Making Cloud Spot Instance Interruption Events Visible

KyungHwan Kim
Department of Computer Science
Kookmin University
Seoul, South Korea
bryan9801@kookmin.ac.kr

Kyungyong Lee
Department of Computer Science
Kookmin University
Seoul, South Korea
leeky@kookmin.ac.kr

ABSTRACT

Public cloud computing providers offer a surplus of computing resources at a lower price with a service of a spot instance. Despite the possible great cost savings from using spot instances, sudden resource interruption can occur as resource demand changes. To help users estimate cost savings and the possibility of interruption when using spot instances, vendors provide diverse datasets. However, the effectiveness of using the datasets has not yet been quantitatively evaluated, and many users still rely on the guess when choosing spot instances. To help users lower the chance of interruption of the spot instance for reliable usage, in this paper, we thoroughly analyze various datasets of the spot instance and present the feasibility for value prediction. Then, to measure how the public datasets reflect real-world spot instance interruption events, we conduct real-world experiments for spot instances of AWS, Azure, and Google Cloud. Combining the dataset analysis, modeling, and the real-world spot instance interruption experiment, we present a significant improvement in reducing the possibility of interruption events.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**; • **Information systems → Web log analysis**.

KEYWORDS

cloud computing, spot instance, interruption modeling, enhancing reliability, spot instance datasets

ACM Reference Format:
KyungHwan Kim and Kyungyong Lee. 2024. Making Cloud Spot Instance Interruption Events Visible. In *Proceedings of the ACM Web Conference 2024 (WWW '24), May 13–17, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3589334.3645548

1 INTRODUCTION

Cloud computing provides compute resources elastically without the burden of system operation overhead and changes the way we consume compute resources. The core success of cloud computing is due to its on-demand billing model that allows users to dynamically start and stop instances based on its application needs and pay for what they have actually used. To support elastic resource usage, cloud service providers should prepare abundant resources to meet resource usage spikes. Such a plentiful resource preparation inevitably results in resource wastage when computing demand is low. To encourage instance usage when demand is low, public cloud vendors provide the surplus computing resources at a discounted price which can be more than 90% cheaper than the on-demand price in some cases. The billing model is called spot instances, and they are offered by most public cloud service providers, AWS [1], Azure [54], Google Cloud Platform (GCP) [18], Alibaba [36], Oracle Cloud Preemptible Instances, and IBM Transient Virtual Servers.

In the early days of spot instance offering, most vendors adopted an auction mechanism where a user sets a bidding price and a provider sets a spot price. If the bidding price is higher than the spot price, a user gets granted an instance and pays for the spot price, not the bidding price. Based on the compute resource demands, a service provider changes the spot price, and if the spot price becomes higher than the initial bidding price, a spot instance is revoked, which is generally referred to as instance interruption. Such an instance interruption can negatively impact applications' reliability, and users should be prepared for the event. To help users improve reliability when using spot instances, cloud vendors provide diverse datasets publicly. One of the most representative datasets is the spot instance price, which presents the spot price at a specific time. Many research was conducted using the spot instance price dataset to analyze the spot price itself [1, 13, 21, 36, 51], using the spot instance dataset to enhance reliability for diverse applications [2, 11, 19, 30, 34, 43, 45, 49], or suggesting an optimal bidding price using the spot price prediction [3, 15, 25, 26, 44, 60].

The spot price dataset was a precious source of information when using spot instances reliably, especially for AWS, but it changes with the new spot instance operation policy. With the new policy, the advertised spot price does not reflect the spot instance interruption anymore, and the frequency of the change in data becomes very low [6, 20] which greatly degrades the application of the dataset and invalidating the result of previous research. Meanwhile, new data sets of spot instances are released on the web to help users build a reliable resource pool of spot instances [31]. The new datasets include the interruption ratio for the previous month provided by Azure and AWS, and the instant availability information provided by AWS. The new datasets can be of great help for spot instance users, as they provide the historical interruption ratio and instantaneous availability information. However, thorough investigation of the newly provided datasets has not yet been performed, and quantitative evaluation on the correlation of the dataset and the spot instance interruption has not been carried out yet. Given that sudden spot instance interruption is a major hurdle when adopting a spot instance as the main compute resource pool, it is crucial to

WWW '24, May 13–17, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0171-9/24/05…$15.00
https://doi.org/10.1145/3589334.3645548

estimate the likelihood of a spot instance being interrupted at least in the near future. However, predicting future spot instance interruptions or building a statistical model for interruption behavior can be very challenging from outside of cloud service vendors' view as interruption events detail is not accessible.

To quantitatively evaluate and model spot instance interruption events, one should know when the interruption events happen for various instance types that are located in global regions and availability zones. Service providers may have such records, but the information is not publicly available. For example, Yang et. al. [54] proposed a spot instance interruption prediction model for Azure spot instances using the internal dataset of interruption records that are not publicly available, and it limits the development of publicly available research results. To model spot instance interruption events without proprietary information, SciSpot [23] and Pham et. al. [39] conducted real-world experiments to observe spot instance interruption events by running spot instances until an interruption occurs for GCP and AWS, respectively. Even after gathering logs of interruption events, it can be challenging to uncover features that are related to the interruptions. In the case of AWS, the spot price was a good indicator, but it is no longer valid after the change in the operation policy [6, 20].

Based on the observation that the prediction and modeling of spot instance interruption events is challenging even with newly released spot instance datasets, we first analyze the characteristics of the spot price, prior period interruption ratio, and instant availability dataset that are provided by AWS, Azure, and GCP. To validate whether public datasets reflect the real behaviors of spot instance interruptions and compare spot instance reliability of multiple vendors, we conducted spot instance interruption tests for AWS, Azure, and GCP. Based on the public dataset analysis and the result of the real-world spot instance interruption experiment, we argue that precise prediction of the value of the instant spot instance availability dataset can lower the probability of spot instance interruption and confirm the argument quantitatively. Overall, we conclude that the Azure spot instances showed the highest reliability, followed by GCP and AWS. Though AWS showed the lowest reliability, we find that the data set provided by AWS is really helpful in conjecturing spot instance reliability. Using a proposed dataset value predictor, the running time of AWS can increase by 63.2% for instances with an initial high score and by 168% for instances with an initial low score. However, for Azure and GCP, which provide limited datasets, we could not find a strong correlation of spot instance reliability with the provided datasets, as we could with the AWS dataset.

In summary, the main contributions of this paper are as follows.

- Comparison of the reliability of AWS, Azure, and GCP. To the authors' best knowledge, this is the first work to compare the spot instance reliability of major multiple vendors.
- Thorough analysis of spot instance datasets including price, interruption ratio, and availability, to discover correlations with the spot instance interruptions mainly for AWS.
- Quantitatively presenting the effectiveness of a simple model when predicting spot instance dataset values.
- Proposing a heuristic to recommend spot instance types to lower the interruption ratio without using secretive internal operation logs exclusively owned by cloud vendors.

2 CLOUD SPOT INSTANCE AND DATASETS

When using spot instances, the cost saving ratio over the on-demand instance and the reliability of a spot instance are the most important metric for most users. To help users estimate the benefit and risks when using spot instances, the cloud vendors provide various datasets; the spot instance price from the beginning and the more recent interruption ratio and instant availability information.

2.1 Depreciation of Price Dataset

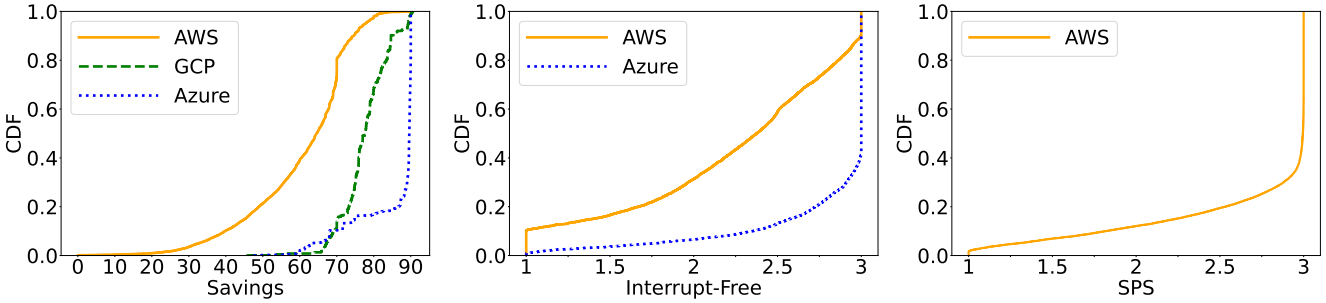
Since the introduction of the spot instance service in 2009, the spot price dataset is publicly available and triggered many research from various perspectives. Statistical analysis of the spot instance price dataset was performed in a comprehensive way [1, 13, 15, 21, 33, 36, 40, 51, 52]. Other works focused on proposing an optimal bidding price to reduce the risk of instance interruption [3, 17, 26, 32, 42, 44, 46, 57, 60]. Another type of work focused on running various applications on spot instances reliably using spot instance price datasets, such as web server [2], big data processing [53], deep learning training [30, 48] and inference [59], batch processing [34, 45], and scientific high-performance computing applications [61]. Other works are carried out using a dataset from other vendors, Azure [54, 55], GCP [18, 23], and Alibaba cloud [36].

One of the reasons that many researches have been conducted using AWS spot instance and its price dataset is owing to its dynamically changing price patterns, which reflect the spot instance interruption events very well. By comparing the advertised spot price and bidding price, a user can easily conjecture the interruption possibility. However, modeling spot instance interruption using a price dataset becomes impossible since the spot instance operation policy change [6, 20]. With the new change, the spot price does not change as often as before. More importantly, the spot price does not indicate an instance interruption; though the advertised spot price is lower than a bidding price, an interruption can still happen. The operation policy change makes the AWS spot instances similar to other vendors' spot instances which implies the spot price rarely changes, and it makes most of the previous research that relied on the spot price datasets becomes useless.

2.2 Appreciation of Availability Dataset

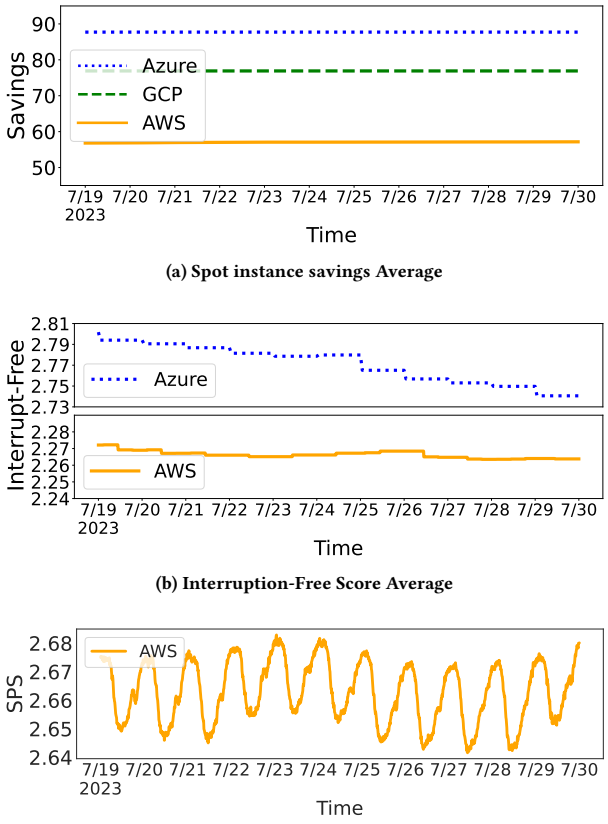
As the usefulness of the spot price dataset decreases from the perspective of inferring spot instance reliability, the service vendors started to provide new types of datasets related to instance availability. AWS and Azure provide the interruption ratio of an instance in the previous time frame, for example, 30 days. The dataset classifies interruption ratios into five categories, less than 5%, between 5% and 10%, between 10% and 15%, between 15% and 20%, and more than 20%. By using the dataset of the previous interruption rate, users are expected to infer spot instance's future reliability.

Different from the interruption ratio dataset which simply provides the statistics from the previous period, AWS provides a new data set called Spot Placement Score (SPS). The vendor did not disclose the internal details of how the score is calculated, but it is known to present a timely spot instance's availability. In the SPS, a type of spot instance is assigned an integer score ranging from one to three; the higher score implies greater availability. An unique SPS is assigned for each instance type in an availability zone. Users



(a) Savings ratio (b) Interruption-free score (c) Instant availability score

Figure 1: Spot instance dataset value distribution from multiple vendors



(a) Spot instance savings Average

(b) Interruption-Free Score Average

(c) Instant Availability Score Average

Figure 2: Spot datasets that have been changed during the observation period

can access dataset through the AWS console or API service, but it imposes many restrictions when querying the value. To resolve the query restriction, SpotLake [31] implemented many heuristics and served the dataset as raw files. Note that Azure and GCP do not provide a similar dataset, and the analysis on the availability dataset will be based only on SPS.

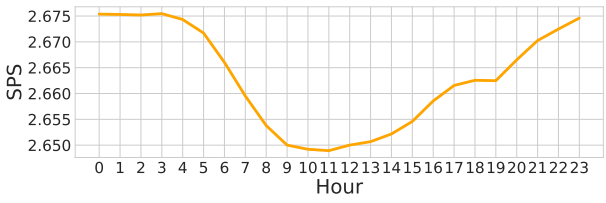
3 ANALYSIS OF SPOT INSTANCE DATASET

To better understand the characteristics of different spot instance datasets, we present empirical analysis results of publicly available spot instance datasets provided by SpotLake [29]. From SpotLake, we could download the spot price dataset of AWS, Azure, and GCP. For the spot instance interruption ratio dataset, we get datasets from AWS and Azure. For instant availability information, we get the AWS SPS dataset. In the analysis, datasets from 1 November 2022 to 31 August 2023 were used.

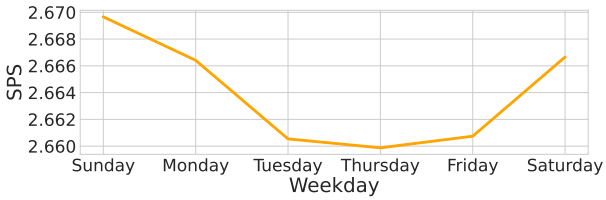
Figure 1 shows the cumulative distribution function (CDF) of the values of the multiple spot instance dataset. Figure 2a compares the savings ratio, calculated as $(1.0 - \frac{SpotPrice}{On\text{-}demandPrice}) \times 100$, of AWS, Azure, and GCP. Figure 2b compares the instance interruption ratio of AWS and Azure. Both vendors provide the dataset with values in the range of five categories, and we match each category to a numeric value between increments of 1.0 and 3.0 by 0.5. The most frequent interruption ratio of more than 20% is matched to 1.0, and the least value of less than 5% is matched to 3.0. We name the converted value as *interrupt-free score* following the notation proposed in SpotLake [31]. Figure 2c presents the distribution of the instant availability dataset provided by AWS. We use the SPS dataset with the score range provided. To distinguish vendors, we use yellow-solid, blue-dotted, and green-dashed lines to indicate AWS, Azure, and GCP, respectively. In the figures, the vertical axis shows the distribution, and the horizontal axis shows the score. For all three sub-figures, the large score means more reliable status.

In the figures, it is apparent that AWS has the lowest cost savings (Figure 2a) and a higher frequency of interruption of the spot instance (Figure 2b). GCP provides only the price dataset and shows a higher median cost savings ratio than AWS. Regarding the interrupt-free score, the median score of AWS is around 2.5, which means the interruption ratio between 5% and 10%, while that of Azure is around at 3.0, which is less than 5%. Only AWS provides instant availability information, and we can observe that most of the spot instances have been allocated near 3.0, which is the best score.

Figure 2 shows the pattern of temporal change of three spot datasets averaged for all instance types collected in each subfigure. Figure 2a compares the savings ratio for the spot instance, and we can observe that regardless of the vendor, the savings value rarely changes over time. The low frequency of the price change of the



(a) Hourly change pattern



(b) Daily change pattern

Figure 3: The change pattern of instant availability dataset shows a strong hourly and daily change pattern

AWS spot instance agrees with previous work that analyzed the recent change in the behavior of the AWS spot instance [6, 20]. Figure 2b presents the pattern of temporal change of the interrupt-free score. To check whether there exists a visually noticeable change pattern, we separated the graph of AWS and Azure to present them in a micro-scale. Similarly to cost savings, both AWS and Azure do not show a noticeable change pattern over time. Figure 2c presents the temporal change pattern of the AWS SPS dataset. Unlike the cost savings and interruption dataset, the SPS dataset shows a sinusoidal periodic pattern for different days. From the analysis, we can uncover the unique and noticeable pattern of SPS change that intrigues further research and analysis. Note that the range of the vertical axis is different for distinct datasets. We closely observed the pattern of savings and interruption-free score in a small scale but could not find a pattern similar to SPS.

To further analyze the pattern of change in the SPS value, Figure 3a shows the SPS values averaged over an hour of a day. In the analysis, we used resources located globally in multiple regions and used the local time zone of each availability zone. From the figure, we can discover that the SPS is lower in the morning. Then the SPS gradually increases and reaches its maximum at night, which is the time when cloud usage is expected to be low. To see whether the change in the SPS value has a weekly pattern, Figure 3b presents the average SPS value by day, shown on the horizontal axis. We can observe that the SPS value is higher on weekends and lower on weekdays. Further analysis of SPS is presented in the Appendix. Note that we analyzed the spot price and the interruption ratio, but we could not observe a noticeable pattern.

We could observe a more frequent and periodic data update pattern of the AWS SPS dataset. To further analyze how often spot instance dataset changes, Figure 4 shows the CDF of the frequency of data change. The vertical axis shows the distribution and the horizontal axis shows how long a value remains unchanged. The larger value, located on the right side of the figure, implies that

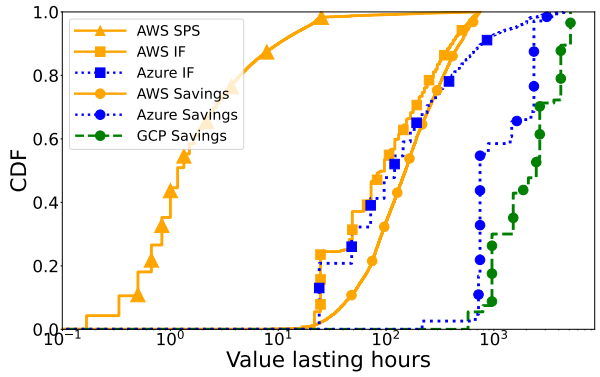


Figure 4: The CDF of value change frequency. The SPS dataset provided AWS shows the most frequent data update

the value does not change for a longer period of time. In the figure, the distribution of all datasets is presented. Among them, the AWS SPS with a yellow solid line with triangle markers shows the most frequent update compared to other dataset. For the cost savings dataset, we assume that a cost savings value has changed when the ratio has increased or decreased by more than 3%.

From the analysis of the various spot instance datasets, we found that spot instances from Azure and GCP show relatively higher cost savings than AWS. The interruption ratio of AWS is higher than that of Azure. Only AWS SPS shows a pattern of daily and hourly changes. The values of AWS SPS changed frequently, and the frequency of change in other datasets is greater than one day.

4 PREDICTING SPOT DATASET CHANGE

When using spot instances, the prediction of future interruption events and probability estimation can be of great help to improve reliability. To achieve the goal, it is important to discover appropriate features that correlate with the interruption events. The analysis of the spot instance dataset reveals that the price and interruption ratio information has little change over time, while the instant availability information of AWS SPS changes often and regularly. Taking into account the recent research outcomes about spot instance price [6, 20, 31], the spot price has little correlation with the interruption of the spot instance. The spot instance interruption ratio presents the prior 30 days of interruption statistics, and it can be far from predicting future availability. In this context, we propose a model to predict the instant availability dataset expecting that the precise prediction can help to improve the reliability of the spot instance, which will be evaluated later.

4.1 Modeling Instant Availability Dataset

The instant availability dataset prediction model uses the SPS of the prior period as input features and predicts future SPS values. Formally speaking, the input dataset and features of $\boldsymbol{X}$ are composed of N distinct instance types in different availability zones, which is the unit of the distinct SPS being provided, and D features which represent an SPS value at different timestamp. Assuming that data collection of the SPS dataset occurs with a period of p minutes, we can use the previous D collected dataset for training, which

	3H	6H	12H	24H	48H	72H
LR	0.96	0.97	0.98	0.97	0.97	0.96
RF	0.96	0.98	0.98	0.98	0.97	0.97
XGboost	0.97	0.99	0.98	0.98	0.97	0.97
SVC	0.96	0.97	0.97	0.98	0.96	0.96
Prophet	0.96	0.96	0.97	0.98	0.97	0.97
ARIMA	0.95	0.95	0.97	0.97	0.96	0.96

Table 1: F1-score of SPS prediction using different models

	Standard Deviation			
	Over 0.1	Over 0.25	Over 0.5	Over 0.75
LR	0.73	0.70	0.62	0.54
RF	0.75	0.72	0.63	0.54
XGBoost	**0.77**	**0.74**	**0.66**	**0.57**
SVC	0.76	0.73	0.64	0.53
Prophet	0.75	0.72	0.64	0.53
ARIMA	0.74	0.71	0.63	0.53

Table 2: F1-score of SPS prediction for different degree of value change

means that we use datasets collected in the last $D \times p$ minutes. Similarly, the target dataset to predict is indicated as Y that has N different instance types and next K SPS values to predict. x_{id}, where $i = 1 : N, d = 1 : D$, means the i-th instance type and the d-th index SPS, and y_{ik}, $i = 1 : N, k = 1 : K$, means the i-th instance type and k-th index SPS. Train dataset is defined as follows.

$$\mathcal{D} \triangleq \{(x_{id} \rightarrow y_{ik}) | (x_{id}, y_{ik}) \in \{1, 2, 3\}\} \quad (1)$$

Each value of x_{id} and y_{ik} takes one of $\{1, 2, 3\}$ that is a possible SPS value. In defining the training dataset, we need to decide how many previous measured datasets to use, which is D, and how many future dataset to predict, which is K. Our thorough empirical analysis reveals that the selection of D and K does not have much impact to the overall prediction accuracy (Figure 11 in the Appendix).

4.2 Accuracy of Availability Dataset Predictor

Using the train dataset, $\mathcal{D}$, in equation 1, we apply various classification models of the Linear Regressor (LR) [37], Random Forest (RF) classifier [8], XGBoost [10], Support Vector Classifier (SVC) [5], Prophet [47], and ARIMA [7]. Table 1 compares the F1 score for different prediction models to assess both precision and recall [41]. In the train, we used the SPS dataset from 16 July 2023 to 31 July 2023. For testing, we used the dataset from 1 August 2023 to 6 August 2023 which is exclusive to the training dataset period. We divide the test dataset in half and use the previous half-period as the model input. Of the next half-data set, we vary the prediction period, k, to the next 3, 6, 12, 24, 48, 72 hours, and they are shown in the columns. Among many models, the XGBoost shows the best performance regardless of the future prediction window size, although the difference is marginal.

We can observe that the prediction performance is very similar although the prediction model is different. Our investigation reveals that in many cases the SPS values stay constant over time, which makes it easy to predict. To compare the prediction performance when the SPS value changes quite abruptly, Table 2 presents the prediction performance categorized by the frequency of the change in the SPS value. We classified instance types whose standard deviation of the SPS value during the test data period was larger than 0.1, 0.25, 0.5, and 0.75. Of the total test data cases, 14%, 11%, 6% and 2% of the cases. respectively, belong to each category.

We can observe that, for the SPS that changes more frequently (*Over 0.75*), XGBoost showed the best prediction performance. To use the SPS prediction, it is important to understand the operation overhead, such as the train and the inference time. We compared the operation overhead in Table 4 in the Appendix. Considering prediction performance and operation overhead, using the XGBoost model makes the most sense.

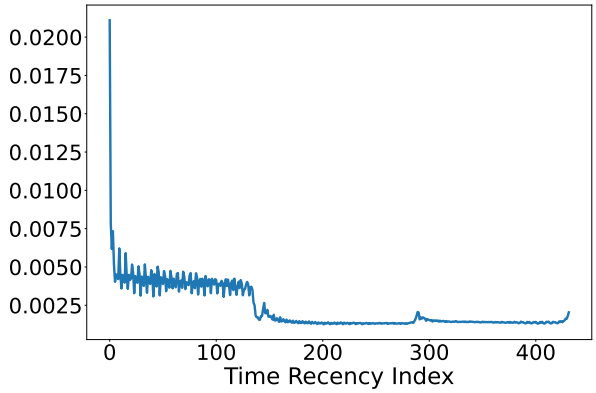


Figure 5: The importance of features when building a SPS prediction model. The smaller X-axis value implies more recent dataset from the train time

To analyze the impact of the recency of the SPS values when building a prediction model, Figure 5 presents the importance of the feature extracted from a built XGBoost model. The vertical axis shows the importance of each feature, and the horizontal axis shows the index of the time-series feature. The lower index number implies a more recent data set when training a model. For example, assuming that a model is built at an arbitrary time t, the index of 1 means the SPS value observed in the time unit $t - 1$. As shown in the figure, the recent SPS values dominate a model, where SPS values until 24 hours taking 79% of total importance.

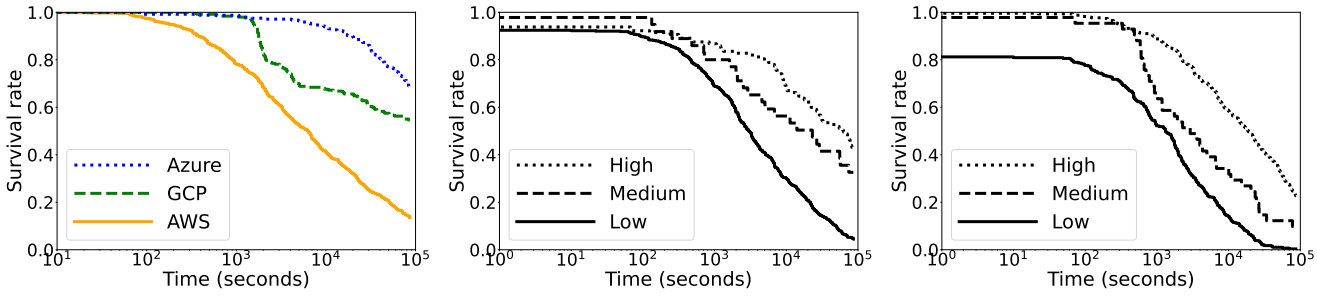
5 EVALUATION

This section evaluates the effectiveness of a heuristic to improve the reliability of the spot instance through a detailed analysis of spot instance interruption events. More specifically, we would like to answer the following research questions.

RQ-1 For spot instances offered by the major public vendors, which vendor provides the most reliability with respect to the interruption?

RQ-2 Of many spot instance datasets, which dataset is most correlated with spot instance interruption?

RQ-3 Can the prediction of the SPS value lower the chance of spot instance interruption?



(a) Vendor (b) Interrupt-free Score (c) Instant Availability (SPS) Score

Figure 6: Kaplan-Meier Estimator to compare the probability of spot instance survival rate with respect to different spot instance datasets and Vendor

5.1 Spot Instance Interrupt Analysis

We first compare the interruption behaviors of multiple cloud vendors (**RQ-1**). To examine interruption events, we performed real-world experiments by running a spot instance until an interruption event occurred. We choose 875 different types of instances in AWS, Azure, and GCP located worldwide. When choosing arbitrary instance types, we try to distribute evenly for different values in the cost savings, interruption ratio, and instant availability datasets. The experiments had been conducted for 24 hours for each type of instance in the period between July 2022 and March 2023. When an interruption occurs, we mark the event and continue to request the spot instance until it is fulfilled again. We share the source code for conducting interruption experiments as an artifact [28].

To quantitatively compare the survival rate of the spot instances, the probability that a spot instance remains running, we applied the Kaplan-Meier Estimator [24] which is a non-parametric statistic to estimate the survival rate of a lifetime dataset. It is generally used in the hospital environment to measure the fraction of people who live after treatment or the length of time that people remain unemployed after losing a job [35]. Considering the general usage, applying it to model spot instance survival probability is well suited. Kaplan-Meier Estimator is calculated as follows.

$$\widehat{S}(t) = \prod_{i:\, t_i \le t} \left(\frac{n_i - d_i}{n_i} \right)$$

The survivor function ($\widehat{S}(t)$) at time t is defined as the probability that a life (spot instance running time) is longer than t. n_i is the number of survived (running spot instances) until time t_i, and d_i means the number of deaths (spot instance interruption) at time t_i.

Figure 6 compares the Kaplan-Meier estimator distribution grouped by different spot instance dataset and vendor. In the figure, the horizontal axis shows the spot instance running time (survival time), and the vertical axis shows the distribution.

Figure 6a compares $\widehat{S}(t)$ of AWS, GCP, and Azure. The yellow-solid, green-dashed, and blue-dotted lines present the distribution of AWS, GCP, and Azure respectively. From the figure, it is evident that AWS shows the lowest survival rate, followed by GCP and Azure. The median running time of the AWS spot instance is 1.2 hours, and more than half of the GCP and Azure instance did not experience interruption during the 24 hour experiments. The shortest $P90$ run time of AWS, GCP, and Azure is 0.02, 0.5, and 5 hours, respectively.

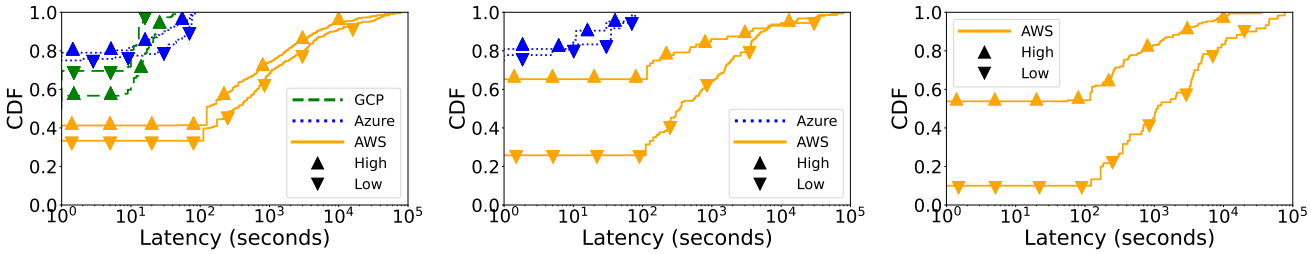
Next, we evaluate the correlation of various spot instance datasets and the interruption events (**RQ-2**). Figure 6b compares the survival time distribution with respect to different interrupt-free scores, and Figure 6c compares with respect to the SPS scores. Both figures are the AWS experiment results grouped by different spot instance datasets. For both datasets, the higher score implies greater availability. The scores of *Low* (1.0), *Medium* (2.0), and *High* (3.0) are presented with solid, dashed, and dotted lines, respectively. From both figures, we can observe that higher score values show a higher survival rate than lower score values. We omit the savings figures, and the interrupt-free score for Azure because they do not show a noticeable pattern as the AWS interrupt-free score and SPS do.

Figure 7 compares the time elapsed between the start of a spot instance interruption and the node becoming fulfilled again. The shorter time implies that a spot instance becomes available again shortly after an interruption occurs, which means a higher availability. The sub-figures group instances according to the spot instance dataset. In each figure, the yellow-solid, blue-dotted, and green-dashed lines indicate the results of AWS, Azure, and GCP, respectively. In each line, we mark a symbol of $\triangle$ for *High* and $\triangledown$ for *Low* value after categorization.

The cost savings ratio dataset (Figure 7a) does not show a noticeable difference among *High* and *Low* values for all AWS, Azure, and GCP. GCP and Azure show a similar latency distribution, while that of AWS shows much less availability than the others. Regarding the interrupt-free score (Figure 7b), Azure did not show a different pattern between *High* and *Low* values, but AWS shows a distinct pattern that follows the advertised score. The high interrupt-free score of AWS spot instances shows a faster fulfillment time after an interruption. The instant availability data (Figure 7c) show a noticeable difference between *High* and *Low* that the higher SPS value shows lower latency for fulfillment after an interruption (higher availability) and follows the advertised score characteristic well.

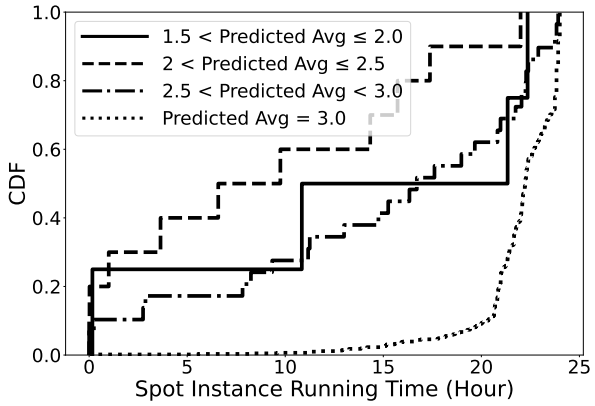
5.2 Effectiveness of SPS Prediction

We have discovered that the instant availability dataset of the spot instance provided by the AWS SPS dataset is beneficial to model the interruption events and availability of the spot instance. To

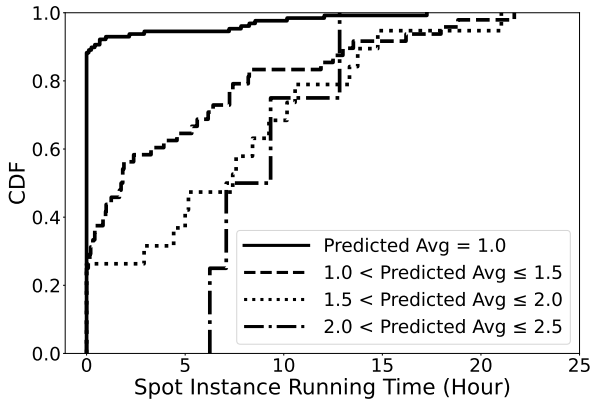


(a) Cost Savings Ratio (b) Interrupt-free Score (c) Instant Availability (SPS) Score

Figure 7: A CDF of latency until a request becomes fulfilled for AWS, Azure, and GCP spot instances categorized by different dataset values of High (△) and Low (▽). The lower latency implies the higher availability.



(a) Initial SPS is High



(b) Initial SPS is Low

Figure 8: Distribution of spot instance availability when using only the current SPS value and predicted values

go one step further and answer **RQ-3** of whether predicting future SPS value can help increase the probability of predicting the interruption of the spot instance, we compare the distribution of the interruption of the spot instance with different predicted SPS values in Figure 8. The vertical axis shows the distribution, and the horizontal axis presents the spot instance running (survival) time. Figure 8a presents the life time of spot instances whose SPS value is high (3.0) when a spot instance request is made. The SPS value can change over the course of the experiment, and we categorize the predicted SPS values in a bin size of 0.5. The dotted line shows when the average predicted SPS value is 3.0, which means that the SPS value is expected to remain constant at the high value. A dashed-dotted line indicates when the expected SPS average is higher than 2.5 but lower than 3.0. The other lines are expressed in a similar way. From the figure, we can discover that, despite the same initial high SPS value of 3.0, ignorant of the future SPS value can significantly affect the reliability of a spot instance. For example, the median survival time for the spot instance when the predicted SPS is high is 22 hours, but that of the lower predicted SPS value is 16 hours.

Figure 8b shows the survival time of spot instances whose initial SPS value is low (1.0). Similarly, the SPS value can fluctuate, and we show the running time by grouping instances by the predicted SPS values. The solid line presents when the SPS value is expected to remain low at 1.0, and it shows a very short run time. Otherwise, even if the initial SPS value is low, instances with higher expected SPS values show a longer survival time. For example, the median running time of the spot instance whose SPS is expected to be consistently at 1.0 is 0 hours, while that of the SPS expected to be between 2.0 and 2.5 turns out to be about 8 hours. This finding is especially helpful when a user has a limited list of spot instance types and their initial SPS is low, such as GPU instances. In that case, the prediction of SPS can greatly improve the reliability of the spot instance, although its current SPS is low.

When choosing spot instances, referencing predicted SPS values can greatly improve the reliability. This approach is unique compared to the previous work, which directly attempted to predict spot instance interruption events. The work carried out by Azure researchers [54] used internal spot instance interruption logs and showed excellent prediction performance. However, the dataset is not publicly available and limits public research. The work presented in SpotLake [31] tried to model the interruption events of the spot instance using a random forest [8] using logs from interruption experiments, but shows poor prediction performance due to the lack of sufficient training data that require prohibitive experimental

cost. This proposed approach can improve spot instance reliability with a low budget without requiring internal information.

In summary, we can answer the research questions raised before as follows.

RQ-1 For spot instances offered by major public vendors, which vendor provides the most reliability with respect to interruption? **Answer** : The Azure spot instance showed the most reliability followed by GCP and AWS.

RQ-2 Of many spot instance datasets, which dataset is most correlated with spot instance interruption? **Answer** : The AWS SPS dataset shows the highest correlations with the spot instance interruption. The cost savings and interrupt-free score do not show a high correlation as AWS SPS does.

RQ-3 Can the prediction of the SPS value reduce the chance of spot instance interruption? **Answer** : Yes, the prediction of SPS dataset can definitely help to increase the spot instance reliability. When the initial SPS value is high, selecting an instance whose predicted value is consistently high can increase the expected runtime of the spot instance by 63.2%. Even when the initial SPS value is low, selecting spot instances with higher expected SPS value can increase the expected spot instance running time by 168%

Overall, the spot instance datasets offered by Azure and GCP are not really useful when choosing appropriate instance types. However, the reliability of Azure and GCP spot instances is relatively high, and users may experience a lower interruption regardless of instance selection. The spot instance offered by AWS shows the most interruptions. However, the spot instance datasets provided by AWS reflect spot instance interruptions very well. Thus, different from Azure and GCP, when using a spot instance from AWS, users should be more cautious to choose appropriate instance type for reliable execution.

6 RELATED WORK

Modeling and Using Spot Instance Dataset : From the inception of the spot instance, the spot price dataset is widely used and analyzed to enhance the reliability of the spot instances. There were attempts to characterize the price dataset of the spot instance and relate the analysis to the interruption of the spot instance to decrease the chances of interruption [13, 21, 33, 36, 40, 40, 51, 52]. Using the spot price and the analysis result, Ali-Eldin et. al. proposed to deploy a web-server [2], Son et. al. proposed DeepSpotCloud [30] for DNN training tasks using GPU spot instances located globally. SeeSpotRun [11] for Hadoop [16] MapReduce [12], Flint [43] and Tr-Spark [53] for Apache Spark [58] are proposed for big data processing. Online web services [2, 19], batch processing jobs [34, 45], and parallel processing of independent tasks [49] are proposed while mitigating the straggler effect due to transient servers [4]. The aforementioned work relied on the spot price dataset, and the work becomes obsolete due to the spot instance operation change [6, 20]. The findings in this paper provide a new opportunity to improve the reliability of spot instances by predicting instant availability without relying on the spot price dataset.

Spot Instance Price Prediction : We showed that predicting the spot instance instant availability dataset is predictable using a machine learning classifier model with decent performance. Before this work, there are attempts to predict the spot price value to predict future cost savings and interruption events. Khandelwal et. al [25] used the Random Forest, Fabra et. al [15] used a deep neural net model, and Alkharif et. al [3] used various statistical methods of time series analysis for price prediction. Due to the change in the spot price operation policy [6, 20], the previous work becomes obsolete, and new approaches, as in this paper, should be provided.

Analyzing Spot Instance Interruption : Compared to the analysis of the spot price dataset, the analysis and experiments of spot instance interruption and correlating it with the spot instance dataset were not carried out much. Pham et. al. [39] and Lee et. al. [31] conducted spot instance interruption experiments for only AWS instances to analyze the interruption pattern. For Azure, Yang et. al. [54] proposed using an instance interruption prediction model based on a Transformer model by using the internally available interruption trace of Azure. For GCP, Haugerud et. al. [18] and Kadupitiya et. al. [23] conducted interruption tests to model behavior. To the best of the authors' knowledge, this paper is the first work to conduct spot instance interruption experiments for three major vendors of AWS, Azure, and GCP. Comparing the behavior of spot instances can greatly help users to choose the most appropriate spot instances in a multi-cloud environment that is considered to be widely adopted [9, 38, 56].

7 CONCLUSION

Cloud spot instances provide significant cost savings when using cloud resources with the risk of sudden instance termination. To help users better use the spot instance, public service vendors provide diverse spot instance datasets, such as price, interruption ratio in the past period, and instant availability information. Despite the diverse publicly available information, they are neither widely used nor analyzed except for the spot price dataset which is irrelevant to the spot instance interruption and reliability. To handle this issue, this paper thoroughly analyzes characteristics of various spot instance datasets and proposes a model to predict instant availability dataset to enhance reliability. To uncover the relationship of the publicly available dataset and the spot instance interruption behavior, we performed real-world experiments for spot instance interruptions in the AWS, Azure, and GCP cloud. We discovered that the Azure spot instance shows the highest reliability, followed by GCP and AWS. Though AWS showed the worst reliability, we discovered that the instant availability dataset offered by AWS can be helpful in predicting interruption events in the near future, which was not possible by using datasets offered by Azure and GCP. Finally, using the proposed instant availability score prediction, we showed that the median running time of spot instances can improve by 63.2% for initial high-score instances and by 168% for initial low-score spot instances. We believe that our study will enable users to use cloud resources more efficiently, ensuring reduced costs and increased reliability, thereby optimizing their overall system performance.

ACKNOWLEDGMENTS

This work is supported by the National Research Foundation (NRF) Grant funded by the Korean Government (NRF-2020R1A2C1102544), AWS Cloud Credits for Research program, and the SW Star Lab (RS-2022-00144309) of IITP.

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/10.1145/2509413.2509416

[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* (Phoenix, 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 Hyeokman Kim. 2018. Time-Series Analysis for Price Prediction of Opportunistic Cloud Computing Resources. In *Proceedings of the 7th International Conference on Emerging Databases*, Wookey Lee, Wonik Choi, Sungwon Jung, and Min Song (Eds.). Springer Singapore, Singapore, 221–229.

[4] Pradeep Ambati, David Irwin, Prashant Shenoy, Lixin Gao, Ahmed Ali-Eldin, and Jeannie 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] Mariette Awad and Rahul Khanna. 2015. *Efficient Learning Machines*. Apress. https://doi.org/10.1007/978-1-4302-5990-9

[6] Matt Baughman, 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

[7] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

[8] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32. https://doi.org/10.1023/A:1010933404324

[9] Sarah Chasins, Alvin Cheung, Natacha Crooks, Ali Ghodsi, Ken Goldberg, Joseph E Gonzalez, Joseph M Hellerstein, Michael I Jordan, Anthony D Joseph, Michael W Mahoney, et al. 2022. The Sky Above The Clouds. *arXiv preprint arXiv:2205.07147* (2022).

[10] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[11] Navraj Chohan, Claris Castillo, Mike Spreitzer, Malgorzata Steinder, Asser Tantawi, and Chandra Krintz. 2010. 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

[12] 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, 10–10. http://dl.acm.org/citation.cfm?id=1251254.1251264

[13] Nnamdi Ekwe-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/CCGRID.2018.00059

[14] Shereen Elsayed, Daniela Thyssens, Ahmed Rashed, Lars Schmidt-Thieme, and Hadi Samer Jomaa. 2021. Do We Really Need Deep Learning Models for Time Series Forecasting? *CoRR* abs/2101.02118 (2021). arXiv:2101.02118 https://arxiv.org/abs/2101.02118

[15] Javier Fabra, Joaquín Ezpeleta, and Pedro Álvarez. 2019. Reducing the price of resource provisioning using EC2 spot instances with prediction models. *Future Generation Computer Systems* 96 (2019), 348–367. https://doi.org/10.1016/j.future.2019.01.025

[16] Apache Software Foundation. 2004. Apache Hadoop. http://hadoop.apache.org/

[17] Weichao Guo, Kang Chen, Yongwei Wu, and Weimin 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

[18] Hårek Haugerud, Jonas Krüger Svensson, and Anis Yazidi. 2020. Autonomous Provisioning of Preemptive Instances in Google Cloud for Maximum Performance Per Dollar. In *2020 5th International Conference on Cloud Computing and Artificial Intelligence: Technologies and Applications (CloudTech)*. 1–8. https://doi.org/10.1109/CloudTech49835.2020.9365879

[19] 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

[20] 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/ICCCN.2019.8846933

[21] Bahman Javadi, Ruppa K. Thulasiram, and Rajkumar Buyya. 2013. Characterizing spot price dynamics in public 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.

[22] JCS Kadupitige, Vikram Jadhao, 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) *(HPDC '20)*. Association for Computing Machinery, New York, NY, USA, 41–52. https://doi.org/10.1145/3369583.3392671

[23] JCS Kadupitiya, Vikram Jadhao, and Prateek Sharma. 2022. SciSpot: Scientific Computing On Temporally Constrained Cloud Preemptible VMs. *IEEE Transactions on Parallel and Distributed Systems* 33, 12 (2022), 3575–3588. https://doi.org/10.1109/TPDS.2022.3157272

[24] E. L. Kaplan and Paul Meier. 1958. Nonparametric Estimation from Incomplete Observations. *J. Amer. Statist. Assoc.* 53, 282 (1958), 457–481. https://doi.org/10.1080/01621459.1958.10501452 arXiv:https://www.tandfonline.com/doi/pdf/10.1080/01621459.1958.10501452

[25] V. Khandelwal, A. Chaturvedi, and C. P. Gupta. 2017. Amazon EC2 Spot Price Prediction using Regression Random Forests. *IEEE Transactions on Cloud Computing* (2017), 1–1. https://doi.org/10.1109/TCC.2017.2780159

[26] Mikhail Khodak, Liang Zheng, Andrew S. Lan, Carlee Joe-Wong, and Mung Chiang. 2018. Learning Cloud Dynamics to Optimize Spot Instance Bidding Strategies. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 2762–2770. https://doi.org/10.1109/INFOCOM.2018.8486291

[27] KyungHwan Kim and Kyungyong Lee. 2024. Making Cloud Spot Instance Interruption Events Visible Artifact Dataset. https://doi.org/10.5281/zenodo.10633186

[28] KyungHwan Kim and Kyungyong Lee. 2024. Making Cloud Spot Instance Interruption Events Visible Artifact Source Code. https://doi.org/10.5281/zenodo.10652941

[29] Kyunghwan Kim, Subin Park, Jaeil Hwang, Hyeonyoung Lee, Seokhyeon Kang, and Kyungyong Lee. 2023. Public Spot Instance Dataset Archive Service. In *Companion Proceedings of the ACM Web Conference 2023* (Austin, TX, USA) *(WWW '23 Companion)*. Association for Computing Machinery, New York, NY, USA, 69–72. https://doi.org/10.1145/3543873.3587314

[30] K. Lee and M. Son. 2017. DeepSpotCloud: Leveraging Cross-Region GPU Spot Instances for Deep Learning. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*. 98–105. https://doi.org/10.1109/CLOUD.2017.21

[31] S. Lee, J. Hwang, and K. Lee. 2022. SpotLake: Diverse Spot Instance Dataset Archive Service. In *2022 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE Computer Society, Los Alamitos, CA, USA, 242–255. https://doi.org/10.1109/IISWC55918.2022.00029

[32] Markus Lumpe, Mohan Baruwal Chhetri, Quoc Bao Vo, and Ryszard Kowalcyk. 2017. On Estimating Minimum Bids for Amazon EC2 Spot Instances. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. 391–400. https://doi.org/10.1109/CCGRID.2017.76

[33] Aniruddha Marathe, Rachel Harris, David Lowenthal, Bronis R. de Supinski, Barry Rountree, and Martin Schulz. 2014. Exploiting Redundancy for Cost-Effective, Time-Constrained Execution of HPC Applications on Amazon EC2. In *Proceedings of the 23rd International Symposium on High-Performance Parallel and Distributed Computing* (Vancouver, BC, Canada) *(HPDC '14)*. Association for Computing Machinery, New York, NY, USA, 279–290. https://doi.org/10.1145/2600212.2600226

[34] Ishai Menache, Ohad Shamir, and Navendu Jain. 2014. On-demand, Spot, or Both: Dynamic Resource Allocation for Executing Batch Jobs in the Cloud. In *11th International Conference on Autonomic Computing (ICAC 14)*. USENIX Association, Philadelphia, PA, 177–187. https://www.usenix.org/conference/icac14/technical-sessions/presentation/menache

[35] Bruce D. Meyer. 1990. Unemployment Insurance and Unemployment Spells. *Econometrica* 58, 4 (1990), 757–782. http://www.jstor.org/stable/2938349

[36] Danielle Movsowitz Davidow, Orna Agmon Ben-Yehuda, and Orr Dunkelman. 2023. Deconstructing Alibaba Cloud's Preemptible Instance Pricing. In *Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing* (Orlando, FL, USA) *(HPDC '23)*. Association for Computing Machinery, New York, NY, USA, 253–265. https://doi.org/10.1145/3588195.3593001

[37] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[38] Dana Petcu. 2013. Multi-Cloud: Expectations and Current Approaches. In *Proceedings of the 2013 International Workshop on Multi-Cloud Applications and Federated Clouds* (Prague, Czech Republic) *(MultiCloud '13)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/2462326.2462328

[39] Thanh-Phuong Pham, Sasko Ristov, and Thomas Fahringer. 2018. Performance and Behavior Characterization of Amazon EC2 Spot Instances. In *2018 IEEE 11th International Conference on Cloud Computing (CLOUD)*. 73–81. https://doi.org/10.1109/CLOUD.2018.00017

[40] Gustavo Portella, Genaina N. Rodrigues, Eduardo Nakano, and Alba C.M.A. Melo. 2019. Statistical analysis of Amazon EC2 cloud pricing models. *Concurrency and Computation: Practice and Experience* 31, 18 (2019), e4451. https://doi.org/10.1002/cpe.4451 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/cpe.4451 e4451 cpe.4451.

[41] D. M. W. Powers. 2011. Evaluation: From precision, recall and f-measure to roc., informedness, markedness & correlation. *Journal of Machine Learning Technologies* 2, 1 (2011), 37–63.

[42] A. Sarah, K. Lee, and H. Kim. 2018. LSTM Model to Forecast Time Series for EC2 Cloud Price. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*. 1085–1088. https://doi.org/10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00067

[43] Prateek Sharma, Tian Guo, Xin He, David Irwin, and Prashant Shenoy. 2016. Flint: Batch-Interactive Data-Intensive Processing on Transient Servers. In *Proceedings of the Eleventh European Conference on Computer Systems* (London, United Kingdom) *(EuroSys '16)*. Association for Computing Machinery, New York, NY, USA, Article 6, 15 pages. https://doi.org/10.1145/2901318.2901319

[44] Prateek Sharma, David Irwin, and Prashant Shenoy. 2016. How Not to Bid the Cloud. In *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*. USENIX Association, Denver, CO. https://www.usenix.org/conference/hotcloud16/workshop-program/presentation/sharma

[45] Supreeth Subramanya, Tian Guo, Prateek Sharma, David Irwin, and Prashant Shenoy. 2015. SpotOn: A Batch Computing Service for the Spot Market. In *Proceedings of the Sixth ACM Symposium on Cloud Computing* (Kohala Coast, Hawaii) *(SoCC '15)*. Association for Computing Machinery, New York, NY, USA, 329–341. https://doi.org/10.1145/2806777.2806851

[46] ShaoJie Tang, Jing Yuan, and Xiang-Yang Li. 2012. Towards Optimal Bidding Strategy for Amazon EC2 Cloud Spot Instance. In *2012 IEEE Fifth International Conference on Cloud Computing*. 91–98. https://doi.org/10.1109/CLOUD.2012.134

[47] Sean J. Taylor and Benjamin Letham. 2018. Forecasting at Scale. *The American Statistician* 72, 1 (2018), 37–45. https://doi.org/10.1080/00031305.2017.1380080 arXiv:https://doi.org/10.1080/00031305.2017.1380080

[48] John Thorpe, Pengzhan Zhao, Jonathan Eyolfson, Yifan Qiao, Zhihao Jia, Minjia Zhang, Ravi Netravali, and Guoqing Harry Xu. 2023. Bamboo: Making Preemptible Instances Resilient for Affordable Training of Large DNNs. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 497–513. https://www.usenix.org/conference/nsdi23/presentation/thorpe

[49] P. Varshney and Y. Simmhan. 2019. AutoBoT: Resilient and Cost-Effective Scheduling of a Bag of Tasks on Spot VMs. *IEEE Transactions on Parallel & Distributed Systems* 30, 07 (jul 2019), 1512–1527. https://doi.org/10.1109/TPDS.2018.2889851

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[51] Cheng Wang, Qianlin Liang, and Bhuvan Urgaonkar. 2017. An Empirical Analysis of Amazon EC2 Spot Instance Features Affecting Cost-Effective Resource Procurement. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering* (L'Aquila, Italy) *(ICPE '17)*. Association for Computing Machinery, New York, NY, USA, 63–74. https://doi.org/10.1145/3030207.3030210

[52] Rich Wolski, John Brevik, Ryan Chard, and Kyle Chard. 2017. Probabilistic Guarantees of Execution Duration for Amazon Spot Instances. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (Denver, Colorado) *(SC '17)*. Association for Computing Machinery, New York, NY, USA, Article 18, 11 pages. https://doi.org/10.1145/3126908.3126953

[53] Ying Yan, Yanjie Gao, Yang Chen, Zhongxin Guo, Bole Chen, and Thomas Moscibroda. 2016. TR-Spark: Transient Computing for Big Data Analytics. In *Proceedings of the Seventh ACM Symposium on Cloud Computing* (Santa Clara, CA, USA) *(SoCC '16)*. Association for Computing Machinery, New York, NY, USA, 484–496. https://doi.org/10.1145/2987550.2987576

[54] Fangkai Yang, Bowen Pang, Jue Zhang, Bo Qiao, Lu Wang, Camille Couturier, Chetan Bansal, Soumya Ram, Si Qin, Zhen Ma, Íñigo Goiri, Eli Cortez, Senthil Baladhandayutham, Victor Rühle, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2022. Spot Virtual Machine Eviction Prediction in Microsoft Cloud. In *Companion Proceedings of the Web Conference 2022* (Virtual Event, Lyon, France) *(WWW '22)*. Association for Computing Machinery, New York, NY, USA, 152–156. https://doi.org/10.1145/3487553.3524229

[55] Fangkai Yang, Lu Wang, Zhenyu Xu, Jue Zhang, Liqun Li, Bo Qiao, Camille Couturier, Chetan Bansal, Soumya Ram, Si Qin, Zhen Ma, Íñigo Goiri, Eli Cortez, Terry Yang, Victor Rühle, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2023. Snape: Reliable and Low-Cost Computing with Mixture of Spot and On-Demand VMs. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 631–643. https://doi.org/10.1145/3582016.3582028

[56] Zongheng Yang, Zhanghao Wu, Michael Luo, Wei-Lin Chiang, Romil Bhardwaj, Woosuk Kwon, Siyuan Zhuang, Frank Sifei Luan, Gautam Mittal, Scott Shenker, and Ion Stoica. 2023. SkyPilot: An Intercloud Broker for Sky Computing. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. USENIX Association, Boston, MA, 437–455. https://www.usenix.org/conference/nsdi23/presentation/yang-zongheng

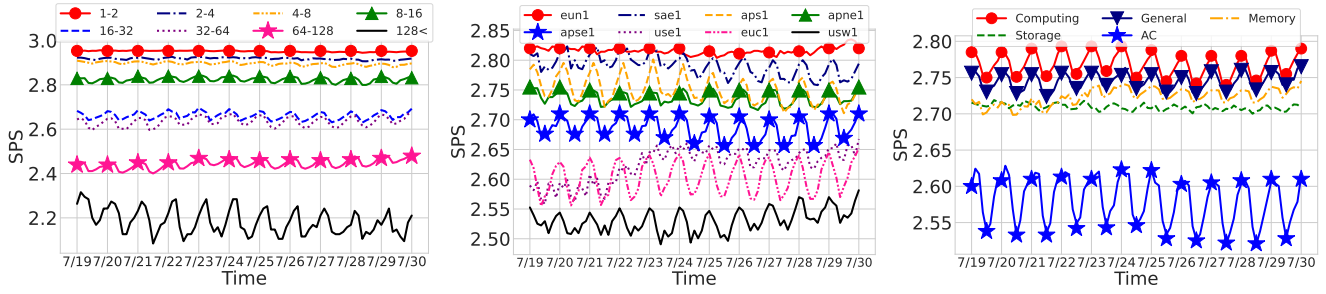
[57] Murtaza Zafer, Yang Song, and Kang-Won Lee. 2012. Optimal Bids for Spot VMs in a Cloud for Deadline Constrained Jobs. In *2012 IEEE Fifth International Conference on Cloud Computing*. 75–82. https://doi.org/10.1109/CLOUD.2012.59

[58] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauly, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2012. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. In *Presented as part of the 9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*. USENIX, San Jose, CA, 15–28.

[59] Chengliang Zhang, Minchen Yu, Wei Wang, and Feng Yan. 2022. Enabling Cost-Effective, SLO-Aware Machine Learning Inference Serving on Public Cloud. *IEEE Transactions on Cloud Computing* 10, 3 (2022), 1765–1779. https://doi.org/10.1109/TCC.2020.3006751

[60] Liang Zheng, Carlee Joe-Wong, Chee Wei Tan, Mung Chiang, and Xinyu Wang. 2015. How to Bid the Cloud. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 71–84. https://doi.org/10.1145/2785956.2787473

[61] Amelie Chi Zhou, Jianming Lao, Zhoubin Ke, Yi Wang, and Rui Mao. 2022. FarSpot: Optimizing Monetary Cost for HPC Applications in the Cloud Spot Market. *IEEE Transactions on Parallel and Distributed Systems* 33, 11 (2022), 2955–2967. https://doi.org/10.1109/TPDS.2021.3134644



(a) Categorization by CPU core numbers (b) Categorization by region (c) Categorization by instance category

Figure 9: SPS score during the observation period categorized by different criteria. SPS values show a noticeable distribution and pattern differences.

AWS				Azure			
Region	HH(%)	LL(%)	Ratio HH	Region	HH(%)	LL(%)	Ratio HH
ap-ne-1	48.8	7.1	0.87	ja-e	63.9	2.3	0.96
ap-ne-2	46.1	9.9	0.82	kr-c	75.2	0	1
ap-s-1	24.2	14.7	0.62	in-c	68.7	2.2	0.96
ap-se-1	8.9	17.1	0.34	ap-se	68.9	0.4	0.99
ca-c-1	48.0	7.7	0.86	ca-c	64.3	1.1	0.98
eu-c-1	22.2	20.8	0.51	uk-s	74.2	1.9	0.97
eu-n-1	48.8	3.3	0.93	eu-n	65.6	1.7	0.97
eu-w-3	43.1	9.9	0.81	fr-c	70.9	1.8	0.97
sa-e-1	36.4	14.0	0.72	us-c	69.7	2.6	0.96
us-e-1	2.2	50.6	0.04	us-e	35.1	16.1	0.68
us-w-2	2.4	24.3	0.09	us-e-2	32.5	15.8	0.67

Table 3: The percentage of Savings and Interrupt-free score combination for different regions.

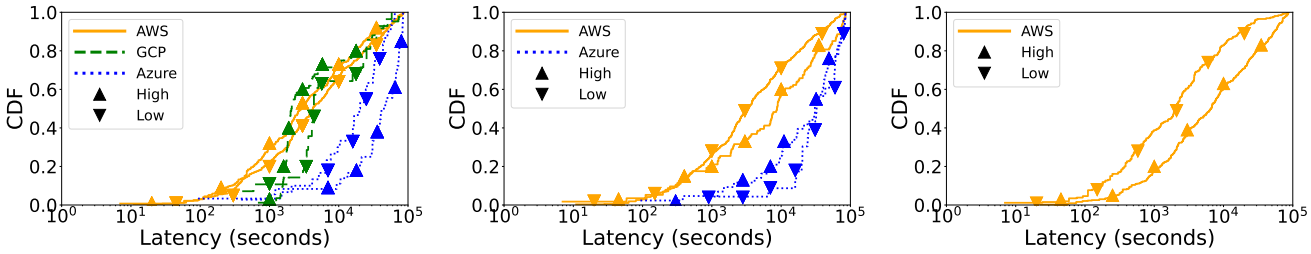
A ANALYSIS OF SPOT INSTANCE DATASET

The spot instance instant availability dataset, AWS SPS, provides crucial information on the reliability of spot instances. To further analyze the characteristics of the dataset, Figure 9 presents the temporal changes of the mean SPS value grouped by different criteria. In the sub-figures, the vertical axis shows the SPS value, and the horizontal axis presents the timestamp. Figure 9a presents the SPS value pattern with respect to the number of CPU cores. The number of cores is grouped into ones with more than 128 cores (solid black line) while decreasing the core numbers in half. From the figure, it is evident that the SPS score is inversely proportional to the number of CPU cores. For example, the mean SPS of when there are more than 128 cores is 2.1, while that of one or two CPU cores is 2.9. This result is consistent with the spot instance interrupt analysis experiments conducted for GCP [22, 55]. Figure 9b shows the mean SPS values for selected representative regions. We can discover the daily pattern for most regions with different scores. This result necessitates proper choice of regions to use spot instance reliably. Figure 9c groups the types of instances by categories. We can observe that the SPS in the accelerated computing category, which is presented with a solid line with star markers, shows lower values. It is an understandable situation that the recent popularity of deep neural networks, which requires significant computing power in the accelerated computing category, caused such a lack of idle resource, which is also presented by DeepSpotCloud [30].

We further analyze the data distribution of the interruption ratio and cost savings. To make the analysis beneficial to understanding the relation of two events, we focus on comparing the proportion of preferred to non-preferred cases. We first investigate the cases of high cost savings and high interrupt-free score, which a user prefers, and mark it as *HH*. In another case, both the cost savings and interrupt-free scores are low, a case that an user wants to avoid, and we mark it as *LL*. We mark savings larger than 70% as *High*. For the interrupt-free score, 2.5 and 3.0 are *High*, and 1.0 and 1.5 are *Low*. Table 3 shows the percentage of each case separated by AWS and Azure regions. Note that GCP does not provide an interruption ratio dataset. The *Ratio HH* column shows the ratio of *HH*, which is calculated as $\frac{HH}{HH+LL}$. The higher ratio means that more cases belong to the *HH* category, which is ideal. In AWS, the regions of Japan, Canada, and Stockholm show a high ratio. Meanwhile, the most of the regions in US show a lower ratio, especially US West 2 (Oregon) and US East 1 (N. Virginia), which are the biggest region in US. This result reflects the high resource demand in some regions. Comparing Azure and AWS, we can observe that Azure shows a much higher *Ratio HH*, which reflects more reliable spot instance operation with higher cost savings. This result coincides with the findings in 1a and 1b. One interesting observation of Azure is that the US east region shows a comparatively low ratio value.

B FURTHER ANALYSIS OF THE SPOT INSTANCE AVAILABILITY EXPERIMENTS

Figure 10 presents a distribution of spot instance running time after fulfillment categorized by different dataset. The vertical axis shows the distribution, and the horizontal axis shows a running time without interruption in log-scale. The larger x-axis value (to the right-side) implies more availability. Different sub-figures group instances to different criteria; cost savings ratio (Figure 10a), interrupt-free score (Figure 10b), and instant availability (SPS) score (Figure 10c). In each figure, the yellow-solid, blue-dotted, and green-dashed lines indicate the results of AWS, Azure, and GCP, respectively. In each line, we mark a symbol of $\triangle$ for *High* and $\triangledown$ for *Low* value after categorization.



(a) Cost Savings Ratio (b) Interrupt-free Score (c) Instant Availability (SPS) Score

Figure 10: A CDF of time until an interruption event happens for AWS, Azure, and GCP spot instances categorized by different dataset values of High (△) and Low (▽). A graph in the right side means higher availability.

Model	Train Time	Inference Time
LR	46	0.7
RF	75	0.9
XGBoost	806	1.2
SVC	626	15
Prophet	0.05	0.002
ARIMA	302	0.17
Transformer	2384	4.7

Table 4: Train and Inference Time (seconds) of a SPS prediction model

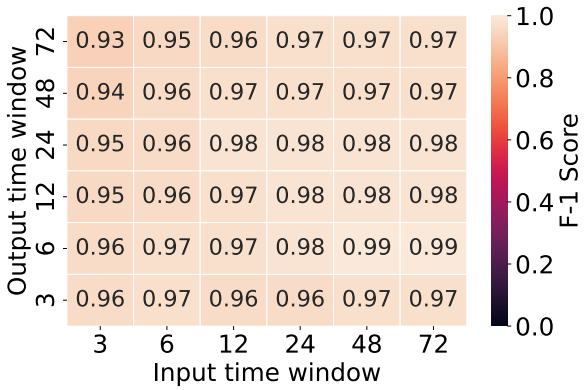


Figure 11: Heatmap of F1-score for XGBoost for modeling with respect to different input and prediction output time window

From the figure, we can observe that the cost savings ratio (Figure 10a) does not show a noticeable pattern between *High* and *Low* value for the running time of the spot instance. This observation concurs with the previous work that the price of the spot instance is not a good indicator of the reliability of the spot instance [6, 20]. We can also observe that Azure shows the longest running time than the other vendors followed by GCP and AWS. For the interrupt-free score (Figure 10b), the AWS spot instance shows a noticeable pattern that the higher interruption-free score is more reliable than the lower one. However, the Azure does not show such a pattern. For the instant availability dataset (Figure 10c), which is provided by AWS only (SPS), instances with high scores show much more reliability than lower ones. On average, spot instances with the score of 3 run for 4.7 hours, while that of the score 1 runs only for 1.8 hours.

C ANALYSIS OF SPS PREDICTION MODEL

We demonstrated that AWS SPS dataset is predictable with decent accuracy and recall. In SPS prediction, we applied multiple models and present the overhead of training and inference in Table 4. When measuring the time, we used the SPS dataset gathered between July 24th. and July 31st. 2023. For inference, we used the dataset from August 1st. to August. 3rd. For the time measurement, we used a dataset from a single instance type for comparison. We can observe a quite difference in training time due to the model's complexity, especially the Transformer [50]. We tried various Transformer optimization heuristics. However, considering the huge training overhead and lower prediction quality, it does not seem to be an appropriate approach to deal with time-series dataset, which coincides with the result presented by Elsayed et. al. [14]. The Prophet shows a very short training time but shows a lower prediction quality compared to XGBoost, especially when the data changes abruptly, as presented in Table 2. The train time difference is owing to the fact that XGBoost and other machine learning classifiers build separate models for each different output, which is the number of output windows, 432. Meanwhile, Prophet builds a single model based on statistics of train dataset, and train time did not get impacts from the output time window size. Considering that the train and inference can occur off-line, we argue that using XGBoost for a SPS value prediction service is appropriate.

Figure 11 shows the heatmap of F1-score of a SPS prediction model built using XGBoost with respect to different input time window (the horizontal axis) and the different output time window (the vertical axis) presented in hours. In the figure, lighter region implies higher F1-score. As we can see from the figure, regardless of the input and output time window, the F1-score shows consistently high scores except when predicting further future ($72H$) using only recent values ($3H$).

All datasets and source codes for the figures in this article are available as an artifact [27].