How Not to Bid the Cloud

Prateek Sharma, David Irwin, Prashant Shenoy
University of Massachusetts Amherst

Abstract

Cloud providers have begun to allow users to bid for surplus servers on a spot market. These servers are allocated if a user's bid price is higher than their market price and revoked otherwise. Thus, analyzing price data to derive optimal bidding strategies has become a popular research topic. In this paper, we argue that sophisticated bidding strategies, in practice, do not provide any advantages over simple strategies for multiple reasons. First, due to price characteristics, there are a wide range of bid prices that yield the optimal cost and availability. Second, given the large number of spot markets, there is always a market with available surplus resources. Thus, if resources become unavailable due to a price spike, users need not wait until the spike subsides, but can instead provision a new spot resource elsewhere and migrate to it. Third, current spot market rules enable users to place maximum bids for resources without any penalty. Given bidding's irrelevance, users can adopt trivial bidding strategies and focus instead on modifying applications to efficiently seek out and migrate to the lowest cost resources.

1 Introduction

Cloud platforms now sell surplus idle server capacity at discounted prices to users to gain additional revenue. Amazon EC2 uses a market mechanism to sell this capacity where users place a bid for servers, and EC2 allocates them if the bid is higher than the spot price, which varies continuously based on supply and demand. When the spot price rises above a user's bid price, EC2 revokes the servers. EC2 determines the spot price by running a sealed-bid multi-unit uniform price auction [3]. Note that the underlying supply of surplus servers in the spot pool also changes, since EC2 may take resources from the spot pool to allocate new on-demand or reserved instances. Thus, the spot price changes dynamically both as users submit new bids, and as the spot pool's capacity changes.

Amazon conducts a second-price auction for their spot instances. Users place a single, fixed bid, which represents the maximum hourly price that they are willing to pay. The market price is based on all the bids and the available supply. Importantly, all users pay the same market price, which may be lower than the bid. If the market price increases above the user's bid, then the spot instance is revoked after a small (120 second) warning.

Spot price dynamics and the potential of unexpectedly losing resources introduces additional new complexities, which applications are typically not designed to handle. Addressing these complexities is an active research area. In particular, there has been substantial research on "optimal" bidding strategies for various applications and scenarios [12, 13, 18, 20]. In general, a bidding strategy determines the lowest bid price that ensures an application satisfies a performance target with high probability, e.g., finishing within a deadline. EC2 publishes three months of spot price history—and there are archives over multiple years—which prior work analyzes extensively to model price characteristics [5, 8, 15, 17, 19].

Designing bidding strategies can be highly complex, especially if a workload is distributed and users have to bid on many resources. In this case, requesting multiple units of the same resource with the same bid is risky, since all resources are governed by the same spot price, such that if one resource is revoked, they all are revoked. To reduce the probability of concurrent revocations, users might either spread their requests across many different resource types with different spot prices or place many different bids for different units of the same resource type. Bidding's complexity may be one reason why, despite its extremely low prices (50-90% less than on-demand instances), the spot market has low utilization [4].

EC2's cloud has attempted to reduce complexity by introducing tools, such as SpotFleets, which enable users to specify bidding policies that apply to large groups of resources from different markets. SpotFleets also includes default bidding policies for users that do not want to design their own policy. However, while bidding is a complex problem in theory, we argue that it is not a significant problem in practice due to at least three reasons.

Wide Range of Optimal Bids. Our spot price data analysis shows there is a wide band of bid prices that all yield optimal results, such that any bid within this range has a similar cost and availability as highly sophisticated bidding strategies. One reason this is not readily apparent is that prior work often compares the cost and performance of a bidding strategy to using higher-priced on-demand servers. However, in today's market, with low and stable prices, bidding strategies need not be sophisticated to reap significant savings compared to on-demand servers. Related work should instead compare their performance and cost with "dumb" bidding strategies.

Resources Always Available. Due to the large number of spot markets and their size, there are always many markets available where prices are low and stable, even when some markets are experiencing price spikes. Hence, upon revocation, a simple strategy that provisions a new spot server in another spot market and migrates an application to it is better than waiting for a spot price spike to subside. This migration approach nearly eliminates the unavailability of spot servers and reduces the practical impact of using bidding as a tool to control availability.

No Penalty for High Bids. Current spot market rules permit users to bid the maximum allowed bid price within each market with no penalty. Thus, sophisticated users can ensure extremely high availabilities on spot instances by placing maximum bids with little or no probability of paying a high price if the spot price were to rise.

Finally, not only do different bidding strategies yield little difference in their performance and cost, but some of our insights above are reflected in the default bidding strategies for EC2's SpotFleets tool [2]. Thus, Amazon is already nudging users to employ simple bidding strategies [1]. Based on these insights, we argue that users should ignore the potential complexity of bidding, and simply procure cheap EC2 spot servers using simple bidding strategies that we outline (or using Amazon's tools to employ such strategies). Rather than focusing on bidding, researchers should instead focus on modifying applications i) to gracefully handle unexpected resource revocation and allocation and ii) to efficiently seek out and migrate to the lowest cost resources. Selecting the best spot server to use at any time, i.e., the one with the lowest cost and best performance, is the primary problem that applications must address when using variable-priced resources. That is, if a resource's price rises significantly, then applications should be flexible enough to simply migrate to lower cost resources elsewhere in the cloud. For applications willing to adopt it, this approach can yield significant cost savings with little performance impact.

2 Background and Related Work

Since EC2 introduced its spot market, there has been significant research both on analyzing and modeling spot prices and developing bidding strategies based on real data and models. One of the first papers analyzing spot price data raised questions about whether EC2's mechanisms for setting the spot price were market driven [3]. However, as the authors note later, the characteristics of the spot price changed, making it consistent with a market driven allocation [3]. A number of related papers also analyze spot price data to better understand its statistical characteristics [5, 8, 13, 15, 17, 19]. Analyzing and modeling spot price data is a prerequisite to developing bidding strategies that select the optimal bid to ensure a target level of performance at the minimum cost.

Recent work focuses on optimal bidding for MapReduce jobs. In [20], the authors focus on selecting a bid such that, with high probability, the completion time on spot instances is less than twice the running time on on-demand instances. The paper examines multiple scenarios: quitting job execution upon revocation, or making persistent requests, i.e waiting until price drops to resuming execution. In all variants, the work only considers bidding in a single spot market: if the price rises too high and instances are not available the MapReduce job must either quit or continue processing with fewer resources.

As we discuss, EC2 (and the cloud in general) is large enough that resources are nearly always available somewhere. Thus, unless an application is highly optimized for specific types of server architectures (which MapReduce is not) or has geographical constraints, waiting for the price of resources to drop is unnecessary. Related work makes similar assumptions about market constraints but focuses on different applications. For example, prior work develops bidding strategies for jobs with deadlines [18], such that it chooses a bid for a particular spot market so the job finishes before its deadline with high probability.

Restricting the problem to only a *single* spot market has also resulted in prior research focusing on the wrong price characteristics. Specifically, if restricted to a single spot market, the only important characteristic is availability, or the percentage of time the bid price is below the spot price. However, if we assume applications should not restrict themselves to only a single spot market, then availability is no longer important, as other cloud resources are available in other markets. In this scenario, the frequency of revocations is the primary attribute that affects performance, since every revocation incurs an overhead to request a new instance and migrate to it.

Unfortunately, modeling revocations is not as straightforward as modeling availability. Modeling availability simply requires fitting a probability density function (PDF) to a histogram over different spot prices, which

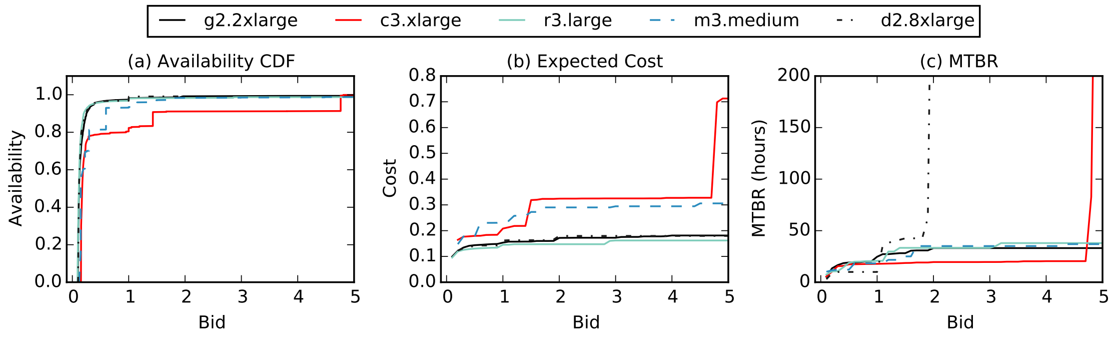

Figure 1: *The effect of bidding on availability, expected cost, and MTBR for selected instance types. Bids and the expected costs are normalized to a factor of the corresponding on-demand price.*

gives a probability the spot price is equal to a particular value. The corresponding CDF then directly gives availability, which is equal to the probability the spot price is above a given value. Prior work models availability using both Pareto and exponential distributions [20]. In contrast, revocations are discrete events with inter-arrival times that are not cleanly captured by a single number. As in any queuing model, the distribution of inter-arrival times is also important. However, the frequency and distribution of revocation events is a function of the bid, and may be different at different bid prices. Even though revocations are the primary attribute that affects performance, we know of no prior work that models the distribution of these events at different bid prices in EC2.

Finally, in many cases, as in [12, 18], bidding strategies are with respect to idealized spot price distributions, e.g., mixed Gaussian, exponential, Pareto, etc., and not real data. These idealized models are often based on examining only a few markets even though thousands of spot markets exist, which have vastly different characteristics. These characteristics are not likely captured by a one-size-fits-all model. Further, as [3] notes, price characteristics may change frequently due to changes in EC2's supply, demand, or its pricing algorithm, which may render models based on prior data unreliable. In many cases above, proposed solutions actually depend on the type and attributes of the particular model used in the analysis. As we discuss, though, the bidding problem in today's market (and possibly in future markets) is a red herring that is not particularly important for maximizing performance and minimizing costs using spot instances.

3 Do Optimal Bidding Strategies Matter?

To understand whether (and how much) optimal bidding strategies matter in EC2, we conduct a data-driven analysis of spot price data over a six month period from March to August 2015 (and longer periods where stated), as well as show aggregate statistics from *every* EC2 spot market. For ease of exposition, we focus on the most popular instance types in the most popular region, i.e., Linux instances in the us-east-1 region.

Bidding strategies optimize the cost-availability tradeoff for spot instances: as a user increases their bid, they may pay more per-hour, but their availability also increases. However, spot price data across many markets shows that there is a wide range of "optimal" bids that essentially yield the same availability for the same cost. To illustrate, Figure 1(a) shows a CDF of availability for instance types in five different markets over our six month period, where the x-axis is a user's bid normalized to the on-demand price, i.e., 1 is 1$\times$ the on-demand price, 2 is 2$\times$ the on-demand price, etc. As expected, availability monotonically increases with the bid. However, in each case, the CDF has a steep incline followed by an extremely long tail, such that there is little increase in availability after some bid threshold and only bids that fall within the steep range of the incline yield different availabilities. As the graph shows, this range of bids is quite small, providing only a narrow window where changing a bid will have a significant effect on availability.

Similarly, Figure 1(b) shows the cost a user would pay for the same instance types and the same bids. In this case, the cost on the y-axis is a fraction of the on-demand cost, i.e., 0.5 means the expected cost is 0.5$\times$ the on-demand price. As with availability, the cost is monotonically increasing with the bid amount. However, just as with availability, the cost curve has a long tail, such that higher bids result in little or no increase in cost. The only exception in these markets is the c3.xlarge instance type, which experiences two abrupt increases in cost at bid levels of 1.2$\times$ and 4.75$\times$ the on-demand price. The other instance types have nearly the same cost regardless of the bid level. This occurs because most markets always have a low and stable spot price, with the average spot price $<$0.2$\times$ the on-demand price. Just as with availability, bidding has little effect on the cost of spot instances.

Finally, as we discuss in the previous section, the frequency of revocations, as indicated by their mean-time-between-revocations (MTBR), is another important metric, since revocations incur overhead for applications that migrate to other available resources. Thus, Figure 1(c) shows the MTBR for different bids. The figure shows that

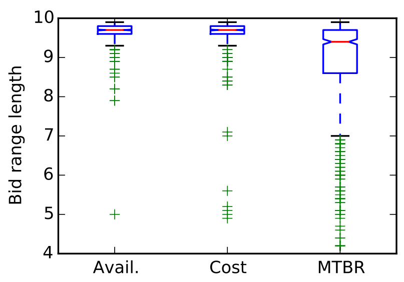

Figure 2: *Range of bids for which availability, cost, and MTBR is within 10% of optimal across 1500 markets.*

MTBRs range from tens to hundreds of hours. In addition, the MTBRs also have a long tail in all but one market, such that bidding high does not significantly increase the MTBR and there is a wide range of bids that effectively yield the same MTBR.

While the analysis above uses only five spot markets as illustrative examples, we analyzed these properties in over 1500 spot markets over our six month period. Figure 2 plots the range of bids such that any bid within the range is within 10% of the optimal bid for availability, cost, and MTBR. The optimal bid is simply the bid that yields the highest availability and MTBR for the lowest cost. Thus, we consider every bid within the range as effectively optimal that yields near the same result. As above, the y-axis is the length of the bid range as a factor of the on-demand price. Thus, a bid range length of 2 indicates a range of $[b, b+(2*D)]$ for some bid b where D is the on-demand price. A smaller range indicates higher bid sensitivity, where an application should carefully select a bid from a small range of near-optimal bids. In contrast, a larger range indicates a low bid sensitivity.

We see from Figure 2 that the bid ranges for the availability, cost, and MTBR are generally quite large, with a bid range near 9. Note that EC2 imposes a maximum bid of 10$\times$ the on-demand price. These results suggest that picking nearly any bid within the range of allowed bids yields the same optimal result. Put another way, users would need to "try hard" to make a "bad" bid by selecting a bid price that is exceedingly low compared to the average spot price. *Thus, in today's market, due to low prices (resulting in high availability) and price stability (resulting in long MTBRs), spot revocations are rare, but unavoidable, regardless of a user's bid.*

4 Beyond Bidding

Based on our analysis in the previous section, we argue that users should focus less on bidding and instead adopt the following simple strategy when using spot instances: i) employ a simple bidding strategy that selects a high bid price equal to the on-demand price, when requesting one or more spot servers; ii) if a server is revoked, simply seek out a different spot market with a lower price and request new servers (and if no spot markets have low prices, request an on-demand instance); iii) migrate application state to the newly acquired server and resume. We examine the feasibility and benefits of this simple strategy above as an alternative to sophisticated bidding.

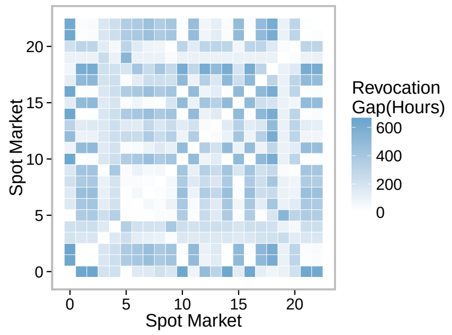

Figure 3: *Revocation gap between different EC2 availability zones and instance types in the us-east-1 region.*

As mentioned in Section 1, SpotFleets partially encodes our simple bidding strategy, as its default policy is to bid the on-demand price. If users are willing to migrate applications, bidding above the on-demand price is not cost-effective, as users can simply migrate to an on-demand instance once the spot price rises above the on-demand price. The only reason to bid above the on-demand price would be to gain advance notification and additional time to migrate to an on-demand instance.

Interestingly, under current spot market rules, users can actually bid the maximum bid price without penalty by simply monitoring the spot price and migrating to an on-demand instance once the price exceeds the on-demand price. Since EC2 only charges based on the spot price at the start of each hour, a user would never incur their high bid price, and, even if EC2 charged at a finer granularity, the user would only incur the high price for the small time window required to vacate the spot instance. The only way to prevent such gaming is to hide real-time spot prices from users, and only publicly release them much later. However, hiding spot prices would likely further discourage users from using the spot market.

Market Correlations. We examined the correlations of price variations between markets by studying price histories for the same spot server across different availability zones within a region, as well as across different types of servers within the same availability zone. We found that price of the same server type across availability zones and those across server types is largely uncorrelated. A consequence of the lack of price correlation is that revocation events are separated in time across spot markets—when one spot market experiences price volatility and price spikes, other markets are often unaffected. This is demonstrated in Figure 3, which shows the average revocation gaps between pairs of markets for the us-east-1 region.

The revocation gap is on the order of hundreds of hours between some markets—thus they are effectively independent, and can be treated as independent failure domains. This observation has two important consequences. First, if a single node application running on a spot server sees a revocation, it is feasible to find another spot server type (of equal or greater size) at a low price with high probability. The application can then be resumed on the new server. Second, for distributed applications that run on multiple nodes, it is beneficial to distribute the application components across *different* (and uncorrelated) spot server types. Doing so ensures a revocation event only results in the application losing a fraction of the nodes, rather than all of its nodes with homogeneous servers. The distributed application can procure new servers from other spot markets while continuing to execute on the unrevoked servers.

Migration Strategies. Migration strategies are key in the light of our proposed strategy to request new spot servers in a different market and to resume the application on the new servers. We can treat these revocation events as fail-stop failures. By periodically checkpointing state to network storage, the application can resume from the most recent checkpoint [9, 6, 7, 14]. Migration strategies are also feasible using live [10] or bounded-time migration [11] of nested VMs [16]. Thus, research should instead focus on determining efficient checkpointing and migration strategies, rather than optimal bidding strategies, to exploit cheap revocable spot servers.

Since job completion time can increase substantially due to frequent revocations, research should also focus on reducing revocation frequency by judiciously choosing between several different markets and choosing one (or more) with a low volatility. The expected completion time, revocation frequency, and cost can be estimated based on spot price data and bid prices, enabling applications to minimize revocations by determining how to optimally use the mechanisms above.

Exploiting Arbitrage. Our analysis also shows that current spot markets are "inefficient" at pricing resources. For example, we observe long periods where larger servers have normalized spot prices that are lower than smaller servers (presumably since demand in the latter market is greater than the former). From Figure 1, `r3.large` spot instances are about 0.15× their on-demand price, whereas the `d2.8xlarge` spot servers are about 0.25× their on-demand price. This price differential can serve as an arbitrage opportunity, enabling the use of more powerful spot servers at low cost. Judicious selection and migration to spot servers with the lowest cost are important in exploiting such arbitrage.

5 Future of Spot Markets

Will markets get more volatile? We have examined price data over the past six years (in addition to our six month traces) and found that bidding has never been a significant problem throughout the history of EC2's spot market. For example, as shown in Figure 4, while the average spot price of the `m1.large` instance type since its inception has decreased (in accordance with decreasing on-demand prices), the spread of spot prices has not increased significantly either. However, while our analysis of historical spot price data leads us to conclude that bidding has never been an important problem, it is possible that it may become an important problem in the future if price characteristics change.

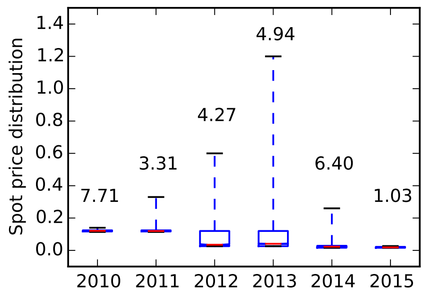

Figure 4: *Spot price distribution for `m1.large` over the years. The number above each boxplot denotes the skewness of the distribution.*

Will prices rise? A substantial increase in demand will undoubtedly cause an increase in average spot prices and any substantial price increase will cause price-sensitive spot users to become "priced out" of the market (which in turn may reduce demand and cause prices to drop). The second-order effects due to widespread adoption of the migration strategies we propose remains unclear, and a rigorous analysis, through game-theoretic or other means, is an open question. However, anecdotal evidence suggests that such effects may not come to pass—due to the significant capacity additions being made by all cloud providers on a regular basis, implying that there may always be some surplus capacity despite increasing demand in both the spot and on-demand markets.

6 Conclusion

In this paper, we dispel the notion that bidding significantly affects the availability and cost of spot instances. In particular, we show that the availability, cost, and revocation rate of spot instances based on spot price history are largely constant across a wide range of bids. Thus, instead of optimizing bidding strategies, we argue users should focus instead on modifying applications to efficiently seek out and migrate to the lowest cost resources.

Acknowledgements. We thank all the reviewers for their insightful comments. This work is supported in part by NSF grants #1422245 and #1229059 and by a Google faculty research award.

References

[1] Ec2 Spot Bid Advisor. https://aws.amazon.com/ec2/spot/bid-advisor/, September 2015.

[2] Ec2 Spot-fleet. http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-fleet.html, September 2015.

[3] BEN-YEHUDA, O., BEN-YEHUDA, M., SCHUSTER, A., AND TSAFRIR, D. Deconstructing Amazon EC2 Spot Instance Pricing. *ACM TEC 1*, 3 (September 2013).

[4] HIGGINBOTHAM, S. Bidding Strategies? Arbitrage? AWS Spot Market is where Computing and Finance Meet. Gigaom, October 8th 2013.

[5] JAVADI, B., THULASIRAM, R., AND BUYYA, R. Statistical Modeling of Spot Instance Prices in Public Cloud Environments. In *UCC* (December 2011).

[6] KHATUA, S., AND MUKHERJEE, N. Application-centric Resource Provisioning for Amazon EC2 Spot Instances. In *EuroPar* (August 2013).

[7] MARATHE, A., HARRIS, R., LOWENTHAL, D., DE SUPINSKI, B. R., ROUNTREE, B., AND SCHULZ, M. Exploiting Redundancy for Cost-effective, Time-constrained Execution of HPC Applications on Amazon EC2. In *HPDC* (2014).

[8] MIHAILESCU, M., AND TEO, Y. M. The Impact of User Rationality in Federated Clouds. In *CCGrid* (2012).

[9] SHARMA, P., GUO, T., HE, X., IRWIN, D., AND SHENOY, P. Flint: Batch-Interactive Data-Intensive Processing on Transient Servers. In *EuroSys* (April 2016).

[10] SHARMA, P., LEE, S., GUO, T., IRWIN, D., AND SHENOY, P. SpotCheck: Designing a Derivative IaaS Cloud on the Spot Market. In *EuroSys* (April 2015).

[11] SINGH, R., IRWIN, D., SHENOY, P., AND RAMAKRISHNAN, K. Yank: Enabling Green Data Centers to Pull the Plug. In *NSDI* (April 2013).

[12] SONG, Y., ZAFER, M., AND LEE, K. Optimal Bidding in Spot Instance Market. In *Infocom* (March 2012).

[13] TANG, S., YUAN, J., AND LI, X. Towards Optimal Bidding Strategy for Amazon EC2 Cloud Spot Instance. In *CLOUD* (June 2012).

[14] VOORSLUYS, W., AND BUYYA, R. Reliable Provisioning of Spot Instances for Compute-Intensive Applications. In *AINA* (2012).

[15] WEE, S. Debunking Real-Time Pricing in Cloud Computing. In *CCGrid* (May 2011).

[16] WILLIAMS, D., JAMJOOM, H., AND WEATHERSPOON, H. The Xen-Blanket: Virtualize Once, Run Everywhere. In *EuroSys* (2012).

[17] XU, H., AND LI, B. A Study of Pricing for Cloud Resources. *Performance Evaluation Review 40*, 4 (March 2013).

[18] ZAFER, M., SONG, Y., AND LEE, K. Optimal Bids for Spot VMs in a Cloud for Deadline Constrained Jobs. In *CLOUD* (2012).

[19] ZHANG, Q., GÜRSES, E., BOUTABA, R., AND XIAO, J. Dynamic Resource Allocation for Spot Markets in Clouds. In *Hot-ICE* (March 2011).

[20] ZHENG, L., JOE-WONG, C., TAN, C. W., CHIANG, M., AND WANG, X. How to Bid the Cloud. In *SIGCOMM* (August 2015).