Information Models: Creating and Preserving Value in Volatile Cloud Resources

Chaojie Zhang  
Dept. of Computer Science  
University of Chicago  
Chicago, United States  
chaojie@uchicago.edu

Varun Gupta  
Booth School of Business  
University of Chicago  
Chicago, United States  
guptav@uchicago.edu

Andrew A. Chien  
Dept. of Computer Science  
University of Chicago & Argonne National Lab  
Chicago, United States  
achien@cs.uchicago.edu

Abstract—Volatile resources are surplus cloud resources not consumed by high priority foreground (reserved/on-demand) load. These resources are exploited by a growing number of users. Today, cloud operators provide no statistical characterization of volatile resources. We consider how releasing such statistics could improve user value by studying Amazon’s 608 EC2 Spot Instance types. Results show that as little as two parameters such as (average, 90pctile) can increase user value by 30%. These results are robust over four-fifths (475 of 608) of instance types.

Beyond competitive concerns, cloud operators are reluctant to share volatile resource statistics because they might be considered a service-level agreement (SLA), and thus constrain their ability to serve foreground load. We show that clever resource management can allay such concerns. We study two plausible classes of foreground load changes, showing one class where such a concern is indeed valid and another where it is not. We design two online resource management algorithms that detect foreground load variation and adapt to maintain a statistical SLA. The algorithms not only improve the ability to maintain guarantees and user value but also improve user experience, reducing job failures by 50%. These results apply to the Stable and Transition classes of resource pools, which account for nearly all of the instance types (577 of 608).

Keywords—cloud computing; resource management; transient resources;

I. INTRODUCTION

Modern cloud datacenters are continually expanding their computing resources to meet growing needs for e-commerce, web search, social networking, enterprise IT, and big data analytics. Offerings include traditional reliable resource models such as on-demand and reserved instances from Amazon’s Elastic Compute Cloud and Google Compute Engine [1][2]. Operators build capacity to match peaks of a dynamically varying load, including short-term fluctuation (minutes, hours, days, weeks), and to match long-term growth (years). Accurate forecasting is difficult, producing fluctuating quantities of excess resources. To increase revenue and resource utilization, cloud providers sell excess resources as unreliable cloud instances (Spot Instances [3], Preemptible virtual machines [4]) that can be revoked.

All cloud providers revoke volatile resources to meet increased foreground (priority) load, and a range of additional cluster management functions [5][6]. In other dimensions such as pricing, revocation warning, and more, volatile resource design varies across providers (see Section II).

Since the introduction of Spot Instances in 2009, extensive research has explored the Spot markets with the goal of increasing the utility and cost-effectiveness of Spot Instances. For example, several studies explore bidding strategies to ensure duration until revocation while minimizing cost [7][8]. Still, others develop sophisticated statistical characterization and prediction by observing revocations and market prices from an external point of view (outside the cloud provider), and recommend bid prices to ensure duration [9][10]. These approaches depend on pricing information and thus apply to only Amazon’s volatile resources. Due to changes that removes guarantees on revocation in price order, it eliminates predictability, these approaches may no longer be viable [11][12][13].

We consider a different approach – a designed statistical characterization of volatile resources we call an “Information Model” – that is released by the cloud provider. A good information model has two properties: 1) it enables applications to make good use of the volatile resources (user value) and 2) it minimizes the information released (provider confidentiality and flexibility).

We ask the question - what is a good information model? To answer, we study a variety of information models, exploring the quantity (number of parameters) and specific statistical parameters included, and evaluating their impact on user value. Using historical data, we study the effectiveness of four information models across 608 Amazon Spot Instance types. Our results show that information models with only two statistical parameters can significantly increase user-value.

Because volatile resources are designed not to interfere with foreground load, information models should not constrain resource management flexibility. We consider the question - If an operator provides an information model based on typical history, what happens when foreground load changes? Specifically, we consider what foreground load change alters the statistics, and thereby incur violation of the implied service-level agreement (SLA) for volatile resources. We further consider – Even with those load changes, can resource management maintain the SLA?
create and study two online resource management algorithms, and their ability to maintain statistical guarantees, and support good user experience in the face of extreme foreground load changes. These algorithms can reduce job failure rates significantly, and maintain overall resource pool value. Specific contributions include:

1) Evaluation of four information models (MTTR, 10pc-tile, 90pc-tile, Full) across a range of volatile resource management algorithms (VRM) that shows the 90pc-tile model with two statistical measures (average, 90 percentile) enables a 30% increase in user value that is robust over 80% (475) of Amazon’s 608 Spot Instance types.

2) Design of online VRM algorithms, additive-increase multiplicative-decrease (AIMD) and Distribution Targeting (DT), that seek to maintain a tacit SLA under disrupted foreground load (frequency shift). Evaluation shows they succeed, eliminating violations for moderate and extreme frequency shifts for two-thirds of all instance types. For the next 30%, the online algorithms are helpful, but not fully successful in preserving guarantees.

3) Study of user-job success rate and user-value under disrupted foreground load shows the online resource management algorithms (AIMD and DT) can shape resource pool statistics to improve user experience. Benefits include maintaining pool value, reducing job failures by 50%, and reducing guarantee violations. These benefits improve the usability of over 95% of the instance types (577 / 608).

The rest of the paper is organized as follows. We cover background – volatile resources and resource statistics – in Section II. In Section III we describe proposed information models and evaluate them. In Section IV, we consider the impact of changes in foreground load and resource management to preserve statistical guarantees. We discuss related work in Section V and summarize in Section VI.

II. BACKGROUND

A. VOLATILE RESOURCES AND INFORMATION MODELS

Cloud operators have introduced volatile resource services to utilize and garner revenue from idle resources. Their defining characteristic is that they are often revoked (for higher paying customers, software upgrades, etc.). However, the features of volatile resources differ greatly across cloud providers [3, 14, 15]. For example, pricing differs – Google’s Preemptible VM’s have a fixed discount [4] and Amazon Spot Instances are dynamically priced by auction [5]. Table I summarizes several major volatile resource services. Cloud providers provide no statistical characterization for these volatile resources; their Information Model is “no information”. The sole exception is Defined Duration instances that provide a minimum duration guarantee for an increased price [16].

<table>
<thead>
<tr>
<th>Duration Info</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spot Instance</td>
<td>None</td>
</tr>
<tr>
<td>Defined Duration</td>
<td>Select duration</td>
</tr>
<tr>
<td>Preemptible VM</td>
<td>≤ 24hrs</td>
</tr>
</tbody>
</table>

Table I: Cloud provider volatile resource services

B. RESOURCE AVAILABILITY PROFILES

Volatile resource availability is defined by capacity and foreground load. Both are trade secrets for cloud providers, consequently, only limited summary information is available; that information insufficient for our broad studies, such as those in this paper [17, 6]. Instead, we exploit pricing information for Amazon Spot Instance markets to infer the resource availability traces.

Amazon EC2 Spot markets contain four regions in North America, each of which is consisted of several availability zones. Each instance type is a resource type with an arbitrary amount of resources associated to it. We treat each instance type in one availability zone from one region as independent. We use 90-day traces of Amazon EC2 Spot markets from 5/5/2017 to 8/3/2017 that include market clearing prices for each 5-minute interval for 608 instance pools.

For each instance type, this comprised 25,920 price samples or 15.8 million prices overall. None of the cloud providers share or release information about resource availability. To enable study, we use the pricing data to infer an availability profile based on a simple linear price-supply relationship, assuming the market is always cleared. At the maximum clearing price in the trace, volatile resources are assumed to be zero; at the minimum price, volatile resources are assumed to be maximum (arbitrarily set at 5,000 units, with a total data center capacity of 10,000 units)². Hence, volatile resource (VR) availability at time $t$ is:

$$VR(t) = (\maxPrice - \text{price}(t)) \times \frac{\maxVol\text{atile Resources}}{\maxPrice - \minPrice} \tag{1}$$

Figure 1 illustrates price traces from which we derived availability profiles for three exemplar instance types ($r3.4xlarge$, $i3.4xlarge$, and $r3.xlarge$) in the us-west-2 availability zone over a 3-month period. Spot Instance prices are shown at 5-minute intervals in blue with price/hour scale. The corresponding on-demand prices are shown by the dark red horizontal line. Data center evolution activities such as large-scale upgrades, system migrations, and system

---

²See Section VI for a more detailed discussion of Defined Duration instances.

---

2This choice affects the absolute value of available resources but has no impact on our results which are all comparative.
reconfigurations, require basic resource flexibility, so we cap maximum interval length at 48 hours.

III. INFORMATION MODELS

![Volatile Resource Management (VRM) Diagram]

Figure 2: Volatile resource managers (VRMs) operating with the same foreground load can generate pools with different interval distributions. Information models provide statistical characterizations to pool users.

We study the question: what information enables users to target volatile resources to extract most value? We define and evaluate a set of information models for volatile resource interval duration statistics (below). Generally, cloud operators do not disclose statistics for volatile resources. All of these information models contain more information than provided by cloud operators today.

### Information Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTTR</td>
<td>Mean, MTTR (mean-time to revocation) of the instance type.</td>
</tr>
<tr>
<td>10ptile</td>
<td>Mean and 10th percentile (10% of intervals are shorter)</td>
</tr>
<tr>
<td>90ptile</td>
<td>Mean and the 90th percentile (90% of intervals are shorter)</td>
</tr>
<tr>
<td>Full</td>
<td>Full histogram (the full probability distribution function of interval durations).</td>
</tr>
</tbody>
</table>

Given a distribution, users select job runtime targets to maximize expected value. We evaluate information models on their efficacy in increasing user value.

A. Methodology

We describe the methodology for evaluating information models: (i) cloud-provider’s resource management algorithms (shapes interval durations), (ii) user value (utility) functions (maps job execution time to user-value), and (iii) user targeting (how users select a job duration). Afterward, we describe the metrics used to compare performance.

**Volatile Resource Management (VRM):** Given foreground load and the resource capacity, the critical choice for resource managers is selecting which resources to revoke. We use four basic resource management algorithms; each chooses different resources for revocation, producing different interval length distributions (see Figure 2). (Random) revoke randomly from the active volatile resources, (FIFO) revoke resources in oldest-first order, (LIFO) revoke resource in last-first order, and (LIFO-Pools) separate volatile resources into five pools; use LIFO within each and LIFO across them.

Of these algorithms, LIFO-Pools bears explanation. A VRM can create multiple resource pools. LIFO-Pools creates a stack of N pools, using LIFO algorithm; the topmost has the shortest intervals and lowest availability. Further down the stack, these properties improve. Pools are divided in “stack” order based on fixed absolute quantities of volatile resources of available. Distribution for each pool is narrower (lower standard deviation) due to the separation, enabling better targeting. In experiments, we use N=5 pools – more pools provided negligible improvement.

**User Value Functions:** A value function defines the benefit a user derives for a job that executes for \( L \) minutes. We assume that the user can pick from jobs (parameterized by a scalar target runtime). Thus, a value function maps the pair target and \( L \) to a value. We will compare the information models under two classes of value functions.

**Step value function:** depends on target duration, achieves a fixed value when duration reaches target.

\[
f(L) = \begin{cases} 
0 & L < \text{target}, \\
V_0(\text{target}) & L \geq \text{target}
\end{cases}
\]  

where the parameter target denotes the target duration, and \( V_0(\text{target}) \) the job value if runtime \( L \) exceeds target. Further execution accrues no further value. The step value

\[\text{Compared to LIFO, for example.}\]
function is a good model for batch and workflow computations. Maximum value \( V_0(\text{target}) \) scales linearly in \( \text{target} \).

**Step-with-ramp value function:** depends on the \( \text{target} \) duration (below), after the runtime exceeds \( \text{target} \), job value increases linearly.

\[
f(L) = \begin{cases} 
0 & L < \text{target}, \\
V_0(\text{target}) + 0.5 \times (L - \text{target}) & L \geq \text{target},
\end{cases}
\]

(3)

The step-with-ramp value function models machine learning and numerical optimization workloads where an initial \( \text{target} \) duration is needed to get a feasible solution and subsequent iterations give steady solution improvement at 0.5x rate.

These two user-value functions emphasize the value of effective targeting, supporting our goal to compare information models. We do not consider submission of additional jobs into residual interval durations beyond the \( \text{target} \) or checkpointing; with such models user-value depends largely on volatile resource capacity and only weakly on the information model.

**User Targeting:** Users select the target duration that maximizes value statistically. Experiments assume sufficient demand to utilize all volatile resources [3].

1) **MTTR:** \( \text{target} = 0.9 \times (\text{Mean interval length}) \) If StDev is 10% of MTTR and distribution is Normal this target gives a success rate of 67%.

2) **10pctile:** Select \( \text{target} \) to maximize expected value assuming Normal distribution

3) **90pctile:** Select \( \text{target} \) to maximize expected value assuming Normal distribution

4) **Full:** Select \( \text{target} \) to maximize expected value given the Full distribution of interval durations

**Metrics:** We use several metrics:

1) **Total Available Volatile Resources (resource-hours):** Sum of available instance hours for the study period (in percentage of total hours for the study period), used to compare different instance types.

2) **Total Value:** Sum of the value of all the jobs run on the intervals of the instance type.

**B. Trace-driven Evaluation**

We employ trace-driven simulation to evaluate information models under a variety of resource management algorithms. For each, we compare the achieved user value. We use 3-month industrial cloud price traces for all of the Amazon EC2 Spot Instance types North America (608 types) as mentioned in Section II-B and each instance type of one availability zone from one region is seen as independent. While we have comprehensive results for these instance types, they are too large to present completely. From these, we select two exemplar instance types for exposition, **Stable 1 and Stable 2**, using the 2-dimensional statistical characterization shown in Figure 4. The plot places all 608 instance types by volatility (standard deviation of the resource availability) and periodicity. These two measures capture important characteristics commonly found in cloud resources, such as diurnal periodicity. We later show that these two exemplars are typical of nearly all of the EC2 instance types (see Section III-D). We also consider one additional exemplar — **Periodic (diurnal)** that exemplifies the

---

*Diurnal Periodicity measures the 24-hr periodicity. It is computed as the coefficients of fast-fourier transform (FFT) of the resource availability profile at one over 24-hours frequency to the average across the entire frequency domain.*
daily variation common in many user-responsive workloads. Key statistics for each exemplar are shown in Figure 3. The interval statistics for each instance type vary with VRM, and quite extremely for Periodic. The two graphs at right present averages across VRM’s, and show that Stable 1 has both generally longer intervals and more available resource hours than both Stable 2 and Periodic. A similar relationship holds between Stable 2 and Periodic.

C. Information Models and User Value

We compare four information models (MTTR, 10th percentile, 90th percentile, Full distribution) for Stable 1 and Stable 2 exemplars, using trace-driven simulation, and report total user value. Results are shown in Figures 5 and 6.

In Figure 5, step value function experiments show that all information models increase value relative to MTTR for Stable 1 and Stable 2. The simple information models (10pctile, 90pctile) combined with a good VRM such as LIFO or LIFO-pools yield 20% to 45% value increase. Of these, 90pctile is best, producing up to 45% increase in value. To our surprise, the use of the Full statistical distribution increases value only modestly beyond 90pctile. Full is considered unacceptable by cloud providers. The rightmost graph in Figure 5 summarizes the average value across exemplars and VRMs for each information model. The average smooths out the individual VRM effects but shows the general superiority of 90pctile over the other limited information models.

The step-with-ramp value function experiments (Figure 6) show higher value because the value function is more generous. However, the results clearly show that the addition of ramp mutes the performance differences between information models. Despite that, the ordering of the information models remains the same. In both cases, all of the information models increase value, but amongst the information models that disclose a small amount of information, the 90pctile gives the highest benefit. The size of benefit suggests that cloud providers should consider providing statistical characterizations of their volatile resources.

D. Relating Exemplars & All Instance Pools

We have done full studies of interval statistics, VRM, and information models on 608 Amazon EC2 resource pools. These results show that the majority of the pools are substantially similar to our two stable exemplars. In Figure 7, we present the mean and median results across the 475 resource pools closest to Stable 1 and Stable 2 in our 2-dimensional volatility, periodicity space [5]. These results show that many of these pools are similar to the two Stable exemplars. For example, for mean user-value, the results are the same ordering for VRMs and information model as in Stable 1 and Stable 2, and likewise, but with smaller gaps.

5 Of the 608 pools, we removed 5 that had no price changes, and the two Stable exemplars, so 475 is approximately 80% of the remaining pools.

for the median. Standard deviations across the collection of instance pools are small, showing the significance of the differences.

In Figure 8, we present results for the 130 resource pools furthest from our two exemplars. These pools are much more unstable, and this is reflected in the generally lower values achieved as well as the larger standard-deviation to value ratios. Remarkably, these pools show an even stronger benefit for 90pctile.

IV. Maintaining Statistical Guarantees

To explore the question, how does the information model (statistical guarantee) constrain resource management?, we consider changes in foreground load and their impact on interval statistics. Subsequently, we consider several resource management approaches that seek to preserve the statistical guarantees and thereby improve users’ experience. The goal is to address a cloud provider’s concern that sharing statistical characterization of volatile resources (information models) would limit flexibility. We focus on the 90pctile information model with LIFO resource management because together they produced robust, good performance in Section III-B. We study two plausible classes of foreground load changes in Section IV-A and propose online resource management algorithms to maintain statistical SLA in Section IV-D using Stable 2 and Periodic for exposition. We evaluate them in detail in Section IV-E and return back to 608 instance types in Section IV-F.

A. Varied Foreground Loads

Consider two types of foreground load changes, frequency (more rapid change) and magnitude (higher average foreground load). Increasing the frequency of variation alters the interval length distributions radically (see Figure 9a and 9b). Higher frequency (F) produces more short intervals, reducing the 90pctile of a 2-week sliding window dramatically, and violating the 90pctile statistical guarantee, shown in Figure 9c and 9d. Interestingly, increasing the foreground load magnitude to reduce available volatile resources to one-half and one-third its prior quantity has no impact on interval length distribution (see Figure 10). Magnitude changes only reduce the number of resources (intervals) available. The 90pctile graphs in Figure 10 all show that the 90pctile statistical guarantee is maintained.

<table>
<thead>
<tr>
<th>Frequency</th>
<th>Increase variation frequency F-fold by contracting the availability trace (F = 2, 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magnitude</td>
<td>Change magnitude of foreground load to reduce volatile resources at each time to 1/Kth (K = 2, 3)</td>
</tr>
</tbody>
</table>

So, if a cloud provider released statistical characterization of an instance type under the 90pctile information model, they need not worry about magnitude changes in foreground load – the statistics will not change. However, if foreground load shifts to higher frequency of variation, 90pctile statistics
will change. We next explore if clever resource management can help to maintain the statistical guarantees and user experience in the face of frequency change.

B. Methodology

We study F1 (base), F2 (2x faster), and F3 (3x faster) scenarios for increased frequency variation of foreground load, and three resource management algorithms. As in Section III-B we use the 90pctl information model based on the 3-months 90pctl interval duration. We evaluate the resource management algorithms using the following metrics.

**Metrics**

1) **2-week 90pctl**: (2W90pctl) 90th percentile interval duration for a 2-week sliding history window.
2) **Guarantee Fraction**: Fraction of time the 2-week 90pctl (2W90pctl) meets the 90pctl guarantee.
3) **Guarantee Violations**: Fraction of time the 2-week 90pctl (2W90pctl) fails to meet the 90pctl guarantee.
4) **Waste:** Our guarantee-preserving algorithms can withhold resources (delay of X), these resources are wasted (users cannot exploit). The units are resource-hours.

5) **Job Failures:** number of jobs that are terminated before they can achieve any value (return value of 0).

6) **Success Rate:** The fraction of jobs started in an instance type that delivers value > 0. (that is, they don’t fail)

**C. Offline Static Preserving Algorithm**

To explore the *feasibility* of preserving guarantees under foreground load change, we use an offline algorithm that searches for a single value X, and each volatile resource is withheld for X minutes before being released to users. Because we are using LIFO, this withholding reshapes the interval distribution by eliminating short intervals and reducing the durations of the remaining intervals by X (see Algorithm 1). For each resource withheld, we waste X resource-minutes. Algorithm 1 finds the smallest value of X, and thereby minimizes the resource waste.

Withholding resources improves the guarantee fraction. To minimize waste, we find the smallest X that achieves 100% guarantee fraction as shown in Figures 11 and 12. The algorithm eliminates short intervals, shifting the 90th percentile to a longer duration. Static Offline restores the statistical guarantee, increasing the 90th percentile duration to 2,880 minutes (see Figure 13) at the cost of 7% resource waste.\(^6\)

```
Algorithm 1 Offline
1: X ← 0
2: procedure DELAY(X)
3: define A_i := Volatile resource availability for time T = i
4: for i in (0, T_max) do
5:   d = A_i - A_i-1
6:   if d < 0 then revoke resources
7:   else if d > 0 then more usable volatile resources
8:     at T = i + X, make d resources available
9:   end if
10: end for
11: end procedure
12: while (∃i: 2W90pctile(i) ≤ 90pctileGuarantee) do
13:   X = X + 1
14:   DELAY(X)
15: end while
16: return X
```

**D. Online Dynamic Preserving Algorithms: AIMD and Targeting**

The Offline algorithm is not implementable, as choosing X requires prescience. However, Offline algorithm results show it is feasible to fully preserve the statistical guarantee by withholding each resource for X minutes. Of course, trimming X idle time causes original intervals between length [target, target+X] to become job failures, so the value of X is critical. Therefore, we further explore two algorithms: choose X based on history (online), and adapt X with time (dynamic).

*Online Dynamic AIMD:* performs additive increase and multiplicative decrease for X in response to whether

\(^6\)Note that all of the intervals wasted are less than one hour, below Amazon’s billing threshold for Spot Instance.
Figure 11: Stable2 F3 - Guarantee Fraction and Waste vs X (Offline)

Figure 12: Periodic F3 - Guarantee Fraction and Waste vs X (Offline)

the 2-week sliding window distribution meets the statistical guarantee (see Algorithm 2). It is inspired by TCP [19]. Specifically, Online AIMD updates X at each interval as follows: (1) If distribution fails to meet the guarantee, X=X+c, where c=1 time unit and (2) If distribution meets the guarantee, X=X/2.

Online Dynamic Targeting: adjusts X towards what would have been optimal for the past two weeks – the smallest X that would have met guarantee (see Algorithm 3). If X was too small, X is adjusted three-fourths of the way to optimal. If X was too large, X is decreased one-fourth of the way to optimal. For both online algorithms, we cap X at 12 hours (1/4 of max interval duration).

E. Evaluation: AIMD and Targeting

We evaluate two online algorithms based on metrics defined in Section IV-B to measure the ability to restore statistical guarantees and maintain the quality of user experience.

Reducing Periods of Guarantee Violation: We evaluate the online algorithms, using the metric guarantee fraction for the 90pctile information model. Both online algorithms can increase the 90pctile, and restore the guarantee for Stable 2 (Figure 14) for F3. For Periodic (Figure 15), the situation is more difficult; both algorithms move the distribution toward restoring the information model guarantee for substantial periods, but cannot perfectly restore the guarantee. Overall, online algorithms are able to double the 90pctile to near the original guarantee of 2,880 minutes for F2. The online algorithms triple the 90pctile for F3.

Reducing Job Failures: Recall that user targeting selects a job duration for each interval; if the interval is too short, the job fails. Frequency change in foreground load changes the interval distribution, causing many more failed jobs in F2 and F3 (see Baseline in blue in Figure 8).

Algorithm 2 Online AIMD

\[
\begin{align*}
X &\leftarrow 0 \\
2: & \text{procedure AIMD}(i) \\
& \text{define } A_i := \text{Volatile resource availability for time } T = i \\
4: & \quad d = A_i - A_{i-1} \\
6: & \quad \text{if } d < 0 \text{ then} \\
8: & \quad \text{revolve resources} \\
10: & \quad \text{else if } d > 0 \text{ then} \\
12: & \quad \text{at } T = i + X, \text{ make } d \text{ resources available} \\
14: & \quad \text{if } \text{flag}[i] = (2W29pctile(i) \geq 90pctileTarget) \\
16: & \quad \text{else if } \text{flag}[i..i-4] == \text{true then} \\
18: & \quad X = \min(X + 1, 12\text{hour}) \\
20: & \quad X = X/2 \\
22: & \quad \text{end if} \\
24: & \text{end procedure}
\end{align*}
\]
Algorithm 3 Online Targeting

\[ X \leftarrow 0 \]

procedure FINDINC(CurX, Dist, 90pctileTarget)
3: \textbf{return} optimal \( X_i \) that increases \( \text{Dist} \)'s 90pctile to match 90pctileTarget
end procedure

procedure FINDDEC(CurX, 90pctileTarget)
6: \textbf{return} optimal \( X_d \) that decreases \( \text{Dist} \)'s 90pctile to match 90pctileTarget
end procedure

procedure TARGETING(i)
9: \textbf{define} \( A_i := \text{Volatile resource availability for time } T = i \)
\[ d = A_i - A_{i-1} \]
\textbf{if} \( d < 0 \) \textbf{then} revoke resources
12: \textbf{else if} \( d > 0 \) \textbf{then}
\[ \text{at } T = i + X, \text{ make } d \text{ resources available} \]
end if
15: \textbf{flag}[i] = (2W90pctile(i) \geq 90pctileTarget)
\textbf{ADJUST} = distribution(2-week sliding window)
\textbf{if} \textbf{flag}[i] == false \textbf{then}
18: \( X_i = \text{FINDINC}(X, \text{ADJUST}, 90\text{pctileTARGET}) \)
\( X = \min(X + 3/4(X_i - X), 12\text{hour}) \)
\textbf{else}
21: \( X_d = \text{FINDDEC}(X, \text{ADJUST}, 90\text{pctileTARGET}) \)
\( X = \max(X - 1/4(X - X_d), 0) \)
\textbf{end if}
24: \textbf{end procedure}

Figure 15: 90pctile, Online algorithms improvement (blue) and remaining (gray) for Periodic

(a) Periodic - F2, Online AIMD  (b) Periodic - F2, Online Targeting  (c) Periodic - F3, Online AIMD  (d) Periodic - F3, Online Targeting

Figure 16: Disruption dramatically increases user job failures; Online algorithms reduce failures.

Figure 17: User Job Success Rate - Online Algorithms (All Pools)

Figure 18: Average user-value over all 608 instance types, all bars normalized to the F1 baseline.

\[ L_6 \] representing results with different frequencies and no algorithms applied.

The frequency change increases the number of short intervals, and because applications have no information about the changed behavior of the instance type, their targeting remains the same. Consequently, they experience a dramatic increase in failures. The online algorithms eliminate short intervals (sure to cause job failures), dramatically reducing job failures by nearly 50%. Reducing failures also improves user job success rate (see Figure \[ L_7 \], with significant increases of 20-30\% for F2 and F3 with online algorithms\[ L_8 \]).

Value: In Figure \[ L_9 \] we present average user-value over all 608 instance types, all bars normalized to the F1 baseline. The baseline bars show a steady modest decrease in total value from F1 to F2 to F3. As we saw earlier, applying the online algorithms increases job success rate and the quality of the application experience. Here Figure \[ L_{10} \] shows that these benefits and the shifted interval distribution come at the cost of only a slight decrease in user value. That is, the online algorithms are able to preserve much of the users' "quality of experience" in maintaining the same targeting and increasing success rate while maintaining resources' user value\[ L_{11} \].

\[ L_7 \] Note that it is correct that these metrics are not inverses, as the number of intervals is changing significantly.

\[ L_8 \] Note that the small decrease in user value arises from rare excursions where the online algorithms cause \( X \) to become large.
Figure 18: User Value - Online Algorithms (All Pools)

Figure 19: Classifying Instance Types: Stable, Transition, and Unstable classes have distinctive guarantee violations and 90pctile guarantee behaviors

F. Drilling Down into types of Spot Instance Types

To gain further insights, we divide the 608 Spot Instance types into three classes – Stable, Transition, and Unstable – based on their ability to deliver the 90pctile guarantee for 2-week sliding windows. As shown in Figure 19, the Stable and Transition instance types have high (48 hours) 90pctile guarantees. The 400 Stable instance types, always deliver the guarantee for a 2-week sliding window, including example Stable 2. The 177 Transition instance types deliver the guarantee some of the time and include the Periodic example. The 31 Unstable ones (rightmost in Figure 19) both have very low 90pctile and consistently fail to deliver it. Because the Unstables are largely unusable, we focus on the 577 Stable and Transition types.

Figure 20: Guarantee Violations percentage for Stable and Transition instance types

Figure 21: User Job success rate, Online algorithms

Applying the Online algorithms reduces Guarantee violations significantly (see Figure 20), with over 50% reductions in the Stable instance types for both F2 and F3. Significant reductions in Transition instance types are achieved for F1, F2, and F3.

One of our most important goals is to maintain the quality of user experience in the face of foreground load change. To explore online algorithms effectiveness, we present user success rates for the Stable and Transition instance types in Figure 21. The Stable types experience little disruption, so they see only modest improvement from the online algorithms. However, the Transition instance types see a larger disruption, and the data shows they see a large improvement. Consequently, they account for a large fraction of the overall benefit.

V. Related Work

Research exploring volatile resources covers use and resource management – statistical characterization and duration prediction, efficient exploitation, and efficient cloud resource management. We take the perspective of a cloud operator and explore VRM and information models as statistical guarantees.

Volatile Resource Characterization No commercial volatile resource services provide statistical characterizations of resources; numerous academic efforts have sought to do so. Studying Amazon’s Spot markets empirically, deriving models from price [15] and revocation behavior [20]. Brevik and Wolski [9, 10] used time-series models to predict the bid price needed for spot durations of a given reliability. SpotLight [21] is an information service that probes the cloud platform to infer and quantify availability. Our efforts explore how different information models affect achievable user value, complementing their work. Amazon’s recent defined duration instances allow specification of 1 to 6 hours [16] – a limited information model. Amazon’s Spot Instance advisor [22] provides the information of frequency of interruption. However, the strategy for users can adapt based on this type of information is yet unclear. Cloud providers provide little and appear to be reducing available pricing information [11, 12, 13], fearing loss of resource
management flexibility, and leakage of competitive information.

**Engineering Reliable Resources** Studies have employed mechanisms such as checkpointing [23, 24], replication [25], and migration [26] to shield applications from resource volatility. Historical empirical characterization of volatility is used to reduce the overheads. For example, Carvalho et al. [27] construct “economy class” computing from unused data center resources using statistical characterization of foreground load to create nearly reliable resources. In contrast, our work uses VRMs to engineer the volatile resource properties and explores viable information model design for intelligent user-management.

**Value of Information** Shastri et al. [28] use LIFO resource management to divide resources on a single Google data center trace into volatile resources characterized statistically using MTTR (called “transient guarantee”).

Our work goes further. We study four information models (four different “transient guarantees”), showing how the choice affects user value and that 90pctile yields much greater value. We continue further, exploring guarantee-preserving VRM.

**Volatile Resource Management to Preserve Statistical Guarantees** Traditional resource managers do not address this problem, matching resource requests to a fixed set of resources based on heuristics with little guarantee [29, 30]. We know of no other efforts to build statistical guarantee preserving resource management algorithms – one of the key new ideas in this paper. Our AIMD control approach is inspired by TCP’s congestion control [19]. The insight that preserving statistical guarantees requires wasting resources also comes across in user-side resource management through conservative bidding [9], and workload placement in clusters for latency guarantees, e.g., [31]. Careful management of volatile resources for value is similar to background load management in systems such as [32].

More extensive results and background for the work contained in this paper can be found in [33, 34].

VI. SUMMARY AND FUTURE WORK

We have studied 608 Amazon resource pools, characterizing their periodicity and availability based on spot price traces. Using these insights, we evaluate four information models (statistical guarantees) for users, showing that even limited information (MTTR + 90pctile) can have a large impact on achieved value, up to ~2X. These results hold for the vast majority of resource pools, approximately 475 of 608 (about 80%). Cloud operator release of a small amount of information can dramatically increase users ability to capture value.

To go further, we address the cloud provider concern that any statistical information will constrain future ability to serve foreground load. We show two types of foreground load change – one violates information model guarantees and one that does not. For the violated case, we study offline and online algorithms that seek to maintain a 90pctile guarantee. The algorithms not only help maintain guarantees and user-value but also improve the user experience by reducing job failures by 50%. Our results show the online algorithms give benefit for 95% of the 608 resource pools.

While our results are a promising first step, they suggest design of an information model that maximizes value whilst protecting cloud provider’s internal proprietary secrets is an interesting topic for future research. We study volatile resources from one provider on two disruption models in one window and time with two value functions, and further study could explore a broader range within the space. Others future directions include more complex models of user utility and predictive models to more finely characterize resources.

On the guarantee preserving side, one could further exploit using algorithms to increase current statistical guarantees for better value.

ACKNOWLEDGEMENTS

This research was supported by in part by the National Science Foundation under Awards CNS-1405959 and CMMI-1832230. Support from the CERES Center for Unstopable Computing is also gratefully acknowledged.

REFERENCES


