Risk Aware Resource Allocation for Clouds*

Muntasir Raihan Rahman
Department of Computer Science
University of Illinois at Urbana-Champaign
mrahman2@illinois.edu

Yi Lu
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
yilu4@illinois.edu

Indranil Gupta
Department of Computer Science
University of Illinois at Urbana-Champaign
indy@illinois.edu

ABSTRACT

Cloud computing offers on-demand access to large-scale computing resources in a pay-as-you go manner. Market-based resource allocation mechanisms are gaining popularity among commercial cloud providers to deal with dynamically fluctuating resource demands. For example, the recently introduced Amazon EC2 spot instances allow users to bid for computing resources and thus control the cost vs. reliability trade-offs of their workloads. Although this promises significant cost reduction, it comes at an additional risk of price fluctuation. This will get worse as cloud computing gradually moves towards a free market system. We propose a novel approach that utilizes financial option theory to simultaneously mitigate risk and minimize cost for cloud users. We formulate the cloud user optimization problem and mathematically characterize the cost of using European style options for clouds. We also propose a novel on-line policy using American options that outperforms base-line spot policies in terms of price variance reduction against high risk factors. We present trace-driven simulation experiments to support our results.

Categories and Subject Descriptors

C.2.4 [**Computer Systems Organization**]: [Computer Communication Networks - Distributed Systems]

Keywords

Cloud Scheduling, Spot Market, Option Pricing

1. INTRODUCTION

Cloud computing [7] is a distributed computing paradigm offering on-demand access to large-scale computing resources for data intensive computations. Cloud computing has become attractive because clients pay as they use resources on demand (i.e., no upfront costs), while providers are able to present the illusion of infinite resources to such clients (e.g., via virtualization) [7, 1]. We interpret a cloud to mean a public datacenter offering a wide variety of hosting services based, e.g., virtualization, or software services. This includes public cloud providers such as Amazon EC2/S3, Google AppEngine, Microsoft Azure, etc.

Market oriented cloud systems have started to receive much attention [1, 2, 27, 33, 17, 11]. Our focus for cloud management system arises from markets where variable pricing is allowed, e.g., Amazon's Spot Instances market allows clients to bid for spare CPU-hour resources. While variable pricing markets offer advantages shared by economic free markets, they introduce risk into cloud client jobs due to market price fluctuations. When prices vary, a cloud client may spend more or lose resources - the latter could result, for instance, in batch jobs failing before they are completed, or in web services seeing reduced availability and throughput. This situation is made worse by the *Efficient Market Hypothesis* [20] in economic markets, which states that (cloud) clients cannot accurately predict the variation of prices in an open market using past price history. Additionally, the workload seen by web services, which are run on the cloud by clients, can be highly variable across time and it may be difficult to predict this variability (notwithstanding DDoS attacks and flash crowds). To ameliorate these risks, there is a need to provide clients with techniques that allow hedging against risk in cloud markets with variable prices.

Our goal is to imbue variable-priced cloud services with techniques that enable cloud clients to ameliorate risks. The bulk of public clouds today offer fixed pricing for resources, e.g., per CPU-hour, per GB-month stored, and per GB transferred over the network, etc. Some representative examples include Amazon EC2, S3, Google AppEngine and Microsoft Azure. However, a new generation of cloud services is emerging wherein prices are variable and determined by demand-supply market equilibrium. Typically in these settings (e.g., in Amazon's Spot Instances service) clients (i.e., applications) bid for unused capacity. If a clients bid exceeds the current spot price, the bid is granted. Thereafter, if the spot price drops below the bid, the allocated resource is withdrawn from the client. Today, these variable pricing markets are often internally manipulated by the provider in order to de-incentivize usage during peak hours. However, we believe this will move closer to a free market economy in the near future. The historical price data of spot instances is publicly available [3]. Figure 1 shows an example.

*This work is submitted in part by NSF grant CCF 0964471.

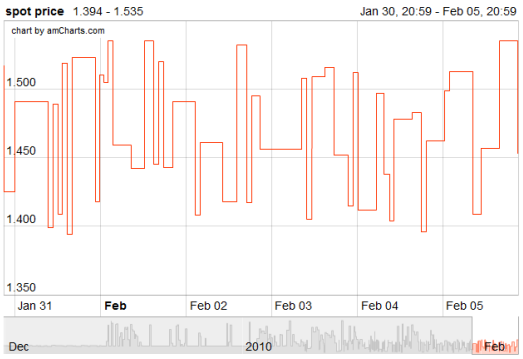

Figure 1: Spot Price Fluctuation.

In this paper we propose to utilize techniques from option pricing [8] to mitigate the risk of price fluctuation in spot markets. The main idea is to use a combination of spot and option instances to schedule the workload[1].The client can purchase a number of options at a fixed price before the job starts. This is like an insurance policy. We say a spot instance fails when the price goes above the user bid. Whenever such a failure event occurs, the client exercises an option which protects the client againts price variation. At other instances, the client can continue using regular spot instances. As a result all the price hikes are smoothed out with a controlled price variation due to the options.

Spot price fluctuation can result in users losing an instance before completion of tasks. In addition to that, users cannot predict price fluctuation on the fly. This results in significant risk for cloud users who want to minimize cost by utilizing spot instances. Our work is motivated by this risk factor and we propose option pricing mechanisms to hedge these risks for cloud users. Since options are sold at any points of time, it is a more flexible form of pricing than on-demand. There can be times when the demand is so high that the on-demand price is low. As option prices also fluctuate, it definitely retains the congestion control property of spot prices.

Using these ideas, we first develop an off-line optimization formulation to find the optimal number of spot and option instances to allocate a given workload. We then propose a cloud provider option pricing model based on the binomial option pricing model. There are mainly two types of options that are more popular. European options can only be exercised at expiration, whereas American options can be exercised at any time before expiration. Since European options are more amenable to mathematical analysis, we use them to statistically characterize the total price for using options for cloud resource allocation. On the other hand, American options are more practical, and we utilize them to develop an efficient on-line resource allocation policy which we compared against base-line policies that use only spot instances. Trace-driven simulation results show that the option policy can significantly reduce total price variation for cloud users.

We adopt the spot instance model proposed in [34], where each spot instance for unused amazon computing resources is considered as a separate spot market. Each physical machine runs multiple types of virtual machine instances, some of which are on-demand or reserved instances, while others are spot instances. All the spot markets share the same unused computational resource pool. In general, Amazon's spot instance mechanism works in a continuous fashion. A spot instance can start running as soon as a request with bidding price higher than the current spot price is submitted. Theoretically this can be implemented by having the instance with higher bid price preempt the one with lower bid price, if there is only enough resources for one instance [34].

The rest of the paper is organized as follows. In Section 2 we discuss some related works. Next in Section 3 we briefly introduce some basics of financial option theory that we utilize in our work. This is followed by Section 4, where we formulate the cloud provider option pricing model, and the cloud user optimization problem. This section also includes a characterization of total cost for using European Options for a cloud workload. Section 5 discusses base-line online policies and introduces the new policy using American options. Next we discuss the results of our trace driven simulation experiments in Section 6. Finally, we conclude in Section 7.

2. RELATED WORKS

Recently, Market oriented resource sharing mechanisms have been popularized for cloud computing environments [10, 27]. Markets provide a decentralized method to deal with supply and demand from multiple users and providers. One common market based approach is using auctions for resource allocation [22]. For example, Mirage [13] is a sensor network testbed with limited resources where users bid using a virtual currency in a closed loop market economy. However the scale of resources in a cloud might prohibit efficient implementation of an auction mechanism scheme. As a result computationally hard combinatorial auctions might be required for auction based resource allocation in clouds [33].

Most of the work on cloud pricing has been based on the assumption of pay as you go pricing models for on-demand VM instances. Although spot instances promise low cost utility computing, few research papers have addressed cloud scheduling using spot instances. The authors in [32] addressed the reliability problem of spot instances using a check-pointing mechanism to periodically save the results of computing. In [6], the same authors develop a probabilistic model for optimization of cost and reliability for using spot instances, and propose a mechanism for cloud users to bid optimally. Dynamic allocation of spot instances for map-reduce computations has been proposed in [12]. The authors in [34] propose to use market analysis for forecasting the demand for each spot market, and develop a dynamic scheduling and consolidation mechanism for allocating resources to each spot market to maximize total revenue. To the best of our knowledge, we are the first to propose the application of financial option pricing [24] to deal with risk management for

[1]In this paper, we only deal with a single type of resource, that is, we are only concerned with allocation of computational resource workloads. Considering multiple types of resources is an interesting generalization that we leave for the future.

cloud users in the presence of a spot market. The low cost of spot instances is coupled with a reliability factor, since users might lose spot instances due to market fluctuation before completion of a scheduled task. Our techniques can be used in conjuction with the revenue maximizing dynamic mechanisms for spot instances proposed in [34].

There is a wealth of literature on pricing financial derivatives in the quantitative finance community [24]. Prominent among these is the celebrated Black-Scholes (BS) model [8]. The BS model is for continuous price jumps and requires solving a partial differential equation which captures the price movement. On the other hand, the binomial lattice is a commonly used approximation for discrete price jumps [24]. One of the most prominent applications of financial options has been the deregulated electricity market in order to design transmission pricing schemes that will ensure open access to the transmission networks [19]. The authors in [5] use real options theory to price grid resources. They only use option theory to price the infrastructure, given the uncertainty in demand. They are not proposing to sell or buy options as we do. Whereas the work in [5] is concerned with risk factors like obsolete grid technologies, we are mainly concerned with risk generated from market fluctuations.

In contrast to using ideas from finance, resource allocation in shared computing environments using ideas from economics is not a new topic. Economic issues in shared infrastructures [29, 14] have received a lot of attention in grid computing [28], and large scale experimental testbeds like PlanetLab [30]. SHARP [21] provides a barter economy based framework for secure distributed resource management. Quincy[25] is a flexible global optimization based framework for scheduling concurrent jobs in a cluster, which uses a flow-based scheduling model to jointly optimize fairness and data locality.

FlexPRICE [23] proposes a flexible cloud pricing model where once a user submits a job, the cloud provider uses the job requirements to generate multiple feasible schedules with prices in terms of a price-speed curve. The user can then select an appropriate point in the curve based on the budget and deadline constraints. The authors in [17] propose an economic model for adaptive cloud caching suitable for querying large scientific workloads. The authors in [31] use a micro-economic approach to determine the optimal number of VMs for each cloud user through profit maximization and price discrimination. Their approach continuously monitors the QoS of each user and adjusts the VM allocation accordingly. For multiple users, they show that their approach reaches an equilbrium that ensures proportional resource fairness [26]. The authors in [16] propose auction based resource allocation mechanisms for on-demand grid computing environments with supply adjustment. This work is different from existing mechanisms since it focuses on the seller's profit instead of client profit by dynamiclly adjusting the supply of resources in the system.

3. BASICS OF FINANCIAL OPTIONS

In this section we describe some essential basics of financial markets. We refer to [24] for a more details. One of the basic underlying principles in mathematical finance is the Efficient Market Hypothesis which states that stock prices already incorporate all available information. Otherwise predictable price movements would generate possibilities for speculators to gain risk free profits. In efficient markets, such speculators always exist and they always take advantage of the presented opportunities, as a result in the end all such opportunities have been taken and all available information has been incorporated into the current market price. It is generally assumed that stock prices follow brownian motion [24]. Let S denote the stock price. Then the rate of change dS is defined as

$$dS = \mu S dt + \sigma S dz \tag{1}$$

here, μ is the expected rate of return and σ is called the stock price volatility. Basically the first term represents the constant and predictable rate, where as the second part is the random component which is usually modeled as a wiener process or a ito process. This results in a Geometric Brownian Motion (GBM):

$$\frac{\Delta S}{S} = \mu \Delta t + \sigma \epsilon \sqrt{\Delta t} \tag{2}$$

The basic underlying idea of any option pricing scheme is as that *to value an option, one must form a self-financing hedging strategy that replicates the payoff of the option.* Volatility is a measure of risk of a stock price, and the celebrated black-scholes formula [8] gives an alternative method for computing volatility as opposed to usual statistical computations. Black Scholes assumes that assets are perfectly divisible, which holds for VM instances.

The binomial model is a simple approximation of the black-scholes model: $P[S \rightarrow Su] = p$, $P[S \rightarrow Sd] = 1 - p$. As $\Delta S \longrightarrow 0$, the binomial model converges to the geometric brownian motion model.

A *call option* gives the holder the right to buy the underlying asset by a certain date (called expiration date, *exercise date* or maturity) for a certain price (called exercise price or *strike price*). *American options* can be exercised at any time up to the expiration date, whereas a *European option* can only be exercised on the final expiration date. American options are more practical and widely used, but European options are easier to analyze mathematically. An option gives the holder a right to exercise the option, but it is not an obligation. This is different from forwards and futures, where the holder is obligated to buy or sell the underlying asset. However, forwards and futures are free, where as an investor must pay to purchase an option contract.

4. PROBLEM FORMULATION

In this section, we first formulate the cloud user optimization problem using European style options. We also develop the option pricing model adopted by the cloud provider.

4.1 Cloud Resource Allocation Using European Options

The *Cloud User Problem* is concerned with finding a work schedule that minimizes cost and mitigates risk at the same

time. Some design decisions in this space include the optimal number of options to buy, and among the options purchased, which options should be exercised.

Suppose the total workload of the task is W instance hours, the time horizon is T. We assume all instances have the same size, e.g. all are small EC2 instances. At each time instance, only a single instance is allowed to run. c is the price of a on-demand instance, c_i is the spot price at time i, v_i is the bid at time i. We also assume that to mitigate risk, the user purchases *European style Call Options* at each time instance: $\mathcal{O}_i =< \mathcal{B}_i, k_i, T_i >$, where $\mathcal{B}_i$ is the price of buying the option, k_i is the strike price, and T_i is the strike date. We now have the following cloud user optimization problem:

$$\text{Minimize} \sum_{i=1}^{T} \mathcal{B}_i + c\sum_{i=1}^{T} d_i + \sum_{i=1}^{T} c_i s_i + \sum_{i=1}^{T} k_i o_i \qquad (3)$$

$$s_i + d_i + o_i \leq 1, \quad \forall i \in \{1, \ldots, T\} \qquad (4)$$

$$\sum_{i=1}^{T} d_i + (1-p_s)\sum_{i=1}^{T} s_i + (1-p_o)\sum_{i=1}^{T} o_i \geq \mathcal{W} \qquad (5)$$

$$o_i \geq \mathcal{I}(v_i < c_i), \quad \forall i \in \{1, \ldots, T\} \qquad (6)$$

$$s_i \geq \mathcal{I}(v_i \geq c_i), \quad \forall i \in \{1, \ldots, T\} \qquad (7)$$

$$s_i, d_i, o_i \in \{0, 1\}, \quad \forall i \in \{1, \ldots, T\} \qquad (8)$$

The objective function minimizes the total cost of using on-demand instances, spot instances and option instances. The first constraint ensures that only one type of instance is selected at each time instance. The second constraint ensures that the schedule covers the required W units. p_s and p_o denote penalties associated with spot and option instances. The third and fourth constraints ensure that an option is exercised when a user loses the bid, other wise a spot instance is used. Here we have a mixed integer program, which can be solved using an open source solver like glpk [4]. Even though we can solve this numerically, it is not that useful since it assumes a static model, where the price variations are known in advance. This is not true in general for a dynamic spot market, which prohibits exact price prediction due to the efficient market hypothesis.

4.2 The Cloud Option Pricing Model

The cloud provider is concerned with the optimal pricing strategy for options that can maximize revenue. Since options can increase the number of customers by offering risk mitigation services, we assume that the cloud provider follows an existing binomial option valuation method without any further revenue maximization strategies.

Consider a stock with current price S. S goes to uS with probability P_u, and to dS with probability P_d at each time step. So let $T = n\Delta t$, where T is the option expiry date and Δt is the time step. So We can create a lattice of spot prices as $S_{j+1} = uS_j$ w.p. p_u, and $S_{j+1} = dS_j$ w.p. p_d. We know the value of the option at the expiration time. So we can move backwards using dynamic programming:

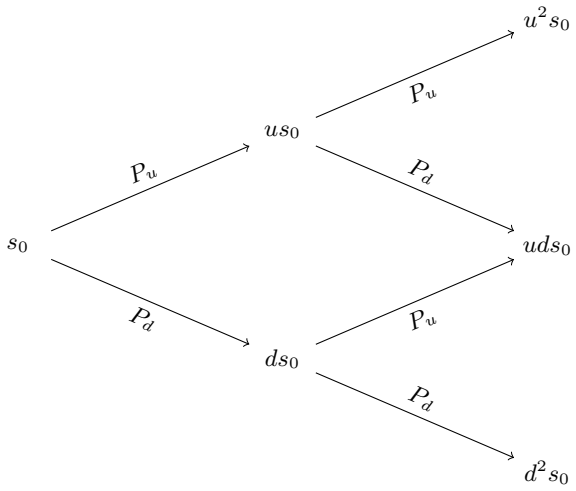

Figure 2: Snapshot of a Binomial Price Tree

$$V_j = \max(P_u V_{j+1}^u + P_d V_{j+1}^d, \max(S_j - K)). \qquad (9)$$

This gives us the value of the option at each time instance before the expiry date. It should be noted that an American option that pays no dividend is never exercised early. As a result its value coincides with a similar European option. This is also confirmed by our simulation results presented later.

4.3 Characterizing the Cost of European Options

As mentioned earlier, european options are only exercised at expiration, whereas american options can be exercised at any time before the expiration. This makes european options easier to analyze mathematically. On the other hand, american options are more practical. As a result, in this section we mathematically analyze the performance of european options for spot resource allocation. Recall that we have european options at each time instance $\{1, \ldots, T\}$. We use the binomial price model to simulate the spot price evolution. A snapshot of the price evolution is shown in Figure 2.

The cost of a european option consists of the fixed base price to purchase the option, and the exercise price at the expiry time. We have the following theorem that characterizes the total price of using european options, where we utilize a european option at each time instance.

Theorem 1 *Let T be the total job cpu requirement. We have T european options at each time instance $\mathcal{O}_i =< \mathcal{B}_i, k_i, T_i >$, where $\mathcal{B}_i$ is the price of buying the option, k_i is the strike price, and T_i is the strike date. Let s_i denote the spot price at time i, which follows the binomial model $< u, d, P_u, P_d >$, where u is the upward price jump factor, d is the downward price jump factor, P_u is the probability of an up jump, and $P_d = 1 - P_u$. Then the total cost for using european options is $\mathcal{R} = E(s_i)$.*

The proof uses the following simple lemma.

Lemma 1 *For any* $a, b > 0, \max(a-b,0) + \min(b,a) = a$.

Proof.

If $a > b, \max(a-b,0) + \min(b,a) = a - b + b = a.$ (10)

If $a < b, \max(a-b,0) + \min(b,a) = 0 + a = a.$ □ (11)

Proof of Theorem 1. We will show the result by proving it for 1 and 2 iterations. The result will become obvious from these two cases.

Case 1: At time $i = 2$, the option $\mathcal{O}_2$ expires. In this case s_0 can go to either us_0 w.p. P_u, or to ds_0 w.p. P_d. So the expected value of the spot price is:

$$E[s_i] = us_0P_u + ds_0P_d \tag{12}$$

Now the option price consists of two components: (1) the base price computed using the binomial option pricing formula, and (2) the exercise price at expiration. We know the option price at the expiration. If the price goes up, then the price is $\max(us_0 - k)$, and if the price goes down, then it is $\max(ds_0 - k)$, where k is the strike price. So, if we backtrack to the start, then the option price is

$$\mathcal{B}_2 = \max(us_0 - k, 0)P_u + \max(ds_0 - k, 0)P_d \tag{13}$$

The cost of the option at the expiration is:

$$\min(k, us_0)P_u + \min(k, ds_0)P_d \tag{14}$$

By using Lemma 1, we get the total option cost:

$$\begin{aligned}\mathcal{R} &= P_u(\max(us_0 - k, 0) + \min(k, us_0)) \\ &+P_d(\max(ds_0 - k, 0) + \min(k, ds_0)) \\ &= P_u us_0 + P_d ds_0 \\ &= E[s_i]\end{aligned}$$

Case 2: At time $i = 3$, the option $\mathcal{O}_3$ expires. The spot price variation in this case can be seen in the binomial tree shown in Figure 2. So the spot price at the expiration will be u^2s_0 w.p. P_u^2, uds_0 w.p. $2P_uP_d$, and d^2s_0 w.p. P_d^2. So we have

$$E[s_i] = u^2s_0P_u^2 + uds_02P_uP_d + d^2s_0P_d^2 \tag{15}$$

Similar to the previous case, the option base price will be

$$\begin{aligned}\mathcal{B}_3 = \max(u^2s_0 - k, 0)P_u^2 &+ \max(uds_0 - k)2P_uP_d \\ &+ \max(d^2s_0 - k, 0)P_d^2\end{aligned}$$

The cost of the option at the expiration is:

$$\min(k, u^2s_0)P_u^2 + \min(k, uds_0)2P_uP_d + \min(k, d^2s_0)P_d^2 \tag{16}$$

Again using Lemma 1, the total option cost is:

$$\mathcal{R} = u^2s_0P_u^2 + uds_02P_uP_d + d^2s_0P_d^2 = E[s_i] \tag{17}$$

Clearly the result holds at all time instances. This completes the proof. □

5. ON-LINE POLICIES

The formulation shown in subsection 4.1 is off-line, since we assume we know the spot price values at each time instance. Realistically we need to find on-line strategies and compare them against the optimal off-line solution. We identify mainly two classes of on-line policies: (1) Periodic and (2) Reactive. A periodic policy makes decision at fixed time intervals, for example, if we track price changes in Δt units, then a periodic policy could make decisions every $D\Delta t$ units. A decision would be to either re-bid or utilize an option at that time instance. On the other hand, reactive policy makes a decision whenever a failure event occurs.

Previously we only allowed European style options, where an option holder can only exercise the option at the specified strike date. This is inflexible and may not allow us to get maximum benefit from using options. Instead, if we allow American style options, then the user can exercise it any time before the strike date, thus allowing more flexibility.

We first describe two base-line policies using only spot instances, and then we propose an on-line policy using American options.

5.1 Baseline On-line Policy: Using Pure Spot Instances

The first baseline policy is the simplest and at each time instance it just uses the current spot instance at the current spot price. For this policy we assume there is a check-pointing mechanism in place that can periodically save the work. As a result the total expected price for a workload of T hours is $E(\sum_{i=1}^{T} s_i)$. This is the default policy used by current spot market providers like Amazon EC2.

5.2 Baseline On-line Policy: Spot Restart Policy

For this strategy, we assume the user has a cut-off price C, such that for a spot price $s_i > C$, the user can no longer afford the instance. This policy is applicable for map-reduce type computations [18], which do not support check-pointing

mechanisms. The risk measure $r(C, s_0)$ denotes the probability that the spot price random walk $< s_i >$ goes over C. The strategy is to keep trying until the entire workload is complete. As an example, if the first attempt finishes $T/2$ hours before a failure, the second attempt finishes $T-2$ hours before a failure, and the third attempt completes T hours, then we have success in 3 trials. With probability $r(C, s_0)$, there is a failure. And the success probability is $1 - r(C, s_0)$. So we now have a geometric distribution with mean $k = \frac{1}{1-r(C,s_0)}$. So the expected work completion time is

$$kT = \frac{T}{1 - r(C, s_0)}. \tag{18}$$

The total price is

$$E[\sum_{i=1}^{T} s_i \ \mathcal{I}(s_i < C)]. \tag{19}$$

5.3 On-line Policy Using American Options

We propose a new allocation policy based on American options. The main justification for using American options for the online policy is that they can be exercised at any time before the option expires. We can embed the spot instances in american style call option instances. We want to schedule the workload for the user using option instances. When the spot price is below the user bid, the user can use spot instances. When it goes above the bid, the user exercises an option. The key difference is that now the user does not need a separate option at every time instance. Risk r is defined as the probability of the spot price moving above the user bid. We want to find the option schedule with a risk level $< r$. r will be a parameter specified by the cloud user. Assume we have the following options: $\mathcal{O}_i =< \mathcal{B}_i, k_i, T_i >$. For simplicity, now assume $k_i = k \forall i$. So the total cost for the pure options case is:

$$\sum_{i=1}^{T} [P_i + \min(k, s_i)], \tag{20}$$

where P_i is the option valuation for $\mathcal{O}_i$ calculated using the binomial option pricing model for american options. We have the following on-line policy using american options:

Algorithm 1 On-Line Policy Using American Options

```
1: procedure OLAP(r, v, P(s > v))
2:     m = ⌈(1 − r)(P(s > v))T⌉
3:     Purchase m American options
4:     for i = 1 to m do
5:         if s_i > v then
6:             Exercise option instance
7:         else
8:             Use regular spot instance
9:         end if
10:    end for
11: end procedure
```

We estimate the number of American options required using an approximate value for $P(s > v)$, which specifies the probability of the spot price going over the bid. This value can be calculated analytically using martingale techniques, or empiricially using simple statistical calculations. Then, whenever a failure occurs, we simply exercise an American option. Otherwise we use a regular spot instance.

6. PERFORMANCE EVALUATION

In this section we describe the simulation environment followed by the main evaluation results. Our evaluation focuses on quantifying the minimization of price variation by using options compared to base-line approaches. We examine the average and standard deviation of the total price for workloads using the proposed on-line option policy, the baseline spot policy, and the baseline spot-restart policy.

6.1 Experimental Setup

We have implemented a discrete event simulator to evaluate our proposed heuristics. The simulation parameters include the total workload requirement T, the initial spot price S, the user bid differential dV (user bid $V = S + dV$), the upward jump probability p_u ($p_u + p_d = 1$), the upward jump factor u, the downward jump factor d, and the strike price K[2]. We compute the average and standard deviation of the total price over a 1000 sample paths or trials for the on-line policies. For the spot-restart policy, we divide the total price by the number of attempts until success for each trial. We assume that the spot price follows a binomial price evolution. We vary three simulation parameters: p_u, u, and T in our experiments. The initial spot price S is fixed at 0.039, the user bid differential is $dV = 0.0005$, $d = 0.8$, and $K = 0.039$ in all the experiments.

For the first two plots (Figures 3, 4), we fix T to 10, u to 1.4, and vary p_u from 0.5 to 0.9 at 0.001 increments. For the next two plots (Figures 5, 6), T is fixed at 10 as usual, p_u to 0.8, and u is varied from 1.1 to 1.9. We can consider p_u, and u as risk factors, since the higher the values of these parameters, the higher the chance of the spot price going over the user bid. For the graphs against workload duration T (Figures 7, 8), we set p_u at 0.8, and u at 1.4, and vary T between 2 and 20.

6.2 Performance Metrics

We use two performance metrics for evaluation purposes in our experiments. We measure the average and standard deviation of total price over a 1000 trials or sample paths. For each sample path we compute the binomial option prices and compute the total price for the three heuristics. The standard deviation is an approximate measure of risk since it is proportional to the price variance. As a result the standard deviation of price against p_u, and u gives an approximate characterization of the risk-price trade-off. On the other hand, the plots of price against T measures the performance of the heuristics against increasing workload requirements. The Table 1 summarizes the parameters and performance metrics for our experiments.

We compare the performance of the proposed on-line policy using American options against two base-line policies using

[2]In our experiments we set $K = S$.

Table 1: Simulation Parameters and Performance Metrics

Notation	Description
Parameter: p_u	Upward jump probability in binomial model.
Parameter: u	Upward jump factor in binomial model.
Parameter: T	Number of cpu-hours of computation requested by client.
Metric: $E[p]$	Average price for the entire workload.
Metric: $\sqrt{V[p]}$	Standard deviation of price for the entire workload.

Table 2: Compared Policies

Policy	Description
Option	Proposed On-line policy using American options.
Spot	Default base-line policy using only spot instances.
Cutoff	Base-line spot-restart policy using a cutoff parameter.

only spot instances. The compared policies are summarized in the Table 2.

6.3 Experiments

Our first set of experiments is designed to evaluate the average price and standard deviation of price for using option and spot instances[3]. Our goal for these experiments is to show the effectiveness of options in reducing the overall price variation against increasing risk factors and workload duration. We compute both the average price and standard deviation of price against u, p_u, and T. T represents the workload duration, whereas u, p_u represent risk factors. These experiments assume an underlying binomial price variation model. The evaluation results of these experiments are discussed in Subsection 6.4.

The option policy leverages the binomial price model characteristics, whereas the base-line policies are oblivious of the adopted pricing model. We design the second set of experiments to evaluate the proposed policies in a bias-free manner. In these experiments we generate prices using a trinomial model, and compute option prices using the binomial model. This removes any advantage for the option policy in utilizing pricing model information. The results of these experiments are shown in Subsection 6.5.

Finally in the third set of experiments, we evaluate our policies using Amazon EC2 spot trace date obtained from [3]. The results are discussed in Subsection 6.6.

[3]In all our experimental plots, *Options* refers to the online policy using American options, *Spot* refers to the default policy using only pure spot instances, and *Cutoff* refers to the spot-restart base-line policy.

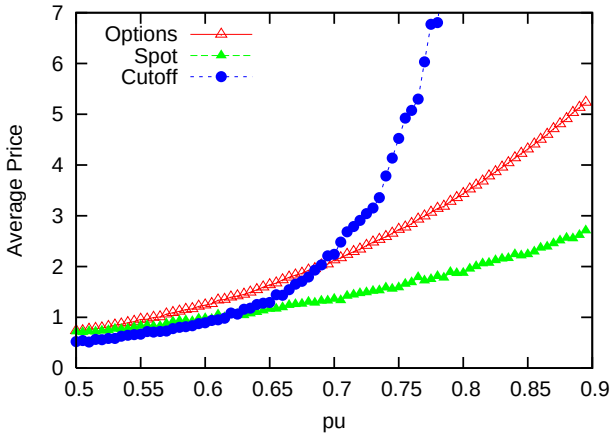

Figure 3: Average Price Against p_u.

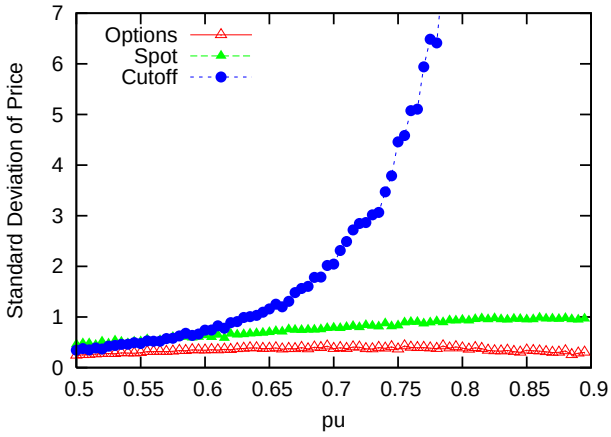

Figure 4: Standard Deviation of Price Against p_u.

6.4 Evaluation Results

We compare our proposed on-line option policy against the baseline spot policies. We present our evaluation results by summarizing the key observations in the following subsections.

6.4.1 *The option policy hedges against risk better than the baseline policies*

From Figure 4, and Figure 6, we see that the option policy has lower price variance against p_u, and u. This indicates that with higher risk, the option policy outperforms the baseline policies in terms of risk mitigation. This is because in case of out of bid scenarios, the option policy exercises an american option and reduces the price to the strike price. The option policy utilizes information about u, d and p_u, p_d to minimize the price variance. On the other hand, the spot and the spot-restart policies are oblivious of the spot price history and perform poorly against higher risk factors.

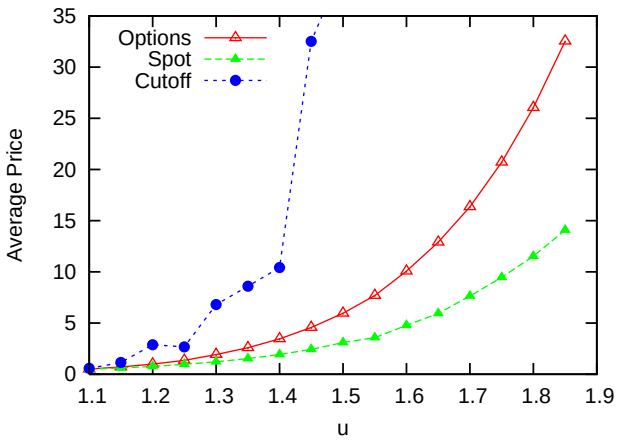

Figure 5: Average Price Against u.

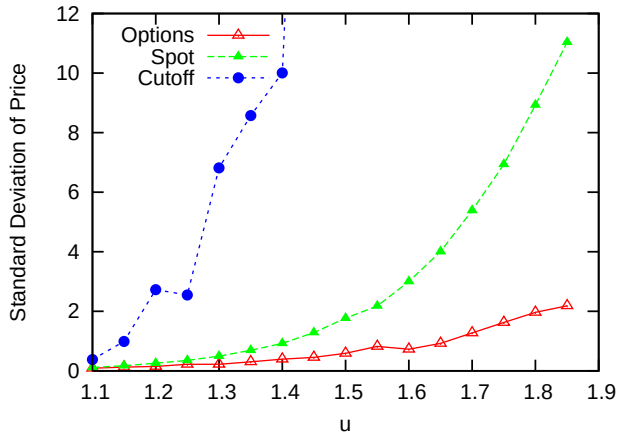

Figure 6: Standard Deviation of Price Against u.

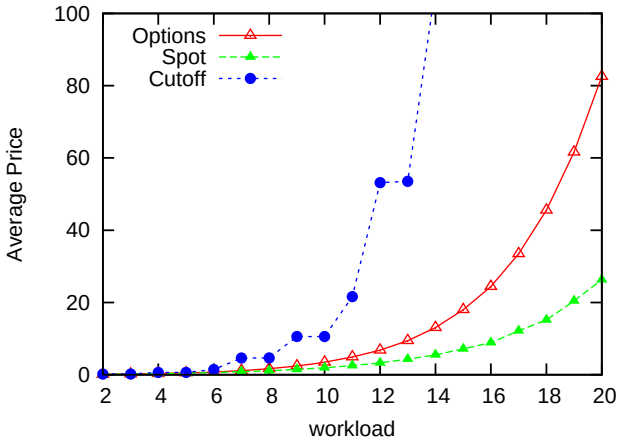

Figure 7: Average Price Against T.

6.4.2 *The average price for option policy is slightly higher the spot policy*

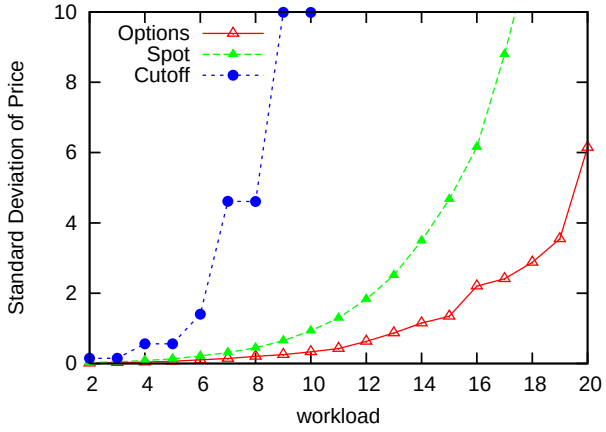

Figure 8: Standard Deviation of Price Against T.

From the Figures 3, and 5, we observe that the baseline spot policy has better average price than the option policy. The option policy has slightly higher average price due to the inclusion of the option premium in the total price. The option policy would have been better if we excluded the option premium that the user has to pay upfront. If we only compared running cost, then the option policy would have outperformed baseline spot policy in terms of average price.

6.4.3 *The average price and the price variance against the workload T show similar trends*

We observe that the average price and the price variation against T result in similar characteristics (7, 8). The only difference is that for the average price the spot policy is better, whereas for the price variance for the option policy outperforms the baseline policies.

6.4.4 *The spot-restart policy performs worse*

We observe from all the experiments that the baseline spot-restart policy performs worse under all scenarios. This is obvious, since it has to restart after every failed attempt. However this policy is still important in map-reduce type computations where there is no failure check-pointing mechanisms in place.

6.5 Testing the Policies using a Trinomial Price Model

The option policy leverages the properties of the binomial model to minimize risk. On the other hand the pure spot policy and the spot restart policy are oblivious of the adopted pricing model. To remove this bias, we simulated the price evolution using a trinomial model instead. A trinomial model [9] is a slight variation of the binomial model, where the price stays the same with a certain probability p_m in addition to moving up or down with probailities p_u, and p_d, respecively $(p_u + p_d + p_m)$. We then ran the same set of experiments using this modified setting. We observed that the results for the option approach and the pure spot approach remain almost the same, whereas the spot-restart policy exhibited slightly worse performance. This is evident from Figures 9, 10, 11, 12, 13, and 14.

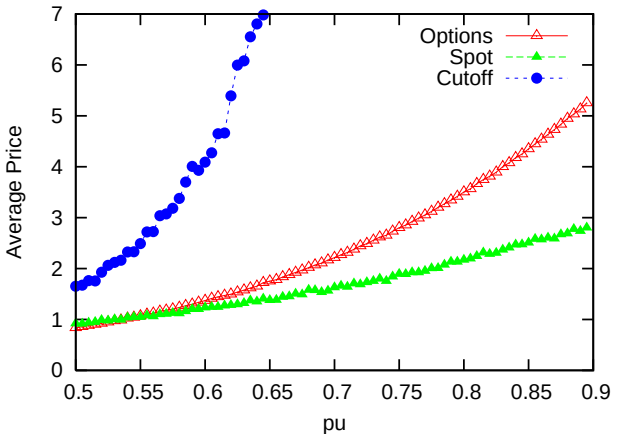

Figure 9: Average Price Against p_u using Trinomial Model.

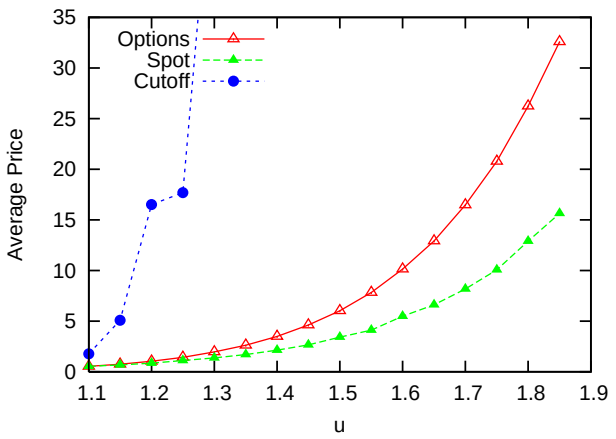

Figure 11: Average Price Against u using Trinomial Model.

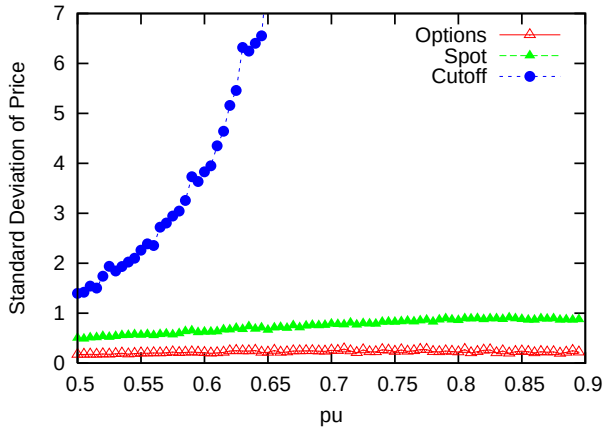

Figure 10: Standard Deviation of Price Against p_u using Trinomial Model.

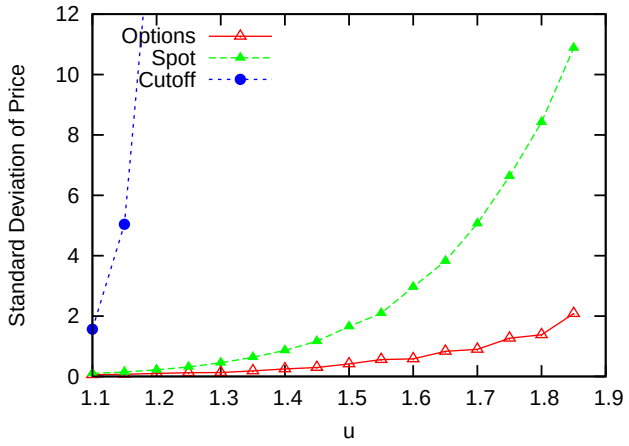

Figure 12: Standard Deviation of Price Against u using Trinomial Model.

6.6 Real Price History from Amazon EC2

In this section we evaluate our results using real trace data. We use the following formulas from [15] to estimate the parameters of the binomial model using real price history of amazon EC2 spot instances [3]:

$$u = e^{\sigma\sqrt{\frac{1}{n}}}, d = \frac{1}{u} = e^{-\sigma\sqrt{\frac{1}{n}}}, p_u = \frac{1}{2} + \frac{1}{2}\frac{\mu}{\sigma}\sqrt{\frac{1}{n}} \qquad (21)$$

From the spot price history obtained from [3], we get $\mu = 0.0385$, and $\sigma = 0.000707107$, and $n = 9518$. So the binomial model parameters are estimated as $u = 1.000007248$, $d = 0.999992752$, and $p_u = 0.779044124$. So $p_d = 1 - p_u = 0.220955876$. Here we see that the real data set has very low volatility which results in a value of $u \approx 1$. As a result, for spot prices 0.0385 ± 0.000707107, a user bid of 0.04 will always be sufficient to avoid all risk. The reason for this is that the percentage of cloud users using spot instances is much less compared to the existing user-base for on-demand instances. The spot instances were introduced fairly recently and does not have a large user base yet. The spot price fluctuation due to demand variation would be much larger if the number of users using spot instances increased drastically. Another issue is that the binomial model assumes that the original data follows a log-normal distribution, which might not be the case for ec2 spot price data.

However we believe that many commercial cloud providers will gradually increase the share of market based cloud instances. With a larger market, the greater amount of supply and demand will nataurally lead to higher levels of price fluctuation. To quantify this effect, we multiplied the spot trace data by a certain factor (in this case 2.5), in order to artificially increase price fluctuation. We also zoomed in the data and only considered high fluctuation regions of the data set. This resulted in similar trends as our simulation experiments using the binomial model. The results are shown in Figures 15, and 16.

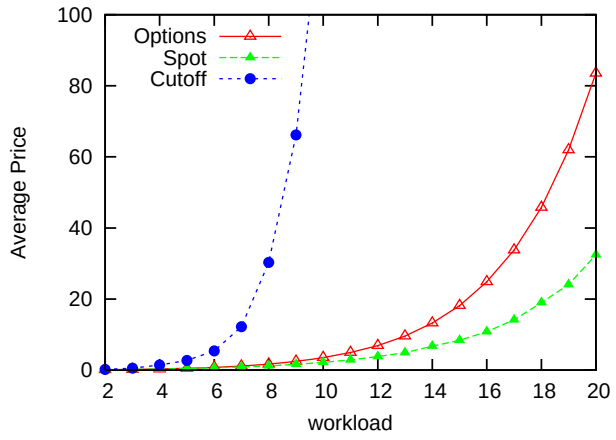

Figure 13: Average Price Against T using Trinomial Model.

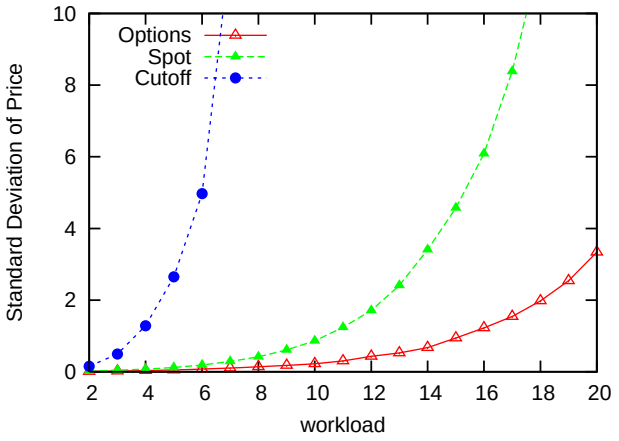

Figure 14: Standard Deviation of Price Against T using Trinomial Model.

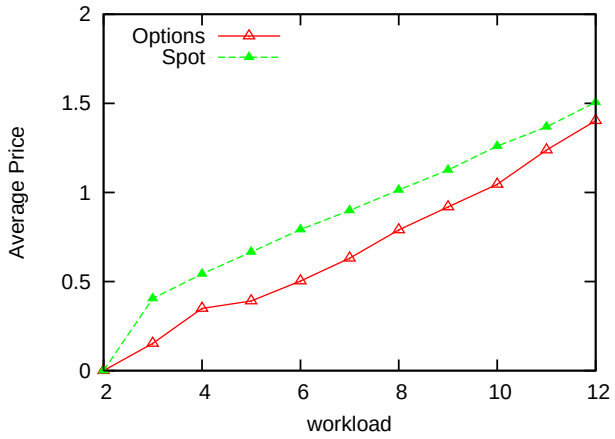

Figure 15: Average Price Against T using real EC2 trace.

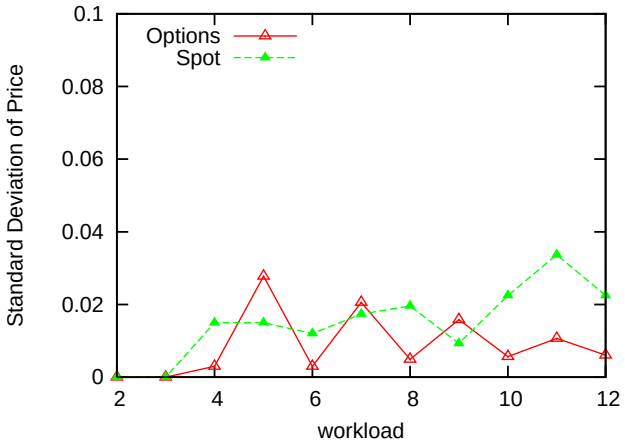

Figure 16: Standard Deviation of Price Against T using real EC2 trace.

We also investigated the optimal magnification factor for the real trace data. That is, we multiplied the trace data by a certain factor, which was varied from 1 to 5 in increments of 1.1. Our goal is to find the optimal factor that best represents the market fluctuation of a spot market in a free economy. Our assumption is that as clouds gradually transition to free markets, the price variation will drastically increase. Since the current trace data does not show enough fluctuations as expected in a free market, we magnified the trace data to reflect a more fluctuating spot price history. Our results in Figure 17, and 18 indicate that there is a sharp threshhold for values from 2.5 to 3. Before this threshhold, both the spot and option policies have negligible average cost and price variation. Afterwards, there is a sharp increase in both average cost and price variation. Our results here would have been more informative if we had trace data for a cloud spot market that is closer to a free economy, rather than a trace from the Amazon EC2 spot market, where the spot price is internally varied by the cloud provider.

6.7 European vs American Options

We know that European options can only be exercised at expiration, whereas American options can be used at any point in time. This makes European options more amenable to theoretical analysis. On the other hand American options are more realistic. This represents a classic modeling tradeoff. We also follow this paradigm by mathematically analyzing cloud resource allocation policies using European options, while using realistic online heuristics for American options. In general the same cloud resource allocation with American options requires much less option instances compared to European options. This is because we need a single Eureopean option at every time instance, since we do not know beforehand when a spot failure will occur. However the price of an American option is generally higher to account for the flexible exercise time. Our theoretical analysis shows that for European options, the total cost of using options is equal to the average spot price. This does not hold in general for American options, which is evident from the

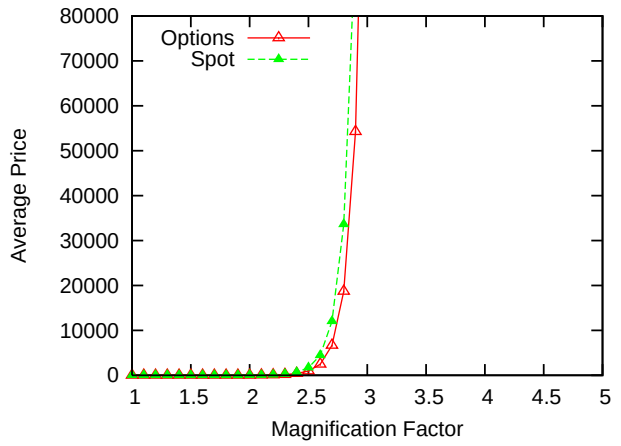

Figure 17: Average Price Against Magnification Factor using real EC2 trace.

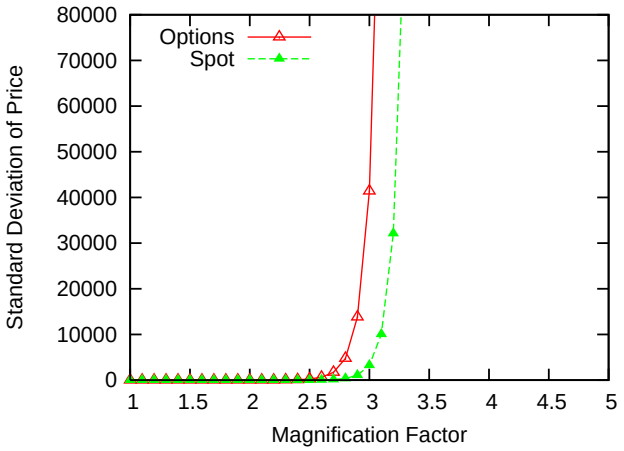

Figure 18: Standard Deviation of Price Against Magnification Factor using real EC2 trace.

simulation results presented in this section. However our simulation results indicate the American options can significanlty reduce the standard deviation of price, which means options can mitigate price variation risk for users.

7. CONCLUSION

Market-based cloud systems with spot instances offer the flexibility of free market economies and the possibility of low cost utility computing. But this comes at a cost of certain price fluctuations which result in significant risk for users. In this paper, we have proposed to leverage financial options to simultaneously minimize cost and mitigate risk for using spot instances in a market-based cloud system. To this end we have formulated the cloud user optimization problem and specified a cloud provider pricing model. We have mathemtically characterized the cost of using European options for cloud resource allocation. We also proposed an efficient on-line scheduling policy using American options which can significantly reduce total price variation against increased risk factors compared to baseline spot policies.

However there are a number of issues that still need to be addressed. First of all, we need to optimize the low u, p_u region by introducing a adaptive policy, such that for low values the policy uses spot instances, and for high values it uses options. This can further reduce the cost of using spot instances. So far, we have tested our heuristics using a homegrown discrete event simulator. On top of that, we are also thinking about implementing our algorithms on a real testbed like emulab. However emulating a pricing model on a testbed is not feasible, since testbed resource usage is free. One possible approach would be to write an agent software that can simulate the price behavior of real cloud users. We could then deploy the agent code on emulab to indirectly simulate pricing dynamics in a cloud environment. We are also interested in testing our models using the java based cloud toolkit cloudbus [11]. Although we envision a cloud spot market with price variations similar to a free market economy, the amazon EC2 data indicates that the current price fluctuation is not yet that volatile. As a result an insurance policy based mechanism for low volatility cloud markets could also be investigated.

8. REFERENCES

[1] `http://cloudharmony.com/`.

[2] `http://aws.amazon.com/ec2/spot-instances/`.

[3] `http://cloudexchange.org/`. [Online; accessed 11-December-2010].

[4] `http://www.gnu.org/software/glpk/`.

[5] David Allenotor, Ruppa Thulasiram, and Parimala Thulasiraman. A novel application of option pricing to distributed resources management. *Parallel and Distributed Processing Symposium, International*, 0:1–8, 2009.

[6] Artur Andrzejak, Derrick Kondo, and Sangho Yi. Decision model for cloud computing under sla constraints. *Modeling, Analysis, and Simulation of Computer Systems, International Symposium on*, 0:257–266, 2010.

[7] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A view of cloud computing. *Commun. ACM*, 53:50–58, April 2010.

[8] Fischer Black and Myron S Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–54, May-June 1973.

[9] Phelim Boyle. Option valuation using a three-jump process. *International Options Journal*, pages 7–12, 1986.

[10] Rajkumar Buyya. Market-oriented cloud computing: Vision, hype, and reality of delivering computing as the 5th utility. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, CCGRID '09, pages 1–, Washington, DC, USA, 2009. IEEE Computer Society.

[11] Rajkumar Buyya, Suraj Pandey, and Christian Vecchiola. Cloudbus toolkit for market-oriented cloud computing. In *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09,

pages 24–44, Berlin, Heidelberg, 2009. Springer-Verlag.

[12] Navraj Chohan, Claris Castillo, Mike Spreitzer, Malgorzata Steinder, Asser Tantawi, and Chandra Krintz. See spot run: using spot instances for mapreduce workflows. In *HotCloud'10: Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, pages 7–7, Berkeley, CA, USA, 2010. USENIX Association.

[13] B. N. Chun, P. Buonadonna, A. AuYoung, Chaki Ng, D. C. Parkes, J. Shneidman, A. C. Snoeren, and A. Vahdat. Mirage: a microeconomic resource allocation system for sensornet testbeds. In *Proceedings of the 2nd IEEE workshop on Embedded Networked Sensors*, pages 19–28, Washington, DC, USA, 2005. IEEE Computer Society.

[14] Costas Courcoubetis and Richard R. Weber. Economic issues in shared infrastructures. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, VISA '09, pages 89–96, New York, NY, USA, 2009. ACM.

[15] John C. Cox, Stephen A. Ross, and Mark Rubinstein. Option pricing: A simplified approach. *Journal of Financial Economics*, 7(3):229–263, 1979.

[16] Amir Danak and Shie Mannor. Resource allocation with supply adjustment in distributed computing systems. In *Proceedings of the 2010 IEEE 30th International Conference on Distributed Computing Systems*, ICDCS '10, pages 498–506, Washington, DC, USA, 2010. IEEE Computer Society.

[17] Debabrata Dash, Verena Kantere, and Anastasia Ailamaki. An economic model for self-tuned cloud caching. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 1687–1693, Washington, DC, USA, 2009. IEEE Computer Society.

[18] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113, January 2008.

[19] Shijie Deng. *Financial Methods in Deregulated Electricity Markets*. PhD thesis, University of California at Berkeley, USA, 1999.

[20] Eugene F. Fama. Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance*, 25(2):383–417, 1970.

[21] Yun Fu, Jeffrey Chase, Brent Chun, Stephen Schwab, and Amin Vahdat. Sharp: an architecture for secure resource peering. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 133–148, New York, NY, USA, 2003. ACM.

[22] D. Grosu and A. Das. Auctioning resources in grids: model and protocols: Research articles. *Concurr. Comput. : Pract. Exper.*, 18:1909–1927, December 2006.

[23] Thomas A. Henzinger, Anmol V. Singh, Vasu Singh, Thomas Wies, and Damien Zufferey. Flexprice: Flexible provisioning of resources in a cloud environment. In *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing*, CLOUD '10, pages 83–90, Washington, DC, USA, 2010. IEEE Computer Society.

[24] John C. Hull. *Options, futures, and other derivatives*. Pearson Prentice Hall, 6. ed., pearson internat. ed edition, 2006.

[25] Michael Isard, Vijayan Prabhakaran, Jon Currey, Udi Wieder, Kunal Talwar, and Andrew Goldberg. Quincy: fair scheduling for distributed computing clusters. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, SOSP '09, pages 261–276, New York, NY, USA, 2009. ACM.

[26] F P Kelly, A K Maulloo, and D K H Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, 1998.

[27] Orran Krieger, Phil McGachey, and Arkady Kanevsky. Enabling a marketplace of clouds: Vmware's vcloud director. *SIGOPS Oper. Syst. Rev.*, 44:103–114, December 2010.

[28] Michael Litzkow, Miron Livny, and Matthew Mutka. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, June 1988.

[29] A. Ozdaglar and N. Shimkin. Socially optimal pricing of cloud computing resources. In *ValueTools 2011, to appear*, 2011.

[30] Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proceedings of HotNets–I*, Princeton, New Jersey, October 2002.

[31] Konstantinos Tsakalozos, Herald Kllapi, Eva Sitaridi, Mema Roussopoulos, Dimitris Paparas, and Alex Delis. Flexible use of cloud resources through profit maximization and price discrimination. In *Proc. of the 27th IEEE Int. Conf. on Data Engineering (ICDE 2011)*, Hannover, Germany, 2011.

[32] Sangho Yi, Derrick Kondo, and Artur Andrzejak. Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud. *Cloud Computing, IEEE International Conference on*, 0:236–243, 2010.

[33] Sharrukh Zaman and Daniel Grosu. Combinatorial auction-based allocation of virtual machine instances in clouds. In *CloudCom*, pages 127–134, 2010.

[34] Qi Zhang, Eren Gurses, Raouf Boutaba, and Jin Xiao. Dynamic resource allocation for spot markets in clouds. In *Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE '11)*, March 2011.