spot Market

4.1.1 Background

In this paper, 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 [37]), 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 Section 3.3 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 [14], 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.

4.1.2 Methodology

We have collected the historical data (published on [35]) 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. In this paper, we only briefly describe the methodology we used to evaluate the predictability of spot pricing. More details can be found in our prior work [38]

First, we trim out the outliers in the data set and plot the daily price update frequency for VM of class *linux-c1-medium* in Figure 7. Because the derived data set is unequally spaced with inconsistent sampling intervals, 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.

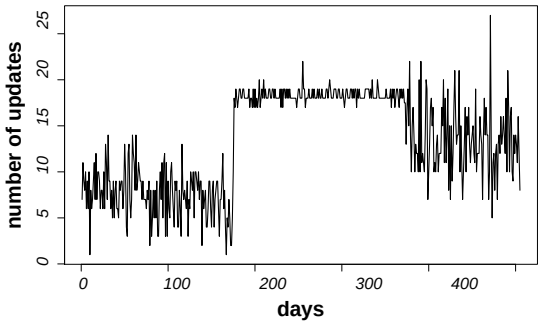


Fig. 7. Variation of daily spot price update frequency for VM class linux-c1-medium

We have performed various experiments on this converted data set, each with different time scale of prediction (both short-term and long-term). In Figure 8, 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 Figure 8 for comparison. Examination of the Shapiro-Wilk test result (omitted here) verifies that the pricing data does not fit the normal distribution.

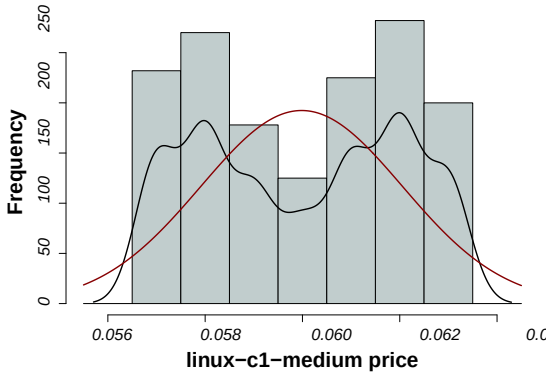


Fig. 8. Histogram plot for the selected spot price history (linux-c1-medium). Black line depicts the density, and the red line depicts the approximate normal distribution sampled from the same mean and variance of the series.

In order to identify patterns in the selected series and perform prediction, we use the ARIMA approach developed by Box and Jenkins [39], 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. The decomposition of the selected series is presented in Figure 9, 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.

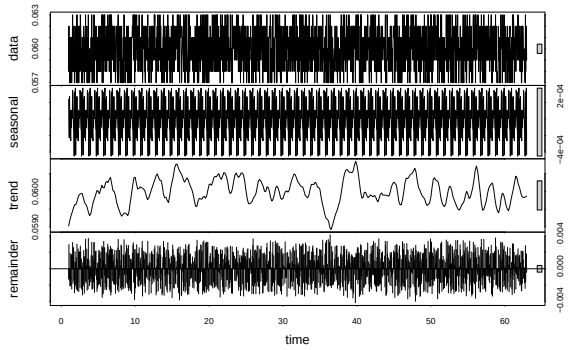


Fig. 9. Data decomposition for the selected data series

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 10. 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 since its value is still far from from 1.0 (perfect correlation).

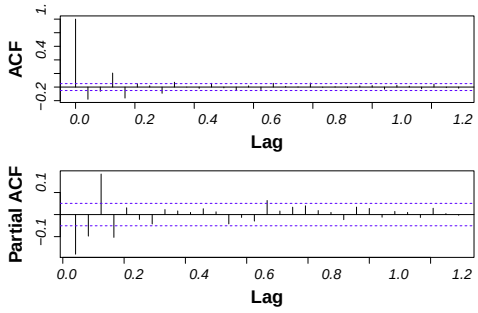


Fig. 10. ACF and PACF for the selected series

Finally, the identification of the most appropriate model parameters is achieved by the forecast package developed in R [40]. The prediction result for the selected series is shown in Figure 11. 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.

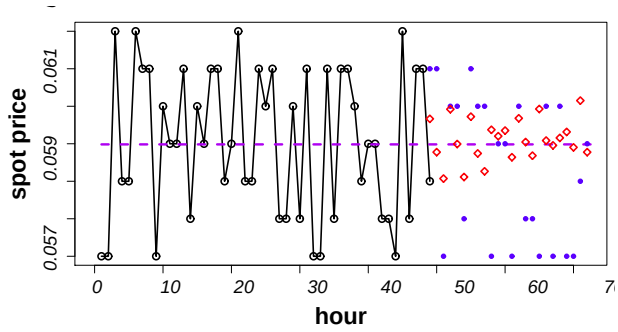


Fig. 11. Day-ahead prediction for the selected series: black line shows the past 48-hour price variation, red solid and blue hollow points represent predict and real prices respectively, purple dashed line denotes the average price in the selected data series.

4.2 Stochastic Planning for Spot Pricing Market

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 $\mathcal{T}$, i.e., C_p for class-i instance is a collection: $\{C_p(i,t) : t \in \mathcal{T}\}$. The true valuations of the spot prices over the planning horizon are represented by set: $\{\widehat{C_p}(i,t) : t \in \mathcal{T}\}$. 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:

$$\begin{aligned}\delta_{exp} = \mathbf{E}_{C_p}\{\sum_{t\in\mathcal{T}}(C_f^+(t)\cdot\Phi_i\cdot\alpha_{i,t} + (C_s(t)+C_{io}(t)) \\ \cdot\beta_{i,t} + C_f^-(t)\cdot \mathrm{D}(i,t) + C_p(i,t)\cdot\chi_{i,t}\},\end{aligned} \tag{8}$$

where δ_{exp} is the expected total cost. The optimization model now becomes to minimize (8), subject to constraints (2), (3), (4), (5), (6), and (7). We summarize our solution to stochastic resource planning as follows.

1) Generate bid prices $\widehat{C_p}(i,t)$ for the class-i VM at every $t \in \mathcal{T}$, based on the true valuations.
2) Calculate the base probability distribution according to the pricing history.
3) Derive new probability distributions at all $t \in \mathcal{T}$ 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.

If the bid price is not sufficient, the planner has to wait until the next auction starts (in this paper we assume that the auction is held regularly at the start of each planning period). Due to the possibility of losing the auction, the actual realizations of spot prices are possibly different at multiple decision points. Step 1), 2) and 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.

4.2.2 Bid-Dependent Dynamic Sampling

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

$$Pr(C_p(i,t) = \lambda_i) = 1 - \sum_{s_i \leq \widehat{C_p}(i,t)} Pr(C_p(i,t) = s_i) \quad (9)$$

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 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.

4.2.3 Transforming Stochastic Planning 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, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, presented in Figure 12. 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 $t \in \mathcal{T}$ 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: $\mathcal{G}(v) = (\mathcal{V}' \subseteq \mathcal{V}, \mathcal{E}' \subseteq \mathcal{E})$ containing all descendants of vertex v. The complete tree is represented by $\mathcal{G} = \mathcal{G}(0)$.

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.

$$X_{i,t} \Rightarrow \{X_{i,v} | \tau(v) = t\}, t \in \mathcal{T} \quad (10)$$

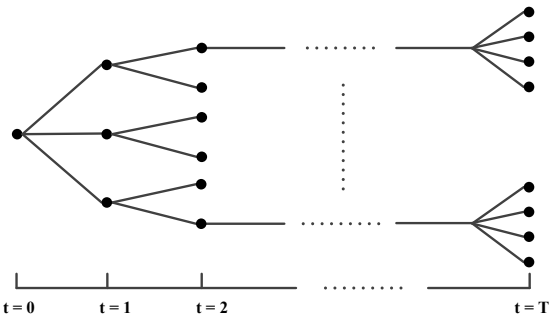


Fig. 12. An example of the multistage scenario tree: each leaf vertex represents a scenario, and each non-leaf vertex represents an intermediate state within the planning horizon. A probability is associated with each branch that represents the likelihood of state transition.

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 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:

$$\{\alpha_{i,v}, \beta_{i,v}, \chi_{i,v}\}, v \in S_j \quad (11)$$

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

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 $\mathcal{G}(0)$. 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 with regard to the scenario tree. The complete deterministic equivalent formulation of SRRP is given below:

$$\begin{aligned} \min \sum_{v \in \mathcal{V}} & p_v \cdot (C_f^+(\tau(v)) \cdot \Phi_i \cdot \alpha_{i,v} + (C_s(\tau(v)) + \\ & C_{io}(\tau(v))) \cdot \beta_{i,v} + C_f^-(\tau(v)) \cdot \mathrm{D}(i, \tau(v)) + \\ & C_p(i, \tau(v)) \cdot \chi_{i,v}) \end{aligned} \quad (12)$$

s.t.

$$\beta_{i,\pi(v)} + \alpha_{i,v} - \beta_{i,v} = \mathrm{D}(i,\tau(v)), \quad i \in \mathcal{I}, v \in \mathcal{V} \quad (13)$$
$$\mathrm{P}(i) \cdot \alpha_{i,v} \leq \mathrm{Q}(i,v), \quad i \in \mathcal{I}, v \in \mathcal{V} \quad (14)$$
$$\alpha_{i,v} \leq B \cdot \chi_{i,v}, \quad i \in \mathcal{I}, v \in \mathcal{V} \quad (15)$$
$$\beta_{i,0} = \varepsilon, \quad i \in \mathcal{I} \quad (16)$$
$$\alpha_{i,v}, \beta_{i,v} \in \mathbb{R}_+, \quad i \in \mathcal{I}, v \in \mathcal{V} \quad (17)$$
$$\chi_{i,v} \in \{0,1\}, \quad i \in \mathcal{I}, v \in \mathcal{V} \quad (18)$$

4.2.5 Polynomial-Time Algorithms for Stochastic Pricing Resource Rental Planning

Since variables at each $t \in \mathcal{T}$ are associated with a number of possible realizations, solving SRRP is equivalent to solving a large-scale MILP. There exists a number of standard techniques to solve this problem, for example, using Benders decomposition [41]. However, due to the huge search space for optimization, they are only suitable for performing short-term resource rental decisions. For example, we used CPLEX to solve SRRP for a 6-hour planning horizon on a machine with Intel Core i5-650 (4M Cache, 3.20GHz) and 4GB memory. It took a few minutes to generate the solution. When more scenarios were incorporated, the computational time became unbearable for practical use.

Fortunately, a number of polynomial algorithms were proposed to solve the stochastic lot-sizing problem (SLSP) in manufacturing and operations research. Among them, the most notable solutions are a branch-and-cut algorithm developed by Guan et al. [42], a cubic-time dynamic programming algorithm developed by Huang and Küçükyavuz [43], and later improved to quadratic time by Jiang and Guan [44]. All of them are polynomial-time algorithms with respect to the number of nodes on the scenario tree. In this section, we demonstrate a polynomial time algorithm [44] using the dynamic programming technique to solve the stochastic pricing resource rental planning problem.

In order to develop a polynomial-time algorithm for SRRP, we need to take care of a few things. First, the general formulation of SLSP includes multiple stochastic variables, e.g., demand, production and inventory cost, and order lead time. In our case, the service demand is known, and the order lead time represents the latency from finishing the boot-up of the computational instance to running of the application. Therefore, our formulation can be treated as a special instance of SLSP with deterministic demand and zero lead time. This greatly simplify the calculation. Second, given a positive initial storage (constraint 16), we can construct an equivalent problem without initial inventory by using ε to satisfy the demands emerged in the 1st, 2nd, ..., and n-th stages until it is depleted. Finally, note that DRRP is a special case of SRPP with leaf node number equal to one. A relaxation of the Wagner-Within property [45], called the semi-Wagner-Within property, becomes the key to develop polynomial-time algorithm for SRRP. The semi-Wagner-Within property is stated as follows.

Proposition 1 (Semi-Wagner-Within Property [43])**.** There exists an optimal solution (α', β', χ') of SRRP such that if $\alpha'_{i,v} > 0$ for $v \in \mathcal{V}$, then

$$\beta'_{i,\pi(v)} + \alpha'_{i,v} = D(i,\tau(m)) - \mathrm{D}(i,\tau(\pi(v)))$$

for some $m \in \mathcal{G}(v)$.

Based on this proposition, we define an optimal valuation function $f(v,w)$ for each node $v \in \mathcal{V}$, such that the total computations completed in nodes from root to v equal to the service demand from root to w. We use $g(v)$ to represent the feasible set of w for $f(v,w)$. Before we establish the backward formula for dynamic programming, we also need to establish a few sets for notational brevity. In particular, let $\Gamma(v)$ be the set of v's child, and $H(v)$ be the set of nodes whose service demands can be satisfied by computations conducted in v, and finally, we define $\Upsilon(v)$ such that $\Upsilon(v) = H(v) \setminus \cup_{x\in\Gamma(v)} H(x)$. Moreover, we use $\varrho(v)$ to represent the path from root to v, and $D_{0,v} = \sum_{i\in\varrho(v)} D_i$ to represent the cumulative service demands from the root node to each node in the scenario tree.

The binary variable, $\chi_{i,v}$, gives us two choices for each $v \in \mathcal{V}$. Therefore, If we use $f_0(v,w)$ to represent the valuation function when bid fails, and use $f_1(v,w)$ to represent the case when bid succeeds, we can derive $f(v,w)$ as follows.

$$f(v,w) = \min\{f_0(v,w), f_1(v,w)\}$$

If bid fails, the value function $f_0(v,w)$ includes: (1) storage and network transfer costs incurred to nodes in $\Upsilon(v)$ triggered associated with prior computation activities, and (2) costs for v's descendants.

$$f_0(v,w) = \sum_{k\in\Upsilon(v)} (C_s(\tau(k)) + C_{io}(\tau(k))\cdot(D_{0,w} - D_{0,k}) + \sum_{k\in\Gamma(v)} f(k,w)$$

Similarly, when bid succeeds, the value function $f_1(v,w)$ includes three components: (1) computational costs on node v, (2) storage and network transfer costs incurred to nodes in $\Upsilon(v)$ associated with prior computation activities, (3) costs for v's descendants.

$$f_1(v,w) = C_p(i,\tau(v)) + \min_{i\in H(v):i>w}\{C_f^+(\tau(v))\cdot\Phi_i \cdot(D_{0,j} - D_{0,w}) + \sum_{k\in\Upsilon(v)} (C_s(\tau(k)) + C_{io}(\tau(k))\cdot(D_{0,j} - D_{0,k}) + \sum_{k\in\Gamma(v)} f(k,j)\}$$

Given the backward recursion formulation for SRRP, we briefly demonstrate a dynamic programming algorithm runs in $\mathrm{O}(N^2)$ time [44].

Initialization Phase

1) For each node, construct sets $\Gamma(v)$, $H(v)$ and $\Upsilon(v)$ following breadth-first order.
2) For each node v, set $\eta(v,j) = 1$ if $j \in H(v), j > w$, and 0 otherwise.
3) Initialize $\sum_{k\in\Gamma(v)} f(k,j)$ for each pair of v and j, if $\eta(v,j) = 1$, initialize to 0. Otherwise set to $+\infty$.
4) Initialize $\sum_{k\in\Upsilon(v)}(C_s(\tau(k))+C_{io}(\tau(k))\cdot(D_{0,j}-D_{0,k})$ for each pair of v and j similarly to step 3).

Backward Induction Phase

1) For each node w in the scenario tree, calculate and store $f_0(v,w)$. When the calculation is done for every $k \in \Gamma(v)$, we obtain the result for $\sum_{k\in\Gamma(v)} f(k,w)$ for each w.
2) For each node w in the scenario tree, calculate and store $f_1(v,w)$. Again, this is achieved through backward induction bottom-up.
3) Calculate and store $f(v,w)$ for every pair of v and w.

The optimal solution is given by the result of the valuation function of the root node. For the computational effectiveness, readers can also refer to [42] for a detailed computational experiments that comparing the branch-and-cut algorithm with the default CPLEX MILP solver.

4.2.6 Evaluation of Stochastic Pricing Resource Rental Planning

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 Section 3.5. 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 Section 4.1.2, paragraph 3. The results are plotted in Figure 13. Here, we use the prediction values obtained from the approach described in Section 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 labelled as "stochastic planning", and the cost of solving its DRRP counterpart and the cost of using on-demand VMs are labelled 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 VM 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 predictive planning. The results shown on Figure 13 demonstrate that stochastic planning has slightly better advantage in terms of cost saving. In addition, compared to fixed price bidding, another advantage of the stochastic planning approach lies in reliability with regard to probability information generated by the multistage scenario tree.

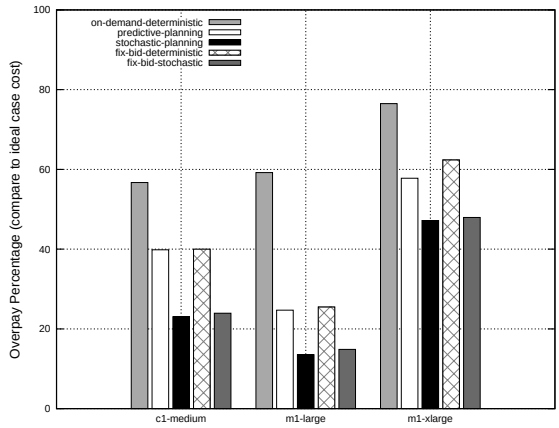


Fig. 13. Cost comparison of predictive and stochastic planning

Next, we investigate the impact of bid price approximation precision to the stochastic planning approach with regard to cost reduction for VM type c1.medium. This evaluation is necessary because according to Section 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%[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 against the baseline cost are plotted in Figure 14. 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 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,

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

the percentage error introduced by approximation is generally acceptable.

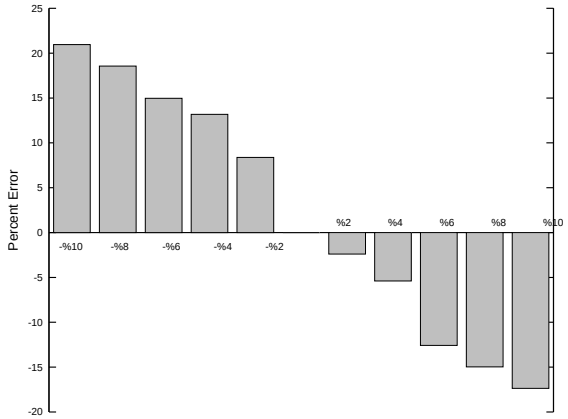


Fig. 14. Impact of approximation precision for SRRP solution

5 Further Discussion

Both the deterministic and the stochastic models provide good estimates of how resources shall be utilized over time for cloud ASPs. However, they come with non-trivial costs. As presented in Section 4.2.5, solving the optimization problem represented by the scenario tree is heavy-weight and takes up tremendous amount of computing resources. For DRRP, although calculating the relaxed form at each step can be done in polynomial time using the Branch-and-Bound technique, the total number of nodes to be evaluated can become exponential to the number of integer variables. For SRRP, we described an $O(N^2)$ algorithm where N is the number of nodes on the scenario tree. Since each decision point branches out multiple future realizations with certain probabilities, the number of nodes grow exponentially as time progresses. In order to make real-time scheduling decisions, it is critical to offload the computation to a separate scheduling module so as to avoid the interference with the cost model. In addition, one can parallelize the searching over the optimization domain using heterogeneous computing units [46], and employ an approximation algorithm to speed up the computation [47].

Another interesting issue is to extend SRRP with other stochastic variables, e.g., the service demand from end users. In practice, the demand often follows some form of distribution which can be summarized over time. This paper presented a simplified model that focused on the resource access uncertainties in the cloud market. However, an optimization model with multiple stochastic variables is more accurate to model realistic scheduling scenario where the optimal solution is driven by multiple factors.

6 Conclusion

In this paper, we investigated the problem of fine-grained resource rental planning in a cloud environment, and developed solutions for both deterministic and stochastic resource pricing settings. Our optimization models were based on a thorough rental cost analysis of elastic application deployment in cloud. When resource pricing is fixed, we observed that the cost tradeoff between computing and storage emerges in time-slotted resource provision scheduling. Based on this observation, we formulated a deterministic optimization model that effectively minimizes rental cost of VMs while covering customer demand over certain planning horizon. In addition, we took one step further to analyze the predictability of spot resource prices using Amazon's spot instance price trace, and proposed an alternative 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 demonstrated the advantage of the stochastic optimization approach over the predictive approach in rental cost reduction. We also studied the impact of various parameter settings on the performance of both models. We believe the proposed fined-grained approaches offer effective means for resource rental planning in practice.

Acknowledgments

The authors would like to thank Ruiwei Jiang and Dr. Yongpei Guan from the Computational Stochastic Optimization Lab at the University of Florida for their assistance and helpful comments to the dynamic programming algorithm presented in Section 4.2.5.

References

[1] "AWS Case Studies," Available: http://aws.amazon.com/solutions/case-studies/.
[2] "Market Trends: Platform as a Service, Worldwide, 2012-2016, 2H12 Update," ID: G00239236, 5 October 2012.
[3] 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.
[4] R. Buyya, R. Ranjan, and R. Calheiros, "Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services," in *Algorithms and Architectures for Parallel Processing*, ser. Lecture Notes in Computer Science, 2010, vol. 6081, pp. 13–31.
[5] G. Feng, S. Garg, R. Buyya, and W. Li, "Revenue maximization using adaptive resource provisioning in cloud computing environments," in *Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing (GRID'12)*, 2012, pp. 192–200.
[6] S. K. Garg, C. Vecchiola, and R. Buyya, "Mandi: A market exchange for trading utility and cloud computing services," *J. Supercomput.*, vol. 64, no. 3, pp. 1153–1174, 2013.
[7] H. Zhao, Z. Yu, S. Tiwari, X. Mao, K. Lee, D. Wolinsky, X. Li, and R. Figueiredo, "Cloudbay: Enabling an online resource market place for open clouds," in *Proceedings of the 2012 IEEE/ACM Fifth International Conference on Utility and Cloud Computing (UCC'12)*, 2012, pp. 135–142.
[8] S. Zaman and D. Grosu, "A combinatorial auction-based mechanism for dynamic vm provisioning and allocation in clouds," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 129–141, July 2013.
[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] 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.

[11] B. Sharma, R. K. Thulasiram, P. Thulasiraman, S. K. Garg, and R. Buyya, "Pricing cloud compute commodities: A novel financial economic model," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'12)*, 2012, pp. 451–457.

[12] H. Jin, X. Wang, S. Wu, S. Di, and X. Shi, "Towards optimized fine-grained pricing of iaas cloud platform," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, 2014.

[13] 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.

[14] 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.

[15] N. Samaan, "A novel economic sharing model in a federation of selfish cloud providers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, pp. 12–21, 2014.

[16] P. Sharma, S. Lee, T. Guo, D. Irwin, and P. Shenoy, "Spotcheck: Designing a derivative iaas cloud on the spot market," in *Proceedings of the Tenth European Conference on Computer Systems (EuroSys'15)*, 2015, pp. 16:1–16:15.

[17] W. Wang, D. Niu, B. Liang, and B. Li, "Dynamic cloud instance acquisition via iaas cloud brokerage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1580–1593, June 2015.

[18] R.-H. Hwang, C.-N. Lee, Y.-R. Chen, and D.-J. Zhang-Jian, "Cost optimization of elasticity cloud resource subscription policy," *IEEE Transactions on Services Computing*, vol. 7, no. 4, pp. 561–574, Oct 2014.

[19] M. Yao and C. Lin, "An online mechanism for dynamic instance allocation in reserved instance marketplace," in *23rd International Conference on Computer Communication and Networks (ICCCN'14)*, Aug 2014, pp. 1–8.

[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] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing amazon ec2 spot instance pricing," in *IEEE Third International Conference on Cloud Computing Technology and Science (CloudCom 2011)*, 2011, pp. 304–311.

[22] B. Javadi, R. K. Thulasiram, and R. Buyya, "Characterizing spot price dynamics in public cloud environments," *Future Gener. Comput. Syst.*, vol. 29, no. 4, pp. 988–999, 2013.

[23] 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.

[24] Y. Song, M. Zafer, and K.-W. Lee, "Optimal bidding in spot instance market," in *IEEE INFOCOM 2012*, 2012, pp. 190–198.

[25] H. Xu and B. Li, "Dynamic cloud pricing for revenue maximization," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 158–171, July 2013.

[26] 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.

[27] Z. Ou, H. Zhuang, A. Lukyanenko, J. Nurminen, P. Hui, V. Mazalov, and A. Yla-Jaaski, "Is the same instance type created equal? exploiting heterogeneity of public clouds," *IEEE Transactions on Cloud Computing*, vol. 1, no. 2, pp. 201–214, July 2013.

[28] B. Farley, A. Juels, V. Varadarajan, T. Ristenpart, K. D. Bowers, and M. M. Swift, "More for your money: Exploiting performance heterogeneity in public clouds," in *Proceedings of the Third ACM Symposium on Cloud Computing (SoCC '12)*, 2012, pp. 1–14.

[29] M. Mao and M. Humphrey, "A performance study on the vm startup time in the cloud," in *IEEE 5th International Conference on Cloud Computing (CLOUD 2012)*, 2012, pp. 423–430.

[30] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: theory, algorithms, and applications*. Prentice-Hall, Inc., 1993.

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

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

[33] 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.

[34] 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.

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

[36] 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.

[37] "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/.

[38] H. Zhao, M. Pan, X. Liu, X. Li, and Y. Fang, "Optimal resource rental planning for elastic applications in cloud market," in *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium (IPDPS '12)*, 2012, pp. 808–819.

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

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

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

[42] 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.

[43] K. Huang and S. Küçükyavuz, "On stochastic lot-sizing problems with random lead times," *Operations Research Letters*, vol. 36, no. 3, pp. 303 – 308, 2008.

[44] R. Jiang and Y. Guan, "An o(n^2)-time algorithm for the stochastic uncapacitated lot-sizing problem with random lead times," *Oper. Res. Lett.*, vol. 39, no. 1, pp. 74–77, 2011.

[45] H. M. Wagner and T. M. Whitin, "Dynamic version of the economic lot size model," *Manage. Sci.*, vol. 50, pp. 1770–1774, dec 2004.

[46] A. Boukedjar, M. E. Lalami, and D. El-Baz, "Parallel branch and bound on a cpu-gpu system," in *Proceedings of the 2012 20th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP '12)*, 2012, pp. 392–398.

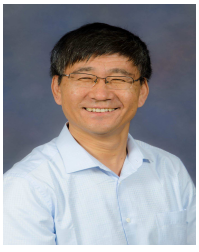
[47] R. Levi and C. Shi, "Approximation algorithms for the stochastic lot-sizing problem with order lead times," *Operations Research*, vol. 61, no. 3, pp. 593–602, 2013.

Han Zhao (S'07-M'14) is a senior research engineer working at Qualcomm Research Silicon Valley. He received his Ph.D. degree (Computer Science) from the Computer & Information Science & Engineering at University of Florida, the M.Sc. degree (Computer Science) from Oklahoma State University, and the B.E. degree (Software Engineering) from Jilin University, China. His research interests include parallel and heterogeneous computing on mobile platforms, and resource management and scheduling in distributed systems.

Miao Pan (S'07-M'12) received his BSc degree in Electrical Engineering from Dalian University of Technology, China, in 2004, MASc degree in electrical and computer engineering from Beijing University of Posts and Telecommunications, China, in 2007 and Ph.D. degree in Electrical and Computer Engineering from the University of Florida in 2012, respectively. He is now an Assistant Professor in the Department of Computer Science at Texas Southern University. His research interests include cognitive radio networks, cyber-physical systems, and cybersecurity.

Xinxin Liu is currently a Privacy Engineer at Google Inc. She received her Ph.D. from Computer & Information Science & Engineer Department at University of Florida, and her M.S. degree from Computer Science Department at Oklahoma State University. Her research interests include mobile networking, and privacy preserving techniques.

Xiaolin Li is an associate professor in Department of Electrical and Computer Engineering (ECE), University of Florida (UF). He received a PhD degree in Computer Engineering from Rutgers University. His research interests include Parallel and Distributed Systems (PDS), Cyber-Physical Systems (CPS), and Security & Privacy (S&P). He is directing the Scalable Software Systems Laboratory (S3Lab). He is a recipient of National Science Foundation CAREER Award in 2010, Internet2 Innovative Application Award in 2013, and best paper awards in ACM CAC 2013 and IEEE UbiSafe 2007.



Yuguang "Michael" Fang (F'2008) received BS/MS degree from Qufu Normal University, China, in 1987, and Ph.D. degrees from both Case Western Reserve University and Boston University in 1994 and 1997, respectively. He joined the Department of Electrical and Computer Engineering at University of Florida since 2000. Dr. Fang received the US NSF Faculty Early Career Award in 2001 and the US ONR Young Investigator Award in 2002, and is a recipient of the Best Paper Award in IEEE International Conference on Network Protocols in 2006. He also received a 2010-2011 UF Doctoral Dissertation Advisor/Mentoring Award and IEEE Communications Society WTC Recognition Award. He served as the Editor-in-Chief of IEEE Wireless Communications and is currently serving as the Editor-in-Chief of IEEE Transactions on Vehicular Technology. He is a Fellow of IEEE.