How to Increase the Value of Volatile Cloud Resources:
Resource Management and Information Disclosure

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

Varun Gupta
Dept. of Computer Science
University of Chicago
guptav@uchicago.edu

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

Abstract

Cloud providers sell unreliable or “volatile” resources that are unused by foreground (reserved/high priority) workloads. The value users can extract from these resources depends on (i) the volatile resource management algorithm, and (ii) the information provided to users about the volatile resources. We describe and evaluate four volatile resource management approaches (Random, FIFO, LIFO, LIFO-pools) using commercial cloud resource traces drawn from 608 Amazon EC2 instance pools. We also consider several information models (MTTR, limited statistics, Full distribution, and Oracle) that statistically characterize the resources for users.

Our results show volatile resource management algorithms can increase user value by 30 to 45%. Slightly richer information models (90pctile) combined with LIFO and LIFO-pools volatile resource management increase user value by as much as 10x. Our results suggest that cloud providers should pay significant attention to what statistical information they provide to users. And, these results broadly characterize the vast majority (475 of 608) of instance pools. Finally, we provide a detailed drill-down showing how the information model shapes user targeting, success rate, and user value.

1 Introduction

Modern cloud datacenters are continually expanding their computing resources to meet growing needs for e-commerce, web search, social networking, and big data analytics. IT leaders have rapidly adopted public cloud resources that match traditional reliable resource models such as on-demand and reserved instances from Amazon’s Elastic Compute Cloud and Google Compute Engine [1, 5]. As the cloud market grows, these operators continue to build capacity to meet unpredictable load spikes, peak, and growing demand. Accurate forecasting is difficult, producing fluctuating quantities of excess resources as in Figure 1.

Figure 1: Cloud operators serve a varied foreground load, producing a variable “excess” resources.

To increase resource utilization, cloud providers sell excess resources to increase revenue. For example, all cloud providers practice overprovisioning, the overcommitment of resources to increase revenue, and if done well it has negligible customer performance impact. Reserved instances guarantee a customer access to a resource, but those resources are frequently idle. Excess resources are often sold as unreliable cloud instances. These resources vary in name (spot instances, Preemptible virtual machines), but share the property that they can be revoked as illustrated in Figure 2. This produces a more complex volatile resource usage interface. Requests for volatile resources can be delayed or fail. Even after a successful request and ongoing use, a volatile resource can be revoked, causing loss of some application work. Because of this potential revocation, we call these resources “volatile”.

The ability to reclaim volatile resources enables cloud providers to meet ramps in foreground (high priority) load. The design of volatile resources varies across cloud providers. For example, prices differ – Google’s Preemptible VM’s have a fixed discounted price[7]; Amazon EC2’s Spot Instances use an auction bidding mechanism[3]. However, all volatile resources exhibit re-
Figure 2: High-level view of how Users and Cloud providers interact on Volatile and Reliable Resources vocation, so users (applications) must monitor resources, respond to revocations, and ensure application progress.

Volatile resources have been available at scale since 2009 [3], so many studies explore efficient application use. For example, a number of efforts focus on bidding strategies in Spot Instance markets [20, 21, 12, 30] or predicting price dynamics [14, 31]. These bidding strategies attempt to keep instances running, but because they fail, other systems employ migration[16, 28] and checkpointing [29, 13, 15, 17, 28] to save application work, and enable application resumption after revocation. Some efforts attempt statistical characterization and prediction [25, 26], and another proposed the idea of using resource management to shape volatile resource properties, providing MTTR as a statistical characterization to increase user value [19]. Our work builds on these ideas.

Volatile resource properties are not natural; they are produced by interaction of foreground load and volatile resource management algorithms. Yet, the temporal, statistical properties of the volatile resources affect their value to users. We call the information that a cloud provider reveals about the volatile resources an “Information Model”. Thus, a cloud provider faces two key questions:

1. What volatile resource management algorithm maximizes the value of my excess resources?
2. How does the information model for a volatile resource pool affect resource value for users?

In this paper, we formulate and explore the question of volatile resource management that creates volatile resource pools. We study several volatile resource management (VRM) algorithms, evaluating their impact on interval distributions, variability, and value. We also vary the information model, exploring the tradeoff between enabling accurate user targeting and the desire to shield proprietary data center specs. Because cloud providers do not release resource management data, we derive resource availability profiles from a large collection of 608 Spot Instance price traces, drawn from the full breadth of Amazon’s EC2.

Our results show that volatile resource management has dramatic effect on the statistical properties of volatile resources, and that good choices can significantly increase user value. Our studies also show the choice of information model is critical for effective volatile resource exploitation.

Specific contributions include:

1. Formulation of the volatile resource management and information model design problem.
2. Using four exemplar instance pools, design and evaluation of four volatile resource management algorithms (Random, FIFO, LIFO, and LIFO-pools), including impact on resource availability intervals and variability. The best-performance VRM algorithm, LIFO-pools, can increase user value by 30-80% (MTTR info model), and up to 2-10x across info models and exemplars.
3. With four exemplars, design and evaluation of information models, MTTR and 10pctile produce poor user value because of the skew in resource interval distributions, but 90pctile approaches the value of providing the Full distribution, outperforming the others.
4. A drill-down on interval statistics, showing VRM resource properties, and how information models inform user targeting and improve success rates for higher user-value.
5. Study of 608 instance pools, Amazon EC2’s full US breadth, characterizing of periodicity and availability. Full analysis of these pools with the four VRMs and info models, showing that our results for the two Stable exemplars are representative of 80% of the instance pools.

The rest of the paper is organized as follows. Key background is covered in Section 2. Section 3 describes our volatile resource management and information model approaches. Methods are covered in Section 4. Section 5 evaluates these approaches using commercial cloud trace data, followed by a drill-down on a number of key questions in Section 6. In Section 7 we discuss related work, and then summarize results and future research directions in Section 8.

2 Background

2.1 Volatile Resources in Commercial Clouds

Cloud operators have introduced volatile resource services to utilize and garner revenue from idle resources. However, the way they are offered differs greatly. Table 1 summarizes several major volatile resource services on the market. Amazon’s Spot Instances are offered and revoked based on a market mechanism where users place bids, and resources are allocated to users in descending
bid price order. Given the quantity of available volatile resources, eventually they are exhausted, and the lowest bid receiving a resource defines the market clearing price. In other words, all bids greater than or equal to the market clearing price receive resources [3]. All active Spot Instances pay the market clearing price for a given hour. Significant academic research has explored the Spot markets [8, 14]. Amazon has disclosed no further detail. We use the 90 days of clearing price history that Amazon provides to inform user bidding.

<table>
<thead>
<tr>
<th>Volatile Resource Service</th>
<th>Duration Info</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spot Instance</td>
<td>None</td>
<td>Revoke on price rise</td>
</tr>
<tr>
<td>Defined Duration Instance</td>
<td>Select duration (≥ 1, ≤ 6hrs)</td>
<td>Pay fixed price per hour</td>
</tr>
<tr>
<td>Google</td>
<td>Preemptible VM</td>
<td>≤ 24hrs</td>
</tr>
</tbody>
</table>

Table 1: Cloud provider volatile resource services

Google offers Preemptible Virtual Machines with a much simpler pricing scheme. Preemptible VM’s have fixed prices, set at 20% of on-demand price for each virtual machine type; no information is provided about how revocation decisions are made, and Preemptible VM’s receive a notification 30 seconds before termination [6].

2.2 Traditional Resource Management

Traditional resource managers or resource management algorithms are important in any large scale computing infrastructure. Such resource managers typically match resource requests (with partial to nearly complete information about requirements) to a fixed set of resources that are owned by the resource manager. To do so, resource managers track resource status, load, and sometimes failure state, and employ a variety of information (lookahead into a queue of requests, priority, wait time, prediction, request characteristics) and sophisticated algorithms (space fitting, back-filling, online, simulation, etc.) to schedule the resource requests with multi-dimensional attributes (compute, memory, storage, parallelism, etc.) onto data center resources.

The problem we consider here differs in that volatile resources vary in quantity rapidly due to external forces (the fluctuations of the foreground workload demand). A volatile resource manager cannot control these, but rather must respond to them by adapting the outstanding set of resources granted to users. In many cases, this adaptation requires resource revocation. Such revocation is a rare, rather than common feature of traditional resource management systems. The lack of awareness or control over foreground load makes the task of volatile resource management significantly different from traditional resource management.

2.3 Resource Availability Profiles

Volatile resource availability is defined by datacenter capacity and foreground resource load. Both of these are considered to be highly proprietary trade secrets by all major cloud providers. Thus only limited summary information is available [24, 11]; insufficient to do the broad studies considered here. On the other hand, plentiful pricing information is available for Amazon Spot Instance markets. We exploit this pricing data to infer the resource availability traces.

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 types. Each instance type is an independent resource type with an arbitrary amount of resources associated to it, from four US Regions. For each instance type, this comprised 25,920 price samples, or 15.8 million prices overall. We used 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, the volatile resources are assumed to be zero; at the minimum price the volatile resources are assumed to be maximum (arbitrarily set at 5000 units, commensurate with a total data center capacity of 10000 units).  

This choice affects the absolute value of available resources, but
Hence, the resource availability at time $t$ is:

$$VR(t) = (\text{max price} - \text{price}(t)) \times \frac{\text{Max volatile resources}}{\text{max price} - \text{min price}}$$

For example, in Figure 3, traces in red show Spot prices for instance type r3.4xlarge,i3.4xlarge,r3.xlarge, and r4.16xlarge for a 3 month period for one availability zone of region us-west-2. The traces in blue show the inferred resource availability (shown in percentage of total capacity) updated every 5 minutes at a datacenter where 50% of the resources are always claimed by foreground load (on-demand and reserved instances).

### 3 Approach

We describe our approach to studying the two key questions posed earlier: the volatile resource management problem, that is, what VRM algorithm maximizes the value of the excess resources?; and the information model problem, that is, what information is needed by users to accurately target volatile resources and extract most value? We begin by decomposing the complexities of the volatile resource management problem, including the key requirements and challenges, and then describe the algorithms for volatile resource management and models for information sharing evaluated in this paper.

Figure 4: Volatile resource managers (VRMs) make decisions to allocate and revoke resources, producing volatile resource intervals of varying lengths.

#### 3.1 Volatile Resource Management

Available volatile resources are determined by the fluctuation of foreground workload (i.e. on-demand and reserved instances) (see Figure 4), and the task of the volatile resource manager (VRM) algorithm is to grant and revoke the available volatile resources. When a resource is needed for foreground load, the VRM must revoke a volatile resource, causing a volatile interval to end. When a resource is no longer needed by the foreground load, the VRM starts a new interval. Different VRMs can produce different interval distributions via their choice of which volatile resource to revoke first. For example, consider two resources $m, n$ that are added to the volatile resource pool at times $t$ and $t + i$, respectively. Subsequently, at times $t + i + j$ and $t + i + k$, there are new foreground load resource requests. By choosing which resource to be revoked first, the volatile resource manager can create two intervals of durations $\{i + j, k\}$ or $\{j, i + k\}$. Thus, choice of volatile resource management produces different volatile resource properties.

We consider four different volatile resource management algorithms:

<table>
<thead>
<tr>
<th>Volatile Resource Management Algorithms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>Revoke a randomly chosen resource from the active volatile resources.</td>
</tr>
<tr>
<td>FIFO</td>
<td>All volatile resources form a single pool. Revoke the volatile resource allocated first in time (FIFO, or oldest-first)</td>
</tr>
<tr>
<td>LIFO</td>
<td>All volatile resources form a single pool. Revoke the volatile resource allocated most recently (LIFO, or last-first)</td>
</tr>
<tr>
<td>LIFO-Pools</td>
<td>Separate volatile resources into five pools. Management within each pool is LIFO, and management across the pools is LIFO.</td>
</tr>
</tbody>
</table>

Of these algorithms, LIFO-Pools bears some explanations. A VRM can create multiple resource pools, designed to create more attractive interval properties for users. In this case, as necessary, the VRM selects a pool and a resource within it to revoke. LIFO-Pools creates a stack of $N$ pools, with the topmost having the shortest intervals and lowest resource availability. We use $N=5$ pools, as our experiments with more pools provided negligible further change. Further down the stack, both of these properties improve. The LIFO-pools VRM seeks to create narrower distributions of interval duration, enabling better targeting.

Data center evolution activities such as large-scale upgrades, system migrations, and system reconfigurations, require basic resource flexibility, so we cap maximum interval length at 48 hours for all our VRM algorithms. Additional key assumptions and requirements for the volatile resource management include:

1. **Cloud neutrality**: All volatile resource customers should be treated equally. That is, there should be no prejudice based on whether job came from MorganStanley or GoldmanSachs or internal cloud properties (analogous to net neutrality).

2. **Foreground load is a quantity requirement**: A foreground load resource request only specifies an instance type (cpu speed, cores, memory, local IO), not a particular physical machine.
3.2 Information Models

Information about the volatile resource interval durations affects users’ ability to capture value from them. The interval lengths for volatile resources in a pool can be characterized statistically, so the users maximize value statistically; for example, enabling users to select job runtimes to maximize the expected value. Current cloud operators disclose essentially no statistical information about their volatile resources\(^2\). However, cloud operators could disclose a statistical characterization of each volatile resource pool, as shown in Figure 5. While one might propose full transparency as a good policy for cloud providers, they are reluctant to reveal such information, viewing their internal resource pool sizes and resource loading as a critical competitive secret.

Users exploit the available information to target job runtimes, tune checkpointing strategy, etc. If the distribution of durations is highly skewed, simple information models may lead users to make poor decisions.

We consider several different information models, and evaluate how they affect the ability of users to derive value from resources.

<table>
<thead>
<tr>
<th>Information Models</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTTR</td>
<td>Basic statistics, MTTR (mean-time to revocation) of the resource pool.</td>
</tr>
<tr>
<td>10pctile</td>
<td>Median and the 10th percentile (duration for which 10% of the intervals are shorter)</td>
</tr>
<tr>
<td>90pctile</td>
<td>Median and the 90th percentile (duration for which 90% of the interval are shorter)</td>
</tr>
<tr>
<td>Full</td>
<td>Full histogram, equivalent to the probability distribution function of interval durations.</td>
</tr>
<tr>
<td>Oracle</td>
<td>Exact duration length for each interval, in advance (unrealizable).</td>
</tr>
</tbody>
</table>

\(^2\)See Section 7 for a discussion of defined duration instances.

3.3 Example VRM: Simple Availability Profile

To build intuition, we consider a simple resource availability profile (see Figure 6a) that includes a periodic daily ramp that might result from diurnal usage. Resources are claimed and released at a constant rate until the maximum or minimum levels of volatile resources is reached. The magnitude of variation is one-half the maximum volatile resource level. On reaching the minimum volatile resource level (2,500 in Figure 6a), the system remains there for a period of time before resuming its variation.

For this availability profile, FIFO volatile resource management would produce a narrow distribution with all intervals of length close to 1,800 minutes. Other resource management algorithms can produce interval distributions that vary in number of intervals, mean interval length, variance in interval length, and the overall distribution (see Figure 6b). In short, the volatile resource management algorithm can shape the interval duration distribution, and thus is a critical element in creating value from volatile resources.

4 Methodology

In this section we outline key elements of our evaluation methodology: user value functions (Section 4.1), how users optimize them for each information model (Section 4.2), and the metrics we study to compare the VRM and information models (Section 4.3).

4.1 User Value Function

The value function for a user’s job maps the actual interval duration assigned to the job to a value. We use a step function that depends on the target duration of the job as defined below:

\[
f(L) = \begin{cases} 
  v_0(\text{target}) & L \geq \text{target}, \\
  0 & \text{otherwise},
\end{cases}
\]

Figure 6: Simple Model
Comparing growth rates for variable value maximize value. is evident, accurate targeting is critical for applications to a good model for batch and workflow computations. As the target accrues no further value. The step function is time

\[ L \]

ure 7a). That is, the job delivers no value until the run-
target linear and v

malize the v

as overhead amortization, etc. For experiments, we nor-

higher value, larger problem sizes, as well as effects such
to the runtime. Polynomial scaling models computations
els computations whose value grows in direct proportion

\[ (t^1.5) \]

as shown in Figure 7b. Linear scaling models computations whose value grows in direct proportion

to the runtime. Polynomial scaling models computations
that benefit from longer runtimes enabling solution of higher value, larger problem sizes, as well as effects such
as overhead amortization, etc. For experiments, we nor-

value so that they are equal for the linear and \( t^1.5 \) scalings for \( t = 1 \) hour.

4.2 User Targeting Strategies

Given a value function, users can select target duration to maximize value based on statistical information available about volatile resources. We assume users employ the following targetting strategies for each information model:

1. **MTTR:** \( t = 0.9 \times \) (Mean interval length)
   Users normally checkpoint or run workloads that are slightly shorter than the given MTTR; Assuming standard deviation is 10% of MTTR and Gaussian distribution, targeting \( 0.9 \times MTTR \) would give a success rate around 67%.

2. **10pctile:** Select \( t \) to maximize expected value assuming Gaussian distribution with given 10th percentile and median

3. **90pctile:** Select \( t \) to maximize expected value assuming Gaussian distribution with given 90th percentile and median

4. **Full:** Select \( t \) to maximize expected value given the Full distribution of interval durations

5. **Oracle:** Users are told the exact interval duration and \( t \) it to extract maximum value (unrealizable)

Experiments assume that demand is sufficient to utilize all volatile resources – same assumption as the AWS market clearing [3].

4.3 Metrics

We use several performance metrics in our study:

**Volatile Resource Availability Profile:** Number of instances available at each 5-minute interval. Induced by foreground load; the input to the volatile resource manager.

**Total Available Volatile Resources (resource-hours):** the total available instance hours over the entire period of study, used to compare different instance pools.

**Interval Statistics:** the VRM induced interval statistics for an instance pool – includes mean (MTTR), %-tiles, distributions, standard deviation.

**Target (minutes):** Target duration selected by user given a resource pool information model.

**Success rate:** Fraction of computations achieving non-zero value.

**Total Value:** Sum of the computation values over intervals of an instance pool.

5 Evaluation

We employ a trace-driven method to evaluate volatile resource management algorithms (VRM), exploring their impact on interval statistics such as mean duration (MTTR) and standard deviation. We then compare achievable user value for each VRM, varying the information models and value function scaling.

Evaluation is based on industrial cloud price traces taken from Amazon’s EC2 for a 3-month period [4, 26]. We studied volatile resource management on 608 instance types, drawn from Amazon’s four US regions, each of which contains 2-6 availability zones. Comprehensive results for these resource pools are too large to be presented, so we identified four exemplars based on two statistical dimensions: volatility (standard deviation of the resource availability) and periodicity (ratio of fourier coefficients for a period of 1 day vs. average)\(^3\) that capture important characteristics commonly found in cloud resources, such as diurnal periodicity. Scatterplots showing the distribution of instance pools along these two dimensions for the four US regions are shown in Appendix A.

Distribution of the 608 instance pools show two key similar characteristics: 1) a wide spread of volatility, but with many instance pools with low volatility, and 2) varied magnitudes of diurnal load variation. To reflect this distribution, we selected four exemplars of instance pools

\(^3\)Computed by taking the fast-fourier transform (FFT) of the resource availability profile, and then the ratio of the frequency bands
Table 2: Key statistics for Instance Pools Exemplars.

- Stable 1, Stable 2, Periodic(diurnal), and Unstable - for detailed study from the us-west-2 region, availability zone a. The scatterplot for us-west-2a is shown in Figure 8, and the selected exemplars are highlighted in red. The two Stable instances reflect the predominance of low volatility pools, and the Periodic and Unstable instances capture the range in those respective dimensions. We return to the large set of instance pools in Section 5.6, showing how the two Stable exemplars are representative of 80% of the Amazon EC2 instance pools.

Key price statistics for each exemplar are shown in Table 2. Price data for each exemplar is processed (as in Section 2.3) creating a resource availability profile (see Figure 3). Statistics for resource availability, and the number of intervals created by each VRM are shown in Table 3.

5.1 Volatile Resource Management and Interval Properties

The VRM algorithm creates availability intervals in a volatile resource pool. In Figure 9, we present the resulting interval duration statistics for the four exemplar pools, applying the four VRMs. The basic statistics include mean, median, and standard deviation. The x-axis shows three VRM’s (Random, FIFO, LIFO) on the left, and LIFO-pools on the right with five lighter-colored bars.

Table 3: Total Resource and Intervals Statistics (Exemplars).

- one for each sub-pool within LIFO-pools. In all cases the whiskers depict standard deviation. The VRM can make a large difference in all of these statistics, and the LIFO-pools’ partition into 5 sub-pools can create differentiated statistics for each sub-pool.

On the left side of Figure 9 is the Stable 1 resource pool. Stable 1 (r3.4xlarge) has small price fluctuations much of the time, and a handful of big price spikes in the 3-month period. Three VRMs (Random, FIFO, LIFO) produce similar mean and median interval durations. They also produce standard deviations close to their mean, so the variability of intervals durations is high. The standard deviation of LIFO is 1.3x greater than FIFO. LIFO creates both more short intervals and long intervals while FIFO creates more medium-length intervals. LIFO-pools differentiates pools. Pool 0 has a much smaller mean, and Pools 1-4 have much higher means and medians than both Pool 0 and the basic VRMs. All 5 sub-pools are significantly less variable, exhibiting lower standard deviation.

Stable 2 (i3.4xlarge) is similar to Stable 1 but has relatively more and stronger price fluctuations. Again, the three VRMs (Random, FIFO, LIFO) produce comparable means, but the medians tell a different story with LIFO producing dramatically more short intervals, and thereby a much smaller median (3x smaller than FIFO). Of course LIFO must produce a number of corresponding longer intervals to achieve a mean close LIFO. The three basic VRM algorithms produce standard deviations close to their means, so interval variability is very high. Consistent with Stable 1, LIFO-pools successfully separates resources into differentiated pools with Pools 0 and 1 exhibiting low means and medians and Pools 2-4 much higher and low variability means and medians compared to both Pools 0 and 1 as well as the basic VRMs.

Third subplot in Figure 9 is Periodic (r3.xlarge, a
Figure 9: Basic Interval statistics, 4 Instance Pool Exemplars (Stable 1, Stable 2, Periodic, and Unstable) and 4 VRMs diurnal load), exhibiting periodic resource availability changes. These patterns produce shorter mean and median interval lengths (∼1000 minutes or approximately 16 hours). As before, LIFO produces a mean similar to Random and FIFO, but much small median (10x smaller), and comparable standard deviation. Again, this effect is due to LIFO producing many more very short and very long intervals. LIFO-pools again successfully creates differentiated resource pools. Pools 0, 1, and 2 with lower mean and median interval lengths. Pools 3 and 4 have much higher means and medians than the basic VRMs (Random, FIFO, LIFO). Combined with a lower standard deviation, this makes Pools 3 and 4 much better resource pools. Note that this is not quite as good as Stable 1 where 4 pools were superior.

Finally the rightmost graph in Figure 9 presents interval statistics for Unstable (r4.16xlarge). Unstable has frequent, extreme price spikes, and many moderate price fluctuations. With such frequent spikes, all four VRM’s produce similar characteristics, comparable means and medians that are consistently lower than other exemplars. LIFO-pools fails to create distinct resource pools.

Thus, we see that VRMs shape the distribution of interval durations, and LIFO-pools can sometimes create differentiated pools. In the next section, we assess the impact of these interval duration properties on the ability of users to extract value.

5.2 Volatile Resource Management Algorithms and User Value

We compare our four volatile resource management algorithms (Random, FIFO, LIFO, and LIFO-pools), using the metric of derived user value for the Step value function as defined in Section 4.1. We simplify this initial comparison, providing only basic instance pool information to the user (MTTR or mean interval duration), and comparing to an ideal (“Oracle”), where the user is given precise duration information for each interval, as it begins, enabling perfect targeting. “Oracle” level performance is of course unachievable.

In Figure 10, we present the achieved user value. Along the x-axis, the major steps are the four exemplar instance pools with each of the VRMs clustered for convenient comparison. The y-axis is the achieved user value for the entire 3-month period. For LIFO-pools we present the aggregated value from the five sub-pools.

Beginning with the Stable 1 and Stable 2 exemplar pools we see that Random, FIFO, and LIFO all achieve comparable value, but LIFO-pools gives a 30% improvement. With the Periodic and Unstable exemplars, overall achieved value is much lower, reflecting the difficulty of exploiting these resources. For Periodic, FIFO and LIFO-pools give best value. For both Periodic and Unstable all of the four VRMs achieve comparable user value. For all exemplars, the best value is well short of “Oracle”, showing the impact of statistical unpredictability in resources.

A deeper look at the interval statistics for each of the VRMs and instance pools in Figure 9 can explain some of the differences in value. LIFO-pools separates small intervals and long intervals in different subpools, so the mean/MTTR captures the pool properties better, enabling more accurate targeting. We will explore this deeper in coming sections, considering different information models.

5.3 Information Models and User Value

Because volatile resources vary statistically, information about interval duration distribution is critical to maximize derived value. As mentioned in Section 4.2, we assume users do optimized targeting based on the infor-
information available to maximize value. Our four information models each reveal a different amount of information, and would require a cloud operator to choose to release such resource characterization information (perhaps leaking proprietary, competitive information). To inform such choices, we compare four information models (MTTR, 10th percentile, 90th percentile, Full distribution, and Oracle), combined with our volatile resource management algorithms.

In Figure 11, along the x-axis, we show four VRMs, and for each in a cluster, we vary the information model for easy comparison. Consider Stable 1, leftmost in Figure 11: the clear trend with all four VRMs is that richer information models increase value. Both Random and FIFO yield modest benefits as the information available increases, peaking at 0.55–0.67 of the ideal potential value (Oracle). Because FIFO has a low standard deviation, the different information models provide similar real information, producing comparable value. LIFO is quite different, increasing nearly 2x from the MTTR information model to Full, and achieving nearly 0.90 of the Oracle value. LIFO-pools is close to LIFO, starting higher, and reaching a similar peak of 0.90 of the Oracle. Stable 2, second from left in Figure 11 shows a more complicated, but similar story. Random and FIFO see small benefits from increased information, but LIFO and LIFO-pools see significant benefits and peak at 0.85 of ideal potential value.

For Periodic (second from right in Figure 11), Random and FIFO have similar properties, benefiting little from increased information. LIFO improves the delivered value as we vary information model, but shows that misleading information (10th percentile for a heavily skewed distribution) causes poor targeting, sharply reducing derived value. LIFO-pools delivers significant improvement, peaking at nearly 0.90 of ideal potential value, and avoids the misdirection for targeting. It is worth noting that the total volatile resources in Periodic is lower than in Stable 1 and Stable 2 (see in Table 3), so the lower ideal potential value is in line with expectation. For Unstable, shown in at far right, all four resource management algorithms do well with a Full information model, peaking at 0.67 of ideal with LIFO and LIFO-pools slightly better. They all do progressively better with 90th percentile models, showing around 0.45 of the Oracle value.

Looking across the full range of exemplars, it is clear that the information model makes a dramatic difference in user value, providing a 30% increase in numerous cases, and more than 10-fold in extreme cases. The Full distribution information model achieves the best user value (within realizable information models), as much as 1.5x better than the next best information model. FIFO and LIFO-pool are insensitive to information models; on the other hand, LIFO’s value depends strongly on the information model. Information about the 90th percentile is robust across both Stable and Periodic instance pools, because LIFO produces distributions that are skewed long, making it an attractive tradeoff of limited information and good value. To achieve value as close to Oracle, a good combination of VRM and information model is needed.

In Figure 12 we present user value results for polynomial scaling of the step value function as described in Section 4.1. Several differences from linear scaling in Figure
11 are worth noting. First, because we scaled the linear and target growth of step value to match at 1 hour, the total achievable value for cases shown in Figure 12 are arbitrarily much higher and should not be compared directly with those in Figure 11. Second, under polynomial scaling, the user value achieved under the “Oracle” information model differs across volatile resource management algorithms within each exemplar. This is because the LIFO algorithms create more long intervals, and as the step value increases superlinearly, this causes an increase in the total potential value for the LIFO algorithms. Third, polynomial scaling does not change the qualitative results comparing information models or volatile resource management algorithms. In all cases, the 90th percentile and Full information models give best user value and achieve 50% to as much as 90% of Oracle value. Comparing the volatile resource management algorithms, LIFO and LIFO-pools are significantly better for the stable and periodic exemplars with LIFO-pools doing particularly well. For the unstable exemplar, results are similar to the linear scaling.

5.4 VRM and Information Model Benefits

Consider VRM algorithms normalized to Random, focused on instance pool Stable 1 (see Figure 13a). Good VRM can increase value beyond Random significantly, by up to 1.7x and LIFO-pools consistently achieves the highest value (1.5x increase), but LIFO is good as well. Consider information models normalized to MTTR, also focused on instance pool Stable 1 (see Figure 13b). All of the information models increase value, including the simple information models (MTTR, 10ptile, 90ptile). Combining a good VRM such as LIFO or LIFO-pools yields 1.2x to 1.45x value increase with simple information models. Further while Full consistently achieves highest value, it may not be desirable for cloud providers. After Full, the best by a significant margin is 90ptile (up to 45% value increase).

5.5 Overall Four Exemplar Results

Our study of four exemplar instance pools demonstrates that both volatile resource management algorithm and information model can make a large difference in achieved user value. LIFO and FIFO vs Random can make more than 2x difference, and LIFO-pools provides the highest performance broadly up to 2-10x better compared to LIFO or FIFO. The information model is also critical suggesting that cloud providers should consider providing statistical characterization of their volatile resources (Amazon, Google, and Microsoft currently do not). Even simple statistics, such as 90th percentile, can increase achievable value by 10% to as much as 5x, and the Full distribution can increase it by as much as 10x. Design of an information model that maximizes value whilst protecting cloud provider’s internal proprietary secrets is an interesting topic for future research.

5.6 Relating Exemplars and All Instance Pools

We have done full studies of interval statistics, VRM, and information models on 608 Amazon EC2 instance pools. We believe these results show that the majority of the pools are substantially similar to our two stable exemplars. To demonstrate, in Figure 14 we present the mean and median results across the 475 instance pools closest to Stable 1 and Stable 2 in the normalized volatility, periodicity space. These results, normalized to Oracle, 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, for median. Standard deviations across the collection of instance pools are small, showing the significance of the differences.

Considering ordinal statistics between different VRM and information model configurations can also show how similar the pools are to Stable. The frequency of various relations that were our key conclusions for the Stable exemplars within the 475 instance pools are shown below.

---

4 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.
Figure 14: Mean, Median, and Standard deviation performance for 475 instance pools closest to Stable 1 and Stable 2 exemplars

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Relationship</th>
<th>Count</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIFO-Pools+Full</td>
<td>Top 1</td>
<td>468</td>
<td>98.5%</td>
</tr>
<tr>
<td>Full Excluded</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LIFO-Pools + 90pctile</td>
<td>Top 2</td>
<td>365</td>
<td>77%</td>
</tr>
<tr>
<td>LIFO + 90pctile</td>
<td>Top 2</td>
<td>388</td>
<td>82%</td>
</tr>
<tr>
<td>LIFO + 90pctile within 10% best</td>
<td></td>
<td>399</td>
<td>84%</td>
</tr>
<tr>
<td>LIFO + 90pctile within 10% best</td>
<td></td>
<td>403</td>
<td>85%</td>
</tr>
<tr>
<td>FIFO + 10pctile</td>
<td>within 10% best</td>
<td>200</td>
<td>42%</td>
</tr>
<tr>
<td>Random + 10pctile</td>
<td>within 10% best</td>
<td>223</td>
<td>47%</td>
</tr>
<tr>
<td>FIFO + MTTR</td>
<td>within 10% best</td>
<td>15</td>
<td>3.2%</td>
</tr>
<tr>
<td>Random + MTTR</td>
<td>within 10% best</td>
<td>14</td>
<td>2.9%</td>
</tr>
</tbody>
</table>

Within 10% of best is defined as greater than or equal to (90% * best value achieved). These results support that LIFO-Pools, Full is best overall in nearly all instance pools. If we exclude the Full information model as infeasible, LIFO-Pools and LIFO do best with the simple information models (90pctile best, 10pctile and MTTR) in more than 80% of these instance pools. The 10pctile and MTTR information models don’t achieve Top 2 performance in the majority of the instance pools. Random does poor uniformly. These ordinal results match our detailed analysis of Stable 1 and Stable 2 well.

6 Drilling Deeper

6.1 Information Models and Optimal Targeting

We have seen that choice of VRM algorithm and information model has direct impact on realizable value. To explore why, we consider how these choices affect optimal target duration and success rate. In Figure 15, along the x-axis, the major steps are the four information models with the four VRM algorithms in each cluster in different colors. For LIFO-pools, one target is shown for each sub-pool for a total of 5 target durations.

Leftmost in Figure 15 is the Stable 1 exemplar r3.4xlarge. Here LIFO-pools’ advantage in separating short and long intervals is clear, enabling better targeting. All other three VRMs have comparable target length, but LIFO pools 1-4 have up to 1.6X longer targets, even under weaker information models. A similar story plays out for Stable 2 exemplar i3.4xlarge (2nd from left), with LIFO-pools targeting quickly (and accurately) growing to 1.75X with even small amounts of information.

The same basic trend occurs for the Periodic exemplar (Figure 15, 2nd from right), but only LIFO-pools 2,3,4 have long targets, whilst targets for other volatile resource management algorithms are relatively low. The benefits of LIFO-pool, separating short and long intervals, and describing them well are clear. However, Unstable (Figure 15, right) is a different story. The VRMs and the information models fail to produce significant differences in target interval length. Even LIFO-pools cannot aid targeting.

6.2 Information Models and Success Rate

The complement of targeting is user computation success rate. Maximizing user value balances the value derived on successful completion (a function of target) against success rate. Thus, maximizing success rate does not necessarily maximize user value. With Stable 1 (see Figure 16, left), Random, LIFO, and FIFO never achieve high success rates, but the better targeting enabled by LIFO-pools achieves nearly 2X higher success rate. The reason for this is large standard deviations of the interval duration (recall Figure 9). For Stable 2 (see Figure 16, 2nd from left), a similar story prevails; LIFO-pools’ differentiated pools are slightly less favorable so the net benefit is approximately 1.5X.

For Periodic (see Figure 16, 2nd from right), FIFO can pick a long target, but because LIFO-pools can target five differentiated pools, the custom target for each one allows a clever trade-off, optimizing for a lower success rate in pools in lieu for a higher payback (larger target). Unstable is a different story (see Figure 16, right) as pool statistics that produce high variability cannot be effectively characterized, even with a range of information model and VRMs. So, the result is uniformly low success rates.
6.3 Exemplars and Realizable Value

A key question for volatile resource users is how the dynamics of the instance pools affect the ability to extract value. First, it is important to take into account the total available resource hours within each instance pool (see Table 3). In Figure 17, the plain green bars show the user value achieved using the Full distribution information model, divided by the total available resource hours in the corresponding pool. This normalizes out the total resource availability. Comparison across the four instance pool exemplars then shows clearly that the Unstable pool’s violent fluctuations make capturing value difficult, causing a nearly 30% decrease. The Stable 1, Stable 2, and Periodic exemplars exhibit a small decrease (< 10%), but are similar to each other. The dotted green bars in Figure 17 show the same ratio, selecting the highest value achieved across information models (the winner is 90th percentile in all cases), and shows a similar trend. This information model is more realistic, requiring only minimal information disclosure by the cloud provider. Despite its attractiveness, it does achieve a lower value due to the lesser information available to users for targeting.

7 Discussion and Related Work

A broad range of research explores use of volatile resources and cloud resource management. Topics include statistical characterization and duration prediction, efficient volatile resource exploitation in public clouds, and efficient cloud resource management from the perspective of cloud operators. In contrast, our work takes the perspective of a cloud operator, assumes foreground resource management for reliable cloud services as given, and explores a variety of volatile resource management algorithms and information models on 608 Amazon EC2 instance pools.

Characterization None of the commercial volatile resource services provide statistical characterization of resources, though numerous external and academic efforts have sought to create such characterization. Many papers have conducted empirical studies of Amazon’s Spot markets, constructing statistical models from historical price [21, 14] and revocation behavior [10]. Examples of key statistical variables are interval durations (MTTR), variation, and prediction of interval duration. Brevik and Wolski [25] observe that stationary statistics are a poor model, and use time-series techniques to predict the bid price
needed for reliable spot interval durations with a given reliability SLO (also see [26]). These efforts are hampered by the sparse information provided by cloud operators. In contrast, our work explores how different information models affect achievable user value, complementing these efforts. Amazon’s recent defined duration spot instances that allow a specification of 1 to 6 hours [2] are an example of a limited information model – where the duration distributions are all longer than the defined duration.

**Engineering Reliable Resources** A number of research studies have explored reliability mechanisms such as checkpointing [29, 13, 15, 17], replication [22, 27], and migration [28, 16] to shield applications from resource volatility. These techniques use historical empirical characterization of resource volatility to reduce the overhead of reliability mechanisms. For example, Carvalho et al. [9] construct “economy class” computing from unused data center resources using statistical characterization of foreground load to predict future volatile resource availability, thus creating a class of nearly reliable resources. In contrast, our work uses VRMs to engineer the properties of volatile resources, and explores the information model space cloud operators could provide to users to enable intelligent user-management approaches.

**Differentiated Pools and Information Models** Shastri et al. [19] describe a statistically characterized volatile resource, using MTTR, called a “transient guarantee”. Using LIFO resource management to divide resources (similar to LIFO-pools) on a single Google data center trace with different MTTRs. Using a superlinear value model, they showed significant resource value increases.

Our work goes further in several key dimensions. First, we consider not just MTTR but four information models (four different “transient guarantees”), showing how the choice affects user value and that two (90pctile, Full) yield much greater value than MTTR. Second, we go beyond LIFO, studying four different VRM algorithms, and showing that they create significant differences in interval distribution and thereby user value. Third, we study the VRMs and information models for over 600 instance pools. And fourth, we use a more challenging user value model. In Shastri, value increases superlinearly with MTTR, so partitioning that produces higher MTTR increases overall user value. However, our linear value and user-targeting is a more challenging hurdle. Increasing value requires reducing interval variability, which we deal with directly in statistical characterization, targeting, and success rate.

**Value Functions** Prior work [9, 19] that quantifies the value obtainable from volatile resources models computations as long-lived. Carvalho et al. [9] model the loss in value of a job assigned to a volatile resource in an ad hoc manner, asserting that availability < 99% entails a loss of 30% (mimicking Amazon EC2 charging scheme). Shastri et al. [19] endogenize the loss in value via the cost of checkpointing due to finite interval durations, and recomputation due to unpredictability of interval durations.

In contrast, we model computations as having finite durations without any checkpointing, assuming no uncertainty in the computation time. Thus, the value obtained by a user is a step function of the interval duration with the only degree of freedom being the maximum value obtainable as a function of the resources required (\(v_0(\text{target})\)), for which we pick two representative cases: linear growth (\(v_0(\text{target}) \propto \text{target}\)), and super-linear polynomial growth (\(v_0(\text{target}) \propto \text{target}^{1.5}\)). In [19], the authors use an exponential value function. In [18], the authors use a linear growth assumption, but assume that jobs have heterogeneous priorities with the slope of the linear growth an increasing function of job priority (however, that paper focuses on utility models that are a function of latency; see also [23]). We instead assume jobs of a single priority that can saturate the volatile resources. This allows a somewhat cleaner comparison of resource management policies and information models.

## 8 Summary and Future Work

The properties of the volatile resources that a cloud provider offers to users with weak service guarantee are not exogenous, but an outcome of the resource management algorithms used by the provider. Further, the ability of users to extract value from these resources depends on the nature of statistical information provided by the cloud provider. Our broad study of 608 Amazon instance pools characterizes periodicity and availability based on spot price traces. Using four exemplars, we study four volatile resource management approaches and four information models that can be provided to the users. Our study is based on Amazon EC2 spot price traces shows that the choice of resource management algorithms can have a large impact on the achieved value, up to \(\sim 2X\). We also show that simple statistical models do not capture real distributions of volatile resources, leading to low value to users. By comparing information models, we show that the ability of customers to extract value are highly-dependent on information model, with difference up to \(\sim 20X\). We also compare all schemes to ideal realizable value, and the results show that one can get very close to the ideal value with a carefully chosen combination of VRM and infomation model. We compared the results of the two Stable exemplars to 80% of the 608 instance pools, showing that results for these exemplars are broadly representative of these 475 pools. Thus, a cloud operator has to deliberately choose the volatile resource management approach and information model for users in order to maximize value.
There are numerous directions in which the present study can be extended. One direction is a more fine-grained value model of users’ utility incorporating uncertain job durations, their ability to checkpoint compute intensive jobs, and manage a portfolio of VMs across multiple instance types for parallel jobs. A second direction in which the study can be extended is to incorporate the cloud provider’s ability to predict idle resources instead of using historical aggregate statistics, and thereby providing more targeted service level guarantees for volatile resources. How such a better targeting can be achieved while not revealing the utilization and other characteristics of the cluster resources is also an open challenge. One resource management algorithm that we have not studied is “defined duration” type volatile resources where the cloud provider offers a hard upper bound on the duration of the intervals on the order of hours. Slicing longer intervals into such smaller intervals allows the provider to trade-off higher service level at the cost of lost value from longer intervals. Finally, taking a holistic view, it would be interesting to explore the cloud provider’s problem of jointly optimizing the menu of guaranteed (reserved/on-demand) resources and high quality volatile resources, the latter of which may cannibalize the demand for former.

References


Appendix A  Summary of All Instance Pools

All of the presented results are based on EC2 Spot Instance price trace[4, 26] over 3 month-long period. We performed a broad comparison of volatile resource management for 608 instance types, drawn from Amazon’s four US regions, each which contains 2-6 availability zones. We use two statistical characterizations, volatility (standard deviation of the resource availability) and frequency ratio (period of 1 day vs. average)\(^5\) The frequency ratio captures the diurnal periodicity of load often seen in cloud workloads. Scatterplots showing the distribution of instance pools for the four US regions are show in Figure 18,19,20, and 21 from all availability zones in us-east-1, us-east-2, us-west-1, and us-west-2 accordingly. These distributions show similar characteristics: 1) a wide spread of volatility, but with many instance pools with low volatility and 2) varied magnitudes of diurnal load.

Note that Figure 8 from Section 5 is a scatter plot of one particular availability zone from us-west-2, so it is a subset plot of Figure 21 here. X-axis denotes the standard deviation of availability, hence the availability volatility, and Y-axis represents the level of periodicity at a diurnal pattern.

\(^5\)We computed by taking the fast-fourier transform (FFT) of the resource availability profile, and then the ratio of the frequency bands.