Empirical Analysis and Offering of Various Historical Spot Instance Datasets

Sungjae Lee
Kookmin University
Seoul, South Korea
sungjae@kookmin.ac.kr

Kyungyong Lee
Kookmin University
Seoul, South Korea
leeky@kookmin.ac.kr

ABSTRACT
Public cloud service vendors provide a surplus of computing resources at a cheaper price as a spot instance. The first spot instance provider, Amazon Web Services (AWS), releases the history of spot price changes so that users can estimate the availability of spot instances, and it triggered lots of research work in literature. However, a change in spot pricing policy in 2017 rendered a large portion of spot instance availability analysis work obsolete. Instead, AWS publishes new spot instance datasets, the spot placement score, and the interruption frequency for the previous month, but the new datasets received far less attention than the spot price history dataset. Furthermore, the various spot datasets can provide contradicting information about spot instance availability of the same entity at the same time quite often. In this work, we develop a web-based spot instance data-archive service that provides historical information of various spot instance datasets. We describe heuristics how we overcome limitations imposed when querying multiple spot instance datasets. We conducted a real-world evaluation measuring spot instance fulfillment and interruption events to identify a credible spot instance dataset, especially when different datasets represent contradictory information. Based on the findings of the empirical analysis, we propose SpotScore, a new spot instance metric that provides a spot recommendation score based on the composition of spot price savings, spot placement score, and interruption frequency. The data-archive service with the SpotScore is now publicly available as a web service to speed up system research in the spot instance community and to improve spot instance usage with a higher level of availability and cost savings.

CCS CONCEPTS
• Computer systems organization → Cloud computing.

KEYWORDS
cloud computing, spot instance, datasets

1 INTRODUCTION
Motivation: Cloud computing has changed how we consume compute resources. One of the most significant changes is its on-demand billing model where users pay for what they have used. Furthermore, excess compute resources are provided at a much lower cost than on-demand pricing, and such instance is commonly referred to as a spot instance, which is provided by the majority of public cloud vendors. Despite the lower price of spot instances, resources may be forced to be shut down as demand changes, and users should be prepared for a sudden interruption of running instances. AWS, the leading cloud service provider, has been providing spot instance price history data since the service’s inception to help users anticipate price changes and the possibility of instance interruption. The spot price change history dataset triggered lots of research work [1, 12, 20, 36, 46] to optimize spot instance usage from various domains, such as big-data processing [48], deep-learning [27], and batch-processing [31, 42].

The spot price change history is not the only spot instance dataset. The spot placement score was released in 2021, and it reflects the likelihood of a spot request success. The spot instance advisor dataset, which was first released in 2015, provides the interruption ratio of a spot instance during the preceding month as well as the cost saving ratio over the on-demand instance. Although the two most recent spot datasets provide useful information when used with spot instances, they have not received as much attention as the spot price change datasets.

Limitation of state-of-art approaches: The spot price change history dataset is very popular in literature for research and system implementation when using spot instances. However, in 2017, AWS changed its spot instance operation policy to make the price change less volatile [7]. The change makes most of the previous spot price analysis work obsolete, and expecting the spot instance reliability using the spot price change history data becomes no longer available [5, 18]. Even in such cases, many spot instance-related works continue to rely solely on the spot price history data only, which is irrelevant to the spot instance availability, and the characteristics of more recent spot instance datasets, such as the spot placement score and interruption ratio, have yet to be thoroughly understood and evaluated.

Key insights and contributions: To help spot instance users to better anticipate resource availability, we thoroughly evaluate and analyze the characteristics of new spot instance datasets, the spot placement score, and interruption ratio from the spot instance advisor dataset. In our comprehensive analysis, it was uncovered that about at 25.53% of cases, the spot instance datasets represent contradicting information in the same moment, and they show low Pearson correlation coefficients [6] which might confuse spot instance users. We conduct real-world experiments to determine which datasets best represent real-world spot instance behavior by measuring the spot request fulfillment and interruption ratio. A spot instance showed noticeably higher availability only when the spot placement score and the interruption ratio indicated high availability. When either one of the datasets indicates low or medium availability, the interruption ratio has significantly increased over 3 times. Based on the experiment result, we propose a new spot instance metric, SpotScore, to help spot users decide spot instance types with superior availability and cost-savings on
a global scale. In the SpotScore calculation, we use the cost-saving ratio, instance fulfillment success probability, and the historical interruption ratio.

Experimental methodology and artifact availability: Other than the contracting information among various spot instance datasets, there exist multiple challenges during the spot instance dataset collection. Unlike the spot price dataset, the two new spot instance datasets do not include historical data. When creating a query, the spot placement score imposes numerous constraints. The interruption ratio dataset is only accessible through the management console and does not provide programmatic access. To expedite research in the related field and enhance reliable spot instance usage by circumventing similar challenges that we have encountered during the data collection process, we have built a spot data-archive service where a user can access the historical dataset of spot placement score, interruption frequency, cost savings, and SpotScore in a single place. The data service will continuously update the SpotScore value considering the correlation of real spot instance availability behavior with the publicly released spot datasets information. Through the web service, all the source codes to collect the datasets and run the experiments are available as well as the dataset itself.

Limitations of the proposed approach: The current implementation of SpotScore calculation reflects only the savings, spot placement score, and interruption ratio. To better prioritize spot instances for various application scenarios, the hardware specifications such as processing unit, generation, memory, network, and accelerator presence should be considered. The current spot dataset only includes information from AWS. Considering other major cloud vendors provide the spot instance service, we expect to provide similar data service from various providers.

In summary, the major contribution of this paper is as follows.

- Implementation of public spot instance datasets archive providing historical information
- Uncovering the real-world spot instance behavior concerning the publicly released spot instance datasets
- Providing a spot instance prioritizing score, SpotScore
- Thorough analysis of the spot placement score dataset for the first time in literature
- Sharing artifacts that solve many challenges encountered during data collection and experiments

### 2 TRANSIENT INSTANCES ON CLOUD

One of the most compelling aspects which result in cloud computing success is its elastic billing model. Before cloud computing, computing power should be purchased in the unit of hardware that is not flexible for varying computing resource demand. Cloud computing’s on-demand pricing mechanism allows users to pay for resources only when they are needed, removing the burden of large-scale hardware purchase. Aside from on-demand pricing, public cloud service vendors offer surplus computing resources at a much lower cost than on-demand instances, which are referred to as spot instances and include AWS Spot Instance, Microsoft Azure Spot Virtual Machines, and Google Cloud Spot VMs.

The spot cloud instance was first introduced by AWS in 2009. Since its inception, it adopts a market-driven auction through a uniform price and sealed bid mechanism. By the uniform price, all spot instance users pay for the same spot price regardless of a bid price, and users do not know the bidding price of other users (sealed bid) [1]. The service vendor determines the spot price based on supply and demand for cloud instances. The spot price varies depending on the instance type and availability zone. When an outbid event occurs or idle resources become scarce, spot instances can be forced to shut down.

The possible state of spot instance request is summarized in Table 1. Upon submission of a valid spot instance request, the request status becomes Pending Evaluation. In the status, if any of the constraints cannot be met, the request status becomes Pending. Possible reasons for the status include spot instance capacity not available in the requested available zone or bidding spot price too low. If all the constraints of a spot request are met, it becomes Fulfilled status. In the status, an instance is started with configurations specified in a spot request. It might take a few minutes for an instance to start. A spot request with a running spot instance can become the Terminal status for the following reasons: spot price out-bid or spot resource capacity not available, and they are generally referred to as spot instance interruption. A user can also terminate an instance voluntarily.

Owing to the uncertainty in the spot instance reliability, users should prepare a plan to deal with instance interruption. To help users better utilize spot instances, service vendors provide various information. For example, AWS provides spot instance price change history datasets during the past three months. Recently, the probability of interruption and placement recommendation scores are provided.

### 2.1 Spot Instance Pricing History

AWS provides spot instance price change history through its management console and Command Line Interface (CLI) library to allow programmatic access. Users can specify the start and end times of queries, as well as availability zones and instance types. The returned output includes the timestamp at which a spot price changes as well as the changed spot price at the time. When using spot instances, the history of spot price for the previous three months provides insightful information, and many studies have been conducted using the dataset. Statistical analysis of the spot price change helps users to better understand the spot instance market [1, 12, 20, 29, 36, 46, 47]. Using the spot price change history,

| Status          | Description                                                                 |
|-----------------|-----------------------------------------------------------------------------|
| Pending Evaluation | A valid spot request is submitted                                           |
| Holding         | Some request constraints cannot be met (price, location, resource availability, ...) |
| Fulfilled       | All the spot request constraints are met, and instance status being updated to running |
| Terminal        | A spot request is disabled possibly by price outbid, resource unavailability, user, ... |

Table 1: Possible spot instance request status and description

1https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-request-status.html
many researches were conducted policy to propose an optimal bidding algorithm [3, 15, 23, 28, 37, 40, 43, 49, 51].

Active and wide usage of spot price change data was feasible because the data update frequency is timely and users could estimate the likelihood of instance interruption and cost savings. However, in 2017, AWS changes its spot instance operation policy [7]. In the new policy, the spot price changes less frequently and becomes more stable [5, 18]. However, the new policy no longer represents the interruption event. Before the update, a user can guess the interruption event by comparing the bid and spot prices. Following the update, the spot price does not accurately reflect the surplus of idle compute resources, particularly when it is low. As a result, if the advertised spot price remains lower than the bid price, the instance interruption can still occur, rendering the results of previous related research work obsolete.

2.2 Spot Instance Advisor
The spot price announcement policy change in 2017 took away users’ capability to estimate the probability of spot instance interruption based on the different bid prices. The Spot Instance Advisor² service can mitigate the limitation to some extent. The advisor service provides the rate at which spot instances had been interrupted for the trailing month. The interruption frequency data is divided into five categories; less than 5%, between 5% and 10%, between 10% and 15%, between 15% and 20%, and more than 20% of interrupt ratio. The interrupt frequency data is provided per a combination of instance type and region. In contrast, to spot instance price data, which is provided based on a combination of instance type and availability zone, interrupt frequency data is provided in a coarse-grained manner. Aside from interrupt frequency, the spot instance advisor provides average cost savings over on-demand instances. The service is only officially accessible via the AWS management console and does not support CLI.

2.3 Spot Placement Score
The information provided by the instance advisor reflects the spot instance reliability during the trailing month, and it might not reflect the current spot resource availability timely which can directly impact the success of a spot instance request. To provide timely spot instance availability, AWS started to provide Spot Placement Score service [19] recently. The placement score’s primary goal is to assist users in estimating the likelihood of a successful spot request before launching an instance type in a specific availability zone. The internal details of how the metric is calculated are not publicly available. Externally, it takes as arguments the desired instance types, regions, and target capacity and returns a placement score ranging from 1 to 10. The higher the score, the more likely the success of the spot request. Data on placement scores can be accessed via the AWS management console and CLI.

3 SPOT INSTANCE DATASETS COLLECTION
Different from the spot instance price history dataset, the spot placement and advisor datasets impose few challenges in the data collection. This section discusses the details of the challenges and how we overcome them.

²https://aws.amazon.com/ec2/spot-instance-advisor/

3.1 Challenges in Data Collection
The spot price change history data was an effective source of information when optimizing spot instance usage before the policy change in 2017 because it provided timely and detailed information through a convenient data access interface. The other two recent spot instance data sources presented in Sections 2.2 and 2.3 have few shortcomings in the data itself and access medium with query constraints.

No historical data: For both spot instance advisor and spot placement score datasets, users can query only the current value, and no historical information is provided. Meanwhile, spot price history data is provided for up to three prior months, and long-term data history was a valuable source for optimizing spot instance usage. Thus, to expand the data usage of the spot instance advisor and placement score, historical data should be provided.

Query limitation: The spot placement score dataset imposes various limitations in the query, and they are described in its official document page [38]. The most significant limitation is the number of unique queries allowed in 24 hours. This can be a significant limitation, especially when a user wants to investigate the availability of multiple instance types across multiple regions. According to our empirical analysis, an account can issue a maximum of 50 unique queries in 24 H. The uniqueness of a query is determined by the combination of regions, instance types, and the number of desired instances. An aggregated placement score value for the specified instance types in a query is returned for each region in the argument for each unique query. Surprisingly, issuing the same query multiple times within a 24 H time frame does not count against the query limit. Thus, an account can issue 50 unique queries multiple times, such as every hour, in a day to record how the score values change.

The spot placement score differs for distinct availability zones even if they are located in the same region. In a query, a user can specify an option of SingleAvailabilityZone as true to get score values for each availability zone in a region specified in the query. Another query regulation is imposed for the number of returned placement scores which is limited to 10. For example, if a query specifies multiple regions with the SingleAvailabilityZone option as true, there can be more than 10 placement scores for different availability zones. In such a case, only 10 placement scores for availability zones with larger values are returned.

At the time of the writing there are about 422 instance types, 17 regions, and 55 availability zones in AWS. To scan the spot placement scores for all possible instance types and regions combination, 422 × 17 = 7,174 queries should be executed at most. Given the limit of 50 unique queries per account, it is impossible to obtain scores for all instance type-region combinations. Users can optimize a query by specifying multiple regions in a query, but the number of placement score results returned from a query is still limited to 10. A large number of possible spot placement scores query dimensions regarding instance types, regions, availability zones, and the number of instances necessitates the use of a specialized data service to provide timely information to help users better utilize spot instances.

Limited query interface: Among many ways to use and operate cloud resources, the programmatic access is preferred over a
management console that uses a graphical user interface [32]. From the context, spot price history and spot placement score data are provided through CLI which allows programmatic management. However, the spot instance advisor data is natively supported only from the management console and hurts operability [24] without programmatic access.

### 3.2 Data Collection Methodology

To increase efficiency when using spot instances, it is important to broaden target instance types if a workload does not have a specific hardware requirement. Furthermore, if a workload does not have a specific geographical requirement, the possibility for cost saving can be further improved [12, 27], and building a compute cluster with heterogeneous spot instances is proven to be an efficient solution for data-parallel analysis tasks [44]. Timely and detailed spot instance status information is required to build a cost-optimal cloud environment while broadening candidate spot resources. However, the limited number of queries imposed for the spot placement score datasets prevents information from being gathered for various resources located around the world.

The major query limitation of the spot placement score is the number of unique queries per account, which is 50, and a query response can obtain a maximum of 10 spot placement scores of an instance type and availability zone combination. A user can specify the instance type, regions, and the SingleAvailabilityZone parameter to get spot placement scores for each availability zone in the region. To make a query optimally operate which can return the maximum 10 responses, we must organize an instance type with multiple regions, each of which may have a different number of availability zones. We must also consider the fact that not all availability zones in a region support a specific instance type. To run spot placement queries as efficiently as possible, we create a nested dictionary whose key is an instance type and the corresponding value is another dictionary whose key is the region and the value is the number of availability zones that support the instance type in the outer dictionary’s key. The problem can be simplified to a bin-packing algorithm [26] which collocates regions together while making the sum of the number of availability zones of each region as the maximum number of returned results. After the problem abstraction, we use Google OR-Tools [34] to solve the bin-packing problem. Among possible libraries, we use Coin-or Branch and Cut (CBC) [13] implementation which is a mixed-integer programming solver. After packing multiple regions with a single instance type to be requested in a single spot placement score query, we could decrease the total number of required queries from 7,796 to 1,796 which is about 4x improvement.

Figure 1 explains the spot placement query optimization example with a sample instance type of `p3.2xlarge`. The regions and number of availability zones that support the instance type are shown on the left side of the figure. Following the bin-packing execution, multiple regions are grouped into a single query to reduce the total number of queries.

### 4 ANALYSIS

In literature, much prior works analyzed the spot price history data thoroughly. However, to the best of the authors’ knowledge, no prior work exists concerning analyzing spot instance advisor and spot placement score datasets. Due to the spot price announcement policy change in 2017 [5], the effectiveness of using the spot price history data faded, and the analysis result of much prior work becomes invalid. In the situation, we are confident that through analysis of other available spot instance information is critical to better utilize spot resources.

For analysis, we gathered the spot instance advisor and spot placement score dataset from November 23. 2021 to January 21. 2022, which is 60 days in total. The spot instance advisor dataset contains interruption frequency data, spot instance price, and savings over on-demand instance information. The spot advisor dataset does not support programmatic access, and we used SpotInfo [25] tool to collect the dataset automatically and used the AWS CLI tool to scan the spot placement score. When using SpotInfo, we set the --region all option to query the entire advisor information using a single command. To gather the spot placement score dataset, we used multiple accounts under the same organization. We applied the CBC algorithm in the Google OR-Tools to allocate instance types and regions to each account. Both datasets were queried every 10 minutes to detect data changes in a fine-grained manner.

The interruption ratio which is included in the spot instance advisor dataset is provided as a categorical value from the least frequency of less than 5% to the most frequency of more than 20%. To improve our analysis, we convert the categorical value to a score value by matching the range with that of the spot placement score, which is between 1.0 and 3.0. We set the least interruption frequency to 3.0 and the most interruption frequency to 1.0 in the score representation. There are three more categorical values in between, and we assign them 2.5, 2.0, and 1.5, ranging from less to more interruption frequency. The converted interruption frequency information is referred to as the interruption-free score. A higher interruption-free score, like a higher spot placement score, indicates better spot instance availability.

#### 4.1 Spot Placement and Interruption-free Score

Figure 2 presents temporal value changes of spot placement score (Figure 2a) and interruption-free score (Figure 2b) using a heatmap.
format with a grayscale color. The brighter color expresses higher spot placement and interruption-free scores which implies higher spot instance availability. The horizontal axis shows the elapsed days since the data collection start date. The vertical axis represents instance classes offered by AWS. They are shown in the order of general instance family (T, M, A), compute-optimized instance family (C), memory-optimized instance family (R, X, Z), accelerated-computing instance family (P, G, DL, Inf, F, VT), followed by storage-optimized instance family (I, D, H). In each instance class, daily average score values are calculated.

From the figures, we can discover that the spot placement score shows a much lighter color (higher score) than the interruption-free score. Overall, the average spot place score across all the instance types is 2.77, and that of the interruption-free score is 2.18. Among many instance types, the accelerated-computing instance family has the lowest scores both for the spot placement and interruption-free, which is 12.79% and 35.47% lower than the average scores, respectively. We can infer such characteristics stems from the recent popularity of the deep-learning where specialized hardware is widely in-use both for training and inference tasks [17, 21]. Among the accelerated-computing instance types, the DL instance shows high scores for both spot placement and interruption-free. The instance type provides a Gaudi processor which is special-purpose hardware for DNN training and inference developed by Habana Labs [30]. It is released recently, and we expect the ecosystem for DNN development using the instance type is not mature yet which may result in the low usage so far. For instance types with GPU devices, G instance class shows higher scores than P instance types. The G instance type equips NVIDIA T4 GPU (G4dni) or AMD V520 (G4ad), and the P instance type equips NVIDIA Tesla V100 GPU (P3) or Tesla A100 GPU (P4). Comparing two instance types, the P instance type shows better performance for DNN tasks [8], but the G instance is more affordable with the lower hourly price. We expect that better affordability and the cost-performance ratio of G instance types result in higher resource capacity and more chances for surplus resources. Concerning the temporal score changes, both spot placement and interruption-free scores do not show significant score variations as date changes in the horizontal axis.

To observe spatial characteristics of the spot placement and interruption-free scores for different instance classes, Figure 3 presents the scores grouped by different regions. To cover a wide range of instance types, we chose 17 regions with the greatest number of instance types supported, which are displayed on the horizontal axis. A region code is expressed in the continent-coordinate-id combination. The color-scale and the instance classes presented in the vertical axis are the same as Figure 2. For instance types that are not supported in a specific region, we mark NA. From the figures, we can visually observe a higher degree of score variations across different regions which coincides with the conclusions from the previous work [12]. Among accelerated-computing instance classes, the general-purpose GPU devices (G and P) show relatively lower scores for most regions. Thus, if users build a DNN environment for a specific purpose either training or inference, one can use a more reliable spot instance environment by using special-purpose instance types located globally [27], such as DL for training and Inf for inference tasks.

To understand the score distribution of spot placement interruption-free, Table 2 shows the percentage of values observed during the measurement period. According to the table, the majority of spot placement scores are 3.0, and only 9.82% of spot placement scores being 1.0, indicating a high likelihood of the spot request being successful. The interruption-free score, as opposed to the spot placement score, exhibits a more uniform distribution across distinct values. About 30.65% of cases shows less than 5% interruption frequency (score value : 3.0), but 22.01% of cases shows more than 20% of interruption ratio (score value : 1.0). From the scores value distribution, we can identify the discrepancy of two spot datasets roughly.

Figure 4 shows the spot placement score and interruption-free scores for different instance sizes which is expressed in the horizontal axis. The solid line represents the spot placement score, and the
multiple instance types. We discovered that for instance sizes with a low number of instance types, the average score value is determined by only a few instance types and does not adequately reflect the impact of size. The number of instance types is expressed using a star marker whose value is shown in the secondary vertical axis. From Figure 4, we can observe that as the instance size increases, both spot placement and interruption-free score decrease which coincides with what Kadupitige et al. [22] discovered. Larger instance sizes necessitate more compute resources and are more likely to cause resource fragmentation without instance migration [11]. Thus, it might have less flexibility than a smaller instance size, and it can result in a low possibility for surplus resources and lower availability scores.

4.2 Spot Placement Score With Diverse Parameters

Different from other spot datasets, the spot placement score dataset supports various query parameters. Users can specify multiple instance types to query a composite spot placement score. Furthermore, a spot placement score of a large number of spot instances can be queried. Despite the spot placement score’s rich query capability, the limited number of unique queries prevents users from making diverse queries. As a result, it is advantageous to be able to estimate the query results of composite query parameters from a single query parameter result.

To understand the characteristics of composite instance type queries, we compare the placement score of a query that specifies multiple instance types and scores of multiple queries, each specifying a single instance type. The goal of this analysis is to determine how a single instance spot placement score affects the score when the instance types are queried together. According to the official spot placement score document, the score value ranges from one to ten. However, the maximum returned score in our experimental queries that specified only a single instance type was three. From the observation, we conjectured that the maximum spot placement score, 10, would be returned when a query specifies multiple instance types, and the placement score of multiple instance types might be the sum of the individual instance’s placement score.

To check the validity of the hypothesis, we issued multiple queries which specify arbitrary three instance types. Before and after issuing a query with multiple instance types, we issue three individual queries specifying each instance only whose aim is to confirm that the spot placement score has not been changed in the meantime. Figure 5 presents the returned placement score of a query with multiple instance types in the horizontal axis. The vertical axis presents the summed spot placement scores when a query is made separately for each instance type. To uniformly distribute the sum of individual spot placement scores, we choose the same number of instance type and availability zone combinations in each summed score value, which ranges from three (all instance score is one) to nine (all instance score is three). The figure is presented in scatter plot format, with the radius of a circle representing the frequency of occurrences in each coordinate.

In the figure, we add a line of \( y = x \) with a slope of one to indicate a case where the returned spot placement score of multiple instance types is the same as the sum of the individual spot placement score of each instance type. In the experiments, about 38.81% of cases belong to this case. The circles lower-right to the \( y = x \) line indicate cases where the spot placement score of multiple instance types
are larger than the sum of individual scores, and about 60.62% of cases belong to this case. Based on the results of this experiment, we can conclude that the sum of the individual instance type scores can be the smallest of the composite instance type spot requests. We observed two cases in the experiments where the composite instance type score was less than the sum of the individual scores, which we consider to be exception cases.

In a spot placement score query, users can specify the number of spot instances to request, and Figure 6 presents how the spot placement score changes when a large number of instances are requested. The number of instances in a query is shown in the horizontal axis, and the instance classes are shown in the vertical axis. In the experiments, we select a few representative instance types in each instance family. To control the total number of queries, we query only the xlarge size where applicable. For instance, for types that do not have the size, P3 and P4, we use the smallest possible size. The returned spot placement scores are expressed using a heatmap where the higher spot placement score is expressed using a lighter color, and the lower score is indicated with a darker color in the grayscale.

Intuitively, specifying more instances in a spot request lowers the chances for the fulfillment, and we can discover such a pattern in the figure. The ratio of spot placement score decrease is quite different across distinct instance types. For example, instance types in the accelerated-computing family, P, G and Inf, shows significant score drops when a large number of resources are requested. The D instance type which belongs to the storage-optimized family also presents a noticeable score drop. Such instance types are armed with specialized hardware internally in a host, such as GPU devices in P and G, AWS Inferentia chips in Inf, and large size local storage disks in D instances, and the supply of such resources might not be as much as other general-purpose instance types.

4.3 Correlations Among Multiple Spot Datasets

So far, we have analyzed the spot placement score and interruption-free score independently. The two datasets are released in real-time, and the spot instance price dataset is also available at the same time. From the spot users’ perspective, three spot data sources exist, and it can be challenging to decide which dataset provides the most accurate information to infer the spot instance availability especially when distinct datasets imply contrasting information. For example, if the spot placement score is high (3.0), and the interruption-free score is low (1.0, which indicates more than 20% of interruption ratio), users cannot conclude whether the target instance type in a region is going to be available shortly after. To understand the correlation among spot instance price, spot placement score, and the interruption-free score, Figure 7 shows the distribution of the Pearson correlation coefficient [6] of the two datasets combination. The Pearson correlation coefficient of two variables, X and Y, is calculated as follows.
In the analysis, two variables are selected among the spot placement score, interruption-free score, and spot price. $R_{XY}$ is calculated for each instance type and availability zone combination. For each combination, the corresponding value at time $t$ and mean value across the entire measurement period, $T$, is used to calculate the correlation coefficient. The range of correlation coefficient values is between 1.0 and $-1.0$. A value close to 1.0 indicates a strong correlation between two variables and can express variable dependency. For instance, if the coefficient of spot placement score and interruption frequency is close to 1.0, we can assume that the two variables contain similar information. Meanwhile, a value close to $-1.0$ indicates a strong inverse correlation. The coefficient value close to 0 indicates that two variables have no correlation and are more likely to be independent. Intuitively, the spot placement score and the interruption-free score should have a strong correlation because the higher spot placement score implies higher likelihood of spot request fulfillment and the higher interruption-free score also implies a lower probability for spot interruption. Meanwhile, the two values are expected to have a negative strong correlation with the spot price because the higher spot price can be an indication of spot instance shortage.

Figure 7 shows the Cumulative Distribution Function (CDF) of Pearson correlation coefficient of any two variables combinations. The horizontal axis presents the correlation coefficient values, and the vertical axis expresses the distribution. The solid line depicts the correlation coefficient between the spot placement score and the interruption-free score, the dashed line depicts the correlation coefficient between the interruption-free score and the spot price, and the dotted line depicts the correlation coefficient between the spot placement score and the spot price. Using a time-series dataset collected every 10 minutes during the $T$ data collection period, we calculated the correlation coefficient for each instance type and availability zone combination. As shown in the figures, most correlation coefficient values are located near 0.0 which implies that any two spot datasets combination has neither strongly positive or negative correlations. In the distribution, it is noticeable that the correlation coefficients which include the spot price have a much higher density around 0.0. It implies that the spot price dataset might have little information regarding the spot instance availability compared to the other two publicly announced spot instance datasets. This observation confirms what Irwin et al. discovered [18] after the spot instance operation policy change in 2017. In addition, the correlation coefficient between the spot placement score and the interruption-free score is also very low. For 61.84%, the absolute coefficient value is lower than 0.25, and 86.05% of cases have lower than 0.5 correlation coefficients.

The discrepancy among spot instance datasets can confuse users when the publicly announced datasets present contradicting information, such as the high spot placement score (3.0) with the low interruption-free score (1.0). To detect how many portions of the spot placement score and interruption-free score differs, we count the difference of two scores at any given time and show the difference as a histogram in Figure 7. The horizontal axis shows the absolute score difference between the two datasets. The maximum and minimum of the scores are 3.0 and 1.0, respectively, and the step of the interruption-free score is 0.5. Thus, the maximum difference value is 2.0, such as the spot placement score being 3.0 and the interruption-free score being 1.0, which is a complete contradiction, and the minimum difference is 0.0 which means two score values are the same. The vertical axis displays the percentile unit ratio of each score difference. As illustrated in the figure, the difference of 0.0 accounts for the vast majority of cases. There are, however, a large number of cases with contradictory information. For example, for about 18.4% of cases, the spot placement and interruption-free score present the opposite meaning. Considering the difference of 1.5 is not a negligible difference, about 25% of cases, spot users might be confused about which datasets to follow to optimally use spot instances.

To understand spot dataset change pattern, Figure 9 presents CDFs of how often each dataset is updated (Figure 9a) and how much value changes in each update (Figure 9b). In the figures, the solid line represents the spot placement score, the dashed line represents the interruption-free score, and the dotted line presents the spot price. The horizontal axis of Figure 9a expresses the elapsed time (hours) between update events in a log scale. The lower the value on the horizontal axis, the more frequently the variable is updated. The spot placement score is updated the most frequently, as shown in the graph, while the interruption-free score is updated the least frequently. The interruption-free score’s low-value change frequency is consistent with its score calculation policy, which uses the interruption ratio observed over the previous month. The frequent update of the spot placement score can be an indication of timely information that can reflect the success of the spot instance request well.

Figure 9b shows the distribution of amount of value changes between update events. For the spot placement and interruption-free scores, the amount of value changes are expressed in the primary vertical axis. For the spot price, the changes of the savings over the on-demand instance in the percentile unit are expressed in the secondary vertical axis. The horizontal axis expresses distributions. The majority of the spot placement and interruption-free scores change with a minimum change value, 96.71% and 87.14%, respectively. However, in some cases, the value changes dramatically where the 3.29% of spot placement score and 1.24% of interruption-free score changes by 2.0. For the spot price, we can see that the
Thorough analysis of various spot instance datasets revealed that by measuring spot instance fulfillment and interruption ratio of different spot datasets represent real-world spot instance behavior. Lack a concrete guideline for which datasets to use to meet application information can be perplexing for spot instance users who do not show strong correlations and do not agree with each other even in the same time window quite often. Such contradictory information can be perplexing for spot instance users who lack a concrete guideline for which datasets to use to meet application demand. We conduct experiments to determine how well different spot datasets represent real-world spot instance behavior by measuring spot instance fulfillment and interruption ratio of various instance types. In the experiments, we hope to learn how different spot placement and interruption-free scores affect spot instance availability. We decide to omit the impact from the spot price because it is known to be no longer a valid indication of spot instance availability [5, 18, 35].

In the experiments, we categorize the spot placement score and interruption-free score to High, Medium, and Low whose value is 3.0, 2.0, and 1.0, respectively. Then, we sampled instance type and availability zone which belongs to one of the H-H, H-L, L-H, M-M, and L-L combinations where the first character indicates the spot placement score, and the second character indicates the interruption-free score. The number of available instances in each combination differs, and we performed stratified under-sampling with the lowest number of available cases which is L-H combination.

With the stratified sampling [33], we tried to distribute the instance type and availability zone uniformly across all the candidates. If we use purely random sampling, the test cases are biased to specific regions and more popular instance types, which we attempted to avoid to make the experiment result representative globally. In total, we choose a 50 instance type and availability zone for each categorized spot placement and interruption-free score combination, for a total of 250 cases. Smaller and less expensive instance types are preferred where applicable to keep the experimental cost within our budget. For all the experiment cases, we issue a single spot instance request after setting the bid price the same as the on-demand price [40] and record the request status every five seconds.

In a spot instance request, we specify the persistent parameter so that an interrupted spot instance is requested again soon after an interruption event. Each experiment scenario is conducted for 24 hours.

Table 3 presents the rate of not-fulfilled and interrupted cases. For the Not-Fulfilled, we count cases which did not become fulfilled at all in the 24 hours during the experiment. For the Interrupted, we count cases which become interrupted at least once during the experiment. It is noticeable that when the spot placement score is high, 3.0, all the requests are fulfilled in the experiments. When both the spot placement and the interruption-free score are high, 12% of cases are interrupted at least once. When either the spot placement score or the interruption-free score becomes medium, 2.0, or low, 1.0, the interruption ratio skyrockets to 42% at most. It is also worth noting that a low spot placement score is a strong indicator of fulfillment failure.

Overall, the success of fulfillment can be solely assumed by the spot placement score which should reflect the most up-to-date resource availability information. This concurs with the score update frequency presented in Figure 9a which represented the timeliness of the spot placement score. When conjecturing the interruption probability, it is more appropriate to consider both spot placement and interruption-free scores where both values should score high.

To analyze how different spot placement and interruption-free scores impact the behavior of fulfillment and interruption, Figure 10 presents the elapsed time from a spot request submission until it is fulfilled (Figure 10a) and the elapsed time from the fulfillment until interruption event happen (Figure 10b). Both figures are represented in a CDF format whose distribution is expressed in the vertical axis. The horizontal axis shows the elapsed time in seconds using the log scale. In each figure, we present distinct distributions after categorizing the spot placement and interruption-free scores into

![Graph](image-url)

**Figure 9:** The distribution of the frequency and amount of the value changes for the spot placement score, interruption-free score, and the savings over on-demand price.

savings over on-demand instance barely changes at once. For less than 98% of cases, the amount of savings over on-demand change is less than 10%, and this observation concurs with the recent spot price analysis result after 2017 [5].

Overall, the fast update frequency of the spot placement score can provide timely information even if the amount of value change is small. The spot price changes frequently, but the amount of savings changes is relatively small, and it is best not to focus too much on the small price change. The interruption-free score expresses the trailing month’s average interruption ratio, and the value does not change frequently.

### 4.4 Fulfilment and Interruption Behavior

Thorough analysis of various spot instance datasets revealed that they do not show strong correlations and do not agree with each other even in the same time window quite often. Such contradictory information can be perplexing for spot instance users who lack a concrete guideline for which datasets to use to meet application demand. We conduct experiments to determine how well different spot datasets represent real-world spot instance behavior by measuring spot instance fulfillment and interruption ratio of various instance types. In the experiments, we hope to learn how

![Graph](image-url)

**Figure 9:** The distribution of the frequency and amount of the value changes for the spot placement score, interruption-free score, and the savings over on-demand price.
| Category | Not-Fulfilled | Interrupted |
|----------|-------------|------------|
| H-H      | 0%          | 12%        |
| H-L      | 0%          | 38%        |
| M-M      | 26%         | 38%        |
| L-H      | 56%         | 34%        |
| L-L      | 48%         | 42%        |

Table 3: The percentage of not-fulfilled and interrupted spot requests for different dataset category. The category is expressed in High, Medium, and Low, where the first character expresses the spot placement score and the other expresses interruption-free score.

High, Medium, and Low. The latency until spot requests are fulfilled which is shown in Figure 10a presents that when both scores are high, the latency until fulfilled is the shortest. About 28.07% of requests are fulfilled within one second, and over 90% of requests are fulfilled within 127 seconds. Intuitively, when both scores are low, it took the longest to fulfill spot requests with a median value of 1322 seconds. When two scores are contradictory, we can see that the higher the spot placement score, the faster the spot instance fulfillment time. We also specified how many fulfillment events occur in each score category in the figure legend. When a spot instance is interrupted and fulfilled again before the experiment duration of 24 hours expires, the number of fulfillments can be greater than the number of total experiments, which is 50. We can also see from the fulfillment event number that the higher the spot placement score, the more likely it is to be fulfilled again even if there is an interruption.

Figure 10b shows the time until a fulfilled spot instance becomes interrupted, and the longer value indicates higher availability. Similar to the latency till fulfillment, the most availability is observed when both scores are high, and the least availability happens with both scores being low. The interruption ratio which is presented in Table 3 shows a similar value when the spot placement and interruption-free scores show distinct value. However, spot instance running time shows noticeable differences where the median running time when the spot placement score is high (High-Low case) is 8994 seconds, while that of when the interruption-free score is high (Low-High) is 2859 seconds.

Overall, we can observe that the spot placement and interruption-free scores faithfully reflect the spot instance behavior especially when both scores indicate in the same direction. When two datasets contain contradictory information, the higher spot placement score should take precedence over the interruption-free score. We can deduce the reason from the inherent characteristics of how two scores are computed. The interruption-free score reflects the interruption ratio over the previous month, and the data may be out of date. Though the mechanism for calculating spot placement scores is not yet public, it should reflect timely spot resource availability as the metric indicates the likelihood of spot request success.

Figure 10: The CDF of latency distribution categorized by the spot placement score and interruption-free score

5 IMPLEMENTATION OF DATA-ARCHIVE SERVICE AND SPOTSCORE

The real-world spot instance experiment result presented in Section 4.4 reveals that combining the spot placement and interruption-free score can provide a quite accurate estimation of spot instance availability. However, the limited number of unique queries of the spot placement score becomes a major bottleneck when building an optimal spot instance resource pool. Furthermore, the absence of historical datasets might slow down new study outcomes from the cloud system research community. To overcome the shortcomings, we implement a data-archive web-service which provides historical information of the spot placement score, interruption frequency, and cost savings over on-demand instances from the spot instance advisor dataset. The architecture of the implemented web service is shown in Figure 11. To relieve resource management burdens of the application servers for infrequent dataset collection tasks, we adopted a serverless architecture. To periodically invoke data collection tasks, we use Amazon EventBridge, which is triggered every 10 minutes to invoke the AWS Lambda service, which is in charge of data collection tasks. The spot instance advisor dataset can be queried with a single Lambda function, whereas the spot
The SpotScore is calculated by multiplying the saving ratio of spot instances over the on-demand instance to the sum of weighted spot instance scores. To decide the weight of different spot instance scores, we introduce a variable, \( \alpha \), which a user can set in querying the SpotScore. The spot placement score (SPS) is multiplied by \( \alpha \), and the interruption-free (IF) score is multiplied by \( (1 - \alpha) \). We set the \( \alpha \) range to be between 0.5 and 1.0. The real-world spot instance experiment results show that the spot placement score accurately represents the fulfillment success probability. We anticipate that spot instance usage with a short duration will be more influenced by the high fulfillment ratio, with \( \alpha \) close to 1.0 being ideal. If either one of the spot placement score or the interruption-free score is low, the likelihood of a spot instance interruption increases dramatically. We expect that spot requests with a long-running time need the \( \alpha \) close to 0.5 to reflect the impact when the value of either dataset becomes low. We fix the minimum value of \( \alpha \) to 0.5 because the spot placement score expresses more timely information than the interruption-free score. In our extensive empirical analysis, setting the \( \alpha \) around at 0.7 results in a reasonable ranking result as which we set the default value. The SpotScore value is calculated in the Timestream database using a user-defined function.

6 RELATED WORK

The spot instance price change history dataset has been widely used for various purposes. Uncovering statistical characteristics of the spot price change [1, 12, 20, 29, 36, 46, 47] allows spot instance users to estimate the resource availability and cost savings when using spot instances. However, the spot instance operation policy change in 2017 [7] made the previous analysis work obsolete and the price change data itself does not provide as rich information as it was before [5, 18]. Irwin et. al. [18] thoroughly discussed the pros and cons of the spot price policy change while suggesting the direction for the spot instance evolution. New spot instance datasets have been released since the policy change, but they have received little attention. To the best of the authors’ knowledge, this is the first study that empirically examines spot instance availability using a composite of multiple spot instance datasets.

By referencing availability information from spot instance datasets, different types of applications can prepare a plan to react to spot instance interruptions. Son et. al. proposed DeepSpotCloud [27] to run DNN training tasks using GPU spot instances located globally. The big-data analysis tasks are generally conducted with Hadoop [10, 14], and SeeSpotRun [9] proposed running Hadoop MapReduce tasks using spot instances. Flint [39] and Tr-Spark [48] proposed a system to run a distributed big-data processing engine, Apache Spark [50], using spot instances. Using spot instances, online web services [2, 16], batch-processing jobs [31, 42], and parallel processing of independent tasks [45] while mitigating straggler effect due to transient servers [4] are proposed. ExoSphere [41] proposed a portfolio modeling for applications with a different level of interruption-tolerance and cost reduction expectation. The findings discovered in this paper are complementary to the aforementioned work which relies on spot instances because the spot placement availability with the proposed SpotScore information can help to improve the accuracy of the spot instance availability prediction which is no longer available with the spot price dataset.

Kadupitige et. al. [22] proposed a statistical model to represent the constrained spot instance preemption through empirical experiments using Google Cloud Preemptible VMs. The Google cloud does not provide transient instance resource availability information as AWS does, and Kadupitige et. al. tried to build a statistical interruption model. Meanwhile, AWS keeps opening new spot instance datasets, and we attempt to analyze the characteristics and verify how the dataset resembles real-world spot instance behavior. Pham et. al. [35] empirically analyze the spot instance availability with the interruption frequency dataset provided by AWS. As presented...
in Section 4.4, the spot instance fulfillment and interruption probability can be better modeled when combining spot placement score and interruption frequency dataset where the SpotScore is expected to be a better representation of spot instance availability.

7 CONCLUSION AND FUTURE WORK

The spot instance provides a way to use cloud instances at a cheaper price, and the spot price history dataset provided by AWS was a valuable source of information to predict spot instance availability. However, with the spot price change in 2017, most prior work which relied on the spot price datasets become obsolete. To provide timely spot instance availability from publicly released spot instance datasets, we built a data-archive service that collects the spot placement and interruption ratio datasets that are challenging to assemble due to various data access limitations. After the data collection, we performed an empirical analysis of the spot instance behavior to analyze how different spot instances represent the instance availability.

The publicly available spot instance datasets provide only the current information, and we collect the datasets and provide historical information as a web service to expedite research for the cloud system community. In addition, to help users decide on optimal spot instances on a global scale, we provide a new metric, SpotScore, which can prioritize different spot instances with distinct geographical locations.

The data-archive service is currently under active development. The current SpotScore metric does not take into account hardware specifications. Different applications, such as DNN training using accelerator devices, may necessitate specialized hardware or performance requirements, and the SpotScore metric with hardware information can provide more useful information. We currently only provide AWS spot instance datasets; however, adding spot datasets from other major cloud vendors, such as Microsoft and Google, would benefit a wide range of cloud users.

REFERENCES

[1] Orna Agmon Ben-Yehuda, Muli Ben-Yehuda, Assaf Schuster, and Dan Tsafrir. 2013. Deconstructing Amazon EC2 Spot Instance Pricing. ACM Trans. Econ. Comput. 1, 3, Article 16 (sep 2013), 20 pages. https://doi.org/10.1145/2350943.2394416

[2] Ahmed Ali-Eldin, Jonathan Westin, Bin Wang, Prateek Sharma, and Prashant Shenoy. 2019. SpotWeb: Running Latency-Sensitive Distributed Web Services on Transient Cloud Servers. In Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing (Phoenx, AZ, USA) (HPDC ’19). Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3307681.3325397

[3] Sarah Alkharif, Kyungyong Lee, and Hyoeok Kim. 2018. Time-Series Analysis for Price Prediction of Opportunistic Cloud Computing Resources. In Proceedings of the 7th International Conference on Emerging Databases, Wooseky Lee, Wonik Choi, Sungwong Jung, and Min Song (Eds.). Springer Singapore, Singapore, 221–229.

[4] Pradeep Ambati, David Irwin, Prashant Shenoy, Linou Gao, Ahmed Ali-Eldin, and Jeanne Albrecht. 2019. Understanding Synchronization Costs for Distributed ML on Transient Cloud Resources. In 2019 IEEE International Conference on Cloud Engineering (IC2E). 145–155. https://doi.org/10.1109/IC2E.2019.00029

[5] Matt Baugham, Simon Caton, Christian Haas, Ryan Chard, Rich Wolski, Ian Foster, and Kyle Chard. 2019. Deconstructing the 2017 Changes to AWS Spot Market Pricing. In Proceedings of the 10th Workshop on Scientific Cloud Computing (Phoenix, AZ, USA) (ScienceCloud’19). Association for Computing Machinery, New York, NY, USA, 19–26. https://doi.org/10.1145/3322795.3331465

[6] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. Pearson correlation coefficient. In Noise reduction in speech processing. Springer. 1–4.

[7] Amazon Web Services Blogs. 2017. New Amazon EC2 Spot pricing model. https://aws.amazon.com/blogs/compute/new-amazon-ec2-spot-pricing/.

[8] Andrew Boutros, Eriko Nuruivatidhi, Rui Ma, Sergey Gribok, Zhigeng Zhao, James C. Hoe, Vaughn Betz, and Martin Langhammer. 2020. Beyond Peak Performance: Comparing the Real Performance of AI-Optimized FPGAs and GPU. In 2020 International Conference on Field-Programmable Technology (ICFPT). 10–19. https://doi.org/10.1109/ICFPT51310.2020.00011

[9] Navraj Chohan, Claris Castillo, Mike Spreitzer, Malgorzata Steinder, Asser Tadesse, and Chandra Kantakesar. 2018. See Spot Run: Using Spot Instances for MapReduce Workflows. In 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10). USENIX Association, Boston, MA. https://www.usenix.org/conference/hotcloud-10/see-spot-run-using-spot-instances-mapreduce-workflows.

[10] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In Proceedings of the 6th conference on Symposium on Operating Systems Design & Implementation - Volume 6 (San Francisco, CA) (OSDI’04). USENIX Association, Berkeley, CA, USA. 10–48. http://dl.acm.org/citation.cfm?id=1252154.1252164

[11] Thuan Duong-Ba, Tuan Tran, Thinh Nguyen, and Bella Bosse. 2021. A Dynamic Virtual Machine Placement and Migration Scheme for Data Centers. IEEE Transactions on Services Computing 14, 2 (2021), 329–341. https://doi.org/10.1109/TSC.2018.2817208

[12] Nnamdi Ekeke-Ekwe and Adam Barker. 2018. Location, Location, Location: Exploring Amazon EC2 Spot Instance Pricing Across Geographical Regions. In 2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). 370–373. https://doi.org/10.1109/CCEGRID.2018.00059

[13] John Forrest, Ted Ralphs, Stefan Vigerske, LouHafer, Bjarni Kristjansson, jphasano, EdwinStraver, Miles Lubin, Haroldo Gambini Santos, rlougee, and Matthew Saltzman. 2015. con-ocr/ch: Version 2.9.9. https://doi.org/10.5281/zenodo.1317556

[14] Apache Software Foundation. 2014. Apache Hadoop. http://hadoop.apache.org.

[15] Weichao Guo, Kang Chen, Yongwei Wu, and Weiming Zheng. 2015. 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 (Portland, Oregon, USA) (HPDC ’15). Association for Computing Machinery, New York, NY, USA, 191–202. https://doi.org/10.1145/2749246.2749259

[16] Xin He, Prashant Shenoy, Ramesh Sitaraman, and David Irwin. 2015. 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 (Portland, Oregon, USA) (HPDC ’15). Association for Computing Machinery, New York, NY, USA, 207–218. https://doi.org/10.1145/2749246.2749275

[17] Qinghao Hu, Peng Sun, Shengen Tan, Yonggang Wen, and Tianwei Zhang. 2021. Characterization and Prediction of Deep Learning Workloads in Large-Scale GPU Datacenters. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE Press.

[18] David Irwin, Prashant Shenoy, Pradeep Ambati, Prateek Sharma, Supreeth Shastri, and Ahmed Ali-Eldin. 2019. The Price Is (Not) Right: Reflections on Pricing for Transient Cloud Servers. In 2019 28th International Conference on Computer Communication and Networks (ICCCN). 1–9. https://doi.org/10.1109/ICCCN2019.8846933

[19] AWS What is New. 2021. Introducing Amazon EC2 Spot placement score. https://aws.amazon.com/about-aws/whats-new/2021/10/amazon-ec2-spot-placement-score/.

[20] Bahman Javadi, Ruppa K. Thulasiram, and Rajkumar Buyya. 2013. Characterizing sporic price dynamics in public and private cloud environments. Future Generation Computer Systems 29, 4 (2013), 988–999. https://doi.org/10.1016/j.future.2012.06.012 Special Section: Utility and Cloud Computing.

[21] Myeongjae Jeon, Shivaram Venkataraman, Amar Phanishayee, Junjie Qian, Myeongjae Jeon, Shivaram Venkataraman, Amar Phanishayee, Junjie Qian, Myeongjae Jeon, Shivaram Venkataraman, Amar Phanishayee, Junjie Qian, Myeongjae Jeon, Shivaram Venkataraman, Amar Phanishayee, Junjie Qian. 2019. Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads. In 2019 28th International Conference on Computer Communication and Networks (ICCCN). 947–960. https://www.usenix.org/system/files/conference/icccn19/presentation/jeon.

[22] JCS Kadupugne, Vikram Jadha, and Prateek Sharma. 2020. Modeling the Temporally Constrained Preemptions of Transient Cloud VMs. In Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing (Stockholm, Sweden) (HPPC ’20). Association for Computing Machinery, New York, NY, USA, 41–52. https://doi.org/10.1145/3365983.3393267

[23] Mikhail Khodak, Liang Zheng, Andrew Chen, and Ahmed Ali-Eldin. 2019. The Price Is (Not) Right: Reflections on Pricing for Transient Cloud Servers. In 2019 28th International Conference on Computer Communication and Networks (ICCCN). 1–9. https://doi.org/10.1109/ICCCN2019.8846933

[24] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. 2009. Pearson correlation coefficient. In Noise reduction in speech processing. Springer. 1–4.

[25] Amazon Web Services Blogs. 2017. New Amazon EC2 Spot pricing model. https://aws.amazon.com/blogs/compute/new-amazon-ec2-spot-pricing/.
