Cloud computing has emerged as a new paradigm for delivery of applications, platforms, or computing resources to customers in a “pay-as-you-go-model”. According to the National Institute of Standards and Technology (NIST): “Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g. networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction” [1].

The Cloud model is cost-effective because customers pay for their actual usage without upfront costs and scalable because it can be used more or less depending on the customers’ needs. Due to its advantages, cloud computing has fascinated and is still attracting a lot of attention both in industry and in academic world offering a multiple range of flexible, on-demand, and highly scalable computing services. The related flexibility and effectiveness of cost make cloud computing a valuable option for organizations in the public and private sectors [2]. Gartner Forecasts worldwide businesses and individuals spending on Cloud Computing services is expected to be $112 billion in 2017 [3]. Profitability and revenue maximization are the most important goals for any cloud service provider, which can be employed through different pricing models; however, end-users are typically more interested in high satisfaction guarantee through Quality of Service (QoS), cost-effectiveness, usability and availability of cloud resources (Users...
maximize utility and CSPs maximize profits) [4].

In cloud, provisioning of computing resources is offered in the form of Virtual Machines (VM), being deployed on physical computing nodes. There are various types of VM instances offered by cloud service providers at different prices, with different compute capabilities, network and storage. In case of on-demand, the instance price doesn’t change (fixed cost) and such instances with relatively high pricing are well suited for deadline constrained jobs. In December 2009, Amazon introduced spot instance pricing system, which allows consumers to bid on spare Amazon EC2 instances and run them whenever their bid exceeds the current spot price based on current supply and demand. The spot instance pricing model complements the on-demand and reserved instance pricing models, providing potentially the most cost-effective option for obtaining compute capacity, depending on user application. Such instances are best suited for optional, delayable or large scale budget constraints jobs.

The aim of this research is to analyze and construct a model that helps to increase business value for the cloud service provider and cost effectiveness for end users. We intend to find out optimal bidding procedures for spot virtual machines in cloud and develop a framework to create a win-win scenario both for consumers and cloud providers. The concept of financial options has been introduced for acquiring on-demand VMs for critical deadline constrained jobs whereas normal jobs are catered through spot instances which results in lower cost. From cloud provider’s point of view, this model optimizes overall resource utilization while cloud consumers may benefit in terms of overall cost.

The study is aimed to address the following two research questions:

1. How can a cost-efficient and resilient resource provisioning framework enhance overall value optimization for the cloud service provider?

2. How can a resource management model for cloud, based on deadlines and constraints, help resource provisioning to be more efficient?

Profit maximization and cost minimization are the main factors in this research. Both results had to be obtained by without violating the SLA constraints.

1.1 FINANCIAL OPTIONS

A Financial Option is an agreement which gives right to the buyer to buy or sell an asset; it is just a right not an obligation. There are two basic types of options:

“*A call option gives the holder of the option the right to buy an asset by a certain date for a certain price. A put option gives the holder the right to sell an asset by a certain date for a certain price*” [4],[5].

The value of an option can be estimated using different quantitative techniques based on the concept of risk neutral pricing and using stochastic calculus. Stephen Ross and Mark Rubinstein developed the original version of the binomial options pricing model [6]. It models the dynamics of the option’s theoretical value for discrete time intervals over the duration of option. The model starts with a binomial tree of discrete future possible underlying stock prices. By constructing a riskless portfolio of an option and stock, a simple formula can be used to find the option price at each node in the tree.

The current spot market price is $S_0$. $S_0$ goes to $S_0u$ with probability of $p$ and to $S_0d$ with probability $1 - p$ at each time step $\Delta T$. Let $T = n. \Delta T$, where $T$ is the option expiration date. The value of the option can be evaluated for each point at the leaf nodes of tree (time $T$). The value of the option at starting node can be calculated through a procedure known as backward induction. A call option is worth $\max(S_T - K, 0)$, where $S_T$ is the spot
market price for underlying asset at time $T$.

The important advantage of buying options in comparison to other future agreements is that it gives the provider the right (not the obligation) to buy resources (outsource requests) in the future. Therefore, if a cloud client does not request the compute instances, the provider will simply let the contract expire without responsibility to buy unnecessary resources. The only cost for providers in such an arrangement is the premium paid at the beginning of the contract. This cost, however, can translate into trust and goodwill by the clients on the provider.

In Financial option theory model for buying VMs, providers transfer the risk of violating SLAs to other providers by buying option contracts and paying option premium. Therefore, sellers of the option contracts must consider the trade-off between the risk and expected profit.

This paper is organized as follows: Section 2 reviews the existing research related to cloud computing, its economic models and financial options theory in cloud pricing. Section 3 narrows down the focus of this research by formalizing the proposed framework. Section 4 is based on implementation of the proposed model in CloudSim simulator. Finally, we present conclusion and possible future directions.

2. RELATED WORK

It is sometimes hard to find a balance in which both sides agree with the price being set. A good pricing model is defined as a price that will carry no loss neither to the provider nor the consumer. From the consumer’s point of view, a better pricing model is one where they will pay a lower price for the resources requested, while from the provider’s point of view, they should not go beyond the lowest price that provides 0% profit for them as well as increasing the utilization.

Wang et al. investigated the problem to adjust spot price to maximize cloud providers’ revenue. They presented a demand curve model to capture the attributes of spot resources for current cloud computing. This model also presumed the impact of pricing on the future into consideration. The proposed algorithm, based on Lyapunov optimization, operates with and without any knowledge of the future requests [7].

Zhang et al. [8] considered the case of a single cloud provider and addressed the question to best match customer demand in terms of both supply and price in order to maximize the providers revenue and customer satisfactions while minimizing energy cost. The problem is modeled as a constrained discrete-time optimal control problem and by using model predictive control (MPC) to find its solution. Simulation studies using real cloud workloads indicated that under dynamic workload conditions, the proposed solution achieved higher net income than static allocation strategies and minimized the average request waiting time.

A Cloud Asset Pricing Tree (CAPT [9]) was designed, simulated and evaluated to find out the optimal premium price for cloud federated options. It provided the benefit to the service provider in terms of decision making i.e., when to buy options in advance and when would be the best time to exercise them, so that maximum savings can be made in provisioning the virtual machines. Financial option theory was employed as an interface to share an extra pool of federated resources whenever needed.

Sharma et al. [10] proposed a novel financial economic model being able to provide a high level of quality of service to consumers. The price was determined using this model represented the best possible price that the service provider is supposed to charge its customers to recover the capital costs. A lower boundary on the price that should be charged to customers is given by the financial option theory. The upper boundary of the price was calculated
using a proposed compounded Moore’s law. If the price is set between these two boundaries, it could be useful for both customers and service providers equally.

Rahman et al. proposed an approach that utilized financial option theory to alleviate risk and reduce cost for cloud users in spot markets at the same time. The cloud user optimization problem was formulated and mathematically characterized the cost of using European style options for clouds. A novel on-line policy using American options was proposed that overtakes standard spot policies in terms of price variance reduction against high risk factors [11].

Our approach differs from the existing approaches of optimal bidding for VM placement across multi-cloud. By employing the financial options model, we not only hedge against resource unavailability for deadline constrained jobs but we also utilize spot VM instances to reduce overall cost of job execution life cycle. No such model has ever been proposed in the literature.

3. MODEL FORMULATION AND METHODOLOGY

In a virtual cloud, cloud provider, underlining public cloud infrastructure, and cloud service provider realizing cloud resources/services may be different vendors. Resources of other cloud providers are normally borrowed to meet end user requirements. Cloud service provider does not itself own networking or data center resources. A cloud consumer can construct virtual cloud by leasing virtual machines from the cloud providers. A central entity, also known as broker, facilities multiple clouds to share resources. Cloud consumers can find best provider and service through the matchmaking process of cloud broker.

The primary design goal of our proposed system is to facilitate user job execution by automating the entire process on hand and achieve economic efficiency by exploiting low-cost spot VM on the other hand. The core components of the system are cloud customizer, broker at user end and server side cloud provider component.

Our proposed cloud customizer involves the following steps:

- **Job Admission**: Jobs are submitted by the users along with necessary information including task(s) to be executed, budget, deadline, deadline etc.

- **Runtime Estimation phase**: When a job is submitted, the broker estimates the job characteristics and schedule amount of processing nodes considering the workload requirements. Our Parallelism profile component uses Downey’s model to extract such features. More details about the Downey’s model is discussed in [14].

- **Discovery phase**: Cloud customizer queries a list of resource/service providers that satisfies the requirements.

- **Resource Selection**: Checks individual resource to confirm the service requirements. The cost of executing a task is obtained by querying cloud provider. Spot instances are given the highest priority for execution of non-critical jobs.

- **Resource switchover**: In case of early termination in case of Spot VM or low QoS in case of on-demand instances, provisioning of resources is re-evaluated to meet QoS and deadline constraints.
The scheduling module assigns task to the pool of virtual clusters according to the job specifications. It is responsible for completion of job execution within the budget and deadline. Since jobs are executed based on priority order list, we introduce the priority level (PL), given as equation 1, the maximum estimated time for job in a wait queue before the deadline is reached. If PL is negative, job deadline cannot be met. The greater the value of PL, the more is the chance to meet the deadline and hence job can be placed in the low priority. A small positive number indicates the resource must be provisioned immediately and job is considered to be of high priority.

\[
PL = \max(0, T_{\text{deadline}} - T_{\text{now}} - (\alpha * T_{\text{est}} + T_{\text{latency}}))
\]

where \( T_{\text{deadline}} \) is the job’s deadline specified by user, \( T_{\text{now}} \) is the current time, \( T_{\text{est}} \) is the estimated time of job as calculated by the job runtime estimation module, \( T_{\text{latency}} \) is the expected time to set up VM instances. \( \alpha \) is the sensitive factor \([14]\); higher values of \( \alpha \) will indicate the scheduler to provision on-demand instances as PL tends to zero. To ensure that jobs with low priority eventually executes, we use the concept of aging. Priority of low level job is gradually increased (upon every failure/ interruption) and hence a low level job may get high priority to complete its execution.

All incoming jobs are received by the job manager agent \( J \). Considering the job priority list, the job is enqueued in one of three job queue agents: \( J_{\text{high}} \) for high priority jobs, \( J_{\text{med}} \) for medium priority jobs and finally \( J_{\text{low}} \) for low priority jobs.

Medium and low priority jobs are executed on spot instances. Pricing bidding strategy is based on Amazon spot price history. \( J_{\text{low}} \) accepts jobs below a specific threshold \( \gamma \) which is set by the user. If \( \gamma = 3 \), user job can run at least twice on spot VM. Keeping track on the average spot pricing history of past 90 days, the bidding strategy for \( J_{\text{med}} \) is relatively aggressive (above average price) as compared to \( J_{\text{low}} \) so that the chance of spot termination due to out of bid would be relatively low. \( J_{\text{low}} \) accepts all jobs above \( \gamma \). Since the job has multiple chances for execution, \( J_{\text{low}} \) bids on relatively low price (around average price) considering the spot history.

For high priority and deadline constrained jobs, financial options are exercised to guarantee the execution of critical jobs. If job execution in spot instance is failed, the job is re-checked by our provisioning algorithm to place it either in spot instance queue or on-demand option queue according to its priority. High priority jobs are critical and deadline constrained and hence on-demand instances from cloud federation using financial options (future contracts) are utilized. We have implemented the proposed framework through simulation. For this purpose we used java based simulator CloudSim, a simulation framework used for modelling and
experimental simulation of Cloud strategies [9]. Since it only provides console based results to perform synthesis, so we have also used CloudReports that is an extension of CloudSim which provides GUI based statistical synthesis of results. CloudReports is wrapped around CloudSim [10]. In our experimental setup, we have tested our four policies by executing simulations multiple times in CloudReports. We performed a set of experiments by creating 10 to 140 VMs by an increment of 10. During the experiments we measured following variables:

1. Cost
2. Performance
3. Time

3.1 Algorithm for VM provisioning and job scheduling

```java
While (true) do
    while current time < next_schedule_time do
        queue incoming job j in J
        vms ← all VMs currently in pool
        for each j in J
            compute priority level (PL)
            if j is JA_{\text{A1}} then
                exercise options
                continue;
            else if j is JA_{\text{A2}} or JA_{\text{A3}} then
                P ← find average spot price history
                if \{j is JA_{\text{A3}}\} then
                    j bid = P + G
                end if
                if \{j is JA_{\text{A2}}\} then
                    j bid = P
                end if
                mwt ← maximum wait time for j
decision ← FindFreeSpot(j, vms), vms)
                end if
            if (decision allocated = true) then
                AllocateJobToVM(j, decision, VM)
            else
                add j to list J
                delay allocate time by mwt
            end if
        end if
    end while
end while
```

3.2 Option Pricing

For to hedge against the unavailability of computational resources for deadline constrained job(s), we used financial option based leasing strategy. In such scenario, resources are leased for time T by paying a premium price known as call options. In our case, for deadline constrained resource provisioning policies, we used options with the following configuration:

![Figure 2: Provided Parameters for Options Calculation](image)

Based on these parameters, the following binomial tree was generated and used in our experiment. Since the option values with ZERO premium prices are not realistic, we used option values ranging from 0.0019 to 0.0102 depending on supply and demand of VM resources.

![Figure 3: Binomial Tree of Options](image)

4. Results and Discussion

We used Amazon pricing data for our experiments given as under:

<table>
<thead>
<tr>
<th>VM Type</th>
<th>On Demand Cost ($)</th>
<th>Spot Cost ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>m3.medium</td>
<td>0.07 plus 0.001 per GB storage and network operations</td>
<td>0.01 -0.015</td>
</tr>
</tbody>
</table>
To evaluate an optimal policy based on deadline or budget constrained and cost effective optimal application centric resource provision scheme, we derived four different policies: (1) on demand resource provision scheme (2) Spot VM resource provision scheme (3) resource provisioning scheme with (user) feedback (4) runtime estimated resource provisioning scheme (without user feedback).

4.1 On Demand Resource Provision Scheme

Under such scheme we used a pool of on demand instances to complete job requirements. In our simulation environment, these resources were ranging from 10-140 VM requests. The simulation results based on CloudSim reports somehow show a linear relationship between number of VMs and overall cost incurred. The results are show in figure 4.

![Figure 4: On-Demand Instances Resource provision Scheme](image)

Although the job requirements were successfully met, overall cost was relatively high and such policy can only be implemented for deadline constrained jobs which is not a very frequent case in cloud computing environment. One important point that can be observed in figure above is the significant increase in price between the VM range 80 to 140. This is quite understandable as with the increase of VMs, the bandwidth and other operations tend to rise and hence as a result overall cost increases.

4.2 Spot VM resource provision scheme

In this policy, all VM requests were executed on Spot instances. In case of spot VM, termination due to relatively low bid, the job was rescheduled unless deadline schedule is not over. Furthermore, with the increase of Spot VM demand, the rejection rate was also more frequent as compared to low workload.

![Figure 5: Cost of VMs for Spot Instances](image)

Although the results are promising in terms of financial cost and many companies have cut down 50%-60% of their expenses using Spot VM instances, this scheme is not preferable for interactive and real time job allocation. A famous example is SEOMOZ’s “Crawler” where all spot VMs were terminated without any prior notice and SEOMOZ suffered huge financial loss [15]. As a lesson learned, the company had to decide a mix strategy consisting of both on demand and spot VMs to achieve better resource provisioning as well as minimizing overall cost.

Hence in this research, we introduced two new strategies to achieve resource utilization efficiency, meeting budget and deadline constraints and while keeping the cost low. These two policies are discussed in subsequent sections.

4.3 Resource provisioning scheme with feedback

In this scheme, an incoming job request can be classified into low, medium or high priority. Based on the job classification, all incoming jobs are enqueued in one of three job queues agents. For high priority jobs, priority level is set to some positive number below one to indicate that job can only
be run once. For such jobs the risk of unavailability of VM resources may result in job failure. For such jobs, options are purchased by paying a premium price and may be exercised, when required. This strategy minimizes the risk of job failure. Medium and low priority jobs are executed on spot instances. This scheme enables cost-aware task scheduling for budget constrained jobs.

The policy effectively utilizes VM resources based on job requirements and overall cost is reduced as compared to On Demand policy. However average wait time is somehow increased as Spot VMs were also used to gain economic benefit. Overall cost trend for 10-140 VMs is depicted in the following figure:

4.4 Runtime estimated resource provision scheme (without feedback)

In this policy, we created three threshold limits for job runtime estimation i.e., low, medium and high. Low threshold value indicates that the submitted job is not time bound and the system should reduce cost as much as possible by utilizing Spot instances. High threshold values indicate that submitted job is deadline constrained and the system must use an aggressive strategy to provision the resources with minimal delay. Overall strategy of resource allocation is in accordance with policy Resource Provisioning Scheme with feedback with the exception of job runtime estimation without user feedback. Figure below represents resource utilization of medium threshold limit:

This strategy further reduces overall cost as compared to resource provisioning scheme with feedback. Based on existing literature, it is obvious that user supplied job runtimes are mostly over estimated and hence job estimation runtime can be further optimized as adopted in this strategy [14]. Overall results of the four policies are presented as follows:
Our results indicate that on demand VM provisioning is not an optimal solution for resource provisioning as it may result in cost overrun. On the other hand Spot VMs, based on its unreliable nature may not be applicable for real time and deadline constrained jobs. Our two proposed policies, by taking advantages of the above, minimize these limitations resulting in better resource provisioning and VM allocation.

The results also indicate that policy four further optimizes overall resource cost by 15%-20% while hedging against the job failure.

5. Conclusion & Future Work

The results showed promising difference in cost of our two proposed policies. Unlike simulated environment, real experiments could be performed on the real data acquired by different cloud providers. Future workload prediction modelling strategies could be designed to predict the future workload of the consumer.

6. REFERENCES


Communications and Networks, pp 127-142
