Providing Statistical Reliability Guarantees in the AWS Spot Tier*

Rich Wolski
Computer Science Department
University of California, Santa Barbara

John Brevik
Department of Mathematics
California State University, Long Beach

Abstract

In this paper, we present DrAFTS – a methodology for predicting the value a user should bid in the AWS Spot tier to ensure that the request lifetime exceeds a fixed duration with a given probability. DrAFTS uses previous price histories for each category of request to determine both the minimum bid value and the concomitant duration associated with that value that is "guaranteed" with a specified probability. We also describe how DrAFTS can be used to determine the duration associated with overbidding the minimum by a specific percentage.

1 Introduction

Infrastructure as a Service (IaaS) is a form of cloud computing in which computational infrastructure componentry (computers, network connectivity, storage capabilities, delegated identity, etc.) can be "provisioned" via one or more web-service interfaces. "Public clouds" such as Amazon's *Amazon Web Services* (AWS) [5, 14, 15], Google's *Google Compute Engine* [9, 13] implement IaaS using virtualized resources as rentals that are available to the public. Customers who rent these capabilities via an e-commerce transaction do so under the terms of a Service Level Agreement (SLA) for each IaaS service they choose.

The predominant public-cloud vendor of IaaS services today is Amazon, and its *Elastic Compute Cloud* (EC2) [6] is the most popular venue for renting virtual machines to the general public.

Virtual machines are available in two quality-of-service service (QoS) tiers from AWS, each governed by a different SLA. The SLA for the *On-demand* tier guarantees at least 99.95% availability during each month of usage or a 10% dollar-cost refund can be issued. Additionally, less than 99% availability results in eligibility for 30% refund, which is the maximum offered. By contrast, in the *Spot-market* tier, the SLA stipulates that a virtual machine will run until the maximum price its owner is willing to pay is exceeded by the "Spot price" set by Amazon as a function of supply and demand [3] (or an outage occurs, for which no probability is given).

Because the SLA for Spot instances does not provide a probabilistic guarantee on instance lifetime, the price of a instance in the Spot tier is often significantly lower than in the On-demand tier. This discount often makes the Spot tier attractive to High-performance Computing (HPC) users, particularly is academically funded research contexts. HPC applications, however, typically require long-running, uninterrupted access to the computing resources they use and resources in the Spot tier carry no guarantee of sustained availability.

In this paper we describe a method for determining probabilistic guarantees of availability for Amazon AWS Spot instances called **DrAFTS** which is an acronym for **Dur**ability **A**greements **F**rom **T**ime **S**eries. DrAFTS uses a non-parametric statistical prediction algorithm, to make predictions of bid prices that will "win" in the AWS spot market. It also quantifies (using back-testing) the duration over which these bid prices will persist with a specified probability. Thus, given such a probability as input, a DrAFTS prediction is an ordered pair consisting of

- a *price* – a maximum bid price that an AWS Spot-market customer should bid in order to *minimize* the cost, and
- a *duration* – the duration over which that bid price will be sufficiently high to prevent the Spot instance from being terminated either because the Spot price has risen or because the supply of available Spot instances (termed the available "Spot pool" by Amazon) has decreased

such that the output price will result in the output duration with a probability at least as high as the one used

*This work is supported in part by NSF (CNS-0905237, CNS-1218808, ACI-0751315)

as input. Assuming the probability of outage is negligible, then, each DrAFTS prediction provides a guarantee that can be used in an SLA (in terms of availability) for a Spot instance and the price that the customer must pay to obtain that guarantee.

DrAFTS Implementation

We describe the DrAFTS prediction methodology and its implementation as an on-line prediction service that can be accessed from the URL given in [22]. DrAFTS is designed to incorporate new pricing data as it becomes available. The DrAFTS service implementation queries AWS Spot-market pricing data [2] periodically (every 15 minutes in this study) and updates its predictions so that they are for the current time frame. We demonstrate, using historical pricing data available from AWS, the efficacy of the methodology. We also discuss the trade-off between accuracy, statistical soundness, and performance in the implementation.

2 AWS EC2 Spot Instances and Prices

Under the "EC2-Classic" style of AWS usage (one that does not use the Virtual Private networking feature) a user requests a Spot Instance by specifying a three tuple of

$$(Region, Instance\_type, Max\_bid\_price) \quad (1)$$

EC2 is organized as independent *Regions*, each of which constitutes essentially a separate instantiation of the EC2 functionality. That is. EC2 resources are not shared across Regions (although cross-Region data sharing is facilitated by the Simple Storage Service [4, 7]). Each Region is divided into *Availability Zones* (AZs), which define pools of resources with purportedly independent failure probabilities so that the joint probability of failure in multiple zones can be determined. Each Spot instance runs in a single Region and AZ. The user must specify the Region, and can specify the AZ or can elect to allow the Spot service itself to select an AZ in the Region. It is also possible to specify that a group of Spot Instance requests be treated as a unit with respect to initiation (they will be put in the same AZ) and termination (they will terminate together).

A user initiating a Spot instance request specifies a maximum bid price. The Spot service keeps the maximum prices hidden from other users. An Spot instance request will be initiated when the maximum bid price exceeds the current *market price* and the user is charged the market price per hour that the instance runs. If the market price exceeds the maximum bid price, the instance is terminated. If the market price is equal to the maximum bid price, the instance may be terminated if the size of the Spot pool decreases. Amazon does not describe the conditions under which capacity will be removed from or added to the Spot pool.

The market price is set as follows. When a new request is issued or the Spot pool shrinks, the outstanding maximum bids are sorted in descending order and resources from the pool are allocated (based on request size) exhaustively in order of bid size until no more requests can be accommodated. The smallest maximum bid price for a request that can be accommodated is the market price.

Thus, the duration that an instance will run before it is terminated is determined (assuming that the hardware failure probability is negligible) by the time until the market price exceeds the maximum bid price for the instance. The goal of DrAFTS is to predict a combination of maximum bid price and duration at the time of an instance request that will ensure that the predicted maximum bid price does not exceed the market price for *at least* the duration of time predicted with the specified probability.

Spot Instance Price Histories

AWS makes the Spot price history for each instance type, in each Region and AZ available via a web-accessible Application Programming Interface (API). The histories record the time when the market price changes and up to 90 days of previous time period are available for each history.

In this study, we have accumulated price histories for the *us-east-1*, *us-west-1*, and *us-west-2* AZs spanning the period from November 2014 to December 2015. The API is queried every 15 minutes for the previous 2 hour period (to account for possible dropout) and duplicate entries are removed. We further restrict the DrAFTS predictions discussed in Section 4 to the *Linux/UNIX* images (Suse and Windows images are also available in the Spot market at different price points).

The price data is updated every 5 minutes, and there are two sources are available for this information. Amazon publishes the lowest spot price for each instance type in each region (*i.e.* the lowest spot price across all AZs in a region) via a publicly accessible web page. However, it also discourages the practice of "scraping" this page for the current regional Spot price for reasons that are unclear to us.

The second source, and the one recommended by Amazon, is via the AWS web-services API. Rather than providing pricing data on a regional basis, the web-service API returns prices for each instance in each AZ of each region. The regional Spot price from this data is the minimum (at any given time in the pricing history) across all AZs in the region. In this study, we use this aggregation method (and not "scraped" data) to determine the regional Spot price. We hope to develop an AZ-specific capability as part of our future work.

Also, the documentation for the API indicates that each element in the price history "merely indicates the last time that the price changed." However, some histo-

ries include consecutive records that do not show a price change. For example, consider the following segment from the price history for the *t1.micro* instance type in the *us-east-1b* AZ.

```
us-east-1b t1.micro 0.0031 2014-12-01T15:17:24
us-east-1b t1.micro 0.0031 2014-12-02T15:18:39
us-east-1b t1.micro 0.0031 2014-12-03T15:21:08
us-east-1b t1.micro 0.0031 2014-12-04T15:23:25
us-east-1b t1.micro 0.0031 2014-12-05T15:25:19
us-east-1b t1.micro 0.0031 2014-12-06T15:26:58
us-east-1b t1.micro 0.0031 2014-12-07T15:28:11
us-east-1b t1.micro 0.0031 2014-12-08T15:31:13
```

Notice that the time stamps occur more frequently than every 5 minutes and that the prices do not change.

The explanation for this discrepancy is not entirely clear. One reason may be that the data is being slightly obscured to help balance the load across AZs. Amazon attempts to prevent its community of users from overloading any one AZ by changing the AZ names that are reported based on the user ID associated with the API request. Apparently, users have a preference for for AZ names that are "lower" and thus occur earlier in a lexigraphic sorting of the names. To prevent such "herd behavior," Amazon ensures that each user gets a different mapping of AZ names to resources. Indeed, it is likely that the reason only regional and not per-AZ data is available via a public web page is that the web request does not require a valid AWS user ID.

One way to circumvent this obscured mapping would be to compare Spot-price histories from different users. It may be that Amazon is changing the histories slightly so that the time stamps cannot be used to determine a globally consistent naming scheme for AZs.

Another alternative possibility is that each time stamp represents either a price change or a change in the size of the Spot Pool. Note that there are two circumstances under which a Spot instance will be terminated: Either the maximum bid price associated with the instance is exceeded by the market spot price, or the maximum bid price is *equal* to the market price and the size of the "Spot Pool" shrinks. AWS does not make data about the size of the Spot Pool available since, presumably, it is affected by demand for non Spot resources and overall AWS load is held as a trade secret. Therefore, in this study we assume that each record in the price history either indicates a price change or is a duplicate that does not convey any new information.

3 Methodology

The DrAFTS prediction mechanism takes, as a parameter, a target probability (say, 0.95) and returns the lowest maximum bid price and the duration the price will meet or exceed the market with *at least* that probability. Thus, from the perspective of a user of the Spot tier, it answers the question

> *For a specified probability, can we meaningfully generate a price and a duration so that bidding that price gives a probability at least as great as that specified that the job will run for that duration before being terminated?*

That is, DrAFTS attempts to find the lowest maximum price to bid that will guarantee a duration of execution such that the probability of "success" – defined as the instance running for at least the time duration returned by DrAFTS – is greater than or equal to the target probability.

Note that as the price approaches the current market price the duration approaches zero since any bid at market price is immediately eligible for termination. Notice also that any bid price above the DrAFTS predicted bid price will also meet the probability target (any probability greater than the target is acceptable). A bid price above the DrAFTS bid will also potentially increase the durability of the bid but with the added risk associated with paying a higher price.

DrAFTS applies QBETS (described in the next Subsection) to the most recent time series of price history data from AWS to determine both the maximum bid price and the duration. It is intended to be used on-line, as part of a service like the one available from [22]. The service queries the AWS for the "latest" price history data for a specific AWS instance type, executes the DrAFTS mechanism using this price history and a target probability as inputs, and produces a predicted maximum bid price and predicted duration for that instance type. If the user makes a request using this "fresh" prediction, then, it should be instantiated and "survive" for at least the quoted duration with the probability given.

In this way, DrAFTS predictions play the same role that a quantified reliability quotation does in a reliability SLA. However, they are slightly more restrictive than the SLAs given by Amazon in that they are for *continuous* availability durations. Technically, the AWS SLA specifies a percentage of availability time that is cumulative. That is, as long as the instance appears available for for 99.95% of the seconds in a month, the AWS SLA is fulfilled. The DrAFTS probability refers not to the cumulative availability, but to the continuous availability of a specific instance request.

QBETS: Quantile Bounds Estimation from Time Series

DrAFTS uses Queue Bounds Estimation from Time Series (QBETS) [17, 19], a non-parametric time series analysis method that we developed in prior work. We originally designed QBETS for predicting the scheduling delays for the batch queue systems used in high performance computing environments but it has proved effective in other settings where forecasts from arbitrary times

series are needed [16, 8, 20, 12]. In particular, it is both non-parametric and it automatically adapts to changes in the underlying time series dynamics making it useful in settings where forecasts are required from arbitrary data with widely varying characteristics.

A QBETS analysis requires three inputs:

1. A time series of data.
2. The quantile for which a confidence bound should be predicted ($p \in (0,1)$). .
3. The confidence level of the prediction ($c \in (0,1)$).

QBETS uses this information to predict an upper bound for the q^{th} quantile of the time series. It does so by treating each observation in the time series as a Bernoulli trial with probability q of success. If there are n observations, the probability of there being exactly k successes is described by a Binomial distribution (assuming observation independence) having parameters n and q. If Q is the q^{th} quantile of the distribution from which the observations have been drawn, the equation

$$\sum_{j=0}^{k} \binom{n}{j} \cdot (1-q)^j \cdot q^{n-j} \tag{2}$$

gives the probability that no more than k observations are greater than Q. As a result, the k^{th} largest value in a sorted list of n observations gives an upper c confidence bound on Q when k is the smallest integer value for which Equation 2 is larger than c.

QBETS also attempts to detect changepoints in the time series of observations so that it can apply this inference technique to only the most recent segment of the series that appears to be stationary. Details of an efficient implementation as well as a fuller accounting of the statistical properties (including correction for autocorrelation) and detailed assumptions are available in [17, 19, 16, 8, 18].

DrAFTS Prediction Methodology

DrAFTS uses QBETS to predict an *upper* bound on maximum bid price and a *lower* bound on duration. It uses the square root of the desired target probability as the q^{th} quantile and, in the study, a value of 0.99 for the confidence level c. While other probability combinations are possible to reach the target probability, experience shows us that using square roots strikes a good balance between keeping a bid low (i.e. near the current price) and yielding a usable duration. Similarly, we choose $c = 0.99$ to illustrate how DrAFTS could be used as the basis for an SLA. If a statistical guarantee that is less stringent is useful, then smaller values of c can be used.

DrAFTS uses QBETS to create a history of predictions, one each each point in the price history series where pricing data is available. It then records the duration in the previous history over which each prediction is "valid," where a valid prediction is one that is greater than or equal to the Spot price it predicts. That is, for each QBETS prediction in the prediction history, DrAFTS computes the duration until that prediction is no longer sufficient to prevent AWS from terminating a Spot-instance due to price if the QBETS-predicted price had been used as the maximum bid price. This procedure generates a history of durations, one per QBETS prediction. DrAFTS then uses QBETS again to predict a lower confidence bound bound (again with $c = 0.99$) on the 0.025 ($= 1 - .975$) quantile of the duration – note here that we use the *conditional* probability that the price allows the instance to run in the first place – based on the duration history.

Backtesting

Both as a diagnostic and to improve the credibility of the bid forecasts, DrAFTS uses *backtesting* for each instance type price history and reports the fraction of successful predictions. That is, each time a DrAFTS prediction is generated, the method also generates a random sample of previous predictions for which the outcomes can be observed. To do so, it repeatedly chooses a time stamp at random in the previous price history and runs the DrAFTS algorithm using the data before that time stamp in the history. The fraction of correct predictions over the sample is reported as the observed success probability.

4 Results

We begin with an example of the predictions generated by DrAFTS. For each region, Table 1 shows the current *predicted* bid price (plain-faced text), the duration in hours (bold-faced text) and the probability that the predicted price will prevent an eviction for *at least* the duration shown based on a back-testing sample size of 100 using the previous months pricing history.

While there are 129 different possible combinations of instance type and region, Amazon supports 120 of them. The 9 that are not supported are represented by blank elements in the table. This particular example was gathered on September 2, 2015 from the on-line service located at the URL shown in [22]. The target probability set for the service is at least 0.95. That is, the table shows the price to bid in the region to achieve at least a 0.95 probability of having the instance "survive" for the duration shown. The probability shown in the table is a verification (through backtesting over the previous month's price history) that the predicted duration is valid. Thus, when the probability in the table is greater than or equal to 0.95, the prediction methodology has been successful at least 95% of the time in identifying a lower bound on the duration associated with a predicted price during the previous month. For these entries, it is then "reasonable" to assume the given predicted maximum bid will generate

Inst. Type	us-east-1	us-west-1	us-west-2
c1.medium	$0.0163, **3.71 (hrs)**, *0.97*	$0.0163, **0.99 (hrs)**, *0.95*	$0.0162, **4.12 (hrs)**, *0.99*
c1.xlarge	$0.0643, **0.68 (hrs)**, *0.95*	$0.0645, **0.88 (hrs)**, *0.98*	$0.0653, **0.11 (hrs)**, *0.99*
c3.2xlarge	$0.1126, **0.05 (hrs)**, *0.95*	$0.1142, **0.04 (hrs)**, *0.97*	$0.0712, **0.08 (hrs)**, *0.96*
c3.4xlarge	$0.2001, **0.04 (hrs)**, *0.99*	$0.2199, **0.06 (hrs)**, *0.95*	$0.2003, **0.08 (hrs)**, *0.93*
c3.8xlarge	$0.7501, **0.09 (hrs)**, *0.97*	$0.3427, **0.03 (hrs)**, *0.97*	$0.3887, **0.05 (hrs)**, *0.99*
c3.large	$0.0196, **0.10 (hrs)**, *0.97*	$0.0178, **0.03 (hrs)**, *0.96*	$0.0180, **0.10 (hrs)**, *0.97*
c3.xlarge	$0.0394, **0.08 (hrs)**, *0.96*	$0.0397, **0.06 (hrs)**, *1.00*	$0.0363, **0.08 (hrs)**, *0.99*
c4.2xlarge	$0.0877, **0.10 (hrs)**, *0.99*	$0.0651, **0.06 (hrs)**, *0.99*	$0.0737, **0.08 (hrs)**, *0.96*
c4.4xlarge	$0.4233, **0.06 (hrs)**, *0.96*	$1.1042, **0.06 (hrs)**, *0.97*	$0.9281, **0.25 (hrs)**, *0.98*
c4.8xlarge	$0.4948, **0.10 (hrs)**, *0.90*	$0.7201, **0.09 (hrs)**, *0.99*	$0.6760, **0.08 (hrs)**, *0.97*
c4.large	$0.0255, **0.10 (hrs)**, *0.96*	$0.0171, **0.12 (hrs)**, *0.97*	$0.0206, **0.10 (hrs)**, *1.00*
c4.xlarge	$0.0414, **0.04 (hrs)**, *0.96*	$0.0551, **0.06 (hrs)**, *1.00*	$0.0658, **0.08 (hrs)**, *0.93*
cc2.8xlarge	$0.3225, **0.10 (hrs)**, *0.95*		$0.2858, **0.08 (hrs)**, *0.98*
cr1.8xlarge	$0.3497, **0.07 (hrs)**, *0.98*		$0.2677, **0.19 (hrs)**, *0.96*
d2.2xlarge	$0.1579, **0.16 (hrs)**, *0.97*		$0.1501, **0.31 (hrs)**, *0.97*
d2.4xlarge	$0.2888, **0.15 (hrs)**, *0.95*		$0.3468, **0.48 (hrs)**, *0.92*
d2.8xlarge	$0.5809, **0.10 (hrs)**, *0.96*		$0.6661, **0.10 (hrs)**, *0.99*
d2.xlarge	$0.0889, **0.67 (hrs)**, *0.97*		$0.0734, **0.24 (hrs)**, *1.00*
g2.2xlarge	$0.1498, **0.00 (hrs)**, *1.00*	$0.4001, **0.03 (hrs)**, *0.99*	$0.6501, **0.08 (hrs)**, *0.99*
g2.8xlarge	$1.1210, **0.10 (hrs)**, *0.99*	$0.9063, **0.09 (hrs)**, *0.95*	$0.4441, **0.10 (hrs)**, *0.99*
hi1.4xlarge	$0.1750, **0.30 (hrs)**, *0.96*		$0.1550, **0.10 (hrs)**, *0.97*
m1.large	$0.0161, **24.65 (hrs)**, *0.97*	$0.0173, **0.21 (hrs)**, *0.97*	$0.0172, **0.08 (hrs)**, *0.96*
m1.medium	$0.0081, **21.38 (hrs)**, *0.97*	$0.0100, **0.23 (hrs)**, *1.00*	$0.0108, **0.08 (hrs)**, *0.97*
m1.small	$0.0071, **31.21 (hrs)**, *0.98*	$0.0082, **5.44 (hrs)**, *0.98*	$0.0082, **0.25 (hrs)**, *0.99*
m1.xlarge	$0.0321, **22.19 (hrs)**, *1.00*	$0.0323, **2.97 (hrs)**, *1.00*	$0.0335, **0.16 (hrs)**, *1.00*
m2.2xlarge	$0.0384, **0.14 (hrs)**, *1.00*	$0.0327, **2.06 (hrs)**, *1.00*	$0.0447, **0.08 (hrs)**, *0.99*
m2.4xlarge	$0.0962, **0.09 (hrs)**, *1.00*		$0.1075, **0.00 (hrs)**, *1.00*
m2.xlarge	$0.0354, **0.04 (hrs)**, *1.00*		$0.0216, **0.16 (hrs)**, *1.00*
m3.2xlarge	$0.0736, **0.10 (hrs)**, *1.00*	$0.0906, **0.03 (hrs)**, *0.99*	$0.0686, **0.08 (hrs)**, *0.98*
m3.large	$0.0204, **0.09 (hrs)**, *0.99*	$0.0171, **0.06 (hrs)**, *1.00*	$0.0167, **0.08 (hrs)**, *1.00*
m3.medium	$0.0095, **0.62 (hrs)**, *1.00*	$0.0181, **0.09 (hrs)**, *1.00*	$0.0121, **0.10 (hrs)**, *1.00*
m3.xlarge	$0.0527, **0.06 (hrs)**, *0.99*	$0.0354, **0.06 (hrs)**, *1.00*	$0.0392, **0.08 (hrs)**, *0.99*
m4.10xlarge	$0.3298, **0.13 (hrs)**, *1.00*	$0.4797, **0.27 (hrs)**, *1.00*	$0.3385, **0.08 (hrs)**, *0.99*
m4.2xlarge	$0.0530, **0.03 (hrs)**, *1.00*	$0.0593, **0.29 (hrs)**, *0.99*	$0.0544, **0.10 (hrs)**, *0.99*
m4.4xlarge	$0.1053, **0.10 (hrs)**, *0.99*	$0.1333, **0.14 (hrs)**, *1.00*	$0.3501, **0.15 (hrs)**, *0.99*
m4.large	$0.0140, **0.00 (hrs)**, *1.00*	$0.0144, **0.03 (hrs)**, *1.00*	$0.0141, **0.72 (hrs)**, *1.00*
m4.xlarge	$0.0270, **0.09 (hrs)**, *1.00*	$0.0287, **0.06 (hrs)**, *1.00*	$0.0270, **0.26 (hrs)**, *0.99*
r3.2xlarge	$0.2501, **0.05 (hrs)**, *1.00*	$0.1667, **0.06 (hrs)**, *1.00*	$0.0893, **0.08 (hrs)**, *0.99*
r3.4xlarge	$0.2401, **0.07 (hrs)**, *0.99*	$0.3288, **0.12 (hrs)**, *1.00*	$0.3453, **0.07 (hrs)**, *0.99*
r3.8xlarge	$0.6780, **0.12 (hrs)**, *1.00*	$0.4947, **0.09 (hrs)**, *1.00*	$0.5188, **0.00 (hrs)**, *0.99*
r3.large	$0.0195, **0.04 (hrs)**, *1.00*	$0.0637, **0.06 (hrs)**, *0.99*	$0.0174, **0.16 (hrs)**, *1.00*
r3.xlarge	$0.2201, **0.09 (hrs)**, *1.00*	$0.0940, **0.06 (hrs)**, *1.00*	$0.0492, **0.08 (hrs)**, *0.99*
t1.micro	$0.0031, **23.00 (hrs)**, *1.00*	$0.0035, **1.16 (hrs)**, *0.99*	$0.0032, **0.75 (hrs)**, *1.00*

Table 1: *DrAFTS predictions for AWS instance types in us-east-1, us-west-1, and us-west-2 gathered on September 2, 2015. Missing entries indicate data not available via the AWS price history API.*

at least the duration of lifetime with at least a probability of 0.95.

These predictions are conservative in several ways. First, as noted in Section 3, QBETS generates a conservative bound. Thus, each QBETS prediction is larger or smaller that the "true" quantile with the (high) probability supplied to it as parameters. Second, the price prediction is intended to serve as the maximum bid for a Spot instance. The actual price a user of DrAFTS will pay is statistically guaranteed to be less than or equal to this maximum bid price. Finally, the times shown are the time until the instance is *eligible* to be terminated because of price. AWS does not document the algorithm it uses to determine which instances will be terminated due to a supply shortfall when the size of the Spot Pool shrinks; this makes it difficult to quantify the degree of underestimation. However, it is noteworthy that even with such conservative estimates, relatively long statistical guarantees are possible for some instance types in some regions. For example, from Table 1 on September 2, 2015 a maximum bid $0.0071 for a *m1.small* instance type in the *us-east-1* region results in the instance starting and being eligible for termination (with probability at least 0.95) no sooner than 31.21 hours after it is instantiated and the fraction of DrAFTS predictions that were successful for this instance type-region combination over the previous month is 0.98.

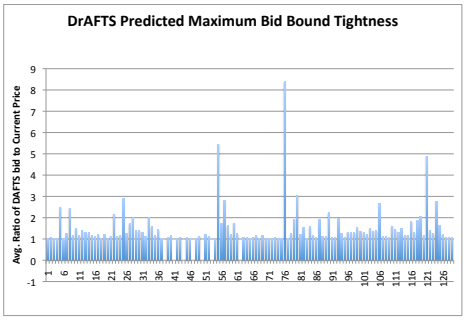

Figure 1: *Average ratio of DrAFTS prediction to current price, month ending September 2, 2015.*

Prediction Correctness versus Tightness

Implicit in this exposition is the notion that a DrAFTS prediction of bid price and duration is **correct** if the duration an instance will experience until it is terminated the predicted duration. The probability quoted with each prediction is the fraction of recent previous predictions (taken from a random sample) that have proved correct in this way.

While this definition is sufficient with respect to the provision of a minimum statistical guarantee of future availability, it does not take into account the degree of over-estimation that this level of statistical certainty engenders. For example, it is possible to construct a predictor that is trivially correct with a fraction of 0.95 under this definition by choosing a large maximum bid price (say, $1000) and a short duration (300 seconds) for 95 out of every 100 predictions made and a low bid price with a long duration for the other 5 predictions per 100. Overall, the predictor would be correct "95% of the time," but the correct predictions would be impractically conservative.

To investigate the **tightness** of the DrAFTS methodology with respect to bid price, we show the average ratio of the DrAFTS-predicted maximum bid price versus Spot price from a sample taken over one month's period in Figure 1. Due to space considerations, we show the data in graphical form rather than tabular form. As a result, the specific instance-type and region names associated with each measurement are not shown in in the figure. Instead, we enumerate the instance-type-region pairs shown in Table 1 using row-major order. That is, *us-east-1-c1.medium* has index 1, *us-west-1-c1.medium* has index 2, *us-west-2-c1.medium* has index 3, *us-east-1-c1.xlarge* has index 4 and so forth. In the figure, the *x*-axis shows the index of each combination and the *y*-axis the average ratio of DrAFTS maximum bid price to

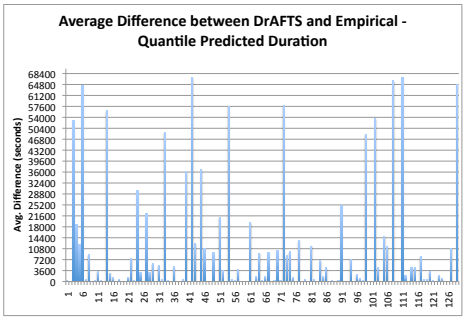

Figure 2: *Average difference in lower bound predictions of duration-until-termination eligibility between DrAFTS and empirical quantiles. The units are seconds.*

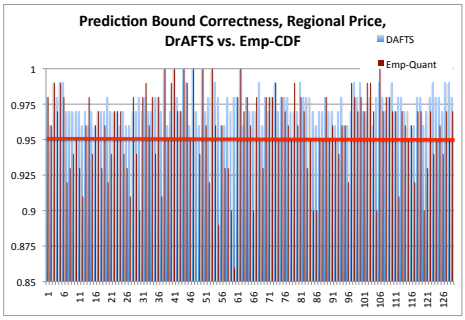

Figure 3: *Correctness Comparison between DrAFTS and Empirical Quantiles,* 500 *samples, for the month ending September 2, 2015. Correctness fractions of* 0.95 *or greater meet the correctness target for the experiment.*

current Spot price over the month under study. A ratio of 1.0 indicates, on the average, that the DrAFTS prediction for the maximum bid price was equal to the Spot price at the time of the prediction. We see from the graph that the average ratio exceeds 2.0 for only 12 of the 120 different possible instance-type-region combinations, and only 3 exceed a ratio of 3.0. For the majority of the combinations, the ratio is substantially below 2.0 indicating that a maximum bid that is less than twice the current Spot price (on the average) is sufficient to achieve a 0.95 probability bound.

Determining the tightness of DrAFTS predictions with respect to duration is more problematic since the "true" duration information is not available from Amazon. Further, running Spot instances to generate a sample of observed durations for all 120 different instance-type-region combinations over the course of a month represents a considerable expense for a significant sample size. Instead, in Figure 2 we compare the DrAFTS duration prediction to a prediction made using the empirical quantile value taken from the data.

The empirical quantile is simply the quantile taken from a sample of the data. For example, the empirical 0.95 quantile from a sample of size 100 is the 95$^{\text{th}}$ largest value in the sample. From a practical perspective, many practitioners often use the empirical quantile from the AWS price history data as an estimate of the "true" quantile. However, since it is drawn from a random sample, it is itself a random variable that varies "around" the true quantile of the distribution from which the sample is drawn.

Figure 2 shows the average difference between the 0.95 lower bound on Spot-instance duration when computed using DrAFTS and using the empirical quantiles in place of DrAFTS predictions (including the correction for equality). For each combination of instance-type and region, we generate a random sample of 100 time stamps from the month preceding September 2, 2015. We then use backtesting (described in Section 3) to generate equivalent analogous predictions using DrAFTS empirical quantiles and show the average difference (in seconds) in the figure. Because the predictions are obtained in identical manners except for the use of the empirical quantile instead of QBETS in each prediction, the result shows the average degree of "conservativeness" in duration prediction that DrAFTS introduces over simply using the observed quantiles at each point in time. That is, the larger the value shown in Figure 2, the *shorter* the time-to-termination eligibility (*i.e.* the durability) predicted by DrAFTS relative to using the empirical quantile to make the same prediction. This difference (which for some combinations is substantial) represents the degree conservativeness DrAFTS adds to the empirically observed quantiles to ensure that its predictions are correct with respect to the specified target probability.

Comparing Correctness of DrAFTS and Empirical Quantiles

Because the DrAFTS durability conservativeness relative to the empirical quantile is so large in some cases, it is worth investigating where the empirical quantile method is a reasonable alternative to the use of QBETS for DrAFTS. Figure 3 shows a comparison of correctness (using backtesting with a sample size of 500 from the month ending September 2, 2015) between DrAFTS duration predictions and duration predictions made using the empirical quantiles. The target probability in both cases is 0.95 and the only difference (as is the case in Figure 2) between the two cases is in how the bounds are determined. The x-axis in the figure shows the index of the instance-type-region combinations (as in Figure 1). The y-axis shows the fraction of correct predictions, from a random sample of 500 time stamps taken from the month ending September 2, 2015, determined during back testing. The light-colored bars are for DrAFTS predictions, and the dark-colored bars depict correctness fractions for the empirical quantile method.

In this experiment, the target fraction is 0.95 (indicated by the thick horizontal line in the graph). Thus

any correctness fraction that is less than 0.95 fails to achieve the desired target correctness set for the experiment. DrAFTS predictions for all 120 combinations for which there is historical Spot price data in this experiment meet the correctness criteria (all light-colored bars are at or above the thick horizontal line).

From the coloration in the figure, it is clear that many, but not all of the empirical quantile predictions are "correct" under our definition (not all dark bars reach the thick horizontal line). Thus by being conservative in the degree shown in Figure 2, DrAFTS is able to achieve correctness for all instance-type and region combinations *automatically* without the need for additional analysis. In contrast, the empirical quantile method is correct for some but does not provide a "guaranteed" correctness for all.

Determining the Price of Durability

Most typically, users of the AWS Spot tier require a way to answer the question

> *What bid minimizes the potential spend while guaranteeing a specific target duration?*

For example, a high-performance computing user who knows that her job will not exceed 1 hour in duration would like to know what to bid to *both* minimize the cost *and* guarantee at least 1 hour before an eviction is possible.

DrAFTS generates answers to such queries by incrementally computing the durations associated with successively larger amount of "overbidding" with respect to the minimum. Note that in the same way DrAFTS computes the minimum time until eviction for the minimum bid, it can also compute the minimum time until eviction for the minimum bid plus a small percentage. Further, as the percentage of "overbid" increases, the minimum time until eviction should either remain the same or increase.

For example, Figure 4 shows the relationship between bid price and guaranteed duration for the *c4.large* instance type in the *us-east-1* region on October 30, 2015. Each duration along the x-axis corresponds to a bid (on the y-axis) which, when submitted, assures that duration with a probability of at least 0.95. To generate this graph, DrAFTS computes the durations associated with overbids up to 400% above the DrAFTS minimum in increments of 5%. We choose 400% because AWS limits the maximum bid that will be accepted to be less than or equal to 400% of the current Spot price.

From the graph, it is possible to determine what maximum bid to offer that both ensures a specific duration and limits the potential expenditure to the minimum achievable by the prediction methodology. For example, a user wishing to guarantee at least 4 hours (14400 seconds) before her *c4.large* instance would be eligible for termination (with probability at least 0.95) would submit a

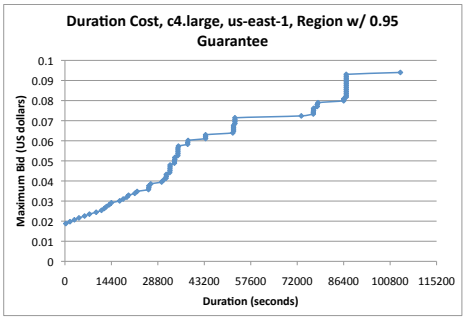

Figure 4: *DrAFTS minimum bid price versus guaranteed duration with probability 0.95 or better for the c4.large instance type in us-east-1 on October 30, 2015*

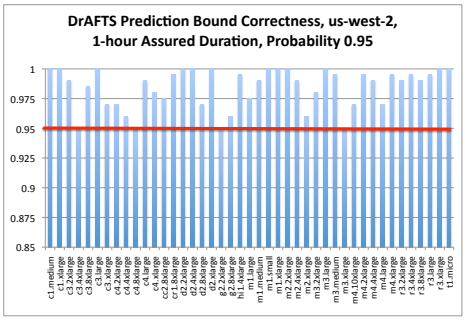

Figure 5: *Correctness of DrAFTS minimum bid price for duration of* 1 *hour with probability 0.95 or better us-west-2 for two weeks ending December 2, 2015*

maximum bid of $0.025 to the Spot tier. Notice that this bid determines the maximum she will pay for each of the 4 hours of occupancy. Thus, using DrAFTS, a user can know what bid is sufficient to assure a desired duration of execution and, in so doing, determine the maximum cost that should be paid to achieve this duration. At the same time, this cost is the minimum achievable according to the underlying price dynamics detected by DrAFTS.

We do not contend that the DrAFTS prediction is "the" minimum bid that is possible. That is, another prediction methodology might be able to provide a lower potential cost while also making the same statistical guarantee of duration, but as of this writing, we are not aware of an alternative approach that is more effective.

In Figure 5 we show the correctness results obtained by backtesting the bid calculation for a 1 hour assured duration and probability at least 0.95 over *all* instance types in the *us-west-2* region for a two-week period ending on December 2, 2015. We show these results for a single region zone (as opposed to all regions as in Figure 1 and Figure 2) so that the correctness fraction for each instance type may be represented explicitly (instead of by row-major index). To generate these results, we captured the price history for each instance type *us-west-2* over a two week period. We then chose 200 non-overlapping intervals in each trace and computed the cost-duration graph (as in Figure 4) at the beginning of

each random interval. We then compared the price corresponding to a 1-hour assured duration to the maximum price that AWS reported during the interval. Figure 5 shows the fraction of correct predictions made. Each fraction of 0.95 or greater corresponds to a correct prediction. That is, DrAFTS determined a correct bid to assure at least 1 hour of occupancy with probability at least 0.95 for every instance type in *us-west-2* over the test period. The results for *us-west-1* and *us-east-1* are similar (and omitted for the sake of brevity).

Summarizing, DrAFTS is able to determine what maximum bid to submit to ensure that a specific minimum duration interval is guaranteed by that bid with a specified probability. For a fixed success probability (0.95 in these experiments) it computes a minimum interval and bid price necessary to achieve that interval for each instance type in each region. It then generates a two-dimensional graph of durations and bid prices by incrementally calculating the durations associated with overbidding for each region-instance type pair. From these graphs, the minimum DrAFTS bid that should be used to ensure a specific duration with the given probability may be determined.

5 Related Work

In [11] the authors examine the question of using "live migration" to avoid downtime when a web service is hosted in the Spot tier. Using nested hypervisors, they describe a scheduler that can migrate a running web service between Spot instances and to do so without incurring an outage, their scheduler must predict when a Spot instance will be terminated in the future. Our work complements this approach in that it attempts to provide a way to determine the probability and duration until a termination may happen.

The work described [10] postulates the use of Paxos (a distributed consensus algorithm) to manage replicated application state Spot instances. It then attempts to solve a cost minimization that is based on a Markovian state model. The authors estimate transition probabilities directly from the Spot price histories.

Our work differs from this work in several ways. First, we focus exclusively on predicting the time until Spot instance termination as a function of the probability target provided to the DrAFTS algorithm. Using QBETS our technique also takes into account the effects of autocorrelation in each Spot price history. However, because it provides a bound on duration, it may be possible to use DrAFTS as a method of estimating the Spot instance failure probabilities that their methodology requires.

The authors of [21] describe a neural-network based approach to predicting prices in the Spot tier. Their approach (based on a mixture of Gaussians and a Box-Jenkins time series methodology) generates one-step ahead predictions (with a granularity of 1.3 hours) for the spot market that are quite accurate. However they point out that predicting the market for longer time frames should be encouraged as future research. DrAFTS constitutes such research in that it combines QBETS predictions of the bounds on price (for the next 5 minute interval) with a duration prediction essentially providing predictive bounds for arbitrary durations into the future. The length of the prediction interval is determined by the probability of the bounds being too high.

Finally, the authors of [1] investigate, at some length, the market dynamics associated with the AWS Spot tier. Their hypothesis is that pricing in the Spot tier is not driven solely by client demand (*i.e.* AWS introduces hidden externalities that affect pricing). We concur with the analysis presented in [1], motivating us to turn to QBETS as an efficient adaptive technique. Again, DrAFTS is only providing a statistical bound predicted price and, thus, need not recover the "true"' underlying market dynamic completely. The efficiency of QBETS combined with its non-parametric nature makes it possible to adapt to any introduced externalities "fast enough" to make on-line prediction possible.

6 Conclusions and Future Work

In this work, we present DrAFTS (**Dur**ability **A**greements from **T**ime **S**eries) as a methodology for predicting bounds on the prices to be bid in the Amazon AWS Spot tier and the durability of these bids. DrAFTS is adaptive, automatic, and non-parametric making it possible to be used without human intervention (*e.g.* as part of an on-line prediction service). We verify that DrAFTS is able to achieve a specified probabilistic bound on future survivability of instances in the Spot tier for a large set of instance-type-AWS region combinations. We also compare DrAFTS to a simple non-parametric approach (widely used colloquially) based on empirical observation of pricing data and find that it is more effective. Finally, we illustrate the relationship between bid pricing and the durability of DrAFTS statistical guarantees.

There are several ways in which we wish to improve upon the methodology. In particular, we focus on DrAFTS illustrations in this paper using regional Spot tier pricing data, but not at the AZ level because Amazon makes AZ pricing data user-specific. DrAFTS does work at the AZ level but the results are only meaningful to the user whose ID requests the pricing data. We wish to expand DrAFTS so that it is useful to users wishing to exploit its features for Spot instances launched in specific AZs.

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. Amazon ec2 spot instance pricing history, 2015. `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances-history.html` accessed August 2015.

[3] AMAZON WEB SERVICES. Amazon ec2 spot instances, 2015. `http://aws.amazon.com/ec2/purchasing-options/spot-instances/` accessed August 2015.

[4] AMAZON WEB SERVICES. Amazon simple storage service, 2015. `https://aws.amazon.com/s3` accessed September 2015.

[5] AMAZON WEB SERVICES. Amazon web services, 2015. `https://aws.amazon.com` accessed September 2015.

[6] AMAZON WEB SERVICES. Elastic compute cloud, 2015. `https://aws.amazon.com/ec2/` accessed September 2015.

[7] ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R., KONWINSKI, A., LEE, G., PATTERSON, D., RABKIN, A., STOICA, I., ET AL. A view of cloud computing. *Communications of the ACM 53*, 4 (2010), 50–58.

[8] BREVIK, J., NURMI, D., AND WOLSKI, R. Quantifying Machine Availability in Networked and Desktop Grid Systems. In *Proceedings of CCGrid04* (April 2004).

[9] GOOGLE COMPUTE PLATFORM. Google compute engine, 2015. `https://cloud.google.com/compute` accessed September 2015.

[10] GUO, W., CHEN, K., WU, Y., AND ZHENG, W. Bidding for highly available services with low price in spot instance market. In *Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing* (2015), ACM, pp. 191–202.

[11] 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* (2015), ACM.

[12] JAYATHILAKA, H., KRINTZ, C., AND WOLSKI, R. Response time service level agreements for cloud-hosted web applications. In *ACM Symposium on Cloud Computing (SoCC)* (2015).

[13] KRISHNAN, S., AND GONZALEZ, J. L. U. Google compute engine. In *Building Your Next Big Thing with Google Cloud Platform*. Springer, 2015, pp. 53–81.

[14] MILLER, F. P., VANDOME, A. F., AND MCBREWSTER, J. Amazon web services.

[15] MURTY, J. *Programming Amazon Web Services: S3, EC2, SQS, FPS, and SimpleDB*. O'Reilly Media, Inc., 2009.

[16] NURMI, D., BREVIK, J., AND WOLSKI, R. Modeling Machine Availability in Enterprise and Wide-area Distributed Computing Environments. In *Proceedings of Europar 2005* (2005).

[17] 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.

[18] NURMI, D., WOLSKI, R., AND BREVIK, J. Model-Based Checkpoint Scheduling for Volatile Resource Environments. In *Proceedings of Cluster 2005* (2004).

[19] 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.

[20] PUCHER, A., GUL, E., WOLSKI, R., AND KRINTZ, C. Using trustworthy simulation to engineer cloud schedulers. In *IEEE International Conference on Cloud Engineering (IC2E)* (2015).

[21] 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.

[22] WOLSKI, R., BREVIK, J., AND UCSB RACELAB. On-line DAFTS predictions for aws spot instances, 2015. `http://predictspotprice.cs.ucsb.edu` accessed September 2015.