Analyzing AWS Spot Instance Pricing

University of California, Santa Barbara Computer Science Technical report Number 2018-02 *

Rich Wolski
Computer Science Department
University of California, Santa Barbara

Gareth George
Computer Science Department
University of California, Santa Barbara

Chandra Krintz
Computer Science Department
University of California, Santa Barbara

John Brevik
Department of Mathematics and Statistics
California State University, Long Beach

Abstract

Many cloud computing vendors offer a preemptible class of service for rented virtual machines. In November of 2017, Amazon.com announced a change to the mechanism it was using to price its preemptible "spot instances" so that prices would change more "smoothly." This paper analyzes the effect of the change in pricing mechanism on spot instance prices. It examines the prices immediately before and after the mechanism change to determine the extent to which prices themselves changed. It also compares the period immediately after the change in mechanism to the most recent period. Our results indicate that in addition to smoothing prices, the mechanism change introduced generally higher prices which is a trend that continues.

1 Introduction

One of the fundamental tenets of cloud computing is that resources (*e.g.* machines, network connectivity, storage, etc.) be characterized by their capacity and capability characteristics rather than their physical construction. Programmers and users reason about the use of cloud resources in terms of these characteristics (in principle) without regard to the physical infrastructure that is used to deliver them. For example, instances (virtual machines) available from Amazon's Elastic Compute Cloud (EC2 [4]) are advertised as rough equivalents to various models of physical processors (in terms of clock speed, cache size, etc.), but Amazon provides no guarantee that these specific processor models will be used to fulfill a specific user's request or are even available in their cloud.

Instead, users enter into a "Service Level Agreement" (SLA) that quantifies the minimum capability that a particular request will receive. Typically, if service provider violates the agreement by failing to meet this "quality of service," the user is entitled to some form of financial compensation. Thus the cloud computing model is one in which users can reason about the capabilities that their computations require, and will receive, in terms of SLAs and not the physical capabilities of specific resources.

Cloud computing vendors may offer different SLAs at different price points so that users can control the value transaction at a fine level of granularity. In particular, vendors such as Amazon and Google [16] offer a preemptible tier of service in which they offer resources at a cheaper price without a reliability SLA.

Until recently (November 2017), Amazon offered these instances as "spot instances" [6], for which the reliability was based (in part) on the maximum amount of money a user agreed to pay for them. As this system operated, a user making a request for an instance having a specific set of characteristics (termed an "instance type") also included a "maximum bid price" indicating the maximum that the user was willing to be charged for the instance. Amazon then created a market for each instance type and satisfied the requests of the highest bidders. Periodically, Amazon recalculated the market price and terminated those instances whose maximum bid was below the new market price. This market-clearing mechanism was public,

*This work is supported in part by NSF CNS-1703560, OAC-1541215, CCF-1539586, ACI-1541215

but individual user bids (and some of the other market parameters) were private to individual users and to Amazon. Thus, each user had to devise an individual bidding strategy that met his or her own reliability needs [26, 18].

Amazon also offers the same instance types under a variety of other pricing plans, each of which carries an availability SLA. These plans are typically significantly more expensive that the prices set for spot instances. Thus users can pay a "fee" for guaranteed reliability or try and use spot instances knowing they could be terminated at any time.

At the end of November, 2017 Amazon announced a change to the spot instance pricing mechanism. While originally described as "price smoothing," the new mechanism is a retail pricing mechanism, in which Amazon sets prices dynamically using an algorithm that it does not disclose. In particular, this new mechanism does not use maximum bids as a market-clearing mechanism: Amazon simply sets the retail price from moment to moment. However it claims that the price changes will be "smoother" than those set using previous, market-based pricing mechanism.

Importantly, Amazon also changed the algorithm it uses to select which instances will be terminated when a price change (or a resource shortfall) occurs. No longer are "maximum bids" used to prioritize terminations. Previous work [28] demonstrated the ability to use price histories for each instance type to determine a maximum bid that would provide some statistical assurance for a desired minimum duration of execution. That is, when spot instances were priced using the market-based mechanism, users could compute the maximum bid necessary to achieve reliability guarantees comparable to those offered with the more expensive pricing plans. Under the new retail pricing mechanism, however, such predictions are no longer possible, because the algorithm used to determine which instances to terminate is now private to Amazon. Thus, the spot-instance product has changed from one that was predictable to one that is not.

In this paper, we examine the price change that accompanies this product change for the spot-instance product. Because spot instances are less predictable, they should be cheaper. The product's reliability has been diminished, and since reliability is the differentiating characteristic of spot instances (relative to other pricing plans) one might expect a decrease in price. On the other hand, smoothing prices might create the need to increase prices to "cover" sharp changes in demand or supply.

By comparing price histories occurring immediately before the change in prices to ones for the same instance types immediately following the switch, this work illustrates the nature of the change in prices. We find that in general, the switch to smoother retail pricing carries a price increase, but this is not universally true. That is, while many instance types became more expensive, some instance types became cheaper as a result of the switch.

We also compare prices from the time period immediately following the switch in November of 2017 to those set in the second quarter of 2018 to determine whether there is a trend in retail pricing implemented by Amazon.

This paper makes the following contributions:

- It describes the previous and new pricing mechanisms for spot instances and discusses techniques for analyzing price histories under each regime.
- It quantifies the change in prices that accompanied the switch in pricing mechanism.
- It identifies trends in the new retail pricing mechanism since the switch.

These contributions demonstrate the depth of analysis that is necessary to understand a change in the spot-instance product from Amazon. More generally, virtualized cloud products are opaque to their users, who must rely on vendor documentation to understand the nature of the value proposition with which they are presented. In this case, the spot-instance product is the *same* product with a different pricing mechanism. However, this work shows that the change in pricing mechanism also changes the reliability characteristics of the product while generally manifesting an increase in price. Thus, this work is a harbinger of the analysis techniques that may be necessary to understand change to current and future cloud resource values.

2 Background

Amazon Web Services (AWS) includes the ability to rent "virtualized" datacenter components that are hosted in Amazon datacenters under the brand name "Elastic Compute Cloud" (EC2). Originally available

only "on demand" on an hourly rental basis, EC2 "instances" (virtual machines) are now available under a variety of fixed-price rental options [7]. These fixed-price rentals all share a common Service Level Agreement (SLA) with respect to availability [5], which guarantees a minimum of 99% over the period of a month. Users who experience less that 99% availability are entitled to a 30% refund (in the form of AWS credits).

Instances are instantiated from "instance types," which describe the CPU, memory, and external storage capabilities associated with a virtual machine. That is, an instance is a running virtual machine that comprises the CPU, memory, and storage capabilities of some instance type.

Amazon uses its datacenters to host instances, but for failure management and scalability reasons the datacenters do not share infrastructure other than the Simple Storage Service (S3) [8]. Instead, AWS is organized as "regions," each of which operates as a separate and independent service venue for EC2. Thus, users of EC2 must select a distinct region in which to request the launch of instances and, once launched, the instance only has access to services in that region (except S3, which is global to all regions).

Amazon further subdivides regions into "availability zones" (AZs), each of which corresponds (roughly) to a single datacenter building. An instance must run in a single AZ within a single region. When making a request to launch an instance, a user must choose a region but can either allow Amazon to select an AZ within that region or specify one explicitly. Almost all services within a region are globally accessible from any AZ in that region, and (because they are virtually identical in this regard) Amazon quotes prices for instances on a per-region basis.

In 2009, Amazon introduced spot instances as a lower-cost, dynamically priced alternative to its fixed-price EC2 offerings. The original spot-instance product offering advertised that prices were set dynamically in separate "spot markets" based on instantaneous supply and demand. Specifically, each user would enter a sealed "bid" indicating the maximum hourly charge she would agree to incur for an instance in a specific AZ. While prices are quoted for each region, for spot instances, each AZ within each region formed a separate market. The region-wide price for a specific instance type was set to the minimum for that type across all AZs in that region. A user could elect either to have Amazon choose an AZ (and it did not guarantee to choose the cheapest) or specify (presumably based on current price, which was published in real time) the AZ in which to launch an instance.

Periodically (approximately every 5 minutes in the original product) Amazon would sort the bids in each AZ in descending order of bid value and assign resources to bidders in that order until all resources were exhausted. The bidders who received resources all paid the lowest bid price among those bids that could be satisfied. The remaining bidders either received no resources or, if they previously had resources assigned, had those resource confiscated so they could be assigned to a higher bidder. Amazon did not publish the available supply but did make available (in real time) the current market-clearing price for each instance type in each AZ.

The conception for spot instances was as a way for Amazon to monetize (at a market price) its own internal idle instance capacity. Thus, price fluctuations were caused by two separate factors: newly arriving bids and unannounced changes in idle capacity. Subsequent research [1] hypothesized a third factor, namely the introduction of a hidden reserve price to help obfuscate the supply and demand signals in the market. However, from a user perspective, the "spot market," as it was known colloquially, offered significantly cheaper hourly rental rates for instances (compared to fixed-price instances), with the proviso that these instances could be terminated at any time due to a change in supply or demand.

At the end of November, 2017 Amazon announced a change from this "market-based" price-setting mechanism to one that uses a dynamically changing retail price set by Amazon that is not based on market inputs [3]. Originally, the product announcement indicated that the change simply smoothed the visible price histories to make spot instances more attractive to users. That is, because demand (and more probably supply) could change suddenly and substantially, users viewing real-time pricing data or price histories could be surprised by sharp changes in price.

However, subsequent reports by heavy spot-instance users indicate that Amazon also changed the way in which it determines which instances to terminate when there is a supply shortfall [24]. Instead of basing the termination decision on sealed bids from users, Amazon switched to using a hidden internal algorithm that may or may not take bid value (now called "maximum value") into account.

This paper is specifically focused on quantifying how the retail prices set by Amazon after the switch to the dynamic retail pricing mechanism compare to the prices that were set before the switch in response to sealed bids and internal idle capacity (henceforth termed the "market-based" mechanism). However, the

change is also significant because it alters the utility of the spot instance product substantially independent of price. Specifically, because the termination algorithm for the retail pricing mechanism is not public [9] and is based on hidden parameters, terminations cannot be predicted in the way they once were [28]. Thus, the switch to retail pricing changes both the price of spot instances and their reliability characteristics.

3 Methodology

We compare spot-instance prices in terms of the expected price per hour of usage. For any given hour, the sum of the price multiplied by the fraction of the hour that price persists is the expected cost for that hour. That is, for each instance type in each AZ we obtain from Amazon the price history for that combination. For each hour h of the price history, we compute the expected hourly price $E(price)_h$ as

$$E(price)_h = \sum_{i=1}^{D_h} P_i \times d_i, \tag{1}$$

where D_h is the number of prices listed during hour h, P_i is the i^{th} price listed for the hour, and d_i is the fraction of an hour for which price P_i persisted.

Intuitively, $E(price)_h$ is the expected spot-instance revenue for a single instance of that instance type and AZ during hour h. Equivalently, it is the expected cost a user arriving at a random time during hour h would expect to incur for that hour of usage.

However, under the previous market-driven pricing mechanism, it appears that Amazon would sometimes set a reserve price for some instance types that was so high that no instances of that type would be run. For example, consider the time series of prices for the *c4.8xlarge* instance type in the *us-west-1b* AZ shown in Figure 1. The series shows the spot-instance price history at 5 minute intervals from 6:00 AM until 6:50

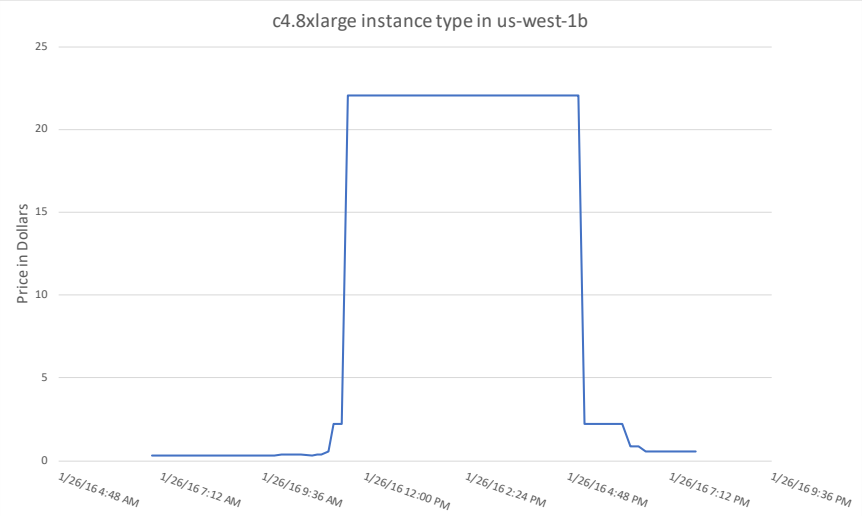

Figure 1: Amazon price history for *c4.8xlarge* instance type in *us-west-1b* AZ. The history begins at 6:00 AM Pacific time and ends at 6:50 PM Pacific Time on January 26, 2016.

PM Pacific time on January 26, 2016. The x-axis is time and the y-axis price in US dollars. From 6:00 AM until approximately 10:10 AM, the average price for a *c4.8xlarge* spot instance in the *us-west-1b* AZ was approximately \$0.31. At that time, the price rose to \$2.028 for approximately 20 minutes and then spiked to \$22.08 at approximately 10:30 AM. This price (which was two orders of magnitude higher than the average price before 10:10 AM) remained fixed until approximately 2:00 PM, when it dropped to \$2.028 once again. At approximately 5:00 PM, the price dropped once again, fluctuating between \$0.87 and \$0.52 with an average of \$0.59 for the remaining 2 hours.

Note that between the beginning of the series and 10:10 AM and then again after 5:00 PM until the end of the series, the price was fluctuating but remained well under \$1.00. When at 10:10 AM it jumped to

$2.028, it remained there with no fluctuation, and again when it jumped to $22.08 (an order or magnitude jump in price) it remained fixed. While it is conceivable that some customer of AWS bidding for a *c4.8xlarge* spot-instance in *us-west-1b* would be willing to pay $2.028, it is unlikely that any user would have consented to paying an hourly rate of $22.80 for an instance. Indeed, the current *on-demand* price (at the time of this writing) for a *c4.8xlarge* in *us-west-1* is $1.19. That is, the most expensive price Amazon charges for a $c4.8xlarge$ in *us-west-1* that does not carry a termination possibility is only $1.19.

Thus, while a bid of $2.028 might be reasonable in order to prevent an early termination of a spot instance, we (like the authors of [1]) speculate that the $22.08 price is a hidden reserve price, and indeed not an actual price that a customer would expect to pay nor one that would appear as a rational bid for the instance.

3.1 Removing Hidden Reserve Prices

Our goal is to have $E(price)_h$ capture the prices that a rational customer would pay in the spot market customers would pay in a rational market. Thus, to compute $E(price)_h$ for spot instances where prices were set by the market-driven mechanism, we attempt to filter out automatically the obviously prohibitive hidden reserve prices from each price history.

To do so, we use a one-dimensional clustering technique based on the Generalized Likelihood Ratio Test (GLRT) [27]. Our implementation of GLRT starts with a sorted list of prices, which it then tries to divide into two sorted lists based on likelihood calculated under the assumption that the original list and each of the two sub-lists is drawn from a pre-specified family of distributions. Once the best split is determined, both of the lists are added to a sorted queue of clusters based on the likelihood associated with the cluster. The process repeats considering subdivision of those clusters with the lowest likelihood statistics until some maximum number of clusters is determined (we use a maximum of 30 in this study). For each clustering it considers, the algorithm computes a Bayesian Information Criterion (BIC) [22] score. It then chooses the clustering among all of those considered with the largest BIC score.

For the likelihood computations we used exponential distributions. While this is in part a matter of numerical expediency, since likelihood calculations can be heavily optimized for these distributions, such an approach can also be justified by the flexibility of hyperexponential models for a wide class of distributions. As a sanity check, we also ran several of the same clusterings assuming normal distributions (and at substantially greater computational cost), and the clustering outcomes for this sample do not differ substantially from our overall results.

To filter suspected hidden reserve prices, we then remove all of the values in the cluster determined by GLRT that contains the maximum value from the series. From the remaining values, we compute the mean and standard deviation. If the maximum value is more than 5 standard deviations above the mean of the values with this high cluster removed, we remove the high cluster from the series under the assumption that it contains hidden reserve prices.

For example, applying this approach to the price history for the *c4.8xlarge* instance type described above for the period beginning January 11, 2015 and ending November 20, 2017 (just before the switch to retail pricing) yields a clustering in which the cluster containing the largest values ranges from $3.50 to $22.80 (using either an exponential distribution or a normal distribution for the likelihood ratio test.) Thus, to compute $E(price)_h$ for each hour h in the price series shown in Figure 1, we first remove all prices that are between $3.50 and $22.80 (inclusively) from the series and then compute the mean and standard deviation of the the remaining prices which are 0.45 and 0.24 respectively. Because the maximum price of $22.80 is greater than $0.45 + (5 \times 0.24)$ (the mean plus five standard deviations, our method removes the values from the series that are $3.50 or greater as they are likely hidden reserve prices.

By way of comparison, Figure 2 shows the price history for the same *c4.8xlarge* spot-instance type in *us-west-1b* from 4:00 PM Pacific time on November 30, 2017 until 4:00 PM, April 16, 2018. This time period corresponds to the time immediately the following Amazon's change from the market-driven pricing mechanism to the retail pricing mechanism. Note that at no time during the almost five-month period does the spot-instance price "spike" to a large fixed value. Indeed, for the retail pricing mechanism, we assume that there are no hidden reserve prices: In this case Amazon does not use "bid" price as a determining factor when deciding to run or terminate a spot instance, so hidden reserve prices have no utility.

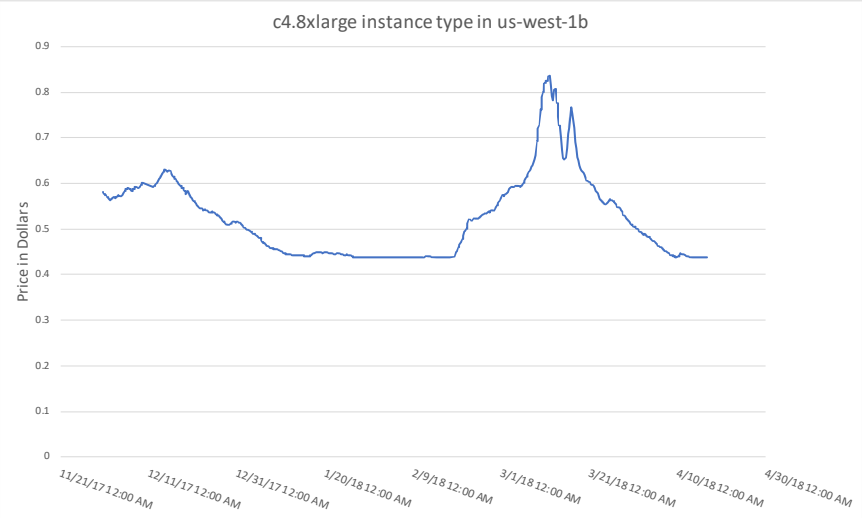

Figure 2: Amazon price history for *c4.8xlarge* instance type in *us-west-1b* AZ. The history begins at 4:00 PM Pacific time on November 30, 2017 and ends at approximately 4:00 PM Pacific Time on April 16, 2018.

4 Results

In this study, we use spot instance price history data [2] that we have been gathering from Amazon since January of 2015 as part of the Aristotle federated cloud project [14]. The data is for the "Linux/UNIX" product only in the North American regions (*us-east* and *us-west*). Because Amazon introduced new instance types (and, indeed, a new region – *us-east-2*) during this period not all of the instance-type histories extend back to January of 2015.

To determine the degree to which the expected prices changed as a result of the switch from a market-based mechanism to a retail pricing mechanisms, we compare the average $E(price)_h$ values for the 90 days immediately preceding the switch to those observed during the 90 days following the switch. The two periods do not abut one another in time because (it appears from the data that) there is a brief period of transition during which some of the instance type-AZ combinations were priced using the market-based mechanism while others were priced using the new retail mechanism. Specifically, the market-based mechanism period extends from August 2nd, 2017 to October 31st, 2017 and the comparative retail period extends from November 30, 2017 to February 28, 2018. We omit the month of November 2017 due to the possibility that Amazon was "phasing in" the new retail mechanism.

Because of the introduction of new products, it is not possible to compare all current instance types in all AZs before and after the switch. There are 862 combinations of instance-type and AZ for which we have sufficient pricing data to make this comparison. Similarly, not all instance types that were present before the switch may be supported in all AZs so in Subsection 4.2, we compare the period immediately after the switch to the most recent period. In this latter case, there are 1272 combinations of instance type and AZ with sufficient data from both periods.

Also, our system for compiling ongoing price histories runs as an ordinary AWS user. Thus all AZ names refer to the AZ naming that Amazon exposes to that user (*i.e.*, another user might be shown the same AZ under a different name within the region due to Amazon's AZ name-obfuscation policy).

Table 1 shows a comparison of average $E(price)_h$ values for the North American regions before and after the pricing mechanism change. The change in price is computed as the difference between the average value of $E(price)_h$ after the change minus the average value of $E(price)_h$ before the change. Thus, a positive value indicates a price increase and a negative value a decrease. Also, we compute the percentage change as the average $E(price)_h$ value before the change divided by the price change (multiplied by 100) *for each instance type* and then take the average over all instance types in a region to obtain the average overall percentage change.

Note that in *us-east-2* and *us-west-2* the average $E(price)_h$ value decreased, but in all regions the

AWS Region	Price Change	Percentage Change
us-east-1	-1.74	37.28%
us-east-2	-4.42	43.49%
us-west-1	0.38	61.00%
us-west-2	-1.29	59.47%

Table 1: Comparison of average $E(price)_h$ values for North American AWS regions before and after the pricing mechanism switch. The units of price change are US dollars. Positive values indicate an increase and negative values a decrease.

average percentage change is substantially positive. That is, some expensive instance types became cheaper in absolute terms, but overall the average price change is an increase of between 37% and 61%. To effect a retail pricing curve that is smoother than the market-driven curve, Amazon raised prices substantially.

Table 2 shows the 0.1, 0.25, 0.5, 0.75, and 0.9 quantiles for the percentage change in average $E(price)_h$ value broken out by region. Note that the median change for all regions is positive, with increases ranging

Region	Quantile				
	0.1	0.25	0.5	0.75	0.9
us-east-1	-89.99%	-29.54%	2.94%	25.11%	84.74%
us-east-2	-25.34%	-1.26%	30.07%	99.07%	132.98%
us-west-1	-29.51%	-6.44%	20.45%	84.48%	159.36%
us-west-2	-31.90%	-16.42%	8.59%	32.95%	122.05%

Table 2: Quantiles from distribution of average $E(price)_h$ value percentage change for North American AWS regions after the pricing mechanism switch.

from approximately 3% for *us-east-1* to 30% for *us-east-2*.

4.1 Correcting for Unlikely Price Changes

While the analysis we present is consistent with Amazon's characterization of the the new retail spot-pricing mechanism [3, 11, 12], we notice that some instance types became significantly more expensive after the change in pricing mechanism.

For example, consider the price series shown in Figure 3 for the *x1e.32xlarge* instance type in the *us-west-2b* AZ. The left-hand side of the figure shows the price series using the market-driven mechanism (*i.e.*, from August 2nd to October 31st, 2017). It clearly includes a price mode at approximately $245.00, which we interpret as a hidden reserve price. However, for short periods, it appears that Amazon removed the reserve price and the price dropped to approximately $1.5. In contrast, the right-hand side shows the price history after the switch to a retail mechanism (*i.e.*, from November 30th 2017 to February 28, 2018), during which time the price remained stable at approximately $24.00. Because the *x1e.32xlarge* instance type was indeed available before the switch to retail pricing at a price of $1.5, this change represents more than a 1000% increase in price. As is clear from the figure, however, the period of time during which a user could have obtained an instance at this price is relatively brief.

We observe similarly large price increases based on the same pattern for the *x1e.32xlarge* instance type in *us-west-2a* and *us-east-1d*. On the other hand, in *us-east-1b* and *us-east-1c* we see a substantial price drop, from approximately $244.00 before the switch to just over $24.00 after. Table 3 shows the comparison of average $E(price)_h$ values for the *x1e.32xlarge* instance type before and after the pricing mechanism change by AZ. Note that we did not find sufficient price history data for the *x1e.32xlarge* instance type in the other North American AZs leading us to suspect that this instance type was in the process of being introduced as a spot instance type and, thus, might have been in a "phase-in" period.

To account for the possibility that the data in Table 1 might be skewed by instance types that appear to be undergoing an introductory or "phase-in" period, we removed the pricing data for the lowest 2.5% and highest 2.5% (in terms of percentage price change) and computed the comparison of average $E(price)_h$

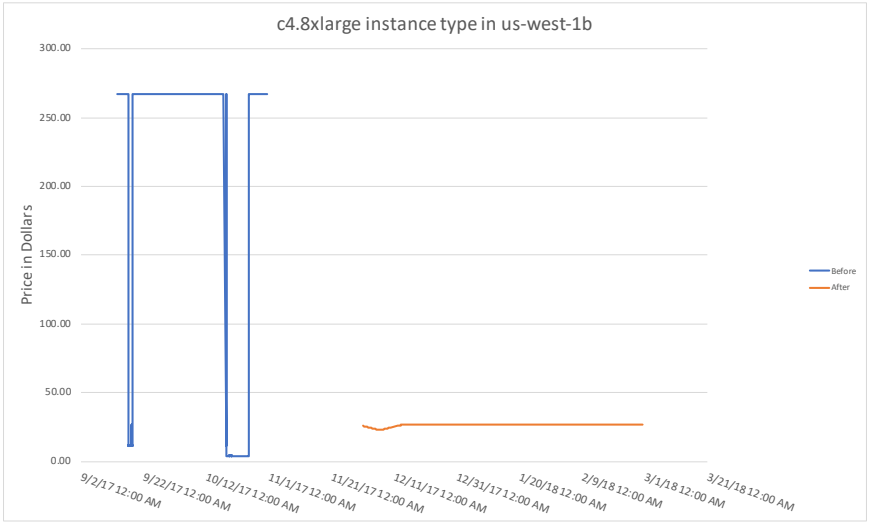

Figure 3: Amazon price history for *x1e.32xlarge* instance type in *us-west-2b* AZ for periods before and after the price mechanism change.

AWS AZ	Expected market-driven price	Expected retail price
us-east-1b	$243.79	$24.11
us-east-1c	$243.97	$24.34
us-east-1d	$1.13	$24.23
us-west-2a	$1.67	$23.33
us-west-2b	$1.54	$24.15

Table 3: Comparison of average $E(price)_h$ values for *x1e.32xlarge* instance type by AWS AZ. change are US dollars.

for the remaining "middle" 95% of the remaining instance types. Table 4 shows the same comparison as that depicted in Table 1 but eliding instance types with the highest increase and decrease percentage change leaving the "middle" 95%. These results indicate that the average $E(price)_h$ rose between 11% and 50%

AWS Region	Price Change	Percentage Change
us-east-1	-1.94	11.46%
us-east-2	-3.06	35.53%
us-west-1	0.27	50.19%
us-west-2	-2.23	18.65%

Table 4: Comparison of average $E(price)_h$ values for North American AWS regions before and after the pricing mechanism switch, eliding 2.5% of those exhibiting the highest percentage increase and 2.5% of those exhibiting the highest percentage decrease. The units of price change are US dollars.

(except for *us-west-1*, which rose slightly less, yielding between a 30% and 50% increase in expected price.

Amazon categorizes its instance types by possible usage and/or optimized capability. The current categories are "General Purpose," "Compute Optimized," "Memory Optimized," "Accelerated Computing," (*e.g.*, GPU and FPGA), and "Storage Optimized." Table 5 compares average $E(price)_h$ values by category, again removing 2.5% of extreme increases and decreases respectively. General Purpose instances became almost 7% cheaper and the average percentage increase for "Compute Optimized" is small. The other instance categories range from 15% increase for "Memory Optimized" to 88% increase for "Accelerated Computing" instances.

Figure 4 shows the distribution of percentage price changes as a "heat map" across all categories and AZs.

Category	Region				Avg. Pct Change
	us-east-1	*us-east-2*	*us-west-1*	*us-west-2*	
General Purpose	-26.03%	21.29%	-9.92%	-4.16%	-4.71%
Compute Optimized	-14.75%	14.32%	28.22%	-10.57%	4.30%
Memory Optimized	1.64%	17.64%	25.85%	17.03%	15.54%
Accelerated Compute	72.94%	32.51%	108.61%	42.30%	64.09%
Storage Optimized	23.52%	91.89%	98.17%	48.62%	65.55%

Table 5: Comparison of average percentage change in $E(price)_h$ values for North American AWS instance type category for each region after the pricing mechanism switch. The last column is the average percentage change across all regions.

We color each cell in the figure to indicate whether the price change was positive or negative. Blue shades

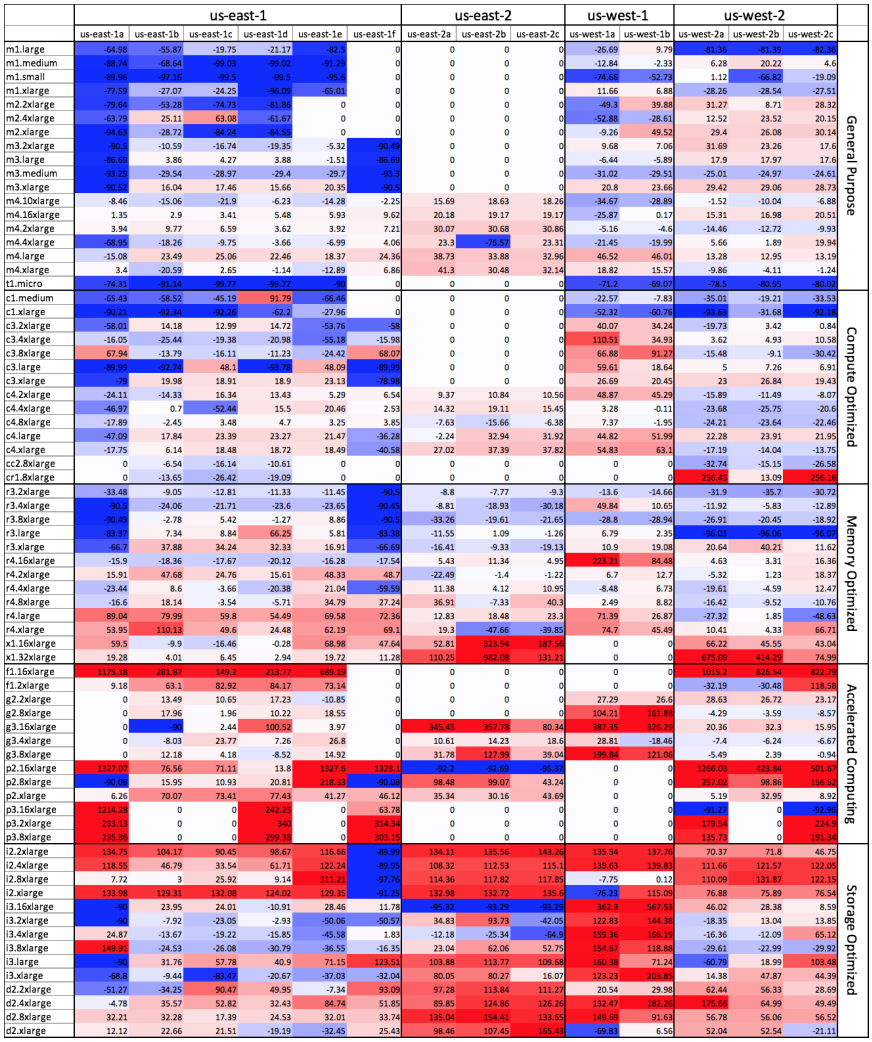

Figure 4: Percentage price changes after the switch to retail pricing shown as a heat map. Blue tones indicate a decrease in price and red tones indicate an increase. Columns correspond to AZs and rows to instance type.

indicate a decrease in price after the switch to retail pricing while red indicates an increase. In addition, the darker shades indicate larger magnitudes (positive or negative). Columns in the figure correspond to AZs

that are grouped by region. Rows in the figure correspond to instance types grouped by category.

While "General Purpose" and "Compute Optimized" instance types in the *us-east-1* region became less expensive on the whole, the other categories and regions combinations indicate price increases. These results seem generally to support those shown in Table 5 in that they show the greatest amount of price increase in the "Accelerated Compute" and "Storage Optimized" categories.

4.2 Price Change Since the Switch to Retail Pricing

Finally, we examine the changes in price for spot instances since the switch to retail pricing. To do so, we compare the 90-day period immediately following the switch (beginning November 30, 2017 and ending on Feb 28, 2018) to the most recent 90-day period to which we have access (beginning April 2nd, 2018 and ending July 1, 2018).

Category	Region				Avg. Pct Change
	us-east-1	*us-east-2*	*us-west-1*	*us-west-2*	
General	-1.13%	11.94%	-1.33%	20.47%	7.49%
Compute	6.46%	9.46%	-6.58%	4.10%	3.36%
Memory	8.19%	9.94%	-10.91%	2.66%	2.47%
Accelerated	-6.00%	-10.14%	-15.62%	2.22%	-7.38%
Storage	13.32%	22.85%	1.84%	7.83%	11.46%

Table 6: Comparison of average percentage change in $E(price)_h$ values for North American AWS instance type category for each region for the 90-day period immediately following the switch to retail pricing versus the most recent 90-day period. The last column is the average percentage change across all regions

.

Table 6 shows the average percentage change in price for each instance category broken out by AWS region. These numbers indicate an increase in *average* price for all instance categories except "Accelerated Compute." However, the quantiles for the distributions of average change shown in Table 7 indicate that the *median* prices, and indeed the middle half of prices, have remained relatively stable for "General Purpose," "Compute Optimized," and "Memory Optimized"; in terms of these statistics, "Accelerated Compute" shows a marked decrease and "Storage Optimized" a similarly pronounced increase.

Category	Quantile				
	0.1	0.25	0.5	0.75	0.9
General	-6.73%	-5.10%	0.00%	3.95%	13.26%
Compute	-14.48%	-6.37%	0.07%	5.10%	18.57%
Memory	-10.99%	-2.55%	1.51%	7.27%	18.41%
Accelerated	-33.92%	-22.42%	-9.82%	0.76%	32.70%
Storage	-2.94%	0.62%	6.89%	20.95%	31.59%

Table 7: Quantiles from distribution of average $E(price)_h$ value percentage change for instance categories in all North American regions regions for the 90-day period immediately following the switch to retail pricing versus the most recent 90-day period.

Note that the quantiles on either side of the median are more or less balanced for the first three instance categories.

Finally, Figure 5 shows the distribution of percentage change in prices when comparing the 90-day period immediately following the switch to retail pricing to the most recent 90-day period. Since the switch to retail pricing, Amazon has generally raised prices modestly with a few exceptions. The *t2* instances types in the *us-east-1* and *us-west-2* regions show a reduction in price as do most of the instance types in the "Accelerated Computing" category. In contrast, on the whole the "Storage Optimized" instance types became more expensive across all regions.

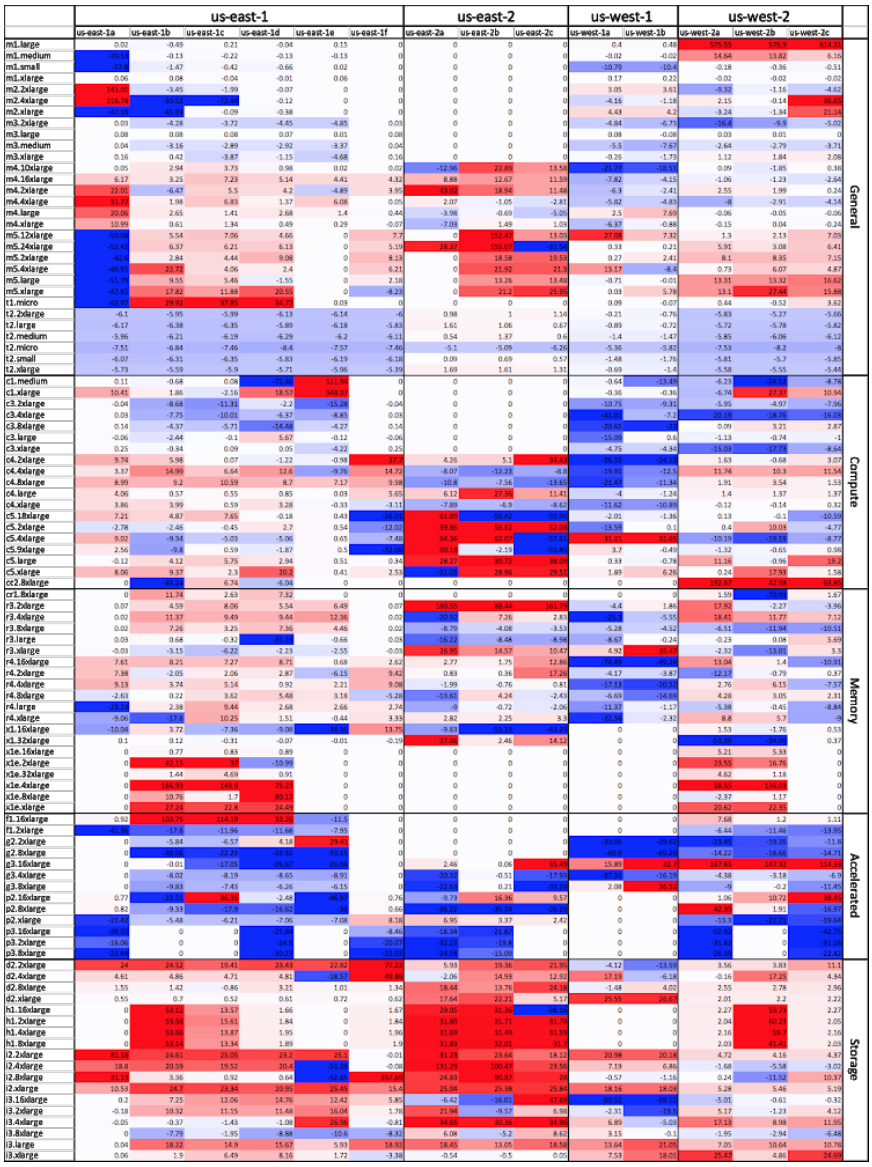

Figure 5: Percentage price changes for the 90-day period immediately following the switch to retail pricing versus the most recent 90-day period. Blue tones indicate a decrease in price and red tones indicate an increase. Columns correspond to AZs and rows to instance type.

5 Related Work

Amazon discloses recent spot price traces but provides only limited information about how spot prices are determined [2]. Much work has been done to understand the characteristics of Amazon's EC2 spot-instance pricing model; however, much of it predates the November 2017 pricing change announced by Amazon[10] and is now unfortunately no longer relevant to the new reduced-volatility spot market.

[19] suggests that the "statistics of Amazon's EC2's spot price history do not correspond to a CP profit-maximizing approach for setting spot price" but rather to Amazon's aim to utilize its spare capacity fully. This is, however, a contentious point according to [1], which doubts these findings due to the mechanics of the hidden reserve pricing it believes Amazon to be using. The paper also suggests a bidding strategy based on moving instances within the same instance family based on minimum expected payment to complete a job on each instance type. The efficacy of this idea may, however, be affected by the recent update as the availability of termination notices [13] changes considerations as risk of job termination is now known through this API.

The authors of [1] reverse-engineer EC2 spot-instance pricing before the November update using historical price data (traces) and suggest that spot prices were not market-driven but rather are generated at random from a dynamic reserve price mechanism. This reserve price mechanism means that bids below the undisclosed and changing reserve price are ignored. They also indicate that Amazon may have set minimum and maximum prices at which instances will sell. In effect, the "spot price is not market-driven but is set by Amazon according to an undisclosed algorithm." [1] They conclude that prices do not represent real client bids 98% of the time.

With Amazon's November update, the wording of [9] changed to indicate that instances might no longer be terminated only when the clients bid price is exceeded but might also reclaimed if "demand for Spot Instances rises, or supply of Spot Instances decreases." [9] This means that instances with bid prices above the market price may be terminated. Amazon seemingly addresses this by providing a new two-minute warning of impending termination with the termination notice API; however, these two factors fundamentally change the considerations for attempts at modeling bidding strategies for spot instances. [13] Much work has been done in the area of developing effective bidding strategies for spot instances; however, the recent changes to the spot market mean that these strategies can now know when instances are to be terminated, but can not necessarily avoid termination by adjusting their maximum bid price.

Various authors have investigated the effectiveness of using checkpointing and live migration to create bidding strategies that meet a range of requirements. In [29] the authors evaluate dynamic checkpointing strategies in an attempt to minimize cost specifically in relation to existing checkpointing strategies that were not cost aware. [17] compares various forms of dynamic checkpointing, specifically "memory checkpointing, memory checkpointing with lazy restore, and live" [17], with the goal of cost optimization.

[15] the authors attempt to use a model based on Markov Chains to recommend a minimal bid price to see a job complete with a certain probability and apply this model to problem of running mapreduce. [25] outlines a service called "SpotOn" with an emphasis on cost optimization for batches of jobs. They employ a service with multiple methods of fault tolerance and select where to run jobs based on the spot market that will offer the lowest cost, factoring in the probability of termination. In [30] the authors also look at the problem of cost optimizing bidding strategies for mapreduce.

In [28] the authors use QBETS [20, 21] to set bounds on a recommended bid price to guarantee a given duration of execution. This paper focuses on reliability, providing a probabilistic guarantee, or SLA as the paper tends to describe it, that an instance will run for a given duration. This type of probabilistic guarantee of duration, however, relies on being able to predict the probability of termination based on historical price data. With the latest market changes this kind of probabilistic SLA unfortunately no longer seems possible.

The paper [23] looks at pricing strategies for cloud providers and bidding strategies for clients. It suggests that a "fixed bidding strategy" might be optimal for clients. It also attempts to model the conditions under which it is worthwhile for a cloud provider to offer a spot market.

6 Conclusions

In this work, we study Amazon's switch to retail pricing for it interruptible "spot instance" class of service. In November of 2017 Amazon changed the mechanism for pricing spot instances from one based on user-

submitted "maximum bids" to a retail pricing mechanism in which the retail prices would change more smoothly than under the market-based mechanism.

Because the new pricing mechanism includes a new algorithm for determining which spot instances will be terminated during a resource shortfall, and that algorithm is private to Amazon, the switch to retail pricing also changed the reliability of spot instances as well. In particular, it is no longer possible to use prediction techniques such as those described in [28] to achieve predictable reliability. That is, accompanying the switch to smoother retail prices is a degradation of spot-instance reliability, which would seem to militate for a decrease in spot-instance price.

On the other hand, price smoothing might require an increase in price so that Amazon could cover sudden supply shortfalls that arise from changes in internal demand for the machine hosting spot instances. In addition, users might also find spot instances with smoother price histories more attractive, thereby increasing demand.

In this work, we analyze spot instance price histories for the North American regions immediately before the switch to retail pricing, immediately after, and for the most recent 90-day period (ending in June of 2018). This analysis shows that the switch to retail pricing (and a decrease in reliability) was generally accompanied by a price increase except for "General Purpose" instances types in the *us-east-1* region. Since the switch, prices have risen moderately except for *t2* instances types in *us-east1* and *us-west-2* and "Accelerated Computing" instance types. The recent price increases are most pronounced for "Storage Optimized" instance types.

More generally, this study demonstrates a challenge faced by users of virtualized datacenter infrastructure products. The change in pricing mechanism was accompanied by an unannounced set of price increases and reductions in reliability. If the infrastructure had been purchased and cited in a datacenter, the datacenter operators would have been able to inform users of upcoming changes so that users could have made appropriate preparations. The spot instance class of service is assuredly Amazon's least expensive class of service which might argue for immediate and unannounced changes in the product's function and price. However for cloud computing use cases (such as scientific research) where fixed-budget accounting is the norm, such changes represent a challenge to adoption.

References

[1] Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., and Tsafrir, D. Deconstructing amazon ec2 spot instance pricing. *ACM Transactions on Economics and Computation 1*, 3 (2013), 16.

[2] Amazon Web Services. AWS EC2 Spot Instance Price Histories. `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances-history.html` Accessed Jul-2018.

[3] Amazon Web Services. New AWS EC2 Spot Instance Pricing. `https://aws.amazon.com/about-aws/whats-new/2017/11/amazon-ec2-spot-introduces-new-pricing-model-and-the-ability-to-launch-new-spot-instances-via-runinstances-api/` Accessed Jul-2018.

[4] Amazon Web Services. Amazon EC2. `https://aws.amazon.com/ec2/`, 2018. [Online; accessed July-2018].

[5] Amazon Web Services. Amazon ec2 service level agreement, 2018. `https://aws.amazon.com/ec2/sla/` accessed July 2018.

[6] Amazon Web Services. Amazon ec2 spot instances, 2018. `https://aws.amazon.com/ec2/spot/` accessed July 2018.

[7] Amazon Web Services. Amazon instance pricing, 2018. `https://aws.amazon.com/ec2/pricing/on-demand/` accessed July 2018.

[8] Amazon Web Services. Amazon simple storage service, 2018. `https://aws.amazon.com/s3` accessed July 2018.

[9] Amazon Web Services. How spot instances work, July 2018. `https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/how-spot-instances-work.html` accessed July 2018.

[10] Amazon Web Services. New amazon ec2 spot pricing model, July 2018. `https://aws.amazon.com/blogs/compute/new-amazon-ec2-spot-pricing/` accessed July 2018.

[11] Amazon Web Services. New amazon ec2 spot pricing model: Simplified purchasing without bidding and fewer interruptions, 2018. `https://aws.amazon.com/blogs/compute/new-amazon-ec2-spot-pricing/` accessed August 2018.

[12] Amazon Web Services. New amazon ec2 spot pricing model: Simplified purchasing without bidding and fewer interruptions, 2018. `https://aws.amazon.com/blogs/aws/amazon-ec2-update-streamlined-access-to-spot-capacity-smooth-price-changes-instance-hibernation/` accessed August 2018.

[13] Amazon Web Services. New ec2 spot instance termination notices, July 2018. `https://aws.amazon.com/blogs/aws/new-ec2-spot-instance-termination-notices/` accessed July 2018.

[14] Aristotle Cloud Federation. `https://federatedcloud.org` [Online; accessed Aug-2018].

[15] Chohan, N., Castillo, C., Spreitzer, M., Steinder, M., Tantawi, A., and Krintz, C. See Spot Run: Using Spot Instances for MapReduce Workflows. In *Usenix HotCloud* (2010).

[16] Google Cloud Platform. Google preemptable virtual machines, July 2018. `https://cloud.google.com/preemptible-vms/` accessed July 2018.

[17] He, X., Shenoy, P., Sitaraman, R., and Irwin, D. Cutting the cost of hosting online services using cloud spot markets. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing* (New York, NY, USA, 2015), HPDC '15, ACM, pp. 207–218.

[18] Javadi, B., Thulasiram, R. K., and Buyya, R. Characterizing spot price dynamics in public cloud environments. *Future Generation Computer Systems 29*, 4 (2013), 988–999.

[19] Khodak, M., Zheng, L., Lan, A. S., Joe-Wong, C., and Chiang, M. Learning cloud dynamics to optimize spot instance bidding strategies.

[20] Nurmi, D., Brevik, J., and Wolski, R. Qbets: Queue bounds estimation from time series. In *Job Scheduling Strategies for Parallel Processing* (2008), Springer, pp. 76–101.

[21] Nurmi, D., Wolski, R., and Brevik, J. Probabilistic advanced reservations for batch-scheduled parallel machines. In *Proceedings of the 13th ACM SIGPLAN symposium on principles and practice of parallel programming* (2008), ACM, pp. 289–290.

[22] Schwarz, G. Estimating the dimension of a model. *Annals of Statistics 6*, 2 (1978).

[23] Song, J., and Guerin, R. Pricing and bidding strategies for cloud computing spot instances. In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (May 2017), pp. 647–653.

[24] Steve Fox. New aws spot pricing model: The good, the bad, and the ugly, July 2018. `http://autoscalr.com/2018/01/04/new-aws-spot-pricing-model-good-bad-ugly/` accessed July 2018.

[25] Subramanya, S., Guo, T., Sharma, P., Irwin, D., and Shenoy, P. Spoton: A batch computing service for the spot market. In *Proceedings of the Sixth ACM Symposium on Cloud Computing* (New York, NY, USA, 2015), SoCC '15, ACM, pp. 329–341.

[26] Wallace, R. M., Turchenko, V., Sheikhalishahi, M., Turchenko, I., Shults, V., Vazquez-Poletti, J. L., and Grandinetti, L. Applications of neural-based spot market prediction for cloud computing. In *Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2013 IEEE 7th International Conference on* (2013), vol. 2, IEEE, pp. 710–716.

[27] Willsky, A. S., and Jones, H. L. A generalized likelihood ratio approach to the detection and estimation of jumps in linear systems. *IEEE Transactions on Automatic Control 21*, 1 (1976).

[28] Wolski, R., Brevik, J., Chard, R., and Chard, K. Probabilistic guarantees of execution duration for amazon spot instances. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2017), ACM, p. 18.

[29] Yi, S., Kondo, D., and Andrzejak, A. Reducing costs of spot instances via checkpointing in the amazon elastic compute cloud. In *2010 IEEE 3rd International Conference on Cloud Computing* (July 2010), pp. 236–243.

[30] Zheng, L., Joe-Wong, C., Tan, C. W., Chiang, M., and Wang, X. How to bid the cloud. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (New York, NY, USA, 2015), SIGCOMM '15, ACM, pp. 71–84.