Deadline-constrained Stochastic Optimization of Resource Provisioning, for Cloud Users

Masoumeh Tajvidi[1], Daryl Essam[1] and Michael J. Maher[2]
[1]*School of Engineering and Information Technology, UNSW, Canberra, Australia*
[2]*Reasoning Research Institute, Australia*

Keywords: Cloud Computing, Stochastic Optimization and Scheduling, Multi-stage Stochastic Programming, Deadline, Google Trace Data.

Abstract: Acquiring computational resources dynamically, in response to demand, and only paying for the resources used, is the main benefit cloud computing may bring for cloud customers. However, this benefit can only be realized when customers can determine the right size of the resources required and allocate such resources in a cost-effective way. While resource over-provisioning can cost users more than necessary, resource under provisioning hurts application performance. To leverage the potential of clouds, a major concern, hence is optimizing the monetary cost spent in using cloud resources while ensuring the quality of service (QoS) and meeting deadlines. Unfortunately, there is still a lack of a good understanding of such cost optimization. The resource provisioning, from the cloud-user perspective, is a complicated optimization problem that consists of much uncertainty, as well as heterogeneity in its parameters. The variety of pricing plans further complicates this problem. There has been little work on solving this problem as it is in the real world and from the end users view. Most works relax the problem by not considering the dynamicity or heterogeneity of the environment. The aim of this paper, however, is optimizing the operational cost whilst guaranteeing performance and meeting deadline constraints, by taking into account parameters' uncertainty and heterogeneity, as well as considering all three available pricing plans, i.e. on-demand, reservation, and spot pricing. The experimental implementation using a real cloud workload shows that, however the proposed model has not the perfect foresight of future; results are very close or in many cases similar to, the full knowledge model. We also analyse the results of various users with different workload pattern, based on a k-means clustering.

1 INTRODUCTION

The key advantage of cloud computing is the dynamic scalability of resources, because of its pay-as-you-go model. Cloud computing providers rely on virtualization techniques to manage the dynamic nature of their infrastructure. Virtualization technologies help cloud providers pack their resources into different types of virtual machines (VMs) with different configurations to satisfy the computing resource needs of a wide variety of application types. Table 1 illustrates a number of VM types and prices available at Amazon Elastic Compute Cloud service (Amazon EC2, 2017). Cloud computing users must use this information to determine the appropriate subset of resource configurations that could run an application cost effectively, while also meeting the Quality of Service (QoS) goals, such as performance. Therefore, the cost effectiveness of cloud computing highly depends on how well a customer can optimize the cost of renting resources from cloud providers.

Table 1: Amazon EC2 instance types, and hourly prices.

VM Type	CPU	Memory	Hourly price($) On-demand	Reservation
t2.micro	1 EC2 Compute Unit	1 GB	0.012	0.008
t2.small	1 EC2 Compute Unit	2 GB	0.023	0.017
c4.large	2 EC2 Compute Unit	3.75 GB	0.1	0.063
c4.xlarge	4 EC2 Compute Unit	7.5 GB	0.19	0.126
c4.2xlarge	8 EC2 Compute Unit	15 GB	0.39	0.252

With regards to pricing, three pricing plans have been introduced for VMs: on-demand, reservation, and spot pricing. On-demand is offered by all cloud providers, in which the user is only charged for the time the VM is running. Reserved VM is provided by only some (big) cloud providers that establish a long-term commitment between the user and the provider with a significant discounted price. On the other hand, spot is a new pricing scheme introduced by few

Tajvidi, M., Essam, D. and Maher, M.
Deadline-constrained Stochastic Optimization of Resource Provisioning, for Cloud Users.
DOI: 10.5220/0006761401790189
In *Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER 2018)*, pages 179-189
ISBN: 978-989-758-295-0

providers and brings more cost saving for the end-users by enabling them to bid on unused instances. However, the instances may be terminated by the cloud provider if their prices increases above the bidding price, which makes this pricing plan unreliable. Choosing the right number of VM with appropriate type and pricing plan is a barrier the end user faces for renting cloud resources. Since future workload is often not known a priori and the on-demand and spot VM prices vary over time, this problem cannot be simply solved by deterministic approaches.

Although the resource provisioning problem can be viewed from different perspectives, like the Infrastructure as a Service (IaaS) provider, the Software as a Service (SaaS) provider, and the cloud end-user view. There is little attention from the end users view, so we focus on this problem from the end-user perspective, and tackle the main issues and complexity a user faces during the resource selection phase.

The main contributions of our research are: firstly, modeling the problem such that deals effectively with the uncertainty of demand and price. Secondly, considering the heterogeneity of pricing plans and VM types. Thirdly, our previous proposed work (Tajvidi et al., 2017) is extended by an enhanced model, which optimizes cost and satisfy the deadline constraints, at the same time. unreliability of spot could hurt the application performance by not finishing the tasks on time; therefore, the optimization model is improved to deal with both cost and deadline, simultaneously. Another improvement we made in this extended model is supporting more hourly time steps (696) ,rather than having only one time step to capture the whole year data (Tajvidi et al., 2017), in order to get more accurate results. More details such as a more realistic reservation pricing plan are also introduced in our current work. Finally, to make the evaluation more valid and reasonable, a real cloud workload demand, Google Cluster Trace (John Wilkes, 2011), is applied in the experiments.

We model the problem as a stochastic cost optimization problem that does not allow the execution time of a particular application goes beyond its specified deadline. We divide it into two phases, each divided into 696 time steps. In the first phase, the number of reserved VMs is determined and in the second phase number of spot and on-demand VMs, based on all constraints, is specified. Our experimental results show that our model outperforms the state of the arts by 20%. Comparing the proposed model with a similar model that has the perfect knowledge of future, we observe that our solutions are close to, and often exactly the same as, the solutions that are based on perfect foresight. To understand user-specific jobs results, we perform K-mean clustering on users attributes of Google trace (John Wilkes, 2011), and find out that different workload patterns affect the final results.

The rest of this paper is organized as follows. In section II we provide an insight into the problem description and model overview, followed by formulation of the problem in section III. Then in IV the model implementation is discussed. The experiments and the experimental results are reported in section V. A brief survey of related work is provided in VI, and finally, in section VII, conclusions is stated.

2 PROBLEM DESCRIPTION

The cloud end-user needs to rent a combination of cost-effective VMs available by cloud providers and various pricing schemes, for a specific period. Since they are not aware of the exact price of spot and on-demand VMs as well as their application workload demand in advance, they need to solve an uncertain optimization problem. The problem is to find the optimal number of reserved VMs for the whole problem time period and the optimal number of on-demand and spot VMs in each time slot (every hour) to fulfill the requested demand and tasks' deadline.

2.1 Pricing Plans

Cloud Providers generally offer various types of VMs with different pricing plans. Amazon is one of the dominant providers that support all three pricing plans of reservation, on-demand, and spot. In experimental evaluation, we use Amazon EC2 VMs data (pricing and configuration). We explain more details in the following sections.

Standard reservation pricing refers to the advance reservation of resources for a specific time, while securing a lower usage charge (up to 75% discount over on-demand instance pricing). It offers consumers three purchasing variants, "all upfront", "partial upfront", and "no upfront" to purchase reserved instances. With the all-upfront variant, users pay for the entire reserved instance with one upfront payment. This variant provides the largest discount. With the partial-upfront variant, users make a low upfront payment and are then charged a monthly rate for their instances, even for instances that are not utilized in this period. The no-upfront variant does not require any upfront payment and provides a monthly rate for the duration of the term.

Convertible reservation pricing, on the other hand is a new reserved instance type recently introduced

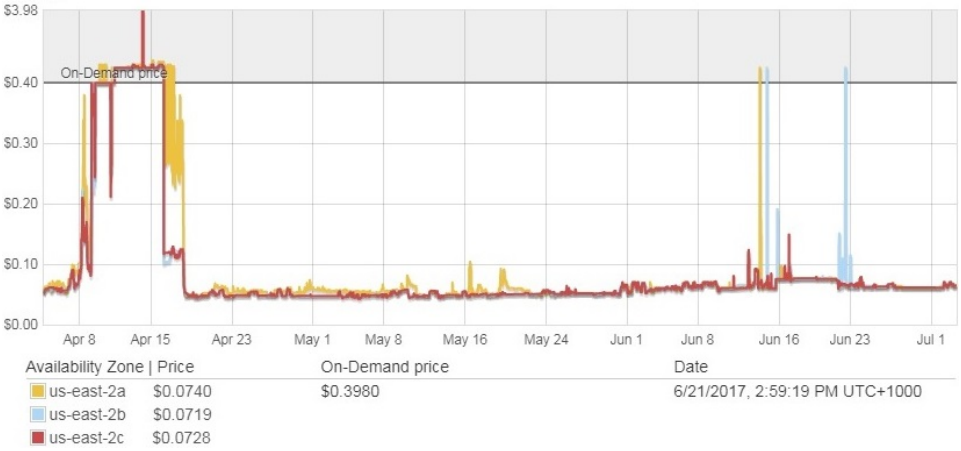


Figure 1: Amazon EC2 spot instance pricing history; c4.2xlarge (us-east regions, Linux/Unix).

by Amazon. It provides customers with additional flexibility for still a very significant discount (around 45% discount over on-demand instance pricing), that can be purchased for a 3-year term. Customers have the option at any time to change the instance family, OS, or tenancy associated with their reserved instance. However, we use standard reservation pricing in our model, since our implementation is running for less than 3 years.

On-Demand pricing lets customers pay for compute capacity by the hour, with no long-term commitments or upfront payments. Depending on the demand of their application, users can simply increase or decrease their compute capacity and only pay for the specified hourly rate for the instances used. Although this pricing model provides convenient flexibility and reliability, it charges customers higher rates than other plans. The on-demand price is not a fixed price and the cloud provider can change it at any time.

Spot pricing enables users to bid for unused Amazon EC2 capacity. This price fluctuates periodically, depending on the supply and demand for spot instances. To acquire spot instances, the users place a spot request, specifying the instance type and the maximum price they are willing to pay per hour per instance. If the customer's bid price meets or exceeds the current spot price, the requested instances are granted and they will run until either the user chooses to terminate them or the spot price increases above the maximum bid price. In the latter case, the instances are terminated by the cloud providers with 2 minutes notice. The actual price users pay for their instances is the spot market price, regardless of their bid price. See Fig 1 for spot price history of a typical VM type. Due to the uncertain availability of spot instances, and the potential interruptions they may bring, the spot instance plan is not reliable and is only practical for fault tolerant applications. In other words, application's downtime or even failure should not adversely impact the operations.

When it comes to spot instances, a big challenge is choosing a good bidding strategy. There are various strategies proposed in the literature (Tang et al., 2012), but generally one can bid high as a means of ensuring to obtain instances with less volatility or bid lower to optimize costs and send any overflow to on-demand or reserved instances. The most common strategy, however, is to bid on-demand price, called "always bidding on-demand price"(AO). With this strategy, customers ensure that they will get a discount over on-demand; in addition, they have a lower chance to be interrupted. The main motivation of this strategy is that if the current spot price is lower than the bid price, customers will be charged the current spot price regardless of their bid. The AO strategy guarantees 1) minimum completion time because the spot price rarely goes beyond on-demand price and 2) being at most 10 percent more than the spot minimum cost (Tang et al., 2014). In our model, we use this simple and effective bidding strategy.

2.2 Model Overview

Having uncertainties in our model, one of the most appropriate techniques to solve it, is stochastic programming (Birge and Louveaux, 2011). In this approach, uncertainty is usually characterized by a probability distribution on the parameters. In practice, it can range in detail from a few scenarios (possible outcomes of the data) to specific probability distributions (Shapiro and Philpott, 2007). The general idea is to divide the problem into at least two stages. In the first stage, a decision is made and the expected cost is optimized, then in the next stage, the consequences of that decision is compensated by a new decision, or the recourse function.

This problem is divided into two phases. In the first phase, we generate some scenarios for the uncertain cost and workload demand to find the optimal number of reserved VMs for the reservation period (the reservation period is one year). The scenarios are generated based on a prediction of the actual workload. Then, in the second phase, which is also called the rolling phase, the actual prices and workload demand become known. So the optimization model run every hour (the billing period of on-demand and spot instances is calculated hourly by the cloud provider) to determine the optimal number of on-demand and spot VM in that time slot. The aim is to minimize the operational cost while the execution time is the shortest possible time. The main constraints of this optimization problem are firstly being capable of giving enough performance to serve the load for each time slot, and not exceeding the deadline specified by the user to complete a task.

Spot instances are not reliable resources, consequently, if a VM terminate, an extra hour (at most) is required to complete a task. Therefore the execution time may extend more than expected if no arrangement for such situations is introduced. In order not to hurt the reliability, the task's deadline is considered as a constraint. Such that the tasks with deadline length of 1 hour or less cannot be assigned to spot VMs. Also, we introduce a limitation that if a task terminates once, spot VMs can no longer be allocated to it because we are trying to greedily minimize execution time with minimum interruption of tasks.

3 PROBLEM FORMULATION

Both the first and second phase of the model have common parameters and constraints, the only difference is the decision variables. The decision variables are the number of each VM type provisioned under different purchasing variants and pricing plans. See Table 2 for notation. The decision variable x_{ik}^R is the number of reserved VM type i, subscribed to purchasing variant k in the first stage, while x_{ik}^O denotes the number of operating VM type i with purchasing variant k. However, for the second phase model, x_{ik}^R converts to a parameter instead of a variable, and its value is assigned by the first phase outcome.

Also decision variables x_i^D and x_i^S, respectively, are the number of on-demand and spot VMs of type i in the second phase. We have three provisioning costs, formulated as follows:

- The total Reservation Cost, or the upfront cost of reserving resources, where c_{ik}^R is the price of VM type i with purchasing variant k:

$$C^R = \sum_i \sum_k x_{ik}^R c_{ik}^R \tag{1}$$

- The total On-demand cost, where c_i^D is the price of VM type i:

$$C^D = \sum_i x_i^D c_i^D \tag{2}$$

- The total Spot cost, where c_i^S is the price of VM type i:

$$C^S = \sum_i x_i^S c_i^S \tag{3}$$

The objective function z is the total expected provisioning cost.

$$Min \;\; z = \sum_i \sum_k c_{ik}^R . x_{ik}^R + IE_\Omega[\Phi(x_{ik}^R, \omega)] \tag{4}$$

subject to:

$$x_{ik}^R \in \mathbb{N} \tag{5}$$

where $IE_\Omega[\Phi(x_{ik}^R, \omega)]$ is the expected cost under uncertainty Ω, which is a combination of all scenarios. Φ is the recourse optimization problem, the objective of $\Phi(x_{ik}^R, \omega)$ is to minimize the cost under uncertainty given scenario ω:

$$Minimize \;\; [\sum_\omega (C^D + C^S)] + C^R \tag{6}$$

subject to ($\forall\ \omega$) :

$$z \leq MaxBudget \tag{7}$$

$$TotalCPU \geq \sum_t Req_t^{CPU} \tag{8}$$

$$TotalMemory \geq \sum_t Req_t^{Memory} \tag{9}$$

$$NonSpotCPU \geq \sum_{t:L_t \leq 1} Req_t^{CPU} + \sum_t UrgentReq_t^{CPU} \tag{10}$$

$$NonSpotMemory \geq \sum_{t:L_t \leq 1} Req_t^{Memory} + \sum_t UrgentReq_t^{Memory} \tag{11}$$

$$x_{ik}^O \leq x_{ik}^R \tag{12}$$

The first constraint (7) states that the total provisioning cost, or the objective function z, cannot be greater than the maximum budget the customer specified for running their application.

Constraints (8) and (9) both ensure that the amount of required resources for all tasks is satisfied by the VMs acquired in the second stage. The instance types comprise varying combinations of CPU

Table 2: Notation of the problem.

Symbol	Description
I	set of VM Types
R	set of VM resources or features; CPU and Memory
T	set of Tasks
Cap_i^{CPU}	CPU Capacity of VM type i
Cap_i^{Memory}	Memory Capacity of VM type i
Req_t^{Memory}	Amount of memory required for completing task t
Req_t^{CPU}	Amount of CPU required for completing task t
$UrgentReq_t^{Memory}$	Amount of memory required for completing task t where t is terminated in the previous stage
$UrgentReq_t^{CPU}$	Amount of CPU required for completing task t where t is terminated in the previous stage
L_t	time (hour) required for completing task t
K	Set of reservation purchasing variants, all-, partial-, or no-upfront
$MaxBudget$	User's maximum budget
c_{ik}^R	Reservation Cost of VM type i subscribed purchasing variant k
x_{ik}^R	Number of reserved Vms type i subscribed purchasing variant k
x_{ik}^O	Number of Operation VM type i, subscribed purchasing variant k
c_i^D	On-demand Cost of VM type i
x_i^D	Number of on-demand VM type i
c_i^S	Spot Cost of VM type i
x_i^S	Number of Spot VM type i

and memory, and give the customer the flexibility to choose the appropriate mix of resources for their applications. Therefore, each task can be run on multiple VMs simultaneously, just as each VM can host multiple tasks of a particular application. $TotalCPU$ and $TotalMemory$ are the available CPU and Memory in stage i, and are defined as follows:

$$TotalCPU = \sum_{ki} x_{ik}^O Cap_i^{CPU} + \sum_i (x_i^D + x_i^S) Cap_i^{CPU} \quad (13)$$

$NonSpotCPU$ and $NonSpotMemory$ are the CPU and Memory of available reserved and on-demand VMs:

$$NonSpotCPU = \sum_{ki} x_{ik}^O Cap_i^{CPU} + \sum_i (x_i^D) Cap_i^{CPU} \quad (14)$$

Number of reserved and on-demand VMs should be enough to fulfill the demand of two type of jobs, constraints (10) and (11). First, jobs that have been terminated in the previous stage, their requirement is defined by $UrgentReq_t^{Memory}$ and $UrgentReq_t^{CPU}$. Second, tasks that are submitted in the current stage, but their deadline time is less than an hour, as spot termination can hurt the application's performance.

The last constraint (12), limits the number of operating reserved VMs (x_{ik}^O) of each type to be less than or equal to the number of reserved VMs (x_{ik}^R). As discussed earlier, the customer reserves a number of VMs in the first stage and pays an upfront fee for them, then in the second stage, these reserved VMs can be used by the customer. So the number of used VMs (operating VMs) in the next phase cannot be greater than the number of reserved VMs in the first phase.

4 MODEL IMPLEMENTATION

4.1 Data Set

Google cluster dataset (John Wilkes, 2011), released in 2011, is measured on a heterogeneous 7000-machine server cluster on a 29-day period involving 672,075 jobs and more than 48 million tasks. Workload demand arrives in the form of jobs. A job is comprised of one or more tasks, each of which is accompanied by a set of resource requirement. The dataset is partitioned into six families, namely, machine events, machine attributes, job event, task event, task usage, and task constraints. Our focus on this paper is on the task and job-related information from the job event and task event categories. The workload dimensions we take into account are task duration and deadline in hour, CPU usage in core, memory usage in Gigabyte, and the associated users to them. The duration of the tasks is calculated as the difference of the time when

it submitted and the time when it finishes the execution (Chen et al., 2014). The trace we utilize does not contain information about task deadlines. Thus, we assigned deadlines based on the time the task state change to "dead" (task completed normally, fails, or killed by the user or provider).

The Google Trace data has been obfuscated to hide exact machine configuration. The resource sizes have been linearly transformed (scaled) by dividing the largest capacity of the resource on any machine in the trace, both CPU and memory are normalized by the same constant value (Reiss et al., 2011). These normalized values are not themselves suitable for our model, firstly because the resource usage values are too small with multiple floating point values, which slows down the optimization execution time. Secondly, this little demand makes the solver to choose one or zero number of reserved VMs, that doesn't allow us to precisely find the correlation of number of VMs with other parameters. Since the absolute values are not available, a reasonable ratio is used. We multiply both CPU and memory by 32, based on an analysis of the largest VM size in Google and Amazon EC2.

In this dataset, each job is associated with a particular user, and since we are viewing this problem from the end-user perspective, we divide the data set into multiple users with different workload demands and attributes. User names are hashed and provided as an opaque base64-encoded string; therefore, we have no information about who the actual users are. Hence, we assign a number to each user to more easily identify them in our analysis. Users with zero task submission during the one-month period in question, are completely ignored in our analysis.

4.2 Minizinc Model

Our stochastic model is implemented in the MiniZinc modeling language (Nethercote et al., 2007), using the COIN-OR CBC solver. We model the problem in two separate MiniZinc models. The first model solves the problem of determining the number of reserved VM before the unknown parameters become known.

To represent the uncertainty of parameters, 20 workload scenarios are generated pseudo-randomly, such that the workload for each hour was randomly chosen from all 29 days at the same exact hour from Google cluster data (John Wilkes, 2011). Similarly, 20 cost scenarios for on-demand and spot prices were generated, based on the extracted data of April to June 2017, from Amazon EC2 official website (Amazon EC2, 2017). The 1-month (29 days) data was split into 696, 1-hour stages, and the scenarios contain the demand and price data for every stage. The reason the stage length is chosen as 1 hour, is because the minimum VM prices are calculated hourly by the cloud providers. The optimization model of the first phase minimizes the operational cost while fulfilling the performance of each task and meeting the deadline for task's execution time. The outcome of this model is hence the optimum number of reserved VMs, thereby allowing a user to guarantee resource availability in advance for an extended period (e.g. 1 year).

In the second phase (rolling phase), the real workload and real prices of on-demand and spot VMs become known. A single set of real workload demand from our Google data set (John Wilkes, 2011) is used with the VM prices of the prevalent IaaS cloud provider, Amazon EC2 (Amazon EC2, 2017) over May 2017. Based on these data and the determined number of reserved VMs from the first phase, the outcome of the second phase is the optimal number of spot and on-demand instances and the actual cost in every single stage, as well as the actual total cost over all stages.

The model contains some approximations. Firstly, not all instances of a particular type, in general, perform to exactly the same standards, because of variations in the physical hardware that is allocated for them and possible multi tenancy (Mao and Humphrey, 2012). We use the minimum guaranteed performance of EC2 instances as the baseline to ensure that sufficient resources are available for each job. Secondly, VM startup is not the same for all VMs and can affect the execution time, however, (Mao and Humphrey, 2012) shows that in EC2, the VM startup time is relatively constant across all instances, 100 seconds in average, where requesting a pool of VMs to start. Alternatively, in (Ali-Eldin et al., 2012) the start-up time for all VMs is considered to be less than 1 minute. In either case, because these times are a small fraction of the 1-hour decision steps, we ignore this factor. Finally, the data transferring cost is not included in our operational cost, because we use VMs from the same regions, and there is no data transfer charge between Amazon EC2 and other Amazon web services within the same regions (Amazon EC2, 2017).

5 EXPERIMENTS

We need to investigate if the proposed model enables end-users to create precise and cost effective provisioning for jobs running in the cloud. To do that, we will pursue and evaluate the following objectives. The

first objective is to show that the results we get in an uncertain environment are very close to the results if the model has the perfect knowledge of future. Second, we show that our model's results are better than other available options. Third, we cluster users and show that our model works better for users with specific workload patterns.

5.1 Experimental Settings

In this case study, we address two popular type of VMs, c4large and c4xlarge within the same region, US-east. The c4 instances are the latest generation of Compute-optimized instances, featuring the highest performing processors and the lowest price/compute performance in EC2 (Amazon EC2, 2017). However, our approach can easily be extended to further VM types.

The model runs 10 times for each user, and the average results are discussed in the next section. A Java program is automating the runs and the time it takes to complete each run, including (i) generating different scenarios of the price and demand for the first phase, (ii) running the first phase Minizinc model, (iii) repeating the rolling phase Minizinc model for 696 time, and (v) writing the output results in an appropriate file, is 5-15 minutes for each individual user.

5.2 Experimental Results

We repeated our experiment for three different options, that are various combinations of Amazon EC2 pricing plans. The options are:

- Reservation-OnDemand-Spot option: This is the main focus of this work. It considers all three pricing plans.
- Reservation-OnDemand (RO): Using only reservation and on-demand instances, a common trend for most related work (Díaz et al., 2017). Therefore, it provides us with a basis comparison with state-of-the-art.
- On-demand (O): The only pricing plan in this option is on-demand, the most expensive and flexible one.

For the ROS model, we consider two alternatives; one is the main model that consists of uncertainty and random scenarios, as explained in the previous section. Another one is an omniscient ROS, which has the full knowledge of the future, i.e. it is similar to the main ROS option, but instead of having multiple uncertain scenarios, its scenarios represent the actual price and workload demand. Therefore, it has an extraordinary advantage of knowing the exact future demand and price. Although it is not a realistic model, it can be used as an evaluation guidance tool that demonstrates the optimal number of VMs and cost.

Figure 2 shows the total cost comparison between the options for individual users. The highest total operational cost belongs to the O and RO options, while ROS and omniscient-ROS have the lowest total operational cost, and the difference between their results is either zero, or very small. The omniscient-ROS only gets around 1.5% better results on average, over all users, than ROS. Based on this comparison, two user categories can be recognized. One with the exact same results for ROS and omniscient-ROS and another with slight differences in the results. We name the first category, perfectly-fitted (around 63% of all users), and the second, imperfectly-fitted users (around 37% of all users). Table 4 provides more details. In the imperfectly-fitted category, the difference between ROS and omniscient-ROS ranges from 9% (e.g. user 51) to 0.09% (e.g. user 86), and is around 3%, on average.

This indicates that although ROS does not know the real future workload and makes a decision based on random scenarios, it is still a reliable model, since its results are very close to the optimal results.

More information can be seen in Table 3, which shows number of VMs, over-provisioned and under provisioned VMs, and total cost on average for all users. Number of over-provisioned is the average number of unused reserved VMs that are paid for but are idle in some stages. On the other hand, number of under-provisioned VMs is the sum of on-demand and spot VMs that are allocated, because the number of reserved instances is not enough to satisfy the workload demand.

Furthermore, ROS outperforms O and RO by about 50% and 20%, on average over all users. This indicates that choosing cloud providers with spot instances, beside on-demand and reservation, can make big cost savings for users. However for some users, this difference is very little or even zero.

By investigating users with a lower than average difference in RO and omniscient-ROS, we find that they have similar steady state workload patterns (Varia, 2012). For such workload patterns, the optimal number of reserved VMs covers all or most of the demand, rather than on-demand and spot, and because these two flexible pricing plans are not involved, we see little difference in RO and omniscient-ROS options. However, no uniform workload pattern for users with differences above the average has been identified.

A similar spiky workload pattern (Varia, 2012) is found in common among users that have a below av-

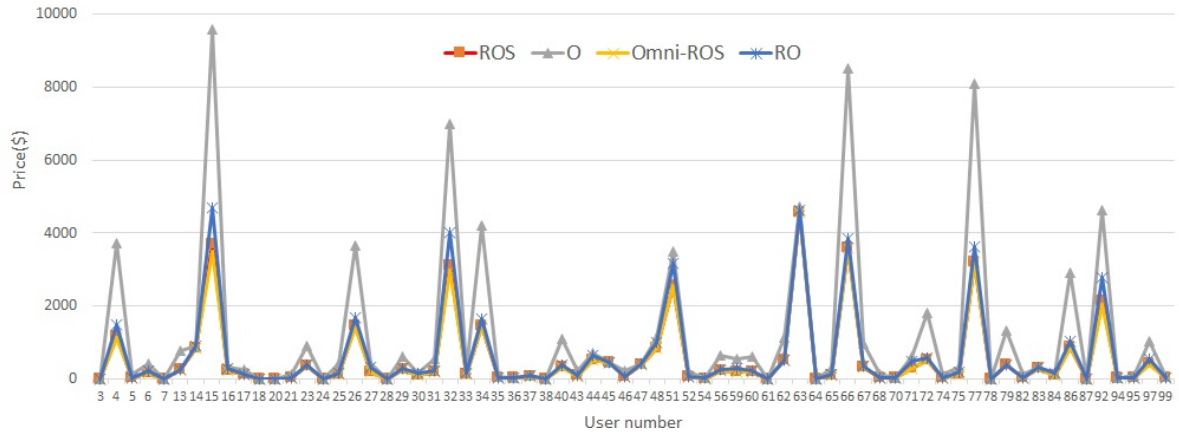


Figure 2: Total costs (USD) for each pricing option, for each user.

Table 3: Comparison of the average number of VMs, number of over/under provisioned VMs, and the total cost for all users during the 696 stages.

Option	Average number of VMs over all users						Average Total Cost
	Reserved	Operating	On-demand	Spot	Over-provisioned	under-provisioned	
Omni-ROS	7	5319	739	1490	23	2229	545.2
ROS	5	4073	746	2076	24	2823	553.39
RO	10	7170	1589	0	76	1589	651.02
O	0	0	6036	0	0	6036	1207

Table 4: Comparison of user characteristics.

User Category	Average over all users		
	Memory Variance	CPU variance	Submission frequency
Perfectly-fitted	8.06	116.1	11.3
Imperfectly-fitted	40.9	142.9	6.5

erage difference between O and omniscient-ROS options. For spiky demand, it seems that the optimal solution reserves no VMs and, instead, rents short-term spot and on-demand VMs for completing tasks. However, this is strongly related to the availability and price of spot instances in the spiked stages. No particular common workload pattern was found for the users with above average difference.

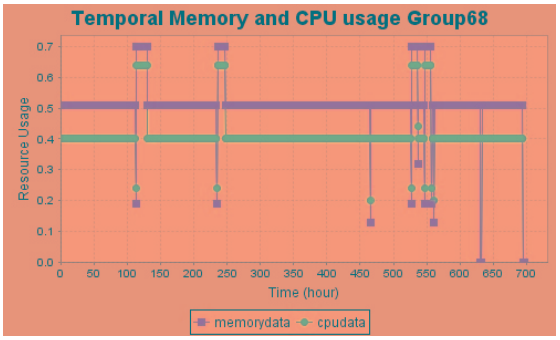
Figure 3 shows two example users with steady state (user 68:a) and spiky (user83:b) workload patterns. The diagrams show the resources requested in each stage during the one-month time slot for each user.

One important observation is the tight correlation of the total cost differences and the number of reserved VM differences, which is 95% for ROS and omniscient-ROS. Figure 4 shows the number of Reserved VMs of different options, divided into the perfectly and imperfectly-fitted categories. The number of reserved VMs for the users in the perfectly-fitted category are very small, and are equal or very close to the reserved VMs by omniscient-ROS.

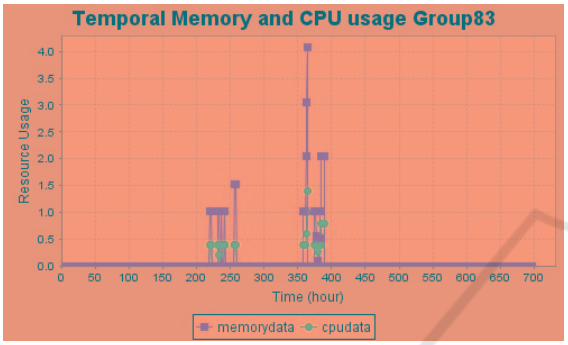
Evidently, the number of reserved VMs chosen by omniscient-ROS is the optimal numbers for this problem, so when other options reserve the same number, they get the exact same results. However, when the solver reserves more or less VMs than omniscient-ROS, the final total cost increases. This is because reserving less VMs results in more occurrence of under-provisioning, while reserving more VMs brings more over-provisioning incidences, and therefore more cost. In Table 3, ROS reserves less VMs than omniscient-ROS, on average, and therefore has more under-provisioned VMs, while RO reserves more VMs, and so has more over-provisioned VMs in comparison to other options.

Another observation, indirectly related to the previous correlation, is that in the imperfectly-fitted category, the resource usage variation is much higher, while the task submission frequency is lower, compared to the perfectly-fitted category. See Table 4. This indicates that variation in CPU and memory demand is an important factor in our optimization model. The reason behind this, is that because uncertain workload is generated randomly in the first stage, and more variance is likely to bring less similarity between random and the real workload, so the first stage decision is less accurate. In other words, the number of reserved VMs are less close to optimal.

In order to further determine user workload characteristics and their correlations with CPU and memory variance, we perform clustering on users, using an off-the-shelf statistical technique, k-means clustering



(a) Steady state workload pattern.



(b) Spiky workload pattern.

Figure 3: Two example users with completely different workload patterns (a) user 68 and (b) user 83.

Table 5: Three classes and their distribution among all users.

Cluster	**rate**	Centroids of users task features		
		CPU usage	Memory Usage	Duration
class1	66%	7.8	6.1	0.5
class2	32%	2.5	0.1	18
class3	4%	446	691	6

(Hartigan, 1975). The clustering was performed on three important attributes: task's CPU usage, memory usage, and duration. We varied the number of clusters from 2 to 10 and found that the best similarity and dissimilarity score, within and between classes, is achieved when 3 clusters were selected. The centroids of the 3 clusters and their share among users are shown in Table 5. Class 1 has medium resource usage and short running tasks, while both class 2 and 3 are long-running tasks, with small and large resource usage, respectively. In each class, 80% of user's task CPU and memory usage are below the centroid provided in Table 5.

In our dataset, no information of application or task types is provided. However, in general, there are two type of long-running tasks. First, user-facing tasks that run continuously, so as to respond quickly to user requests (class2). Second, compute-intensive ones, such as processing web logs (class3). While from Table 5, we see that users with short duration and low resource usage (class1) tasks, such as index look up and search, dominate the user population, which is consistent with related work (Mishra et al., 2010).

As Figure 5 shows, most of the users in class 1 and 3, are among the perfectly-fitted users, around 65% and 100%, respectively, while only 30% of class 2 users are perfectly-fitted.

Analysis of individual user's temporal workload patterns within the same class of users shows that overall, users with lower variances tend to sit in the perfectly-fitted category while users in the imperfectly-fitted category have higher variance.

6 RELATED WORK

The resource provisioning problem has been viewed and solved from different perspectives (Meng et al., 2010), (Li et al., 2015), with little attention from the end-user' view.

The paper closest to our work optimizes the cost of VM provisioning in the cloud computing environment from the end-user's point of view (Chaisiri et al., 2012) by an optimal cloud resource provisioning algorithm (OCRP) while considering both reservation and on-demand pricing plans. Even though the heterogeneity of VMs has been considered in the problem formulation, by specifying the number of VMs as the demand unit in their experiments, heterogeneity of VM has been implicitly denied. In this work, spot pricing was completely ignored and reservation cost is simplified as hourly cost. Another work that also neglects spot VM in their optimization problem is (Díaz et al., 2017). In it, an optimization technique, called LLOOVIA, is proposed to minimize cost, while quarantining the required level of performance. This work has been evaluated with synthetic workloads and Wikipedia users' workloads. The characterization of the workloads are known and therefore the number of reserved VMs are chosen based on a known workload demand. A joint resource provisioning approach that combines both VM and bandwidth allocation is proposed in (Chase and Niyato, 2015). The uncertainty of the problem is also taken into account using stochastic programming. A scenario reduction algorithm is used for scalability of the problem. This work is useful and applicable for both users and cloud providers.

In (Genaud and Gossa, 2011), a satisfactory trade-off between cost and speed to process a set of inde-

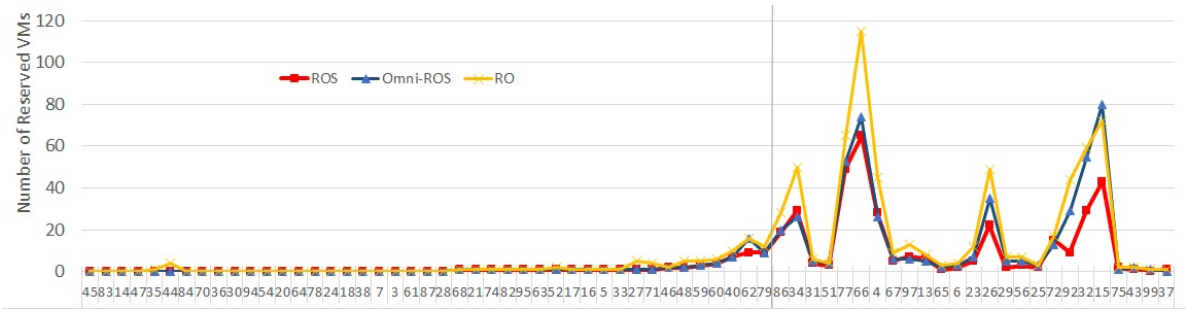


Figure 4: Number of Reserved VMs for individual users for different options.

Figure 5: Percentage classes in each category.

pendent jobs, is conducted from the end-users' side, but only the on-demand pricing model is taken into account. The main focus of (Zhu and Agrawal, 2010) is an automated and dynamic resource allocation approach, in the cloud environment, based on control theory techniques. This problem is solved under constraints of resource budget and a fixed time limit for a particular task. An autonomous elasticity controller is proposed in (Ali-Eldin et al., 2012), it changes the number of virtual machines allocated to a service, based on both monitored load changes and prediction of future work.

Most of the existing literature on cloud resource provisioning focuses on deterministic formulations over fixed horizons, where the scheduler has perfect foresight (Teng and Magoules, 2010). Those that have considered uncertainty in their problem, typically focus on just one aspect (Chaisiri et al., 2012), or use very simple and artificial data (Zafer et al., 2012). Pricing and VM heterogeneity are also neglected in most of them. In our previous work (Tajvidi et al., 2017), we solved a simple version of this problem, while considering parameter uncertainty and heterogeneity of pricing plan and VM types. However we simplified some complexity of the problem, for example, the only optimization objective was cost and the deadline was not modelled. The reservation cost was calculated hourly (to make it comparable with on-demand and spot price), which is not applicable in real-world problems. The workload was not investigated as temporal workload and only two stages of time were modelled.

7 CONCLUSION

In this paper, we have proposed an optimization strategy which determines the number, type, and pricing plan of VMs in multiple stages, to satisfy a user's workload demand and the deadline of tasks.

The experimental results shows that although the user requirements have uncertainty, users can use our model (ROS) to attain provisioning that is close to that obtained with perfect foresight (Omni-ROS). We also conclude that having spot instances beside on-demand and reservation can make a big cost saving for the users. Although the user's workload pattern is an important factor for getting better results, users with lower variance in CPU and memory usage get closer to optimal provisioning, in general. Finally, based on the clustering we performed, we found the majority of the users are allocated to class 1, with short duration and medium resource usage and they are mainly well-fitted to the ROS option. Similarly, the lower variance users within each class get better results.

REFERENCES

Ali-Eldin, A., Kihl, M., Tordsson, J., and Elmroth, E. (2012). Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control. In *Proceedings of the 3rd workshop on Scientific Cloud Computing Date*, pages 31–40. ACM.

Amazon EC2 (2017). Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/.

Birge, J. R. and Louveaux, F. (2011). *Introduction to stochastic programming*. Springer Science & Business Media.

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

Chase, J. and Niyato, D. (2015). Joint optimization of resource provisioning in cloud computing. *IEEE Transactions on Services Computing*.

Chen, S., Ghorbani, M., Wang, Y., Bogdan, P., and Pedram, M. (2014). Trace-based analysis and prediction of cloud computing user behavior using the fractal modeling technique. In *Big Data (BigData Congress), 2014 IEEE International Congress on*, pages 733–739. IEEE.

Díaz, J. L., Entrialgo, J., García, M., García, J., and García, D. F. (2017). Optimal allocation of virtual machines in multi-cloud environments with reserved and on-demand pricing. *Future Generation Computer Systems*, 71:129–144.

Genaud, S. and Gossa, J. (2011). Cost-wait trade-offs in client-side resource provisioning with elastic clouds. In *Cloud computing (CLOUD), 2011 IEEE international conference on*, pages 1–8. IEEE.

Hartigan, J. A. (1975). Clustering algorithms (probability & mathematical statistics).

John Wilkes (2011). More Google Cluster Data.

Li, S., Zhou, Y., Jiao, L., Yan, X., Wang, X., and Lyu, M. R.-T. (2015). Towards operational cost minimization in hybrid clouds for dynamic resource provisioning with delay-aware optimization. *Services Computing, IEEE Transactions on*, 8(3):398–409.

Mao, M. and Humphrey, M. (2012). A performance study on the VM startup time in the cloud. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 423–430. IEEE.

Meng, X., Isci, C., Kephart, J., Zhang, L., Bouillet, E., and Pendarakis, D. (2010). Efficient resource provisioning in compute clouds via VM multiplexing. In *Proceedings of the 7th international conference on Autonomic computing*, pages 11–20. ACM.

Mishra, A. K., Hellerstein, J. L., Cirne, W., and Das, C. R. (2010). Towards characterizing cloud backend workloads: insights from Google compute clusters. *ACM SIGMETRICS Performance Evaluation Review*, 37(4):34–41.

Nethercote, N., Stuckey, P. J., Becket, R., Brand, S., Duck, G. J., and Tack, G. (2007). Minizinc: Towards a standard CP modelling language. In *Proc. Int. Conf. on Principles and Practice of Constraint Programming*, pages 529–543.

Reiss, C., Wilkes, J., and Hellerstein, J. L. (2011). Google cluster-usage traces: format+ schema. *Google Inc., White Paper*, pages 1–14.

Shapiro, A. and Philpott, A. (2007). A tutorial on stochastic programming. *Manuscript. Available at www2. isye. gatech. edu/ashapiro/publications. html*, 17.

Tajvidi, M., Maher, M. J., and Essam, D. (2017). Uncertainty-aware optimization of resource provisioning, a cloud end-user perspective. In *CLOSER 2017 - Proceedings of the 7th International Conference on Cloud Computing and Services Science, Porto, Portugal, April 24-26, 2017.*, pages 293–300.

Tang, S., Yuan, J., and Li, X.-Y. (2012). Towards optimal bidding strategy for Amazon EC2 cloud spot instance. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 91–98. IEEE.

Tang, S., Yuan, J., Wang, C., and Li, X.-Y. (2014). A framework for Amazon EC2 bidding strategy under SLA constraints. *Parallel and Distributed Systems, IEEE Transactions on*, 25(1):2–11.

Teng, F. and Magoules, F. (2010). Resource pricing and equilibrium allocation policy in cloud computing. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 195–202. IEEE.

Varia, J. (2012). The total cost of (non) ownership of web applications in the cloud. *Amazon Web Services whitepaper, Amazon, Seattle, WA*.

Zafer, M., Song, Y., and Lee, K.-W. (2012). Optimal bids for spot VMs in a cloud for deadline constrained jobs. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 75–82. IEEE.

Zhu, Q. and Agrawal, G. (2010). Resource provisioning with budget constraints for adaptive applications in cloud environments. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, pages 304–307. ACM.