Volume 5 Issue 2 December 2017
ISSN: 2321-6840

Cost Optimization Over Amazon EC2 Spot Instances-Research Challenges

Veena Khandelwal[1], Anand Kishore Chaturvedi[2] and Chandra Prakash Gupta[3]
[1]Rajasthan Technical University, Kota, Rajasthan, India.
Email: vn.khandelwal@gmail.com
[2]Rajasthan Technical University, Kota, Rajasthan, India.
Email: chaturvedi101@gmail.com
[3]Rajasthan Technical University, Kota, Rajasthan, India.
Email: guptacp2@redffmail.com

Abstract: Amazon EC2 Spot Instances (SIs) create a competitive low price computing facility. Spot prices are typically far below on-demand prices, and this makes them most suitable to lower the cost of tasks and potentially execute them when there are many spot instances available. However, using spot instances for potential cost savings is a challenging task. The volatility of spot instances deteriorates the QoS of users which in turn limits their usage and escalates the complexity of cloud environment. The surge in utility computing has introduced many trade-offs between price, performance and recently reliability. This paper presents vision and challenges in using spot instances.

Keywords: Amazon EC2, Challenges, Spot instances

I. Introduction

SPOT pricing was introduced by Amazon Elastic Compute Cloud (EC2) in 2009 as a way to help use all of its compute capacity to minimize operational cost, combat under utilization of resources and make more profit. Spot prices do not go beyond a minimum threshold to meet servers sunk cost and can even be tweaked to make people pay more for it. Amazon Web Service (AWS) is not the only participant in the spot instance realm. Google Compute Engine launched its preemptible Virtual Machines on September 8, 2015 designed for such type of workloads that can be delayed and are fault tolerant at the same time. Users can bid for Spot Instances (SIs) where prices are charged at lowest bid price, whereas, pricing on Google Preemptible VMs is fixed at per hour rate. The distinguishing feature of Amazon Elastic Compute Cloud (EC2) SIs is its dynamic pricing. Spot pricing lets companies bid for access to a compute instance for one hour at a set price. The market price of the same spot VM differs in the datacenters across the globe and is based on real time demand and supply of that particular type of resource. Customers whose bid exceeds or equal to it, get a hold of the available SIs and the allocated instances can be run until submitted bid price is higher than market price.

Cloud providers encourage spot instance usage by lowering spot prices when their data centers are lightly loaded. Similarly, cloud users seek to obtain the lowest possible prices for the instances they acquire, thus minimizing their costs. Startups use spot instances to save tens of thousands on certain workloads. This just doesn't cut costs; it also enables a company to do things that were previously cost prohibitive.

However, while gaining significant cost savings over using on-demand instances, users take risks on running spot instances without any availability guarantee, i.e. running SIs can be preempted by Amazon at anytime. When computing capacity is taken up by other instance types, especially the higher reserved instances, the spot capacity diminishes and prices might rise. Reliability of spot instance depends on the market price and the users maximum bid (limited by their hourly budget). Time flexible and interrupt tolerant applications including batch processing tasks, video encodings, testing, compute intensive applications, web crawling are well suited for execution on SIs. However, using SIs for potential cost savings while maintaining QoS is quite challenging. AWS SIs offer cost saving opportunities but might not be accessible when one needs them, making them somewhat inherently unreliable. It is hard to guarantee resource availability. In case of resource shortage, the provider terminates VMs of low bid and replaces them with higher bid VM requests or on-demand requests. Hence cost optimization using SIs becomes more challenging than on-demand or reserved instances.

Amazon EC2 spot pricing is different from other auction models. Unlike in a normal production model of resources where the supply varies with the price, in case of SIs demand does not influence the supply but only the price [1]. In spot pricing model, not all goods are identical: users can bid on different type's instances in different Amazon EC2 regions, running different operating systems. Moreover, once a bid is made for spot instance request, it cannot be changed during the VM instance's life time. Unlike Amazon EC2 spot prices, price traces of free markets do not have a threshold minimum price. Also fluctuations in a spot market are fewer than fluctuations in a free market.

Section 2 presents spot instance operation model. Section 3 presents some key open research challenges in adopting spot pricing model. Section 4 presents conclusion.

II. Amazon Ec2 Spot Instance Operation Model

Amazon charges SIs in integral hours starting from the time that a user's spot instance is granted. For each hour, Amazon charges market spot price at the beginning of that hour. If Amazon itself terminates SIs there is no cost for that hour. However, if user willingly terminates the SIs, Amazon charges the entire hour. A bid one user makes only represents the maximum price the user is willing to pay rather than the actual price the user pays for renting SIs. The actual cost a user pays is determined by market price. The market price is decided depending on the total bids at Amazon EC2 at any time an the spot pool capacity in a region at that point of time. Amazon sorts all received bids in a decreasing order, and use the spot pool capacity to find the cut-off price. Such cut-off price is called market price. All the bids above the cut-off price will be granted SIs. Once a new bid is received, Amazon re-sorts the bids and then a new market price is determined. No matter how high one bids, instances will be interrupted when the size of the spot pool drops to zero.

III. Research Challenges

1. Spot Price Distribution

Several distribution models are presentedby authors to establish a relationship between spot prices at different time for modeling EC2 spot price patterns and simplify research. Mazzucco et al. [2] employ a normal approximation to model the spot price distribution with the same mean and variance over 10,000 samples. The authors find that the normal approximation is adequate but not perfect as the historical data differ substantially from those of the approximations. Using Shapiro-Wilk test for normality Zhao et al. [3] analyzes that normal distribution is inadequate to approximate spot price data set. Javadi et al. [4] perform analysis using one year price history and observe bi-modality in probability densities of hourly and weekly price patterns, time duration between price changes. The authors observe that the ratio between mean and median is 1 and therefore model spot price and inter price time dynamics based on the mixture of Gaussian distribution with 3 or 4 components. According to the authors this model can be used to predict total price of running jobs using Amazon EC2 SIs. According to Bogumił et al. [5] due to the complex spot pricing mechanism this information is insufficient for decision making regarding bidding decisions. Karunakaran et al. [6] show that the lognormal distribution can adequately model the spot-price distribution, although more complex distributions (for example, a mixture of Gaussian) can provide a relatively better fit. Zafer et. al. [7] model spot price evolution as Markov model. However, Mazzucco et al. [2] find that the spot price variation in Amazon EC2 over time does not seem to follow any particular law.

Spot price distribution models presented by several authors are adequate but not perfect due to a number of factors affecting spot pricing mechanism. Firstly, distribution models usually thrust aside important temporal information about the lifetime of the spot price staying below any bid price. Therefore, they cannot capture well the continuity of SIs availability, which is of prime concern particularly for long jobs. Secondly, spot price also depends on the size of the spot pool available at any instant and the aggregate bidding behavior. Assumption that the spot price changes hourly is also unrealistic. In reality spot price changes at any instant. Thirdly, Amazon announces new pricing policies frequently. New regions keep coming up. This leads to the ineffectiveness of the previous analysis. In order to address the issue, it is essential that the newer non-parametric machine learning methods be used for spot price modeling. The methods should be accurate and fast enough to model spot price dynamics.

2. Prediction of Expected Lifetime of Spot Instance

Spot users need to know when to bid for spot instances in order to minimize cost while meeting constraints. To describe stochastic evolution of spot prices Guo et al. [8] model failure probability of spot instances for the next hour based on a semi-Markovian process of spot prices. Dubois and Casale [9] and Chohan et al. [10] describe the stochastic evolution of the spot price and its reliability using Markov model to predict the expected lifetime of SIs. Authors find that when maximum bid is lowered, some small increments in bid price can increase probability to a great extent whereas some increments increase probability by a very small amount. Yehuda et al. [11] show that spot instance availability grows linearly with bid price up to a certain level only. Thereafter, availability grows slowly and non-linearly with bid price. Out-of-bid failure requires applications to adapt their run time to be frequently interrupted. Computation of expected VM lifetimes can determine when to employ SIs for their application and assist in planning for SIs and taking checkpoints also.

3. Spot Price Prediction

Spot price prediction is challenging because there is a little or any information on the underlying operating mechanism of the spot market by Amazon EC2. However, it provides current and past 90 days spot price history data to assist their customers in bidding SIs. Accurate prediction of spot prices can reduce the risk of instance termination due to underbidding thereby increasing the reliability. Few authors have made efforts developing predictive models for spot price variations. Volodymyr et al. [13] present a predictive model for future short-term and middle-term spot price predictions based on neural networks. The authors count a prediction result as an outlier when its relative prediction error is more than 10%.

Their analysis of the short term prediction results shows that relative prediction error is about 4% and outliers up to 5%. Michele Mazzucco [2] makes use of Auto Correlation Function (ACF) to measure correlation between spot prices observed at different times. The analysis of ACF reveals that there is no correlation between spot prices observed at different time periods for different types of instances. The authors focused on achieving availability guarantee with SIs and used a quantile function of the approximate normal distribution to predict when the autocorrelation of current and past price is weak. When the autocorrelation is strong, a simple linear regression prediction model is adopted. However, Zhao et al. [3] find that such an approximation is inaccurate in some test cases and that it cannot be taken as a generic approach. Wolski et al. [14] propose Durability Agreements From Time Series (DrAFTS), a non-parametric statistical prediction algorithm that takes as input the required probability and computes minimum bid value and its availability duration with the given probability based on a time series analysis method QBETS. Zhao et al. [3] use a Seasonal ARIMA (SARIMA) model for spot price prediction using statistical analysis of past price history. It does not provide satisfactory prediction accuracy and cannot be used in practice for bid price. The authors also analyze the predictability of spot price and conclude that spot instance price cannot be well estimated to be used in the deterministic model. Singh et al. [15] make use of gradient descent algorithm to calculate the coefficients of global trend and local seasonality to dynamically forecast one day ahead spot price of any type of spot instances across all Amazon EC2 regions on hourly basis. Their model is capable of predicting the spot price within 9.4% of MAPE for next hour prediction and up to 20% MAPE for one to 5 days ahead forecast. However, gradient descent algorithm can take a long time to converge. Yehuda et al. [11] perform analysis of spot history traces in order to reengineer how spot prices are set so as to assist users in bid decision making. The authors present the need to apply different pricing models in different epochs of the pricing history since pricing mechanism changes notably and qualitatively during these epochs. The authors therefore divide the history traces to three contiguous epochs associated with different pricing policies and evaluate data from a single epoch at a time. Statistical forecasting methods with moving window and employing higher weights to most recent data have also been employed in the literature. Javadi et al. [4] present probability density functions in order to calculate future spot prices and the time interval when spot prices change after studying SIs through statistical models to characterize their behavior.

Spot prices change at unequal intervals. Although, time series forecasting methods after converting data into equally spaced time series data can be used to predict future prices, but since bid once made cannot be changed until the instance is interrupted, forecasts sufficiently ahead in time equal to job length is required in reality. The methods employed for spot price prediction are either slow, predict for the next hour only or are computationally intensive. Real application jobs can have long runtimes spanning many changes in the spot price which requires prediction of not only in the next time slot but also for all future time slots until the job is completed. The challenge can be addressed by smart, accurate and fast price prediction algorithms capable of forecasting spot prices far in advance.

4. Fault Tolerance

While the spot market can provide resources at low cost, jobs are terminated immediately if the current spot price exceeds the bid price. Users need to explicit implement fault tolerance measures to execute their applications. Fault tolerance allows completing task execution in the presence of faults. Volatility of resources affects fault tolerance overhead costs. Higher the availability requirement, higher is the fault tolerance required. Fault tolerance is one of the challenges that spot users are facing in order to achieve cost savings for availability critical jobs such as web applications. High performance computing applications running on the cloud need to ensure that compute intensive applications run smoothly with reduced overhead time and visibility of the cloud infrastructure environment.

a. Check-Pointing

Various check pointing schemes namely Hourly, Rising Edge, Adaptive, Application Centric, and Threshold have been proposed by authors to provide fault tolerance. Each check-pointing scheme suffers from one or more limitations such as storing unnecessary check-points thereby increasing overheads cost, ambiguity in completion time. Some check-pointing schemes are better for higher bid price whereas others for lower bid price while in some check pointing schemes rollback time may become too long. Bogumił et al. [5] find that there is a need of less frequent check-points in lower capacity instance types which are more stable than higher capacity instance types that are highly volatile. Check-pointing can be useful only when there is a high variation in spot price. When the spot prices are stable, there is no need for check-pointing. Moreover, it can be adopted for applications that have significant slack time available and have low check-pointing cost in order to meet both monetary and deadline constraints. Computation states are suspended during check-pointing. Therefore, check-point/restart increases the wall clock time of the application execution which increases the execution cost. As the time for check point varies linearly with memory size of VMs, for applications that have large memory footprints whose check-pointing time cannot be neglected, when to checkpoint and whether to check-point or not are the two major challenges. Frequent check-pointing increase overheads whereas less frequent check-pointing increases recovery time. Bid price adopted also needs to be taken into consideration when adopting any check-pointing strategy.

b. Process Level Redundancy (PLR) Techniques

SIs have the ability and flexibility to provision computer resources for HPC scientific applications that must complete within a user-defined time bound having non zero slack, the

amount of which depends on the context in order to be useful. For HPC applications with little slack or high checkpoint cost, or during times of spot price volatility, bidding only in a single zone can cause low instance availability, high rollback costs. Process level redundancy techniques are used to reduce failure rate and maximize performance of HPC applications on spot instances given a time constraint. Marathe et al. [16] exploit redundancy as the complementary fault tolerance mechanism at different bid prices across multiple Amazon EC2 zones to obtain higher availability. EC2 auctions computational resources at different data availability zones at independent bid prices. The authors show that, despite higher up-front cost, redundancy often results in lower total application cost than their non redundant counterparts because of less frequent out of bid failures and therefore lower check point frequency. Guo et al. [8] bid at lowest upper-bound of bid prices from various zones for spot instances in multiple Amazon EC2 zones in a greedy manner in order to ensure availability of SIs equal to on-demand instances for highly available distributed lock service and reducing cost at the same time while taking advantage of free computation. Designing process level redundancy techniques that reduce overhead cost and at the same time provide fault tolerance to applications is a real challenge.

c. Migration

Virtual machine (VM) migration in multi-cloud scenarios involves moving of a VM from transient sever to stable server in order to achieve availability. Migration time includes VM's memory and disk size. Migration eliminates waiting time of acquiring resources again after failure and therefore effectively reduces the job completion time without significant cost increase. However, it usually causes undesired overheads and performance degradation in the application that runs in the migrated machine or in the whole system. For flexible applications, it allows to change the instance type after a failure and requires bidding for another instance type at a comparable cost. New instance type to bid after out-of-bid failure depends on the application requirements and constraints. Moreover, migration requires the in-memory state of the instance to be transferred consistently and efficiently with integrated consideration of resources for applications and physical servers. Migration is only feasible if an advanced warning long enough to enable transfer is received by the transient server. Presently Amazon EC2 provides 2 minutes warning notice which is insufficient for high infrastructure instances. Efforts are required to minimize the disadvantages of migration which include migration time, application downtime and performance degradation. New solutions for assuring a certain system performance are required that can prevent users from moving the whole set of resources but only moving some part of them keeping the rest unchanged. Spot prices vary along time. Hence, the question is how users can take advantage of price variations for deploying their infrastructure, even knowing that dynamic prices are unknown for them. The algorithms should be based on prediction model which take into account the historical prices of available regions. Using these predictions, calculate for the next hour deployment, the best region to move the VMs to.

V. Bidding Strategy

Bidding strategy plays a significant role when SIs are to be employed for computation. Although spot prices are considerably cheaper than on-demand prices, but there are intermittent time occasions when these prices escalate up to a very large amount. SIs may not be available at user's bid price when one needs them which could lead to poor response time and high deadline misses. User's bid affects the distribution of out-of-bid failures. The major challenge for the customers using SIs is to figure out a bidding strategy that match economic and compute goals. Bid high to ensure volatility or bid low to optimize costs while using on-demand or reserved instances when price rise above a threshold.

A number of static, dynamic, truthful, non truthful bidding strategies have been proposed in the literature. Karunakaran et al. [9] perform an analysis of the various provider recommended bidding strategies. The authors find that albeit bidding above on-demand seems attractive, have fewer interruptions, improve the probability of bid success and reduce wait time but there is a very little reduction in wait time in comparison to bid price increases that are lower than on-demand. A study of bidding strategies reveals that the type of bidding strategy to adopt depends on the application requirements, its fault tolerance, fault tolerance mechanism adopted, availability level desired, deadline and monetary constraints and job lengths to execute on SIs. These factors add to the complexity and make bid decision making a complex task. The bid amount for SIs cannot be easily determined. Spot price depends on the unused capacity in Amazon EC2 which is influenced by many factors like availability zone, time of the day etc. Using a very high bid amount in order to guarantee instance availability can have very unexpected financial implications. Spot prices can rise unexpectedly high from either more demand for SIs or an increase in requests for on-demand instances. This affects cloud users on Amazon EC2 where users are now bidding extremely high prices for scarce compute capacity. Bidding strategy should ensure that the we do not pay more for a server on the spot market than what we are willing to pay. Never bid more than on-demand price because when the price starts to spike it will easily go beyond any rational price. Spikes in spot pricing can be efficiently dealt with by writing price-aware code that can manage cloud infrastructure so that it responds intelligently to spot market price spikes and automatically request an on-demand server when spot prices rise above a certain reasonable price.The graph shown in Figure 1 for c48*xlarge spot* instance pricing from March 2015 to April 2016 in us-east-1a region has spot price a number of times higher compared to on-demand instance price.

Bidding strategies to present bid values at different time (normal, peak and low usage hours) are required that can balance execution cost and reliability for both long and short jobs.

Out-of-bid failure probability can be reduced using dynamic spot price prediction algorithms that could forecast spot price far in advance while at the same time maintaining monetary cost under control. Availability of SIs depends on user's bid, budgetary constraint and job length. The influence of job size on reliability is higher, especially for low bid prices. Reliability of job execution decreases with the job size. Bidding strategies that take into consideration job size can ensure significant cost savings and reliability as well.

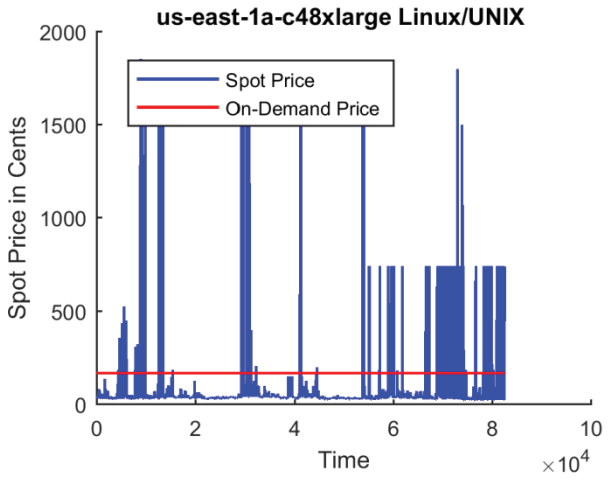

Fig. 1: Spot Price History Traces Showing Unexpectedly High Spot Prices

V. Multiple Regions and Instance Types to Choose

Amazon web service is geographically diversified into multiple regions and each region comprises multiple smaller geographic areas called availability zones in order to reduce latency to end-users based in different parts of the world. A spot capacity pool is a set of available EC2 instances that share the same region, availability zone, operating system (Linux/Unix or Windows) and instance type. Each EC2 spot capacity pool has its own availability which is the number of instances that can be launched at any particular moment in time and its own price, as determined by supply and demand. Multidimensional models such as Markovian models for spot price prediction suffer from scalability problems while considering multiple Amazon EC2 markets for ensuring profitability and availability. Ben-Yehuda et al. [14] showed that different supply and demand conditions exist in different regions. Bogumił et al. [8] in their study of finding the most important factor among zone, machine and reserved price multiplier for bid price observe that the most important variable that influences the bidding decision strategy is the availability zone. Moreover Amazon EC2 provides multiple instance types based on computing/memory capacity, operating System (OS)/ platform type. Identifying the least volatile region to ensure longest continuous running spot instance that can save a bundle using spot instances in the long run is a real challenge. High prices and a high degree of price fluctuations over time clearly indicate that many users are bidding for capacity in the same pool. The issue can be addressed by analyzing past spot price history traces of multiple capacity pools (regions) to seek out pools that have lower prices and more stable prices (both current and historic) to find bargains and lower interruption rates.

Amazon has different types of instances to choose from. Selecting the right type of instance is very crucial in optimizing cost and finishing execution of workload within budget and within deadline constraints. Also it appears that instances are indeed virtual machines that are potentially compatible. However the prices for large and extra-large instances change at different times, and also show uncorrelated price changes. Bogumił et al., [8] find that running short simulations is cost effective on higher virtual machines while long simulations on stable machines that have lower infrastructure due to less frequent check-pointing and its overheads.

VI. Finding Optimal Resource Utilization Strategy for Workflow Scheduling

Workflows define computational components, data and their dependencies in order to make them easier to execute automatically, improving the application performance and reducing the time required to obtain scientific results. Significant cost savings can be achieved for scheduling scientific workflows on cloud infrastructure by effective bidding of SIs and using on-demand instances to meet deadline constraints. Poola et al. [17] propose robust schedules that can minimize the execution cost of the workflow while satisfying deadline constraints by evaluating the critical path and computing the slack. If there is sufficient slack, SIs can be exploited. In order to meet job's deadline, when there is no slack, jobs scheduled on SIs are transferred to on-demand instances to reduce the cost of execution whilst meeting the workflow deadline.

Usage prediction and dynamically provisioning the cloud infrastructure is a research challenge. Provisioning requires strategies that can estimate the likelihood of completion of long jobs on the required instance type with reliability, decide the amount of on-demand and SIs to purchase simultaneously in the most economic fashion. In order to prevent degradation of performance due to VM instance provisioning and booting delay in the cloud, a proactive prediction of future demand of resources to dynamically scale in advance to adapt with the changing workload is required. Several works make use of machine learning and statistical methods like linear regression in order to predict resource requirements that satisfy future resource demands. Even if the future resource demand for different usage hours such as peak, normal and low is accurately predicted and approximate spot price probability distribution during different periods are obtained, usage of too many SIs cannot ensure QoS. There exists a tradeoff between high costs (on-demand) and risk (spot). The challenge is how to combine spot and on-demand instances for job execution.

VII. Conclusion

Market-based cloud systems with spot instances offer significant cost savings in providing cloud infrastructure. However, with the changing market conditions and real time volatility of SIs achieving these cost savings in reality has a number of challenges. We have explored few challenges which need to be overcome for utilizing SIs in a better way. The study opens up several interesting areas of exploration for researchers.

References

[1] D. Saha, "How small and medium enterprises (SMEs) should bid for spot instances of amazon's EC2 cloud," *International Journal of Business Data Communications and Networking (IJBDCN)*, vol. 10, no. 4, pp. 43-59, 2014.

[2] M. Mazzucco, and M. Dumas, "Achieving performance and availability guarantees with spot instances," *2011 IEEE International Conference on High Performance Computing and Communications,* Sep. 2011.

[3] H. Zhao, M. Pan, X. Liu, X. Li, and Y. Fang, "Optimal resource rental planning for elastic applications in cloud market," *2012 IEEE 26th International Parallel and Distributed Processing Symposium,* May 2012

[4] B. Javadi, R. K. Thulasiram, and R. Buyya, "Characterizing spot price dynamics in public cloud environments," *Future Generation Computer Systems,* vol. 29, no. 4, pp. 988-999, Jun. 2013.

[5] B. Kamiński, and P. Szufel, "On optimization of simulation execution on Amazon EC2 spot market," *Simulation Modelling Practice and Theory,* vol. 58, pp. 172-187, Nov. 2015.

[6] S. Karunakaran, and R. P. Sundarraj, "Bidding strategies for spot instances in cloud computing markets," *IEEE Internet Computing,* vol. 19, no. 3, pp. 32-40, May 2015.

[7] M. Zafer, Y. Song, and K.-W. Lee, "Optimal bids for spot vms in a cloud for deadline constrained jobs," *2012 IEEE Fifth International Conference on Cloud Computing,* Jun. 2012.

[8] W. Guo, K. Chen, Y. Wu, and W. Zheng, "Bidding for highly available services with low price in spot instance market," *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing - HPDC,* 2015.

[9] D. J. Dubois, and G. Casale, "OptiSpot: Minimizing application deployment cost using spot cloud resources," *Cluster Computing,* vol. 19, no. 2, pp. 893-909, Apr. 2016.

[10] C. Castillo, M. Spreitzer, M. Steinder, A. Tantawi, and C. Krintz, "See spot run: Using spot instances for mapreduce workflows," *HotCloud,* vol. 10, pp. 7-7, 2010.

[11] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, and D. Tsafrir, "Deconstructing amazon EC2 spot instance pricing," *ACM Transactions on Economics and Computation,* vol. 1, no. 3, pp. 1-20, Sep. 2013.

[12] S. Tang, J. Yuan, and X.-Y. Li, "Towards optimal bidding strategy for amazon EC2 cloud spot instance," *2012 IEEE Fifth International Conference on Cloud Computing,* Jun. 2012.

[13] Turchenko, Volodymyr, et al., "Spot price prediction for cloud computing using neural networks," *International Journal of Computing,* vol. 12.4, pp. 348-359, 2014.

[14] W. Rich, and J. Brevik, "Providing statistical reliability guarantees in the AWS spot tier," *24th High Performance Computing Symposium,* 2016.

[15] V. K. Singh, and K. Dutta, "Dynamic price prediction for amazon spot instances," *48th Hawaii International Conference on System Sciences,* Jan. 2015.

[16] A. Marathe, R. Harris, D. Lowenthal, B. R. de Supinski, B. Rountree, and M. Schulz, "Exploiting redundancy for cost-effective, time-constrained execution of HPC applications on amazon EC2," *Proceedings of the 23rd international symposium on High-performance parallel and distributed computing - HPDC,* 2014.

[17] D. Poola, K. Ramamohanarao, and R. Buyya, "Fault-tolerant workflow scheduling using spot instances on clouds," *Procedia Computer Science,* vol. 29, pp. 523-533, 2014.