

Cross-Vendor Multi-Node Spot Instance Dynamics: A Public Cloud Data Archive Service and Analysis

Jaeil Hwang^{a,1}, Taeyoon Kim^{b,2}, Kyungyong Lee^{b,3,*}

^aDepartment of Artificial Intelligence, Hanyang University, Seoul, 04763, South Korea

^bDepartment of Data Science, Hanyang University, Seoul, 04763, South Korea

Abstract

Cloud spot instances trade unpredictable interruption for substantial price savings, and understanding the availability pattern is a central concern in using them. Existing work largely covers single-node availability on a single vendor and has not characterized multi-node provisioning dynamics, nor has it tested whether vendor-provided availability scores actually predict real outcomes. This paper aims to close both gaps through systematic cross-vendor data collection, multi-node availability analysis, and real-world provisioning validation. To this end, a collection system is first built to continuously monitor spot instance availability across major cloud vendors, with vendor-specific query-optimization algorithms reducing query overhead to enable practical long-term collection within current API rate limits. Using the resulting cross-vendor dataset, the characteristics of multiple spot metrics across vendors are analyzed, revealing that price and interruption frequency act as long-term stability signals, while real-time availability metrics track local supply and demand cycles at both single-node and cluster scale. Fitting a degradation model on both vendors further exposes a structural asymmetry in which availability decreases rapidly with cluster size on AWS while showing minimal degradation and a higher availability floor on Azure. To verify whether the collected metrics actually represent real spot instance provisioning behavior, over 215,000 real provisioning attempts are conducted on both vendors, confirming that the same availability score carries different operational meaning on different providers. The collected dataset and the collection system are released through a public webpage and open-source repository.

Keywords: Spot Instance, Cloud Computing, Multi-node Availability, Dataset Archive, Multi-cloud Systems

1. Introduction

Cloud computing transformed resource utilization paradigms with its on-demand, pay-as-you-go billing model. This model permits organizations to dynamically scale resources without significant upfront hardware investments. Driven by the flexibility of paying only for consumed resources, enterprises have increasingly migrated from traditional on-premises infrastructure to cloud services. To optimize the utilization of idle capacity and offer substantial price reductions, public cloud vendors provide surplus computing resources as spot instances. These instances present a critical trade-off of significant price discounts and possibility for sudden instance interruptions.

Spot instances initially operated on a market-driven auction mechanism, requiring users to manage the inherent volatility and risk of sudden terminations. Cloud vendors provided spot price history as the primary dataset for

predicting availability to help users manage the trade-off between price savings and potential interruptions. Volatile spot price changes triggered extensive research [1, 2, 3, 4, 5, 6] on price prediction models, optimal bidding strategies [7, 8, 9], and interruption mitigation techniques across domains including big data processing [10, 11, 12], deep learning [13], and batch processing [14, 15]. However, a pricing operation policy change stabilized spot prices and reduced volatility [16, 17], making most of this price-based literature obsolete [18].

This paradigm shift necessitated new availability metrics, and cloud vendors introduced real-time availability scores and historical interruption statistics to replace price-based predictions. Recent research efforts have systematically collected and archived these metrics to serve them as a web-service [19, 20]. Although these new datasets offer valuable spot instance availability information, existing systems primarily focus on single-node availability metrics and analysis for a single cloud vendor [21], leaving multi-node provisioning dynamics in multi-cloud environments largely unexplored.

These single-node availability insights are insufficient for production workloads that operate in multi-node cluster environments, where high single-node availability does not guarantee successful provisioning of large clusters [22]. AI-

*Corresponding author

Email addresses: jaeil0328@hanyang.ac.kr (Jaeil Hwang), tykim7@hanyang.ac.kr (Taeyoon Kim), kyungyong@hanyang.ac.kr (Kyungyong Lee)

¹ORCID: 0000-0002-6640-795X

²ORCID: 0009-0003-0705-0713

³ORCID: 0000-0003-0312-4386

though this gap has been demonstrated on a single vendor, whether the same degradation pattern holds across vendors, and whether the degree of degradation differs structurally, remains an open question. Major cloud providers have recently introduced multi-node availability metrics [23, 24] that enable such cross-vendor studies, but answering this question requires cross-vendor multi-node collection, which introduces additional challenges. The query space grows by orders of magnitude and each vendor imposes different API rate limits that make comprehensive monitoring impractical [19].

Building a cost-effective, large-scale spot instance cluster across multiple cloud vendors requires a thorough understanding of how availability behaves at multi-node scale and whether vendor-provided metrics reliably guide resource selection. Moreover, whether these vendor-provided availability metrics actually predict real provisioning outcomes has not been independently validated, and scoring or recommendation systems built on these metrics inherit any unverified assumptions about their predictive power. To that end, this paper addresses the following research questions.

- RQ-1 What are the practical challenges of collecting spot instance availability data across multiple vendors, and what collection strategies are needed to assemble an accurate, long-term multi-vendor dataset within current API constraints?
- RQ-2 How do spot instance cost efficiency and availability characteristics vary across vendors, regions, and instance categories?
- RQ-3 Does spot instance availability show specific temporal change patterns? If so, how do these patterns differ across vendors?
- RQ-4 How does multi-node spot instance availability degrade as cluster size increases, and what are the cross-vendor differences?
- RQ-5 Do vendor-provided availability metrics reliably predict actual spot instance provisioning success?

To answer these questions, we build a serverless collection system that continuously monitors spot instance availability on AWS and Azure, with supplementary price collection on GCP. The proposed system continuously monitors and stores historical availability patterns, establishing the first long-term, cross-vendor dataset for multi-node spot instances (RQ-2). To enable practical data collection within strict API constraints, vendor-specific optimization algorithms reduce query requirements by over 60% through intelligent sampling and aggregation strategies. The system also employs an efficient representation method that captures cluster-level availability degradation patterns (RQ-4) while significantly reducing storage and monitoring overhead (RQ-1).

Using extensive data continuously collected from the proposed system, systematic cross-vendor analysis reveals fundamental differences in how availability degrades as cluster size increases, with some providers maintaining consistent performance across varying node counts while others show rapid degradation in large-scale deployments (RQ-4). Temporal analysis further reveals provider-specific fluctuations between peak and off-peak periods (RQ-3), and examines dataset update characteristics that affect monitoring strategies (RQ-1). To verify whether these collected metrics actually represent real provisioning behavior, real-world experiments with over 215,000 attempts are conducted on both providers, confirming that the same availability score carries different operational meaning on different providers (RQ-5). These findings demonstrate that data-driven resource selection is crucial for optimizing multi-cloud deployments. To support both researchers and practitioners, this work provides a publicly accessible dataset archive service⁴ with comprehensive historical data across multiple cloud platforms, enabling informed decision-making for large-scale spot instance usage.

The core contributions are summarized as follows.

- *Implementation of vendor-specific query-optimization algorithms* that reduce Spot Placement Score (SPS) query budgets and enable long-term multi-node collection within current per-account API rate limits (RQ-1).
- *Construction and public release of the first long-term cross-vendor multi-node spot availability dataset*, hosted through the SpotLake archive service for continuous reanalysis (RQ-1, RQ-2).
- *Discovery of a structural cross-vendor asymmetry in multi-node availability degradation*, captured by an inverse model fitted to multi-node SPS as a function of target cluster size on both vendors and observed again in independent provisioning experiments (RQ-3, RQ-4).
- *First real-world validation of vendor-provided availability scores* across both AWS and Azure at single-node and multi-node scale, including the resulting 215,000-attempt provisioning dataset against which any future analysis of these scores can be checked (RQ-5).

2. Spot Instance Mechanisms and Datasets

Public cloud vendors offer surplus computing resources at reduced prices through spot instances, including AWS Spot Instances, Azure Spot Virtual Machines (VMs), and Google Cloud Spot VMs. Providers can reclaim capacity at any time, so users trade reliability for price savings [2].

⁴<https://spotlake.ddps.cloud/>

Cloud vendors provide three categories of datasets to help users manage this trade-off. First, *spot price history* reflects the cost of spot instances over time. AWS provides up to three months of price change history through its CLI/API, and Azure offers current pricing through REST API endpoints. Second, *interruption frequency (IF) statistics* indicate the historical rate of interruptions over the preceding month. AWS provides this data through a web-based management console, while Azure offers API-based access. Third, *spot placement scores (SPS)* provide real-time availability assessments [23]. AWS’s SPS takes desired instance types, regions, and target capacity as arguments and returns a score from 1 to 3 [25], with higher scores indicating greater likelihood of successful provisioning. Azure’s SPS provides analogous functionality, returning categorical values [24] (*High, Medium, Low*) that can be mapped to a 1–3 numerical scale. Both vendors support multi-node queries, enabling assessment of cluster-level provisioning likelihood. Google Cloud does not provide IF or SPS datasets, limiting cross-vendor analysis beyond price comparisons to AWS and Azure.

Table 1 summarizes the availability of these datasets across vendors.

Table 1: Spot instance dataset availability across cloud vendors.

Dataset	AWS	Azure	GCP
Price	CLI/API	REST API	REST API
Interruption	Web only	API-based	Not provided
Availability	CLI/API	REST API	Not provided

2.1. Related Work

Table 2 summarizes how prior work covers each axis. Cross-vendor multi-node analysis, end-to-end validation of vendor-provided availability scores, and the query-optimization machinery needed to collect such a dataset within current API limits are all absent from the prior literature.

Table 2: Comparison of existing spot instance studies.

Study	Price anal.	Multi-vendor	Multi-node	Real-world valid.	Query optim.
Ben-Yehuda et al. [2]	✓				
Javadi et al. [3]	✓				
Kim et al. [21]					✓
Pham et al. [26]				✓	
Lee et al. [19]				✓	✓
Cheon et al. [22]			✓		
This work	✓	✓	✓	✓	✓

Spot Price Analysis. The spot price dataset has been extensively analyzed for statistical characterization [1, 2, 3, 4, 5, 6, 27] and optimal bidding [7, 8, 9, 28, 29]. AWS later changed its internal spot pricing policy in 2017 [16, 18, 17], after which spot prices became stable and these prior price-based analyses lost their applicability.

Modeling Spot Instance Availability. Kadupitige et al. [30] proposed a statistical interruption model for Google Cloud Spot VMs through empirical experiments. Pham et al. [26] empirically analyzed AWS spot instance availability using interruption frequency data. Yang et al. [31] developed a spatial-temporal eviction prediction model for Azure. However, these studies focused exclusively on single-node availability metrics for individual vendors, leaving multi-node provisioning dynamics and cross-vendor comparisons unexplored (RQ-2, RQ-4).

Dataset Collection and Archive Services. Kim et al. [21] and Lee et al. [19] developed dataset archive services for AWS spot instances, providing historical data for single-node metrics. Cheon et al. [22] implemented multi-node SPS collection for AWS, demonstrating the importance of cluster-scale metrics. These efforts are limited to single-vendor data collection and lack empirical validation of whether collected metrics predict real-world provisioning outcomes (RQ-1, RQ-5).

Applications Using Spot Instances. A range of systems use spot instances for deep learning [13], big data processing [10, 32, 33], batch jobs [14, 15], online services [34, 35], and HPC workloads [27, 36]. ExoSphere [37] proposed a portfolio approach for transient servers. These application-side studies all assume the underlying availability characterization that this work supplies in the multi-node and cross-vendor setting.

3. Data Collection Methodology

Systematic collection of spot instance datasets faces three categories of challenges, summarized in Table 3. First, the absence of historical data for most metrics requires continuous real-time collection. While AWS offers up to three months of spot price history, neither AWS nor Azure provides historical data for availability scores or interruption ratios. Second, restrictive query constraints limit data throughput. AWS restricts unique SPS queries to 50 per account per day, and Azure allows only 10 calls per hour per subscription-location pair. Third, disparate access interfaces across platforms complicate unified collection. For example, AWS provides interruption data only through a web console, while Azure offers API-based access.

Table 3: Query constraints and historical data availability for AWS and Azure spot datasets.

Dataset	Query Constraints		Historical Data
	AWS	Azure	
Price	None	Pagination required	3 months
Interruption	None	Standard API limits	None
Availability (SPS)	50 scenarios/day	10 calls/hr/location	None

For multi-node collection, the query space grows dramatically. Systematic collection for node counts from 1 to

50 across all instance types and availability zones (AZs) would require approximately 192,700 unique queries on AWS alone, necessitating over 3,854 accounts under current API limits.

3.1. Query Optimization Algorithms

The collection challenges above admit an algorithmic solution. A fixed query budget can be exploited by carefully packing each call so that it returns the maximum amount of useful information. AWS and Azure expose different rate-limit shapes, however, so the packing problem takes a different form on each vendor and must be formulated separately.

3.1.1. Single-Node Query Optimization

For AWS, the optimization problem is modeled as a bin-packing problem [38]. Regions are treated as items, with the weight of each region being the number of AZs supporting a specific instance type. The bin capacity is constrained by the API's limit of 10 placement scores per query. This is solved using a Mixed-Integer Programming approach with the COIN-OR CBC solver [39] via Google's OR-Tools [40]. Figure 1 illustrates this optimization with the *p5.48xlarge* instance type as a worked example. The left side of the figure lists the regions that support *p5.48xlarge* along with the number of availability zones each region exposes for this instance type. A naive collector would issue one query per region, returning fewer than 10 placement scores most of the time and consuming the per-account query budget faster than necessary. The bin-packing solver instead groups multiple regions into a single query whenever their combined AZ count fits within the per-call limit of 10, so that each query is densely packed and the total number of queries needed to cover all regions drops accordingly.

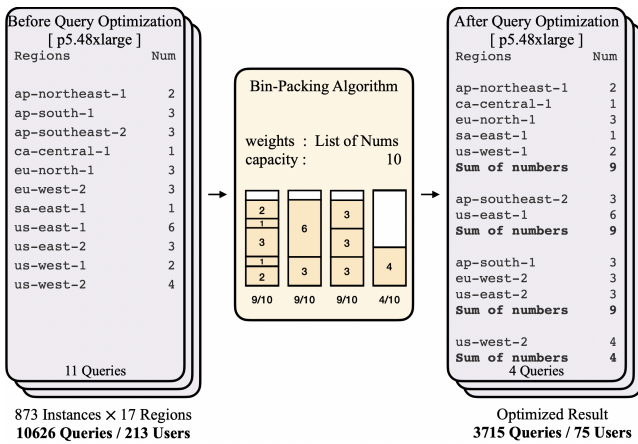


Figure 1: AWS query optimization using bin-packing.

For Azure, where each API call supports up to 5 instance types and 8 regions and returns SPS values for every (instance type, region) combination drawn from the chosen sets (the cartesian product of the two parameters, yielding up to 40 pairs per call), the problem is formulated as

a cardinality-constrained Set Cover Problem [41, 42]. A naive approach of arbitrarily combining regions and instance types in a query would waste call capacity on unsupported combinations, since the availability of specific instance types varies across regions. A greedy heuristic instead iteratively constructs each query by selecting the node (region or instance type) providing maximum incremental coverage of un-queried region-instance pairs, expanding the query until either no further pairs can be covered by adding a new node or the per-call limits of 8 regions and 5 instance types are reached. Figure 2 shows a single step in this iterative construction. The depicted intermediate state has a query that already includes regions $\{R0, R1\}$ and instance types $\{I0, I1, I2\}$, and the algorithm evaluates the incremental coverage gain of every adjacent un-included node. Adding region $R2$ would cover 3 new pairs, while adding either instance type $I3$ or region $R3$ would cover only 1 new pair each. The greedy step therefore picks $R2$ for inclusion in the query, since it offers the largest immediate gain. Ties between candidates of equal incremental coverage are resolved by preferring instance types over regions, and remaining ties by evaluation order.

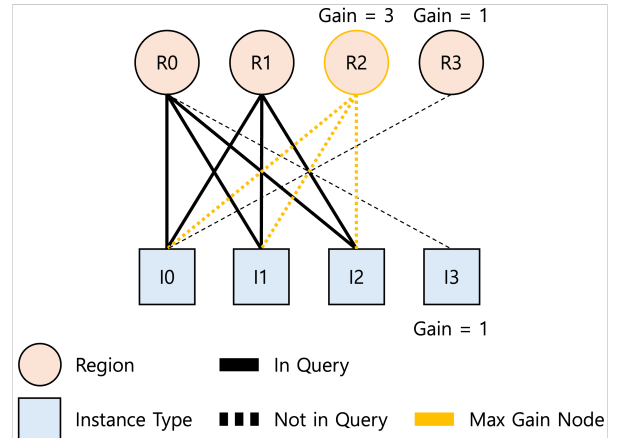


Figure 2: Azure query optimization using greedy set cover.

3.1.2. Multi-Node Sampling Strategy

Exhaustive enumeration of all node counts is infeasible under current API constraints. The collection system extends the single-node optimization above by sampling one target node count per cycle in round-robin order (1, 5, 10, ..., 50) and estimating non-sampled points through an update propagation heuristic based on the monotonicity assumption $SPS(n) \geq SPS(n+1)$. When a query for target count T_c returns score S_{new} , all stored scores for $T_i \geq T_c$ are updated as $SPS[T_i] \leftarrow \min(SPS[T_i], S_{new})$. Together, periodic sampling and monotone-consistent propagation capture availability transitions while operating within practical API query limits.

3.1.3. Compact Representation of Multi-Node SPS

The monotonically decreasing property of SPS admits a compact representation using two transition thresholds. $T3$ is the maximum node count maintaining $SPS = 3$, and $T2$ is the maximum node count maintaining $SPS \geq 2$. These two values reconstruct the full availability profile. The score is 3 if $n \leq T3$, 2 if $T3 < n \leq T2$, and 1 if $n > T2$.

3.2. System Implementation

The collection system, SpotLake, uses a serverless architecture as illustrated in Figure 3. The *Data Collector* executes scheduled jobs via AWS EventBridge, using AWS Batch with spot instances for cost efficiency. A daily query planner at 00:00 UTC retrieves current instance-type-to-region mappings and allocates optimized query scenarios across accounts (AWS) or subscriptions (Azure) using the algorithms described above. Each account is assigned a fixed slice of the query plan in round-robin order so that no account exceeds its per-day quota. The collector employs round-robin sampling across target node counts (1, 5, 10, ..., 50). The *Data Storage Backend* writes each completed cycle’s results first to S3 (the system of record) and then to a time-series database used for low-latency queries. The time-series store is treated as a derived view that can be rebuilt from S3. The *Web Interface*⁵ provides visualization and query capabilities through a static web application.

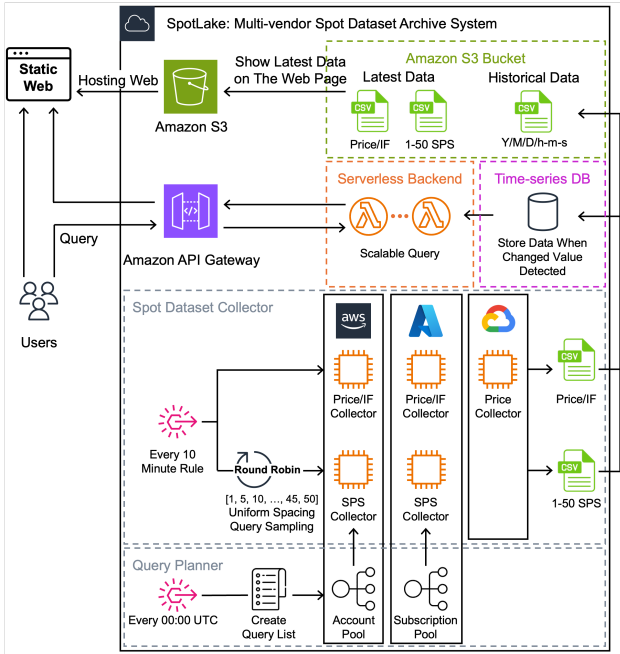


Figure 3: SpotLake system architecture.

4. Dataset and Metrics

This section defines the dataset scope, metric definitions, and temporal analysis convention used throughout the subsequent analysis.

Dataset Scope. The primary analysis uses data collected during June 2025 at 10-minute sampling intervals. The AWS dataset covers 17 regions, 923 instance types, and 55 AZs. The Azure dataset comprises 61 regions, 1,350 instance types, and 129 AZs. GCP price data covers 40 regions and 596 instance types but can only be used for spot price comparisons, since GCP does not provide IF or SPS metrics.

Seven representative regions per vendor, spanning major geographic areas (Table 4), are used for spatial and temporal analyses. This selection balances geographic diversity with analytical depth, covering North America, Europe, and Asia-Pacific. Where the two vendors’ cities differ, both names label the pair. Every pair shares a city except Oregon / Washington, where West US 2 is Azure’s geographically closest region.

Metric Definitions. The categorical IF data are converted to a numerical IF score for consistent analysis. The lowest interruption frequency (less than 5%) maps to 3.0, and the highest (more than 20%) maps to 1.0, with intermediate categories at 2.5, 2.0, and 1.5. Azure’s categorical SPS values (*High*, *Medium*, *Low*) are mapped to numerical scores (3, 2, 1). Instance types are classified into five categories (General, Compute, Memory, Accelerated, and Storage) based on vendor-specific naming conventions.

Non-Availability Response Handling. Azure SPS queries return, in place of a score, one of two structured non-availability responses. **RestrictedSkuNotAvailable** is returned when the Spot SKU is not available for the querying subscription, and **DataNotFoundOrStale** is returned when the data needed to generate a score is missing or stale [24]. Because both responses indicate an access restriction or missing data rather than low capacity, all SPS averages and model fits in the subsequent analysis use valid scores only. Records carrying a non-availability response are excluded, pairs with no valid response during the measurement period do not enter the analysis dataset, and the raw responses are preserved in the archive.

Temporal Analysis Convention. All temporal analyses (hourly and daily patterns) are presented in each region’s local time with daylight saving time adjustments applied for June 2025 (e.g., US East at UTC−4 for EDT, EU Central at UTC+2 for CEST, AP Northeast at UTC+9 for KST). Since all seven representative regions are analyzed together, the reported patterns reflect demand cycles aligned with local business hours in each region. Values represent averages over all 10-minute samples during the measurement period unless otherwise noted.

⁵<https://spotlake.ddps.cloud/>

Table 4: Representative regions for spatial and temporal analyses. City names are used as figure labels.

Geography	City (Label)	AWS Region	Azure Region
N. America (East)	Virginia	us-east-1	East US
N. America (West)	Oregon / Washington	us-west-2	West US 2
Europe (West)	Ireland	eu-west-1	North Europe
Europe (Central)	Frankfurt	eu-central-1	Germany West Central
Asia (Southeast)	Singapore	ap-southeast-1	Southeast Asia
Asia (East)	Tokyo	ap-northeast-1	Japan East
Asia (Northeast)	Seoul	ap-northeast-2	Korea Central

5. Collection System Validation

The collection system that produced the June 2025 single-node dataset described above is evaluated along three axes, namely query optimization effectiveness, operational overhead, and dataset quality.

Query Optimization Effectiveness. Table 5 compares the number of API queries required with and without optimization. In the naive approach, each instance type queries each supported region separately, yielding 10,875 queries for AWS and 4,405 for Azure. The bin-packing optimization for AWS groups multiple regions into single queries by packing AZ counts into bins of capacity 10, reducing queries to 3,875 (64.4% reduction). The greedy set cover for Azure maximizes the coverage of valid region-instance pairs per API call, achieving a 75.1% reduction to 1,098 queries.

Table 5: Query count comparison for single-node SPS collection.

Vendor	Naive	Optimized	Reduction
AWS	10,875	3,875	64.4%
Azure	4,405	1,098	75.1%

Operational Overhead. Each collection cycle completes in approximately 90 seconds on average for both vendors, well within the 10-minute collection interval. The collection pipeline uses AWS Batch with spot instances to minimize compute overhead. Per-query failures (HTTP errors, transient API failures, throttling) are retried within the same cycle with bounded exponential backoff. Cycle-level failures (e.g., the planner itself failing) are retried as part of the next 10-minute cycle, and the most recent successful snapshot per (instance type, region, AZ) is preserved in S3 so that downstream analyses always read from a consistent state.

Dataset Quality. AWS SPS values all fall within the defined $\{1, 2, 3\}$ range, and Azure’s structured non-availability responses (`RestrictedSkuNotAvailable`, `DataNotFoundOrStale`) are stored unchanged, accounting for 45.8% of Azure records (45.6 percentage points

from `RestrictedSkuNotAvailable`, concentrated in special hardware categories and large regions, and 0.2 percentage points from `DataNotFoundOrStale`) and available for downstream analyses to filter or interpret as appropriate. Per-(instance, region, AZ) snapshot counts cluster tightly around the expected 4,320 snapshots over the 30-day month, with the AWS modal value at 4,292 and the Azure median at 4,302.

Propagation Fidelity. Because multi-node collection queries one node count per cycle, a stored profile mixes observations taken up to one rotation apart, and an apparent inversion in that record can reflect either a genuinely non-monotone response or a real capacity change between staggered queries.

Two reference streams separate the two. The first is an Azure stream that queried all eleven node counts within each collection cycle for six instance types across all regions (March through December 2025). The second is an AWS probe that queried six target capacities between 5 and 50 within each cycle for 37 instance types across six regions (January 2024). Measured within a single cycle, non-monotone responses are rare, with one adjacent-count inversion among 2.5 million comparisons on AWS, and 1.7% of profiles on Azure, 98% of them by a single score step, while the round-robin record read at the same instants shows apparent inversions in 0.21% and 11.1% of profiles on the identical data. Most apparent non-monotonicity is therefore an artifact of staggered sampling rather than a property of the scores. The rates measured within a single cycle are themselves upper bounds, since the queries of one cycle are issued in parallel within a short window.

Table 6 scores the candidate reconstruction rules cell by cell against the same-cycle profiles, reporting the share of stored cells that differ from the reference and the mean signed difference in SPS points. Min-propagation is the closest to the reference on both vendors, and its residual error is almost entirely downward, so the stored profile can understate but rarely overstate the observed score (cells recorded above the reference fall from 1.9% to 0.1% on Azure). Isotonic regression yields no gain on either vendor and produces fractional scores that the three-level scale cannot represent, so the one-sided min rule is retained.

Table 6: Reconstruction rules scored against the same-cycle reference profiles.

Rule	Azure (2025-06)		AWS (2024-01)	
	Mismatch	Signed	Mismatch	Signed
Raw record (no update)	3.69%	+0.0001	1.63%	+0.0000
Min-propagation	2.72%	-0.026	1.62%	-0.0002
Isotonic (fractional)	7.35%	+0.0002	1.66%	+0.0000
Isotonic (integer)	3.73%	-0.0004	1.64%	+0.0001

6. Characterizing Dataset Volatility

With the collection system validated and the dataset quality confirmed, the remaining practical question is how frequently the system must sample each metric to avoid missing meaningful changes. This section addresses RQ-1 by analyzing the temporal dynamics of metric updates to inform optimal collection strategies.

6.1. Metric Change Frequency

Update patterns are characterized using two complementary metrics. Let P denote the set of valid instance-AZ pairs and T the set of 10-minute observation intervals over 30 days ($|T| = 4,320$). *Pair-level Coverage* measures the fraction of pairs that changed at least once:

$$\text{Coverage} = \frac{\sum_{p \in P} \mathbf{1}_{\text{changed}}(p)}{|P|} \times 100\% \quad (1)$$

Temporal Change Rate measures how frequently changes occur:

$$\text{Change Rate} = \frac{\sum_{t \in T} \sum_{p \in P} \mathbf{1}_{\text{diff}}(p, t)}{|T| \times |P|} \times 100\% \quad (2)$$

where $\mathbf{1}_{\text{diff}}(p, t) = 1$ if and only if pair p has a valid observation at interval t whose value differs from the most recent preceding valid observation of the same pair. Any change of the recorded price counts for spot price, with no minimum threshold, and any change of level counts for the categorical metrics IF, SPS, and $T3$. Intervals with a missing observation contribute no change. $\mathbf{1}_{\text{changed}}(p) = 1$ if and only if $\mathbf{1}_{\text{diff}}(p, t) = 1$ for at least one t .

Table 7 shows large cross-vendor differences. On AWS, spot price has the highest coverage (99.7%) but a change rate of only 2.5%: most pairs change at some point, but rarely. Azure SPS has the highest change rate (5.8%) and is the most volatile metric. These differences suggest distinct monitoring roles: SPS and $T3$ as real-time indicators of current availability, IF score as a long-term stability assessment.

6.2. Change Interval Distribution

Figure 4 shows the distribution of time intervals between consecutive changes. Azure SPS and $T3$ show 40% of intervals concentrated at 10 minutes, indicating rapid consecutive updates. IF score shows the longest intervals on both platforms, with most beyond 1,000 minutes, confirming its nature as a slowly evolving long-term metric.

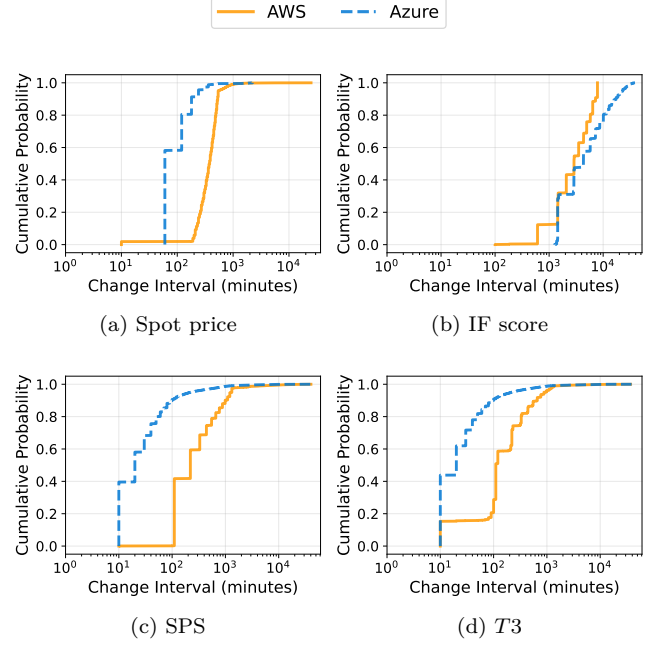


Figure 4: Cumulative distribution of change intervals for each metric on AWS and Azure.

6.3. Impact of Collection Interval on Data Quality

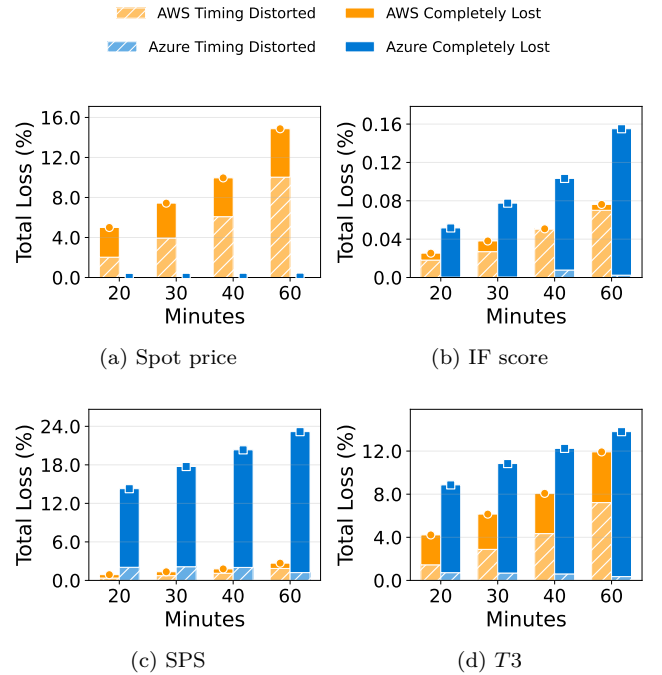


Figure 5: Data loss relative to 10-minute baseline at extended sampling intervals.

Figure 5 quantifies data loss at extended sampling intervals. Azure SPS is the most sensitive, with 14.3% loss at 20-minute intervals and 23.2% at 60 minutes. IF score tolerates extended intervals on both platforms with loss below 0.2%.

Table 7: Dataset volatility over June 2025 with 10-minute sampling.

Provider	Spot Price		IF score		SPS		T3	
	Cov.(%)	Rate(%)	Cov.(%)	Rate(%)	Cov.(%)	Rate(%)	Cov.(%)	Rate(%)
AWS	99.7	2.542	37.0	0.013	38.2	0.451	80.5	2.277
Azure	18.3	0.006	13.9	0.008	54.6	5.759	34.2	3.633

The loss type distribution shows vendor-specific behaviors. On AWS, *Timing Distorted* cases dominate, consistent with gradual metric changes, while on Azure *Completely Lost* cases prevail, consistent with abrupt fluctuations. These findings establish concrete collection guidelines. Azure SPS requires collection intervals of 10 minutes or less, while AWS SPS and IF score tolerate up to 30-minute intervals with less than 5% data loss.

6.4. Temporal Consistency of Dataset Dynamics

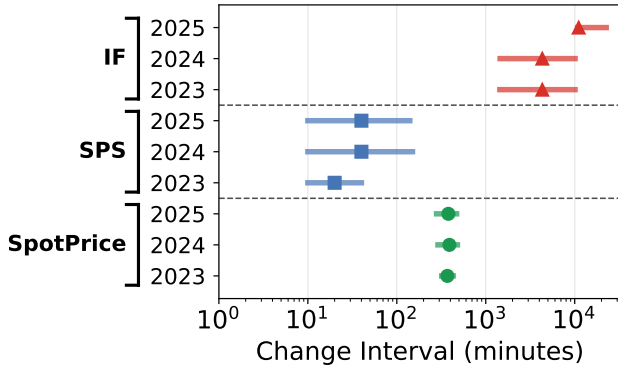


Figure 6: Change interval distributions across January of 2023, 2024, and 2025 for AWS metrics.

Figure 6 examines whether update patterns are stable over time using AWS data from January of 2023, 2024, and 2025. Spot price shows consistent patterns across years, while SPS and IF score exhibit significant year-to-year shifts. These variations indicate that metric change intervals are not static, and collection systems should incorporate adaptive mechanisms to maintain data quality as vendor behaviors evolve.

7. Analysis of Single-Node Spot Datasets

With the collection methodology, system validation, and sampling strategy established, the remainder of this paper turns to analyzing the collected dataset. This section addresses RQ-2 and RQ-3 by analyzing the spatial and temporal characteristics of single-node spot instance datasets across vendors, establishing the baseline for multi-node analysis.

7.1. Cross-Vendor Price Comparison

Figure 7 compares price savings across vendors and instance categories. Azure consistently offers the highest median savings in every category. GCP exceeds AWS in four of the five categories, while their Accelerated Computing medians are nearly equal (66.1% and 66.3%). AWS exhibits substantially wider variance, with numerous outlier instances offering lower savings, necessitating careful instance selection on AWS. Despite high demand for GPU hardware, the Accelerated Computing category does not show a markedly different savings pattern from other categories. These vendor-level differences prompt a finer-grained question of whether the same patterns hold at the regional and availability-zone level.

7.2. Spatial Characteristics

Regional and AZ-Level Price Variations. Figure 8 shows that AWS has the widest inter-regional variance in price stability, with standard deviation ranging from about 7% in *eu-west-1* to nearly 15% in *eu-central-1*. Azure and GCP are more uniform across regions, with regional standard deviations spanning 2.9 and 1.8 percentage points versus 8.0 for AWS. Azure and GCP also keep spot prices uniform across AZs within a region, whereas AWS prices vary across AZs by 9.1% on average and by up to 91% in extreme cases (Figure 9).

IF and SPS Spatial Distribution. Figures 10 and 11 present heatmaps of IF and SPS across instance categories and regions. For both vendors, Storage-optimized instances have the highest IF scores (AWS: 2.09, Azure: 2.48). On AWS, Accelerated Computing has the lowest, consistent with high demand for GPU instances, while Azure returns no valid IF values for that category. The two vendors also differ sharply in data coverage. AWS provides valid IF scores for 99.8% of instance-AZ pairs versus Azure’s 31.2%. For SPS, AWS reaches 100% coverage versus Azure’s 79.5%.

The average SPS of Azure (2.2) is lower than AWS (2.6). Each vendor, however, computes SPS through an undisclosed and proprietary methodology, so the lower Azure score does not by itself imply lower actual availability. Section 9 addresses this question directly through real-world provisioning experiments.

Correlation with Region Size. Using public IPv4 address counts as a proxy for region size [43], with the counts computed from the vendors’ published IP ranges archived in

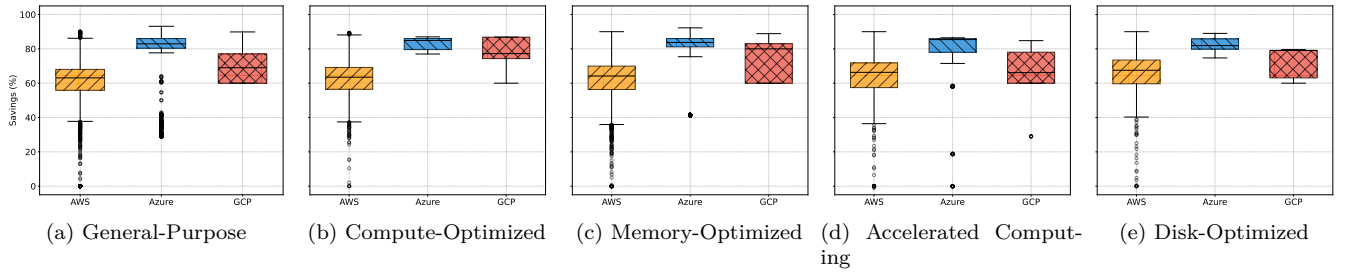


Figure 7: Spot instance price savings by instance category.

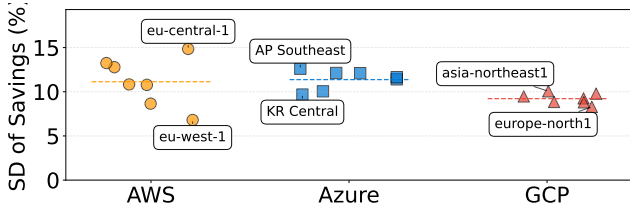


Figure 8: Regional comparison of spot instance price savings standard deviation.

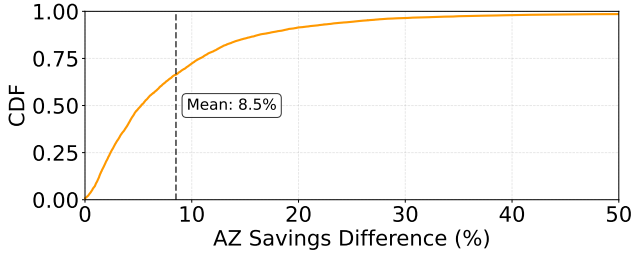


Figure 9: CDF of price savings variability across AZs within AWS regions.

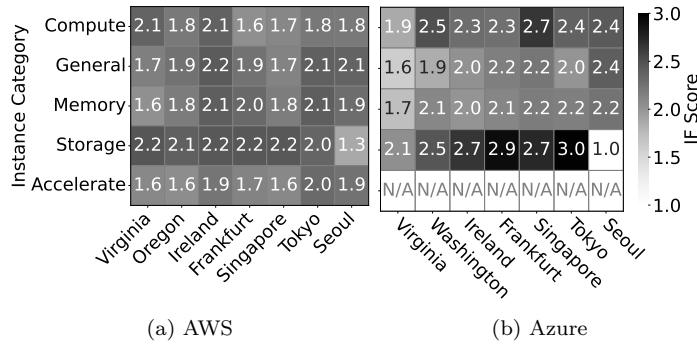


Figure 10: Spatial variations of IF scores by instance category and region.

June 2025 [44, 45], Azure exhibits a strong negative correlation between region size and both IF score ($r = -0.94$) and SPS ($r = -0.75$), indicating that larger, more popular regions have lower availability. AWS shows only weak correlations (IF: $r = -0.26$, SPS: $r = 0.56$), suggesting capacity provisioning that is more uniform relative to demand.

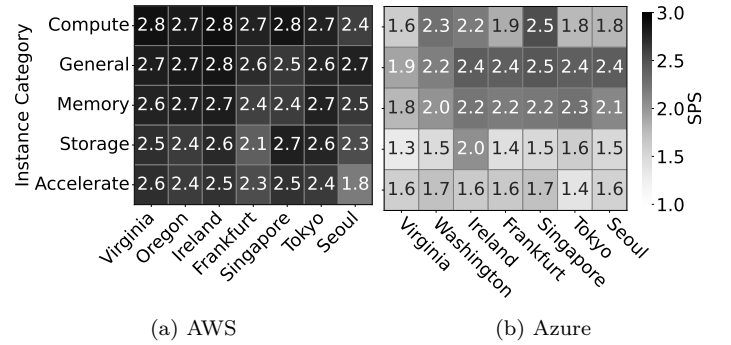


Figure 11: Spatial variations of SPS by instance category and region.

Azure SPS Aggregation Mechanism. Azure SPS API provides both per-AZ and per-region values. To understand the aggregation logic, per-region scores were compared against aggregated per-AZ scores using three common statistics (mean, maximum, minimum). The maximum aggregation achieves the highest match rate at 85.4%, with 71.9% for both the mean and the minimum, indicating that the region-level SPS corresponds to the maximum SPS across AZs. Having characterized the spatial distribution of spot metrics, the next question is whether these metrics also vary over time.

7.3. Temporal Characteristics

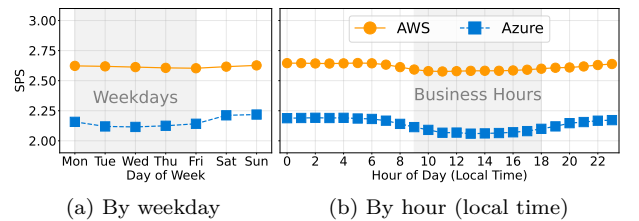


Figure 12: Temporal variations of SPS (local time).

Spot price and IF score are temporally stable on both vendors. Price shows no significant fluctuation consistent with the post-2017 pricing policy [17], and IF score dilutes short-term changes by summarizing interruption frequency

over the preceding month. In contrast, SPS exhibits subtle but discernible cyclical patterns (Figure 12): scores are marginally lower during weekdays compared to weekends and during local business hours compared to off-peak periods. This makes SPS the most responsive real-time indicator among the three metrics. With three metrics showing different spatial and temporal behaviors, a natural follow-up is whether they carry redundant or complementary information.

7.4. Correlation Analysis

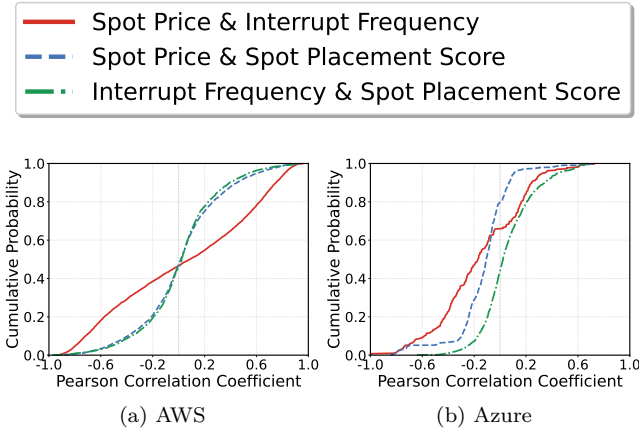


Figure 13: CDFs of pairwise Pearson correlation coefficients among Spot Price, SPS, and IF score.

The Pearson correlation coefficient r measures the strength of a linear relationship between two variables on a -1 to $+1$ scale, where values near zero indicate no linear association. Figure 13 shows the distributions of the pair-level coefficients. Median coefficients are small on both platforms (price-IF -0.17 and price-SPS -0.11 on Azure, $+0.09$ and $+0.02$ on AWS), and no relationship is consistent enough for price to serve as a proxy for availability. Analyses targeting reliability should rely on SPS and IF instead. The preceding analyses treat all instance sizes equally. The final dimension to examine is whether instance size itself affects these metrics.

7.5. Instance Size Impact

Figure 14 shows consistent patterns across both providers: SPS decreases as instance size increases, indicating reduced placement favorability for larger instances. Spot price scales non-linearly, with AWS reaching the best price efficiency at 16xlarge (64 vCPU) and Azure at 64 vCPU, before sharp increases at the largest size classes. IF score remains stable across size classes on both platforms.

The drop of single-instance SPS with increasing instance size already hints that requesting multiple instances at once should degrade availability further. The next section quantifies this multi-node degradation directly and asks whether the resulting curves admit a compact functional form across vendors.

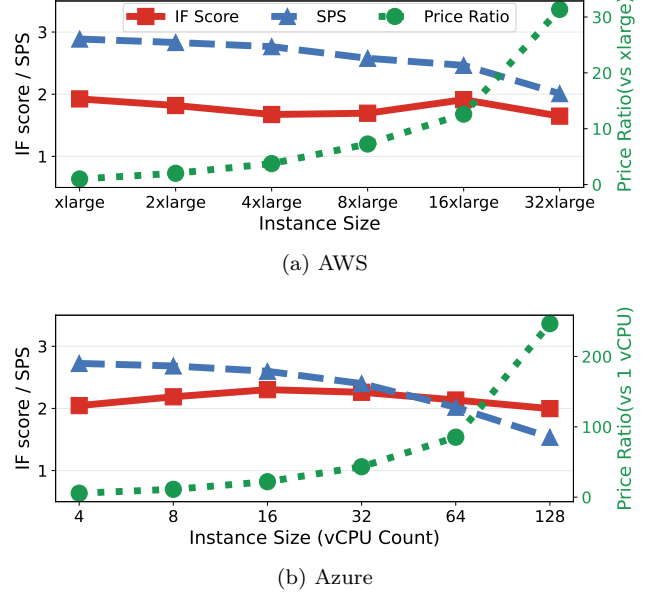


Figure 14: Spot Price, SPS, and IF score as functions of instance size.

7.6. Cross-Vendor Availability Complementarity

The preceding subsections compare AWS and Azure independently on each metric. A distinct question for multi-cloud practitioners is whether the two vendors' availability fluctuations are correlated. If both vendors experience low availability simultaneously, multi-cloud diversification provides no protection. To test this, the average SPS time-series of each vendor is computed per 10-minute timestamp for seven region pairs over the June 2025 observation period, and the Pearson correlation between the two aligned series is measured. Each pair follows the matched regions of Table 4. Correlations use the full 10-minute resolution ($n = 4,281$ timestamps), and uncertainty is quantified with day-block bootstrap 95% confidence intervals, which resample whole days to respect the strong within-day autocorrelation of the series.

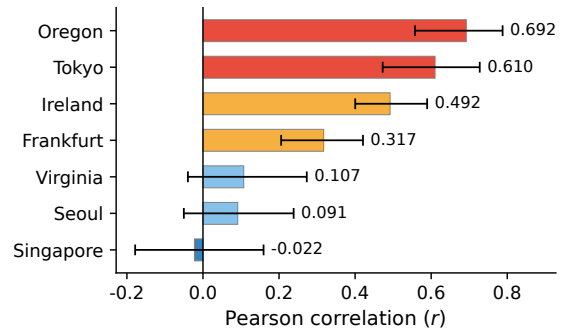


Figure 15: Pearson correlation of region-aggregate SPS between AWS and Azure for each matched region pair. Error bars show bootstrap 95% confidence intervals.

Figure 15 shows that the correlation is region-dependent, ranging from -0.02 (Singapore) to 0.69 (Ore-

gon). For three pairs (Singapore, Seoul, Virginia) the confidence interval contains zero, so the two vendors' availability fluctuations are statistically indistinguishable from independent in those regions. Oregon and Tokyo exhibit strong positive correlation ($r = 0.69$ and 0.61) and Ireland and Frankfurt moderate correlation (0.49 and 0.32), suggesting that demand-driven capacity pressure in those areas affects both vendors simultaneously. Joint low-availability events in which both vendors' region-aggregate SPS falls below 2.0 at the same timestamp do not occur in any pair over the month. This reflects the aggregate levels rather than co-movement, as the AWS region aggregate never falls below 2.0 in the window, so joint events are unreachable regardless of how the vendors co-move, and the diversification argument rests on the near-zero correlations instead. Multi-cloud spot deployments can therefore reduce the probability of cluster-wide provisioning failure, though the degree of diversification benefit is region-dependent and strongest in regions with low cross-vendor correlation.

8. Analysis of Multi-Node Availability

This section addresses RQ-4 by analyzing cross-vendor differences in how spot instance availability degrades as cluster size increases. An inverse degradation model is fitted on both AWS and Azure to enable direct quantitative comparison of degradation shape, steepness, and availability floors.

8.1. SPS Across Target Node Counts

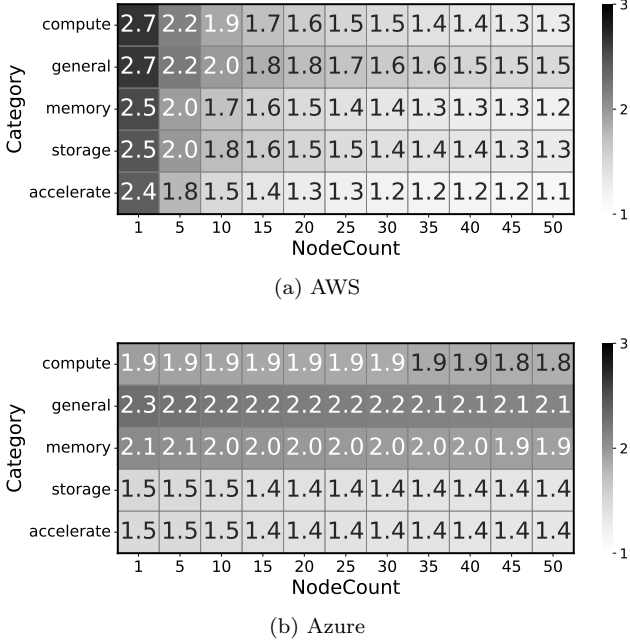


Figure 16: Average SPS by target node count and instance category.

Figure 16 presents the cross-vendor comparison of multi-node SPS. At single-node scale, AWS records higher SPS

values than Azure for most categories. As node count increases, however, AWS SPS decreases substantially and converges toward or falls below Azure's values, while Azure SPS varies by less than 0.2 points across the same node-count range and across categories.

This indicates a structural cross-vendor difference. AWS is more favorable for single-node or small-cluster deployments, while Azure offers more predictable availability for large-scale clusters.

8.2. Availability Degradation Model

To make this qualitative observation precise and enable quantitative cross-vendor comparison, a parametric model is fit to the degradation curves. Six candidate models are fit to the average SPS as a function of target node count n , namely Linear ($a - bn$), Exponential ($a \cdot e^{-\lambda n} + c$), Power-law ($a \cdot n^{-b} + c$), Logarithmic ($-b \cdot \ln(n) + a + c$), Inverse ($a/(n + b) + c$), and Square root ($a - b\sqrt{n} + c$). Model quality is evaluated by two metrics. R^2 (coefficient of determination) measures how much of the variance in the data the model explains, where 1.0 is a perfect fit and 0 is no explanatory power. The Akaike Information Criterion (AIC) [46] balances goodness of fit against model complexity, penalizing a model with more parameters unless it fits substantially better. A lower AIC is better.

Table 8 summarizes the average R^2 and AIC across all five instance categories for each model on both vendors. Among the six candidates, the inverse model achieves the highest average R^2 and lowest AIC on both AWS and Azure. Per (vendor, category) combination, the inverse model is selected as the best by AIC for 7 of 10 combinations, namely AWS Memory, Storage, Accelerated and Azure General, Compute, Memory, Storage. In the remaining three combinations, AWS Compute and Azure Accelerated are best fit by the Logarithmic and Exponential models respectively, but the inverse model remains within $\Delta\text{AIC} \leq 3$ and is statistically indistinguishable from the best. AWS General is fit substantially better by the Logarithmic model ($\Delta\text{AIC} \approx 22$ for the inverse). For consistency of cross-vendor parameter interpretation, the inverse model is adopted as the unified form for subsequent analysis even on AWS General. The resulting fit still has $R^2 = 0.997$ for that category, and the loss in AIC reflects the inverse model's slightly slower decay near the small-cluster end rather than a structural mismatch:

$$\text{SPS}(n) = \frac{a}{n + b} + c \quad (3)$$

where a , b , and c denote the degradation magnitude, offset parameter, and availability floor respectively.

The degradation magnitude a controls how much SPS drops from single-node to large-cluster scale. The offset parameter b governs the initial decline rate. The slope of the fitted curve is $\text{SPS}'(n) = -a/(n + b)^2$, so the initial decline rate at the single-node end equals $a/(1 + b)^2$, and

a smaller b produces a steeper initial decline. The availability floor c represents the minimum SPS that the fitted model converges to as cluster size increases.

Table 8: Average R^2 and AIC of six candidate degradation models.

Model	AWS		Azure	
	Avg R^2	Avg AIC	Avg R^2	Avg AIC
Linear	0.740	-33.5	0.926	-95.0
Sqrt	0.894	-41.8	0.988	-113.5
Power-law	0.992	-71.9	0.922	-93.1
Exponential	0.988	-65.2	0.996	-126.9
Logarithmic	0.993	-81.8	0.929	-94.2
Inverse	0.998	-97.3	0.997	-132.9

Table 9 presents the fitted inverse model parameters for all five instance categories on both vendors.

The fitted parameters reveal three cross-vendor differences. First, the total degradation from single-node to 50-node scale is far larger on AWS, where SPS drops by 1.13 to 1.33 points compared to 0.09 to 0.16 points on Azure. Second, the initial decline rate $a/(1+b)^2$ ranges from 0.15 to 0.29 on AWS and from 0.004 to 0.012 on Azure. The AWS rate is roughly 12 to 45 times larger in every category, confirming that AWS availability drops more rapidly at small cluster sizes. Measured as total degradation instead of initial decline rate, the same gap is 7 to 14 times. The two ratios differ because the AWS curves are convex and decline fastest at small n , whereas the Azure curves are nearly linear over the sampled range. Third, the floor c diverges sharply for General and Memory categories, where Azure floors (near 2.0) indicate near-Medium availability at large cluster sizes while AWS floors (near 1.1) converge toward Low availability. Accelerated and Storage categories show comparable floors on both vendors, suggesting similar worst-case availability for GPU and storage resources.

The widest cross-vendor gap at 50 nodes is in Memory (AWS 1.26 vs Azure 1.94), and the steepest initial decline on either side is AWS Accelerated Computing (initial decline rate 0.29, floor 1.02), where GPU contention dominates at cluster scale. A 70/30 train-test split at the instance-AZ pair level, repeated 10 times with different random seeds, gives a root mean square error (RMSE) below 0.035 on the held-out test set on the 1-3 SPS scale across all categories and vendors. The fit is therefore not specific to the sampled pairs.

The cross-vendor asymmetry is also robust to the treatment of Azure non-availability responses (Section 4). Refitting the Azure curves with every non-availability response treated as the lowest score, an extreme lower bound, flattens the Azure curves further. The AWS initial decline rate becomes 17 to 114 times larger across categories, and Azure’s higher floors for the General and Memory categories persist. Treating only `DataNotFoundOrStale` responses as the lowest score

changes the fitted floors by at most 0.003.

The asymmetry is also stable over time. Refitting on each month of pre-propagation observations from April 2025 through June 2026 preserves every cross-vendor comparison in all fifteen months. AWS degrades more in every category, in total degradation and in initial decline rate alike, and the steepest decline always belongs to AWS. The smallest monthly AWS total degradation (0.75 SPS points) exceeds the largest on Azure (0.15) five-fold. The absolute values change within the window, most visibly for AWS Accelerated Computing, whose initial decline rate halves from 0.30 to 0.17, while the cross-vendor ordering is unchanged in every month.

While the inverse model describes the average degradation shape across all snapshots, $T3$ is a time-varying metric computed at each collection cycle that reflects the current maximum cluster size at high availability for each instance-AZ pair. $T2$ and $T3$ are strongly correlated (mean Pearson $r = 0.692$ for AWS, 0.754 for Azure), justifying the use of $T3$ as the representative multi-node metric in subsequent validation analyses.

Section 9 tests these metrics against real provisioning outcomes.

9. Real-World Validation Experiments

This section addresses RQ-5 by independently testing whether vendor-provided SPS values predict actual provisioning outcomes. Existing studies have not addressed this question. Prior work either uses these metrics as inputs to scoring and recommendation systems without validating their predictive power, or validates availability on a single vendor only. Real-world availability is measured by having each worker issue a spot request, record whether the request is fulfilled, and terminate immediately. This approach is also adopted in recent spot instance studies [47, 48]. The short worker lifetime isolates the provisioning-success outcome that SPS is meant to predict and keeps cloud cost manageable at the scale required for cross-vendor validation. The experiments were conducted on AWS and Azure during March and April 2026.

9.1. Single-Node Validation

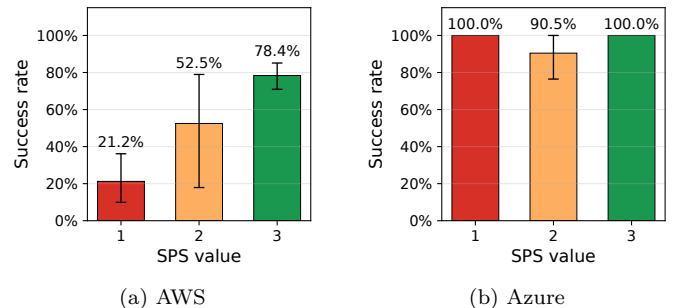


Figure 17: Single-node provisioning success rate by SPS.

Table 9: Fitted inverse degradation model parameters ($SPS(n) = \frac{a}{(n+b)} + c$).

Vendor	Category	a	b	c (floor)	SPS(1)	SPS(50)	Drop	R^2
AWS	General	12.25	8.14	1.30	2.64	1.51	1.13	0.997
AWS	Compute	15.55	8.73	1.06	2.65	1.32	1.33	0.998
AWS	Memory	9.91	5.82	1.08	2.53	1.26	1.28	0.998
AWS	Accelerated	6.77	3.83	1.02	2.42	1.14	1.28	1.000
AWS	Storage	10.10	6.73	1.15	2.45	1.33	1.13	0.999
Azure	General	3.91	17.18	2.07	2.29	2.13	0.16	0.998
Azure	Compute	11.37	54.96	1.74	1.94	1.84	0.09	0.999
Azure	Memory	5.18	22.86	1.87	2.08	1.94	0.15	0.999
Azure	Accelerated	10.46	38.05	1.24	1.51	1.36	0.15	0.992
Azure	Storage	6.79	28.32	1.28	1.51	1.37	0.14	0.998

Three 24-hour single-node provisioning rounds were conducted on AWS (28,224 attempts) and two on Azure (134,069 attempts), each covering the 7 representative regions and 12 to 35 instance types. The SPS value at request time was obtained from the SpotLake archive. Only outcomes that represent a capacity decision enter the analysis. A request counts as a success when the instance is created and as a failure when the vendor declines it for lack of capacity. On AWS every logged outcome is of one of these two kinds. On Azure the excluded outcomes are requests for three VM sizes that Azure does not offer as spot instances (5,040 attempts, all failing deterministically), conflicts over the measurement harness’s own network interfaces, and polls that timed out before an outcome was returned. After matching each remaining request to the snapshot in effect at its issue time, 121,186 Azure attempts and all 28,224 AWS attempts enter the analysis. Figures 17a and 17b show the combined success rates by SPS value. Error bars indicate cluster bootstrap 95% confidence intervals, obtained by resampling the (instance type, region, AZ) pairs within each group: repeated attempts on the same pair are strongly correlated (intra-class correlation 0.51–0.70), and intervals that treat attempts as independent would be narrower than these intervals by a factor of six to more than forty. For the two Azure groups with no failures the resampled interval is degenerate, and their uncertainty is better read from their pair counts (8 and 63 pairs). The chi-squared contrasts reported below likewise treat attempts as independent and are therefore read as descriptive summaries, with uncertainty carried by the cluster-aware intervals.

The two vendors produce qualitatively different SPS behavior. On AWS, success rates rise monotonically from 21.2% at SPS=1 ($n = 3,213$) to 52.5% at SPS=2 ($n = 402$) to 78.4% at SPS=3 ($n = 24,609$). To confirm that these gaps are not due to chance, Pearson’s chi-squared test was applied to each adjacent pair. This test compares observed success/failure counts against what would be expected if both SPS levels had the same underlying rate. Both comparisons yielded $p \ll 0.001$ ($\chi^2 = 186.0$ and 152.5), and both remain significant under Bonferroni correction for

two tests ($\alpha = 0.025$). AWS SPS therefore acts as an ordinal availability predictor. In the SpotLake archive, SPS=2 accounts for only 1.3% of snapshots (vs 7.6% SPS=1 and 91.1% SPS=3), making the scale effectively binary in practice. The cross-cycle swing of SPS=2 (22% to 81%) reflects this rarity: a 24-hour window can land on either side of a transition.

Azure does not exhibit this ordinal structure. SPS=1 and SPS=3 both succeed at 100% ($n = 9,697$ and $87,042$), and failures concentrate only in SPS=2 (9.5% fail rate, $n = 24,447$), tracing to a small set of Storage-optimized instance-region pairs in AP East and KR Central. Applying the same chi-squared test to Azure confirms that SPS=2 differs significantly from both SPS=1 ($\chi^2 = 991.8$, $p \ll 0.001$) and SPS=3 ($\chi^2 = 8479.4$, $p \ll 0.001$). The differences are statistically significant, but the pattern is not ordinal: SPS=1 outperforms SPS=2 while SPS=3 matches SPS=1, so SPS does not rank availability monotonically on Azure. This cross-vendor asymmetry becomes apparent only when both vendors are validated under the same protocol.

The same cross-vendor asymmetry appears in the archive-based characterization of Section 8.2: AWS has steep initial decline rates (0.15–0.29) and low floors c (1.02–1.30), an environment where SPS variation carries meaning, while Azure has near-flat initial decline rates (0.004–0.012) and high floors c (1.24–2.07), where SPS rarely signals more than a localized hot spot.

9.2. Multi-Node Provisioning Validation

Single-node SPS captures only one point on the multi-node degradation curve. To validate whether the cluster-level metric $T3$ (Section 8.2) predicts actual multi-node provisioning, a 10-minute round experiment was conducted. Each round requested 50 spot instances of one (instance type, region, AZ) combination, drawn round-robin from a fixed schedule covering the 7 representative regions and a rotating set of instance types. At the moment each request is issued, the most recent SpotLake snapshot for that (instance type, region, AZ) is read and its $T3$ value

(the largest cluster size whose SPS equals 3 in that snapshot) is recorded as the predictive metric for the round. Each of the 50 workers in a round independently observes its launch result. The round’s success rate is the count of successes divided by the 50 requested results, and a result that is not returned within a 10-minute matching window is counted as a failure.

Three contiguous experiment windows were aggregated into a 96-hour dataset spanning weekend (Saturday–Sunday) and weekday (Monday–Tuesday) operation in early April 2026. The combined dataset contains 528 rounds and 26,400 requested results on AWS, and 529 rounds and 26,450 requested results on Azure. On AWS, 100 of the requested results (0.38%) were not returned and are counted as failures. All of them trace to a fault in the measurement client specific to a single instance type rather than to a capacity shortage. Azure returned every requested result. Because the 50 workers of a round share one (instance type, region, AZ) draw at a single time point, their outcomes are strongly correlated (intraclass correlation 0.66 on AWS and 0.31 on Azure). Interval estimates therefore resample whole rounds (cluster bootstrap, $B = 10,000$) rather than treating workers as independent, which would overstate the effective sample size by more than an order of magnitude.

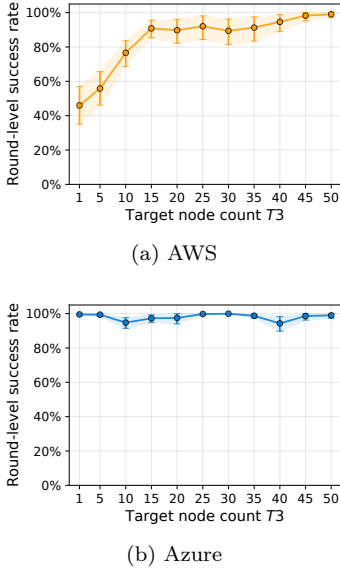


Figure 18: Multi-node provisioning success rate by $T3$. Error bars show bootstrap 95% confidence intervals.

AWS multi-node results. Figure 18a shows that AWS success rate increases with $T3$ from 46.0% at $T3 = 1$ to 99.0% at $T3 = 50$, generally monotonically except for small dips at $T3 = 20$ (89.8%), $T3 = 30$ (89.4%), and $T3 = 35$ (91.3%) whose pooled rates lie within the bootstrap confidence intervals of the adjacent buckets, consistent with sampling variation across rounds rather than a break in the trend. The $T3 = 20$ bucket also contains one round whose results were lost to the fault in the measurement

client and count as failures. Pairs with small $T3$ fail about half of the time in real provisioning, while pairs with large $T3$ succeed almost always. Because the experiment samples a different (instance type, region, AZ) combination each round rather than repeating a single combination, individual rounds at the same $T3$ come from combinations with different baseline availability (std 0.27 at $T3 = 10$, 0.39 at $T3 = 1$). The monotone rise of the aggregate curve with $T3$ therefore demonstrates that $T3$ ordinally tracks real provisioning success across the full set of combinations, not merely within a few specific ones.

Azure multi-node results. The Azure curve in Figure 18b is flat across all $T3$ values: pooled success rate stays in the 94–100% range, and the cluster bootstrap 95% confidence intervals that resample whole rounds are at most 0.085 wide and lie entirely above 0.89, the two widest (at $T3 = 40$ and $T3 = 10$) falling at the two node counts where the mean rate dips. There is no monotone relationship between $T3$ and success rate. Instead, the 20 rounds that fall below 80% are spread over 11 of the 81 sampled instance-region pairs, dominated by Memory-optimized instances in EU West and EU North and General-purpose instances in AP Southeast, EU West, and US East. Only three are Storage-optimized instances, all in AP Southeast. The multi-node experiment therefore supports the single-node observation: Azure’s $T3$ value, like its SPS, is not a reliable ordinal predictor of cluster-level capacity, and the few failures that do occur are localized rather than systemic.

9.3. Cross-Vendor Validation Summary

The single-node and multi-node experiments produce a consistent cross-vendor contrast. On AWS, both SPS at single-node scale and $T3$ at multi-node scale function as ordinal predictors of real provisioning success, with success rates varying from 21% to 78% by SPS value and from 46% to 99% by $T3$. On Azure, neither metric shows an ordinal relationship with success: single-node rates stay within 90.5–100% across all SPS values, and multi-node rates remain within 94–100% regardless of $T3$. A rank test at the round level makes the contrast exact: the Spearman correlation between $T3$ and round success rate is $\rho = 0.53$ ($p < 10^{-39}$) on AWS and $\rho = -0.02$ ($p = 0.72$) on Azure.

These validation results translate into different operational habits. On AWS the SpotLake-derived metrics (SPS for single-node, $T3$ for multi-node) are usable as ordinal availability predictors, and instance selection can rely on them without additional probing. On Azure capacity is plentiful in essentially every instance-region pair, so the same metrics carry almost no signal and users do not currently need to consult them before launching spot instances. The flip side is that if Azure capacity were to deteriorate in a specific instance-region pair, SPS or $T3$ alone would not detect the change. Active probing would be needed as a complementary signal.

10. Discussion

This section synthesizes findings across all analyses and discusses limitations.

10.1. Key Findings

Update dynamics (RQ-1). Metric update behavior is vendor-specific and changes over time. Azure SPS demands high-frequency sampling at 10 minutes or less, whereas AWS SPS and IF score tolerate up to 30 minutes with under 5% data loss. Collection systems should treat these intervals as adaptive parameters rather than constants.

Pricing and spatial structure (RQ-2). Azure offers the highest median spot price savings while AWS shows the widest variance, with intra-region (AZ-level) price dispersion reaching 91% on AWS in extreme cases. Region size correlates strongly with Azure availability (SPS $r = -0.94$) but barely at all with AWS ($r = 0.56$).

Temporal patterns (RQ-3). Spot price and IF score are temporally flat, but SPS and $T3$ track real-time demand. Both metrics dip during local business hours and weekday operation. Temporal analysis must therefore be conducted in local time rather than UTC, otherwise region-specific demand cycles average out.

Multi-node degradation (RQ-4). Of six candidate models, the inverse form $SPS(n) = a/(n+b) + c$ achieves the best average AIC for both vendors and $R^2 \geq 0.99$ for every (vendor, category) combination. AWS SPS drops by 1.13 to 1.33 points between 1 and 50 nodes with initial decline rates ($a/(1+b)^2$) between 0.15 and 0.29, while Azure drops by only 0.09 to 0.16 points with rates between 0.004 and 0.012. The floor parameter c indicates the worst-case SPS that the model converges to at large cluster sizes, ranging from 1.24 to 2.07 on Azure and only 1.02 to 1.30 on AWS. Azure therefore exhibits both a flatter degradation curve and a higher availability floor for most categories.

AWS validation (RQ-5). SPS is an ordinal predictor of AWS single-node capacity, with 21.2%, 52.5%, and 78.4% success at SPS=1/2/3 and all adjacent gaps significant at $p < 0.001$. The cluster-level metric $T3$ extends this property to multi-node provisioning: success rises with $T3$ from 46% at $T3 = 1$ to 99% at $T3 = 50$ across 528 rounds spanning weekend and weekday operation, with small dips at $T3 = 20$, $T3 = 30$, and $T3 = 35$ that stay within the confidence intervals of adjacent buckets, and the overall shape mirrors the cross-vendor structure characterized in RQ-4: a correspondence of pattern rather than of fitted values, one that monthly refits preserve in every month from April 2025 through June 2026.

Azure validation (RQ-5). The same metrics on Azure carry almost no signal. SPS=1 and SPS=3 both succeed at 100%, multi-node success stays between 94 and 100% at every $T3$, and the failing rounds concentrate in a small subset of the sampled instance-region pairs rather than in a single instance family.

Generalizability. The analysis is based primarily on June 2025 data, with supplementary three-year longitudinal observations for update pattern analysis (Figure 6). The core cross-vendor structural differences, the large total degradation and low floor on AWS versus the minimal degradation and high floor on Azure, reflect infrastructure and capacity-management choices by each provider. Monthly refits on the pre-propagation archive show them holding in every month from April 2025 through June 2026. Specific numerical values do drift within that window, most visibly the initial decline rate of AWS Accelerated Computing halving and the single-node lead of AWS General reversing, which motivates continuous data collection through services like SpotLake and periodic reanalysis.

Application Scenarios. The findings translate into deployment guidance for the workload classes that already rely on spot instances. Distributed training pipelines [13] face the steepest availability decline on AWS, whose Accelerated Computing category combines the highest initial decline rate (0.29) with the lowest floor (1.02). Multi-GPU jobs therefore benefit from consulting $T3$ before placement and from keeping an Azure fallback for the same category, where the floor is higher. Delay-tolerant batch and HPC systems [14, 15, 27, 36] can rank vendors and regions by the archived $T3$ directly, which ordinally predicts multi-node success on AWS (46% at $T3 = 1$ rising to 99% at $T3 = 50$). Big-data frameworks that provision clusters of a fixed size [10, 32, 33] can read the expected availability at their target node count off the fitted degradation curves instead of probing. Cross-vendor portfolios [37] reduce risk in the region pairs whose availability fluctuations are statistically independent (Section 7.6). These benefits apply where the spot discount dominates the overheads that multi-cloud operation adds, such as data egress, data locality, image portability, and separate credential management.

10.2. Limitations

The degradation model captures average behavior at the category level and does not account for instance-level variance or sudden capacity events. The multi-node validation experiment uses 12 rounds per $T3$ value within each 24-hour cycle, so individual outlier rounds can shift cycle-level estimates by several percentage points. The 96-hour aggregated dataset reduces but does not eliminate this noise.

11. Conclusion

Two main findings emerge from this study. First, multi-node spot availability degrades structurally differently on

AWS and Azure. An inverse model fits both vendors well, but the AWS curve is roughly an order of magnitude steeper at the small-cluster end, with initial decline rates 12 to 45 times larger across categories, and converges to a much lower floor than Azure, so cluster-size-aware vendor selection is a consequential decision rather than a marginal optimization. Second, the same vendor-provided availability score has different operational meaning on the two providers. AWS SPS and the multi-node T3 statistic are usable as ordinal availability predictors, but the same metrics on Azure are uncorrelated with real success because Azure capacity is plentiful in nearly every instance-region pair. Practitioners should therefore treat the SPS scale as semantically vendor-specific rather than as a comparable cross-vendor index.

A consequence for monitoring is that the absence of useful SPS signal on Azure has a cost of its own. It also means the metric will not flag a future loss of capacity in a specific instance-region pair, and complementary signals such as observed provisioning latency or active probing will be needed in that setting. Techniques developed for extracting patterns from noisy operational data [49, 50] provide candidate building blocks for automating the detection of such capacity losses over the archived metric streams. The SpotLake system, the multi-node dataset, and the fitted models are released to support that follow-on monitoring as well as continued research and reanalysis.

Data availability

The archived multi-vendor dataset is publicly served through the SpotLake web service (<https://spotlake.ddps.cloud/>), and the collection system is open source (<https://github.com/ddps-lab/spotlake>). The analysis scripts, the provisioning-experiment logs, and the reproducibility documentation that regenerates every published table and figure are released as a separate public artifact archive on Zenodo (<https://doi.org/10.5281/zenodo.20351146>).

Acknowledgments

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (RS-2020-II201373, RS-2022-00144309, RS-2025-25441560, and RS-2026-25507506).

References

- [1] R. Wolski, J. Brevik, R. Chard, K. Chard, Probabilistic guarantees of execution duration for amazon spot instances, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '17, Association for Computing Machinery, New York, NY, USA, 2017. doi:10.1145/3126908.3126953. URL <https://doi.org/10.1145/3126908.3126953>
- [2] O. Agmon Ben-Yehuda, M. Ben-Yehuda, A. Schuster, D. Tsafir, Deconstructing amazon ec2 spot instance pricing, ACM Trans. Econ. Comput. 1 (3) (sep 2013). doi:10.1145/2509413.2509416. URL <https://doi.org/10.1145/2509413.2509416>
- [3] B. Javadi, R. K. Thulasiram, R. Buyya, Characterizing spot price dynamics in public cloud environments, Future Generation Computer Systems 29 (4) (2013) 988–999, special Section: Utility and Cloud Computing. doi:<https://doi.org/10.1016/j.future.2012.06.012>. URL <https://www.sciencedirect.com/science/article/pii/S0167739X12001483>
- [4] G. Portella, G. N. Rodrigues, E. Nakano, A. C. Melo, Statistical analysis of amazon ec2 cloud pricing models, Concurrency and Computation: Practice and Experience 31 (18) (2019) e4451, e4451 cpe.4451. doi:<https://doi.org/10.1002/cpe.4451>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.4451>
- [5] C. Wang, Q. Liang, B. Urgaonkar, 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, ICPE '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 63–74. doi:10.1145/3030207.3030210. URL <https://doi.org/10.1145/3030207.3030210>
- [6] N. Ekwe-Ekwe, A. Barker, 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), 2018, pp. 370–373. doi:10.1109/CCGRID.2018.00059.
- [7] L. Zheng, C. Joe-Wong, C. W. Tan, M. Chiang, X. Wang, How to bid the cloud, in: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 71–84. doi:10.1145/2785956.2787473. URL <https://doi.org/10.1145/2785956.2787473>
- [8] P. Sharma, D. Irwin, P. Shenoy, How not to bid the cloud, in: 8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16), USENIX Association, Denver, CO, 2016. URL <https://www.usenix.org/conference/hotcloud16/workshop-program/presentation/sharma>
- [9] M. Khodak, L. Zheng, A. S. Lan, C. Joe-Wong, M. Chiang, Learning cloud dynamics to optimize spot instance bidding strategies, in: IEEE INFOCOM 2018 - IEEE Conference on Computer Communications, 2018, pp. 2762–2770. doi:10.1109/INFOCOM.2018.8486291.
- [10] N. Chohan, C. Castillo, M. Spreitzer, M. Steinder, A. Tantawi, C. Krintz, 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, 2010. URL <https://www.usenix.org/conference/hotcloud-10/see-spot-run-using-spot-instances-mapreduce-workflows>
- [11] A. S. Foundation, Apache hadoop (2004). URL <http://hadoop.apache.org/>
- [12] J. Dean, S. Ghemawat, Mapreduce: Simplified data processing on large clusters, in: Proceedings of the 6th Conference on Symposium on Operating Systems Design & Implementation - Volume 6, OSDI'04, USENIX Association, Berkeley, CA, USA, 2004, pp. 10–10. URL <http://dl.acm.org/citation.cfm?id=1251254.1251264>
- [13] K. Lee, M. Son, Deepspotcloud: Leveraging cross-region gpu spot instances for deep learning, in: 2017 IEEE 10th International Conference on Cloud Computing (CLOUD), 2017, pp. 98–105. doi:10.1109/CLOUD.2017.21.
- [14] I. Menache, O. Shamir, N. Jain, 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, 2014, pp. 177–187. URL <https://www.usenix.org/conference/icac14/>

- technical-sessions/presentation/menache
- [15] S. Subramanya, T. Guo, P. Sharma, D. Irwin, P. Shenoy, Spotton: A batch computing service for the spot market, in: Proceedings of the Sixth ACM Symposium on Cloud Computing, SoCC '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 329–341. doi:10.1145/2806777.2806851. URL <https://doi.org/10.1145/2806777.2806851>
 - [16] A. W. S. Blogs, New amazon ec2 spot pricing model, <https://aws.amazon.com/blogs/compute/new-amazon-ec2-spot-pricing/> (2017).
 - [17] M. Baughman, S. Caton, C. Haas, R. Chard, R. Wolski, I. Foster, K. Chard, Deconstructing the 2017 changes to aws spot market pricing, in: Proceedings of the 10th Workshop on Scientific Cloud Computing, ScienceCloud '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 19–26. doi:10.1145/3322795.3331465. URL <https://doi.org/10.1145/3322795.3331465>
 - [18] D. Irwin, P. Shenoy, P. Ambati, P. Sharma, S. Shastri, A. Ali-Eldin, The price is (not) right: Reflections on pricing for transient cloud servers, in: 2019 28th International Conference on Computer Communication and Networks (ICCCN), 2019, pp. 1–9. doi:10.1109/ICCCN.2019.8846933.
 - [19] S. Lee, J. Hwang, K. Lee, Spotlake: Diverse spot instance dataset archive service, in: 2022 IEEE International Symposium on Workload Characterization (IISWC), IEEE Computer Society, Los Alamitos, CA, USA, 2022, pp. 242–255. doi:10.1109/IISWC55918.2022.00029. URL <https://doi.ieeecomputersociety.org/10.1109/IISWC55918.2022.00029>
 - [20] K. Kim, S. Park, J. Hwang, H. Lee, S. Kang, K. Lee, Public spot instance dataset archive service, in: Companion Proceedings of the ACM Web Conference 2023, WWW '23 Companion, Association for Computing Machinery, New York, NY, USA, 2023, p. 69–72. doi:10.1145/3543873.3587314. URL <https://doi.org/10.1145/3543873.3587314>
 - [21] K. Kim, K. Lee, Making cloud spot instance interruption events visible, in: Proceedings of the ACM on Web Conference 2024, WWW '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 2998–3009. doi:10.1145/3589334.3645548. URL <https://doi.org/10.1145/3589334.3645548>
 - [22] S. Cheon, K. Kim, K. Kim, M. Song, K. Lee, Multi-node spot instances availability score collection system, in: Proceedings of the 34th International Symposium on High-Performance Parallel and Distributed Computing (HPDC '25), ACM, 2025.
 - [23] A. W. is New, Introducing amazon ec2 spot placement score (2021). URL <https://aws.amazon.com/about-aws/whats-new/2021/10/amazon-ec2-spot-placement-score/>
 - [24] Azure, azure virtual-machine-scale-sets spot-placement-score (2025). URL <https://learn.microsoft.com/en-us/azure/virtual-machine-scale-sets/spot-placement-score>
 - [25] A. W. Services, Spot placement score limitations, <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-placement-score.html#sps-limitations> (2021).
 - [26] T.-P. Pham, S. Ristov, T. Fahringer, Performance and behavior characterization of amazon ec2 spot instances, in: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018, pp. 73–81. doi:10.1109/CLOUD.2018.00017.
 - [27] A. Marathe, R. Harris, D. Lowenthal, B. R. de Supinski, B. Rountree, M. Schulz, 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, HPDC '14, Association for Computing Machinery, New York, NY, USA, 2014, p. 279–290. doi:10.1145/2600212.2600226. URL <https://doi.org/10.1145/2600212.2600226>
 - [28] W. Guo, K. Chen, Y. Wu, W. Zheng, 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, HPDC '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 191–202. doi:10.1145/2749246.2749259. URL <https://doi.org/10.1145/2749246.2749259>
 - [29] A. Sarah, K. Lee, H. Kim, 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), 2018, pp. 1085–1088. doi:10.1109/DASC/PiCom/DataCom/CyberSciTech.2018.00067.
 - [30] J. Kadupitige, V. Jadhao, P. Sharma, Modeling the temporally constrained preemptions of transient cloud vms, in: Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing, HPDC '20, Association for Computing Machinery, New York, NY, USA, 2020, p. 41–52. doi:10.1145/3369583.3392671. URL <https://doi.org/10.1145/3369583.3392671>
 - [31] F. Yang, B. Pang, J. Zhang, B. Qiao, L. Wang, C. Couturier, C. Bansal, S. Ram, S. Qin, Z. Ma, I. n. Goiri, E. Cortez, S. Baladhandayutham, V. Rühle, S. Rajmohan, Q. Lin, D. Zhang, Spot virtual machine eviction prediction in microsoft cloud, in: Companion Proceedings of the Web Conference 2022, WWW '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 152–156. doi:10.1145/3487553.3524229. URL <https://doi.org/10.1145/3487553.3524229>
 - [32] P. Sharma, T. Guo, X. He, D. Irwin, P. Shenoy, Flint: Batch-interactive data-intensive processing on transient servers, in: Proceedings of the Eleventh European Conference on Computer Systems, EuroSys '16, Association for Computing Machinery, New York, NY, USA, 2016. doi:10.1145/2901318.2901319. URL <https://doi.org/10.1145/2901318.2901319>
 - [33] Y. Yan, Y. Gao, Y. Chen, Z. Guo, B. Chen, T. Moscibroda, Tr-spark: Transient computing for big data analytics, in: Proceedings of the Seventh ACM Symposium on Cloud Computing, SoCC '16, Association for Computing Machinery, New York, NY, USA, 2016, p. 484–496. doi:10.1145/2987550.2987576. URL <https://doi.org/10.1145/2987550.2987576>
 - [34] X. He, P. Shenoy, R. Sitaraman, D. Irwin, 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, HPDC '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 207–218. doi:10.1145/2749246.2749275. URL <https://doi.org/10.1145/2749246.2749275>
 - [35] A. Ali-Eldin, J. Westin, B. Wang, P. Sharma, P. Shenoy, 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, HPDC '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 1–12. doi:10.1145/3307681.3325397. URL <https://doi.org/10.1145/3307681.3325397>
 - [36] A. C. Zhou, J. Lao, Z. Ke, Y. Wang, R. Mao, Farspot: Optimizing monetary cost for hpc applications in the cloud spot market, IEEE Transactions on Parallel and Distributed Systems 33 (11) (2022) 2955–2967. doi:10.1109/TPDS.2021.3134644.
 - [37] P. Sharma, D. Irwin, P. Shenoy, Portfolio-driven resource management for transient cloud servers, Proceedings of the ACM on Measurement and Analysis of Computing Systems 1 (1) (2017) 1–23.
 - [38] C. C. Lee, D. T. Lee, A simple on-line bin-packing algorithm, J. ACM 32 (3) (1985) 562–572. doi:10.1145/3828.3833. URL <https://doi.org/10.1145/3828.3833>
 - [39] J. Forrest, T. Ralphs, S. Vigerske, LouHafer, B. Kristjansson, jpfasano, EdwinStraver, M. Lubin, H. G. Santos, rlougee, M. Saltzman, coin-or/cbc: Version 2.9.9 (Jul. 2018). doi:10.5281/zenodo.1317566. URL <https://doi.org/10.5281/zenodo.1317566>
 - [40] L. Perron, V. Furnon, Or-tools. URL <https://developers.google.com/optimization/>

- [41] V. Chvatal, A greedy heuristic for the set-covering problem, *Mathematics of operations research* 4 (3) (1979) 233–235. doi: 10.1287/moor.4.3.233.
- [42] S. Khuller, A. Moss, J. S. Naor, The budgeted maximum coverage problem, *Information Processing Letters* 70 (1) (1999) 39–45. doi:https://doi.org/10.1016/S0020-0190(99)00031-9. URL https://www.sciencedirect.com/science/article/pii/S0020019099000319
- [43] PatMyron, IP addresses per region, <https://github.com/patmyron/cloud/?tab=readme-ov-file> (2025).
- [44] Microsoft, Azure IP ranges and service tags – public cloud, snapshot of june 11, 2025, <https://github.com/jensihnow/AzurePublicIPAddressRanges>, commit e9ccd38 (2025).
- [45] A. W. Services, AWS IP address ranges, snapshot of june 11, 2025, <https://github.com/femueller/cloud-ip-ranges>, commit f45d1d5 (2025).
- [46] H. Akaike, Information theory and an extension of the maximum likelihood principle, in: *Selected papers of hirotugu akaike*, Springer, 1998, pp. 199–213.
- [47] Z. Li, T. Xia, Z. Mao, Z. Zhou, E. J. Jackson, J. Kerney, Z. Wu, P. Mishra, Y. Xu, Y. Qiao, S. Shenker, I. Stoica, Skynomad: On using multi-region spot instances to minimize ai batch job cost (2026). arXiv:2601.06520. URL <https://arxiv.org/abs/2601.06520>
- [48] Z. Wu, W.-L. Chiang, Z. Mao, Z. Yang, E. Friedman, S. Shenker, I. Stoica, Can’t be late: Optimizing spot instance savings under deadlines, in: *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, USENIX Association, Santa Clara, CA, 2024, pp. 185–203. URL <https://www.usenix.org/conference/nsdi24/presentation/wu-zhanghao>
- [49] S. Hong, Y. Huang, W. Jiang, J. Zhang, J. Wang, R. Jiao, BGA: A noise-immune neural distillation framework for malicious signature extraction in high-entropy encrypted flows, *Information Processing & Management* 64 (1) (2027) 105040. doi:10.1016/j.ipm.2026.105040.
- [50] J. Ahmad, S. Latif, I. U. Khan, M. S. Alshehri, M. S. Khan, N. Alasbali, W. Jiang, An interpretable deep learning framework for intrusion detection in industrial internet of things, *Internet of Things* 33 (2025) 101681. doi:10.1016/j.iot.2025.101681.



Kyungyong Lee is an associate professor in the Department of Data Science at Hanyang University, Seoul, South Korea. His research interests include big data platforms, distributed systems, and cloud computing. He received the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA.

Author Biographies



Jaeil Hwang is a Ph.D. student in the Department of Artificial Intelligence at Hanyang University, Seoul, South Korea. His research interests include performance optimization, cloud computing, and machine learning. He received the M.S. degree in computer science from Kookmin University, Seoul, South Korea.



Taeyoon Kim is a student in the integrated M.S. and Ph.D. program in the Department of Data Science at Hanyang University, Seoul, South Korea. His research interests include AI platform, cloud computing, software development and resource optimization. He received the B.S. degree in software at Kookmin University, Seoul, South Korea.