Towards Index-based Global Trading in Cloud Spot Markets

Supreeth Shastri and David Irwin
University of Massachusetts Amherst

Abstract

Infrastructure-as-a-Service clouds are rapidly evolving into market-like environments that offer a wide range of server contracts. Amazon EC2's spot market is the clearest example of this trend: it operates over 5000 markets globally where users can rent servers for a variable price. To exploit spot instances, while mitigating the risk of price spikes and revocations, many researchers and startups have developed techniques for modeling and predicting prices to optimize spot server selection. However, prior approaches focus largely on predicting individual server prices, which is akin to predicting the price of a single stock. We argue that researchers should instead focus on "index-based" modeling and prediction that aggregates prices from many markets in each region and availability zone. We show that, for applications flexible enough to select and "trade" servers globally, making decisions based on broader indices lowers costs and improves availability compared to index-agnostic policies.

1 Introduction

Infrastructure-as-a-Service (IaaS) clouds are rapidly evolving into market-like environments that offer servers for a variable price under a wide range of different contract terms that differ in their time commitment, price level, resource guarantees, and risk exposure. For example, EC2's spot market enables users to bid on servers, such that if a user's bid price exceeds the servers' current spot price, the platform allocates the servers to the user, who pays the variable spot price for them. While spot servers typically cost $\sim$50-90% less than on-demand servers, they introduce a new element of risk: if the spot price ever rises above the bid price, the platform immediately revokes the servers after a brief warning [3]. EC2's spot market is similar to other spot markets, particularly for electricity, in allocating real-time capacity that cannot be effectively stored to the highest bidder.

As shown in prior work, spot prices are not purely driven by supply and demand, as Amazon owns all the resources and sets the price [5]. However, regardless of how prices are set, the variability in spot prices across different servers and locations presents a new optimization opportunity to select servers based on their dynamic price characteristics. Prior work leverages this optimization to select servers for various applications that offer the best risk-adjusted returns, which takes into account the cost of performance penalties due to server revocations [12, 13, 18]. While this work offers the potential for significant cost savings, the magnitude of these savings is not guaranteed, is based on future prices, and could ultimately be negative if prices change. In general, prior work simply computes the expected risk-adjusted returns based on historical spot price traces, and thus implicitly assumes that the past accurately predicts the future. As a result, if the future deviates significantly from the past, applications can experience substantial losses.

Thus, accurately predicting future server prices is important in both estimating the potential savings for different servers and in selecting the optimal server. As a result, a number of researchers and startups [4, 8, 11] have proposed more sophisticated techniques for modeling and predicting spot market prices. For example, in a recent whitepaper [17], Spotinst [8] claims to use an "...in-house prediction algorithm..." to "[choose] the most effective and most likely available EC2 Spot instance." Researchers have proposed numerous similar modeling and prediction techniques for EC2 spot prices [1, 2, 5, 6, 9, 10, 16, 19, 20]. As one example, DrAFTS is an online service that, given a bid price and duration, returns the expected probability of acquiring an individual spot instance for that duration [1, 20].

In general, prior spot prediction techniques have focused on predicting prices in *individual server markets*, which dictate a dynamic price for each OS configuration of each instance type in each availability zone (AZ) of each region of EC2. For example, an `m4.large` running Linux in AZ `a` of the us-east-1 region has its own dy-

namic spot price, which is distinct from other server configurations and types in other AZs and regions. In aggregate, there are ∼5000 individual server markets across EC2's global platform. Modeling and predicting the behavior of each of these markets presents multiple challenges. In particular, unlike prices in electricity spot markets, which correlate with weather metrics, such as temperature, and other routine behavioral patterns, e.g., days versus nights, it is less clear if server spot prices correlate with any easily-measured external variables. As a result, price prediction techniques are inherently limited, as the primary information they leverage for prediction is historical prices. In addition, there is no guarantee a one-size-fits-all model exists, as price characteristics are based on local supply/demand conditions that may differ across individual server markets, just as individual stock prices may exhibit widely different characteristics.

Investors face similar issues in financial markets when making investment decisions. Since predicting individual stock prices is challenging, investors base investment decisions, in part, on the characteristics of broader market indices, such as the Dow Jones Industrial, the S&P 500, and the NASDAQ. This paper's hypothesis is that, rather than focus exclusively on predicting prices for individual server markets to guide decision-making, cloud users should also make decisions, in part, based on these broader market indices. This is especially true for applications that are not geographically constrained and are flexible enough to "trade" resources as prices change, i.e., migrate from one server to another. For these flexible applications, any individual server's price is not particularly important, as it can simply trade servers if the price rises too high (or the price of another server drops). As we discuss, accurate price predictions for individual servers are important only if applications are inflexible and must commit to a server for their entire execution.

We provide initial evidence to support our hypothesis by analyzing and characterizing various spot market indices in EC2. We define indices based on the entire market, each region, and each AZ, and show how their broad characteristics differ with respect to each other and to individual server prices. Then, to demonstrate the benefits of index-based global trading, we compare policies that select servers globally based on their individual price characteristics versus selecting them based on their broader index. We show that, since there are non-trivial costs associated with migrating between AZs and regions, selecting a server based on its AZ's index can yield lower costs and higher availability relative to selecting based on individual server prices. In addition, we also show that the aggregated price of these indices is significantly less volatile (and thus more predictable) than individual server market prices, further motivating index-based trading for flexible applications.

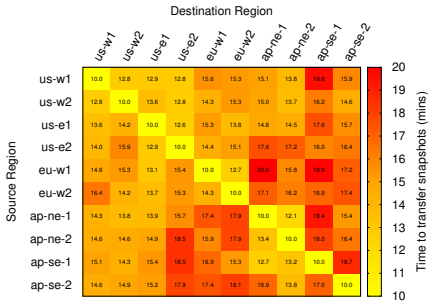

Figure 1: *Overhead of migrating 10GB EBS disk in EC2.*

2 Index-based Market Analysis

Our hypothesis is that users should make decisions on which spot server to request based, in part, on broad market indices, rather than the price characteristics of individual spot servers. Our hypothesis is especially applicable to "flexible" applications that are not geographically constrained, capable of trading servers as prices change, and resilient to revocations, i.e., they are either stateless or employ fault-tolerance mechanisms to enable restarting on a new server. For these flexible applications, any individual server's price is not particularly important, as it can simply trade servers if the price rises too high (or the price of another server drops). Individual server price predictions *are* important for inflexible applications that are incapable of trading servers and are intolerant to revocations. Inflexible applications must commit to a particular server and its price for the duration of their execution, and should select the individual server that exhibits the best risk-adjusted returns.

Cloud applications are becoming increasingly more flexible at both the systems- and application-level. Systems-level migration and checkpointing techniques, e.g., for nested virtual machines and resource containers, are rapidly maturing, while application-level fault-tolerance mechanisms are already embedded into most "big data" frameworks, e.g., Spark, Naiad, Hadoop, TensorFlow, etc., to handle inevitable failures at large-scales. Since flexible applications can trade servers if prices change, their primary constraint is the overhead to trade, which is largely a function of the size of an application's state and the network's characteristics. For example, a stateless application incurs little trading overhead based only on the delays imposed by the platform's API.

In contrast, if an application maintains persistent disk or memory state, this overhead is significant if a trade crosses an AZ or region, compared to trades within an AZ, as disk volumes are only accessible within an AZ. Trading across AZs and regions requires first migrating the server's disk volumes, e.g., in EC2's Elastic Block Store (EBS), from one AZ to another (by storing the data in S3 and then using EBS's SnapshotCopy function). Figure 1 quantifies the overhead to migrate a 10GB EBS

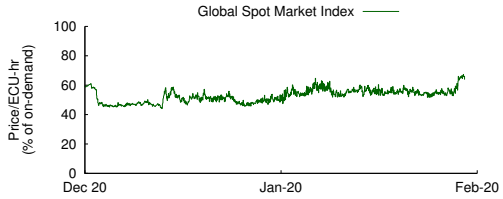

Figure 2: *Market index for all 2287 Linux server markets across all 14 regions in EC2 over the last 2 months.*

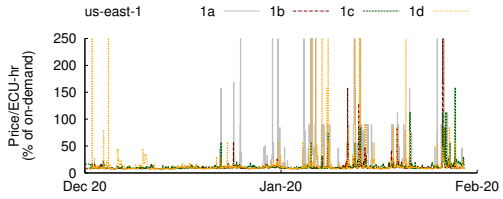

Figure 3: *Price of a representative Linux server (r3.4xlarge) across each AZ of the us-east-1 region.*

disk between AZs both within and across regions. The figure shows that, while overhead is variable, it ranges from ∼1-2min/GB. In addition, migrating between AZs within a region (indicated by the diagonal from upper-left to lower-right) has ∼20-50% less overhead than migrating across regions. In contrast, migrating state within an AZ incurs a small fixed overhead (∼120s) as servers within the AZ can directly mount any EBS volume.

Based on the overheads above, we examine market price indices at the region-, AZ-, and server-level, and characterize their salient attributes. In this paper, we define simple indices across different administrative boundaries, e.g., all of EC2, each region, and each AZ, using the arithmetic mean of the price per ECU-hour for each server normalized by its on-demand price. Note that other indices are also possible, e.g., across each server type or family, and may be relevant to certain classes of applications. For example, stock market indices often use a weighted average based on a company's capitalization or size. Figure 2 shows the global spot market index for Linux servers across all of EC2 over the past 2 months. Here, we plot the price per ECU-hour for all 2287 Linux spot servers across all 14 EC2 regions normalized by the on-demand price per ECU-hour across all Linux servers. Note that, to reduce each graph's size, we only plot the maximum index price each hour. The graph shows that EC2's global spot market is stable, with prices bounded between ∼40-60% of the on-demand price with low variance. This differs from the individual server prices in Figure 3, which exhibit "peaky" behavior with long periods of flat prices interspersed with large spikes.

We can also examine the market indices on a regional basis to get an idea each region's overall behavior with respect to the others. While the global spot market index reveals the general state of the market, per-region indices can inform users' choice of regions in the global market. This choice is important based on the overheads

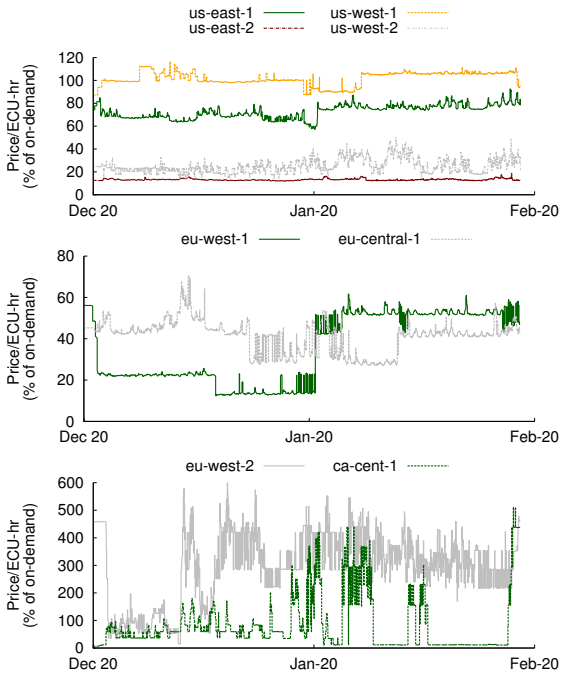

Figure 4: *Select regional spot market indices.*

in Figure 1: since the cost to "trade" within a region is much less than across regions, applications may not be able to exploit low prices in another region. Figure 4 demonstrates that the regional price indices differ widely in both their magnitude and variance. For example, the US regions (top) are highly stable with the us-east-2 index price remaining at ∼13% of the on-demand price for nearly the entire two months. In contrast, the EU regions (middle) exhibit much higher overall variance coupled with abrupt phase changes. In eu-west-1, the index rose by over 2× near January 20th. Such abrupt increases likely reflect internal changes in supply that are not market-driven. We have also found that newer regions often exhibit more volatile prices than older regions, presumably because they are not as well-provisioned and thus have a higher variance in their idle spot capacity. For example, both eu-west-2 and ca-cent-1, which opened in late 2016, exhibited highly volatile spot prices over the past two months (bottom).

The magnitude of regional spot prices also varies widely. Since on-demand prices vary across regions and we normalize the prices in Figure 4 relative to local on-demand prices, and show the average on-demand price per region in Figure 5 to permit a rough comparison. As the figure shows, not only do us-east-2 and us-west-2 have lower spot prices relative to their local on-demand price, they also have some of the lowest average on-demand prices. Thus, us-east-2 and us-west-2 are much more attractive options than either i) us-east-1 and us-west-1 (which are also stable but have much higher overall prices) or ii) eu-west-2 and ca-cent-1 (which have

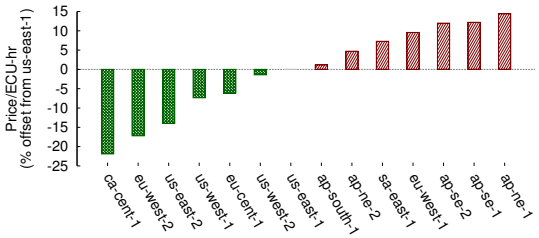

Figure 5: *Average on-demand price in each region.*

lower on-demand prices but more volatile spot prices).

Finally, just as per-region indices reveal more detail than the global market index, per-AZ indices provide a deeper breakdown into each region. Figure 6 shows AZ-level indices for us-east-1, us-east-2, and eu-west-1. We see that, while us-east-1's regional index is largely stable, its AZ-level indices are less stable and have a large variation in their magnitude, ranging from a constant 30% of the on-demand price (1e) to over 100% of the on-demand price (1c and 1d).[1] In contrast, in us-east-2, the AZ-level indices are highly correlated and all similar to the regional index. Thus, while both us-east-1 and us-east-2 have similarly stable regional prices, us-east-1 imposes a much higher risk, as it exhibits much higher volatility at the AZ level. Finally, in eu-west-1, we see that the abrupt spike near January 20th was the result of a correlated spike in only two of the AZs (1a and 1b) with one AZ maintaining low and stable prices.

Our analysis confirms the intuition that, in general, the broader the index, the more stable and predictable its future prices. The global spot market index is generally more stable than the regional indices, which are more stable than the AZ-level indices, which are in turn more stable than the individual server markets. Thus, applications should have more confidence over region- and AZ-level decisions compared to decisions based on expectations of individual server prices.

3 Comparing Global Trading Policies

We demonstrate the importance of index-based global trading using a generic long-running application in simulation. We assume our application i) has no geographical constraints, ii) is capable of consuming whatever resources are available, and iii) executes within a virtualized environment, such as a nested virtual machine or resource container, that makes it capable of trading servers via transparent systems-level migration. We also assume the application can gracefully handle IP address changes when crossing regions and AZs, and employs fault-tolerance mechanisms, such as replication or checkpointing, to make it robust to revocations. Solutions to enabling these assumptions are well-known, as

[1]Note that this analysis includes spot prices that are 10× the on-demand price in the index price. However, since 10× the on-demand price is the maximum bid, these spot servers are effectively unavailable.

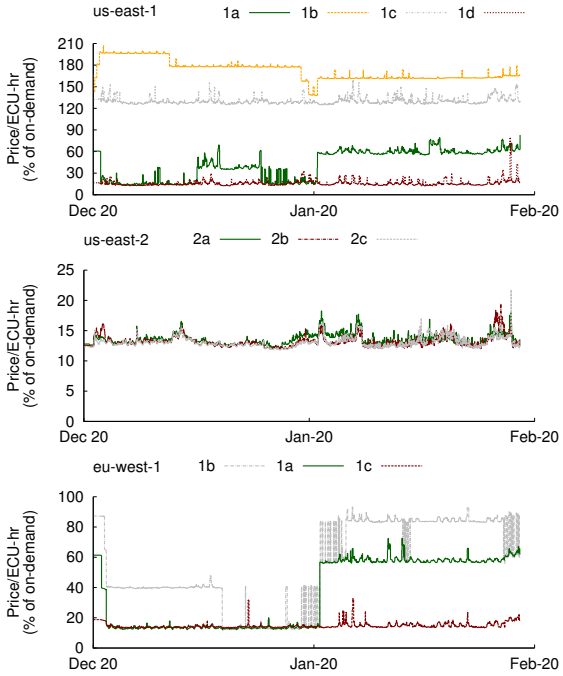

Figure 6: *Select AZ-level spot market indices*

prior work on superclouds has resolved many of these "plumbing" issues [7, 15]. We simulate the application's behavior over the past two months using real spot market prices from EC2's 2287 Linux spot markets. To enable trading, we assume the application monitors spot prices in each of these Linux spot markets, and includes a trading policy that dynamically migrates as prices change to the server with the lowest current price per ECU. Our simulation accounts for the overhead of trading across AZs and regions based on Figure 1.[2] We define multiple global and local trading policies, as outlined below.

- **Market-based No Trading** selects the individual spot server across the global market with the highest Sharpe ratio below, which is a standard measure for estimating an asset's risk-adjusted returns: for an asset i, it is the ratio of the expected difference between the asset's returns R_i and the risk-free returns R_{free} divided by the standard deviation of the returns σ_i. In this case, the on-demand price captures the risk-free returns. As in nearly all prior work on spot instances, this policy commits to its chosen server and never trades, regardless of price changes.

$$S_i = \frac{E[R_i - R_{free}]}{\sigma_i} \quad (1)$$

- **Market-based Local Trading** selects the individual spot server in the global market with the lowest price per ECU, and then actively trades *within that server's AZ* to ensure it always runs on the server with the

[2]Note that the cost to migrate the 10GB state across regions (at $0.20) is negligible compared to the average time between trades.

Figure 7: *Comparison of cost and availability for our generic application using each of our four policy variants.*

lowest price per ECU. Thus, this policy avoids any trading overheads from crossing AZs and regions.

- **Market-based Global Trading** selects the spot server in the global market with the lowest price per ECU, and then actively trades *across the global market* to ensure it always runs on the server with the globally lowest price per ECU. The policy incurs the trading overheads from crossing AZs and regions.
- **Index-based Global Trading** first selects the AZ with an index having the highest Sharpe ratio, then selects the individual spot server in that AZ with the lowest price per ECU, and finally actively trades *within that server's AZ* to the server with the lowest price per ECU. As shown in Figure 4, the AZ with the highest current Sharpe ratio in EC2 is us-east-2.

Note that incorporating risk, in this case using the Sharpe ratio, is most important when committing to a subset of markets (either the individual server market in the first bullet or the AZ in the last bullet). Considering risk is much less important when actively trading, as the application often migrates before revocations ever occur.

Figure 7 shows both the overall cost (left) and availability (right) of running our generic application over the two month period. The cost is normalized as a percentage of the No Trading policy to illustrate the benefits of actively trading servers as market prices change. As mentioned above, this No Trading policy is similar to policies in prior work, which commit to spot servers and only select a new server after a revocation [12, 14, 18, 21]. Since spot prices for individual servers are generally stable, revocations are rare, and thus there are few opportunities for selecting a new server in prior work. However, given the large number of individual server markets, the lowest cost server (across the global market, region, or AZ) actually changes quite frequently. Thus, policies that actively trade servers can reduce their costs relative to policies that do not actively trade. As the figure shows, the No Trading policy incurs a higher cost than all of the active trading policies.

The figure also shows that the Market-based Local Trading policy incurs a much higher cost than the Market-based Global Trading policy. Since the local trading policy only trades within its own AZ to eliminate trading overhead, it cannot take advantage of low prices in other AZs. In general, the individual server in the global market that has the best combination of price and risk, as measured by the Sharpe ratio, is not necessarily contained in the AZ with the best combination. Overall, the Market-based Global Trading policy achieves the lowest cost, even when accounting for its high trading overhead, as it always actively migrates to the globally least-cost server. In comparison, the index-based global trading policy, which commits to the AZ with the highest Sharpe ratio, but then restricts itself to intra-AZ trading to mitigate trading overhead, incurs only a slightly higher cost than the Market-based Global Trading policy.

To quantify availability, we assume the application is unavailable when trading servers according to the benchmarks in Figure 1 with intra-AZ trades incurring an unavailability of two minutes. The figure shows that, while the Market-based Global Trading policy has the lowest cost, it also has the lowest availability (1 nine) due to the high trading overhead imposed by frequently crossing regions and AZs. While it is costly, the No Trading policy exhibits a slightly higher availability (2 nines), since it never trades and only experiences downtime when its spot price spikes. The Market-based Local Trading policy has the highest availability (4 nines), since it also restricts trades to within its AZ; by actively moving to the lowest-cost server it experiences few price spikes that cause unavailability. However, the policy incurs a high cost, since it selects an initial server based on its price characteristics and not AZ-level characteristics. Finally, in this case, our index-based policy achieves the best of both worlds—a high availability (3 nines) at a low cost—by selecting an AZ with an index price that has low magnitude and variability, and then actively trading within it. Of course, the best policy is application-dependent, and varies based on an application's footprint and other availability constraints. We are developing application-specific trading policies as part of future work.

4 Conclusion

This paper highlights the importance of selecting spot servers based, in part, on broad price indices, rather than individual server prices. While predicting individual server prices is a popular research topic, we argue that it is not particularly important for flexible applications that are capable of actively trading servers.

Acknowledgements. This work is supported by NSF grant #1422245 and a Google Faculty Research award.

5 Discussion Topics

Our paper takes a different approach than prior work in this space by advocating decisions based on index-level price analyses, rather than analyses at the individual spot server-level. Ultimately, any system that optimizes its use of low-cost spot instances has to base allocation decisions on (implicit or explicit) predictions of future market prices. Our approach parallels the real world, where investors not only model individual stock prices, but also make decisions based on broader market trends. In addition, just as in investing, there are different overheads associated with trading different types of investments, e.g., based on their liquidity. We believe our approach is becoming more relevant as cloud applications are becoming more flexible, enabling them to actively migrate to new servers as market prices change. We would welcome the community's feedback on this hypothesis.

Our work is only an initial simplistic example of the benefits of index-based trading. Other indices are also possible. For example, some applications might be bound to certain server families in the cloud, such that they would trade based on indices for these server families. Our simulation experiments also make a number of simplifying assumptions due to space constraints, such as no geographic constraints and a workload that can fully saturate any server. In practice, applications are likely to have some geographic constraints and exhibit a variable workload that would also influence decision making. In general, we believe there are many parallels between the financial world and emerging cloud markets. Thus, adapting and modifying methodologies from finance can advance market-based applications. In this case, we noted a difference between existing work on spot markets, which focuses on price prediction for individual servers, and financial investing, which is more sophisticated and generally takes a higher-level view.

We expect the paper to generate interesting discussions at the workshop. One point of discussion could be that EC2's spot market is neither general nor real (having been artificially constructed by Amazon). Thus, the idea could "fall apart" if Amazon either eliminated or altered its spot market. However, we believe index-based trading can apply to other contract variants, such as burstable instances, spot blocks, the reserved marketplace, Google preeimptible VMs, etc. There are also many other clear parallels (and differences) between the diversity of contract types in cloud platforms and in existing commodity markets. For example, companies that operate in commodity markets focus on achieving the best mix of contracts to balance their risk and reward. Of course, there are key differences in the cloud. For instance, server availability is less constrained than other commodities with less volatile prices that generally drop over time.

References

[1] DrAFTS – Durability Agreements from Time Series for AWS Spot Instances. `http://predictspotprice.cs.ucsb.edu/`, Accessed March 2017.

[2] AREVALOS, S., LOPEZ-PIRES, F., AND BARAN, B. A Comparative Evaluation of Algorithms for Auction-based Cloud Pricing Prediction. In *IC2E* (April 2016).

[3] BARR, J. New - EC2 Spot Instance Termination Notices. https://aws.amazon.com/blogs/aws/new-ec2-spot-instance-termination-notices/, January 6th 2015.

[4] Batchly, Inc. http://www.batchly.net/, September 2016.

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

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

[7] JIA, Q., SHEN, Z., SONG, W., VAN RENESSE, R., AND WEATHERSPOON, H. Smart Spot Instances for the Supercloud. In *CrossCloud* (April 2016).

[8] LARDINOIS, F. Spotinst, which helps you buy AWS spot instances, raises $2m Series A. TechCrunch, March 8th 2016.

[9] LIANG, Q., WANG, C., AND URGAONKAR, B. Spot Characterization: What are the Right Features to Model? In *SAC* (June 2016).

[10] MAZZUCCO, M., AND DUMAS, M. Achieving Performance and Availability Guarantees with Spot Instances. In *HPCC* (September 2011).

[11] NOVET, J. Amazon pays $20M-$50M for ClusterK, the startup that can run apps on AWS at 10% of the regular price. VentureBeat, April 29th 2015.

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

[13] SHARMA, P., IRWIN, D., AND SHENOY, P. Portfolio-driven Resource Management for Transient Cloud Servers. In *SIGMETRICS* (June 2017).

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

[15] SHEN, Z., JIA, Q., SELA, E., RAINERO, B., SONG, W., VAN RENESSE, R., AND WEATHERSPOON, H. Follow the Sun through the Clouds: Application Migration for Geographically Shifting Workloads. In *SoCC* (October 2016).

[16] SINGH, V., AND DUTTA, K. Dynamic Price Prediction for Amazon Spot Instances. In *HICSS* (January 2015).

[17] SPOTINST. The State of the Amazon EC2 Spot Market: Research, Conclusions, and Opportunities. `https://spotinst.com/white-papers/the-state-of-the-amazon-ec2-spot-market/`, 2016.

[18] SUBRAMANYA, S., GUO, T., SHARMA, P., IRWIN, D., AND SHENOY, P. SpotOn: A Batch Computing Service for the Spot Market. In *SoCC* (August 2015).

[19] WANG, C., LIANG, Q., AND URGAONKAR, B. An Empirical Analysis of Amazon EC2 Spot Instance Features Affecting Cost-effective Resource Procurement. In *ICPE* (April 2017).

[20] WOLSKI, R., AND BREVIK, J. Providing Statistical Reliability Guarantees in the AWS Spot Tier. In *HPC* (April 2016).

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