MANAGING THE VALUE OF VOLATILE CLOUD RESOURCES: INFORMATION DISCLOSURE AND GUARANTEE-PRESERVING MANAGEMENT

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCE
IN CANDIDACY FOR THE DEGREE OF
MASTER

DEPARTMENT OF COMPUTER SCIENCE

BY
CHAOJIE ZHANG

CHICAGO, ILLINOIS
JULY 2018
# TABLE OF CONTENTS

LIST OF FIGURES ................................................................. v
LIST OF TABLES ................................................................. vi
ACKNOWLEDGMENTS .............................................................. vii
ABSTRACT ........................................................... viii

1 INTRODUCTION ................................................................. 1
   1.1 Volatile Resources in Public Cloud ......................................... 1
   1.2 Challenges with Volatile Resources ........................................ 2
      1.2.1 Exploiting Volatile Resources ...................................... 2
      1.2.2 Managing Volatile Resources ...................................... 3
   1.3 Approach ................................................................. 4
   1.4 Contributions ............................................................ 5
   1.5 Thesis Outline .......................................................... 7

2 BACKGROUND ................................................................. 8
   2.1 Public Cloud Services ..................................................... 8
   2.2 Volatile Resources in Commercial Clouds .................................. 9
   2.3 Applications on Volatile Resources ....................................... 10

3 APPROACH AND METHODOLOGY ............................................. 12
   3.1 Approach ................................................................. 12
       3.1.1 Volatile Resource Management (VRM) ............................. 12
       3.1.2 Information Models .............................................. 15
   3.2 Methodology ............................................................. 16
       3.2.1 User Value ........................................................... 16
       3.2.2 Targeting Strategy ................................................ 18
       3.2.3 Resource Availability Profiles .................................... 18
       3.2.4 Metrics .............................................................. 20

4 INTO 608 INSTANCE POOLS .................................................. 21

5 EXPERIMENTS AND EVALUATIONS ........................................ 25
   5.1 Experiments ............................................................. 25
   5.2 How VRM Shapes Intervals and User Value ............................... 25
   5.3 How Info Models affect User Value ..................................... 29
   5.4 Info Models, Targeting, & Success Rate .................................. 32
   5.5 Exemplars and Realizable Value ......................................... 34
   5.6 Relating Four Exemplars & All Instance Pools ........................... 34
   5.7 Summary ................................................................. 38
# LIST OF FIGURES

1.1 Cloud operators serve a varied foreground load, producing a variable “excess” resources. ................................................................. 1
1.2 High-level view of how Users and Cloud providers interact on Volatile and Reliable Resources ............................................................. 2
3.1 Volatile resource managers (VRMs) make decisions to allocate and revoke resources, producing volatile resource intervals of varying lengths. .... 12
3.2 Volatile resource managers (VRMs) can generate distinct pools. Information models for each can provide users with statistical characterizations. .... 13
3.3 Value Functions ........................................................................ 17
3.4 A 3-month trace of price history and derived resource availability profile. .............. 18
4.1 Scatter-plot of Instance pool pricing behavior from all availability zones ............. 22
4.2 Scatter-plot of Instance pool pricing behavior. .................................... 23
5.1 Basic Interval statistics, 4 Instance Pool Exemplars (Stable 1, Stable 2, Periodic, and Unstable) and 4 VRMs .................................................. 25
5.2 Comparing four VRMs, MTTR info model ........................................ 27
5.3 User value vs. Information model, Linear scaling of Step value ................. 28
5.4 User Value vs. Information model, Polynomial scaling($d_{i,5}$) of Step value .... 28
5.5 Relative Value increases for VRM and Info Model (Stable1) ....................... 31
5.6 Target Interval vs. VRM and Info Model ........................................ 32
5.7 Success rate vs. VRM and Info Model ............................................ 32
5.8 Relative User value per Resource-hour (VRM/Random) for the four exemplars. 34
5.9 Mean, Median, and Standard deviation performance for 475 instance pools closest to Stable 1 and Stable 2 exemplars ................................. 35
5.10 Average count exceeding other VRM+Info Models ............................ 35
6.1 Interval CDF and 90ptile Information Model changes for 4x Magnitude Foreground Load changes .......................................................... 39
6.2 Interval CDF and 90ptile Information Model changes for 4x Frequency Foreground Load changes .......................................................... 40
6.3 90ptile with Offline, Online algorithms for Stable 2 ............................ 42
6.4 90ptile with Offline, Online algorithms for Periodic ............................ 42
6.5 90ptile with Offline, Online algorithms for Stable 2 ............................ 45
6.6 90ptile with Offline, Online algorithms for Periodic ............................ 45
6.7 AIMD Algorithm for Periodic, Delay Value X .................................. 46
6.8 Distribution Targeting Algorithm for Periodic, Delay Value X ................ 47
6.9 Comparing User Value achieved by Offline and Online Statistical-Guarantee Preserving Algorithms ......................................................... 47
LIST OF TABLES

2.1 Cloud provider volatile resource services ........................................ 10
4.1 Key statistics for Instance Pools Exemplars. ................................. 23
4.2 Resource and Intervals Statistics (Exemplars). ............................. 24
6.1 X-value, Resource waste for Offline algorithm. .............................. 42
6.2 Resource Waste, AIMD algorithm .............................................. 46
6.3 Resource Waste, Distribution Targeting Online algorithm ............... 46
ACKNOWLEDGMENTS

I would like to express my deepest appreciation to all those who provided me the possibility to complete this project. A special gratitude I give to my advisor, Prof. Andrew A. Chien, who is always there to guide and support me in every possible way.

Furthermore I would also like to thank all of those with whom I have had the pleasure to work with. In particular, much appreciation goes to Prof. Varun Gupta, who has invested great effort in this project.

I also want to thank my friends, Xiaoan Ding, Chunwei Liu, Chi Li, Tianlin Xu, Junwen Yang, Yutong Yuan, and all the members of LSSG team for their love and support.

Last but not least, this work would not be possible without the love of my parents, Feimin Zhang and Qimei Cao, and my grandparents, Yanggao Zhang and Yayuan Xie.
ABSTRACT

Cloud providers sell unreliable or volatile resources that are unused by foreground (reserved/high priority) workloads to increase revenue while still meeting foreground requests. Hence, volatile resource properties are products of interactions between foreground demands and volatile resource management algorithms. Thus, different algorithms create distinct statistical properties, affecting user value. Also, since current cloud providers do not provide any statistical information or guarantees, it is difficult for users to efficiently exploit volatile resources. As a consequence, cloud providers must consider two key factors to maximize user value: (i) volatile resource management algorithm, and (ii) information provided to users about the resources as statistical guarantees. We describe and evaluate four volatile resource management approaches (Random, FIFO, LIFO, LIFO-pools) using commercial cloud resource traces from 608 Amazon EC2 instance pools, for a 3 month period from 5/2017 to 8/2017 from four AWS US regions. We also consider the value of several information models (MTTR, limited statistics, Full distribution, and Oracle) that statistically characterize the resources.

Our results show volatile resource management algorithms can increase user value by 30 to 45% in four instance exemplars. Information models also provide a 30% increase in numerous cases, and more than 10-fold in extreme cases. Our results suggest that cloud providers should pay significant attention to what statistical information they provide to users. Simple statistics can increase achievable value by 10% to as much as 5x. However, skewed distribution can lead to misleading information, and thus sharply reducing derived value. Then our results of relative user value per resource-hour for the four exemplars show that 90pctile info model is best in all cases and achieves close to the max possible. And, these results broadly characterize the vast majority (475 of 608) of instance pools. The results are the same ordering for VRMs and information models as in exemplars, and the frequency of various relations that are key conclusions for the exemplars are carried along in the vast majority of instance pools. Furthermore, we provide a detailed drill-down showing
how the volatile resource management algorithms affect resource interval durations, and thus potential user value. We further show how the information model shapes user targeting, success rate, and user value.

Cloud providers may concern that providing statistical guarantees constrains resource management flexibility to meet future foreground load. We study two variations of foreground load changes and their impact on statistical guarantees. Then we propose three algorithms attempting to maintain a simple info model. Our offline algorithm results show that statistical guarantees can be fully preserved under foreground load changes with trivial resource waste, increasing user value by up to 134%. Moreover, two online algorithms, AIMD algorithm and Distribution Targeting algorithm, can dynamically preserve guarantees with online knowledge most of the time, and in doing so increase user value by up to 82%. This suggests that further research exploring online algorithms is promising.
CHAPTER 1
INTRODUCTION

1.1 Volatile Resources in Public Cloud

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; the fluctuating demands of traditional reliable resources are marked as foreground load in orange in Figure ??.

Accurate forecasting is difficult, producing fluctuating quantities of excess resources as in Figure 1.1 in blue.

![Figure 1.1: Cloud operators serve a varied foreground load, producing a variable “excess” resources.](image)

To increase revenue and resource utilization, cloud providers sell such excess resources. All cloud providers practice overprovisioning, the overcommittal of resources to increase revenue, and if done well it has negligible customer performance impact. Reserved instances guarantee resource access, but those resources are frequently idle. Excess resources are sold as unreliable cloud instances (spot instances, Preemptible virtual machines) with discounts, up to 90% cost savings in comparison to on-demand resources, attracting users for cost
optimization. Figure 1.2 depicts how users and cloud providers interact and shows the differences between volatile and reliable resources. Once reliable resource users’ requests are accepted by the cloud operator, users keep resources for as long as they want; on the other side, requests for volatile resources can be frequently delayed or rejected. Even after successful requests and ongoing use, volatile resources exhibit unilateral revocation, forcing users (applications) to monitor resources, respond to revocations, and ensure application progress. Such “volatile” resources have a more complex resource usage interface by nature. The ability to reclaim volatile resources enables cloud providers to meet ramps in foreground (high priority) load while still maintain flexibility. The design of volatile resources varies across cloud providers, in pricing and revocation schemes.

Figure 1.2: High-level view of how Users and Cloud providers interact on Volatile and Reliable Resources

1.2 Challenges with Volatile Resources

1.2.1 Exploiting Volatile Resources

Volatile resources have been available at scale since 2009 [3], and have become popular due to its price discount in comparing to other reliable resources. Many scientific and analytic workloads that are not time sensitive are chosen to be computed on them for budget saving. However, their unpredictability and volatility are barriers for users with such workloads. Therefore, a number of studies explore efficient application use on volatile resources. For ex-
ample, a number of efforts focus on bidding strategies in Spot Instance markets [29, 30, 15, 43] or predicting price dynamics [19, 44]. These bidding strategies attempt to keep instances running, but because they fail, other systems employ migration [21, 40] and checkpointing [41, 18, 20, 23, 40] to save application work, and enable application resumption after revocation. Some efforts attempt statistical characterization and prediction [36, 37], and another proposed the idea of using resource management to shape volatile resource properties, providing MTTR as a statistical characterization to increase user value [28]. Our work builds on these ideas.

1.2.2 Managing Volatile Resources

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.

However, volatile resource management algorithms are intuitively different because the resources at the manager’s disposal vary in quantity rapidly due to external forces (foreground demand fluctuation). 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 (or preemption) 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.
Volatile resource properties are not natural; they are produced by interaction of foreground load and volatile resource management algorithms. Yet, their temporal, statistical properties affect their user value. We call the information that a cloud provider reveals about the volatile resources an “Information Model”. Thus, a cloud provider faces three key questions:

1. What volatile resource management algorithm maximizes the value of excess resources?

2. How does the information model, provided as statistical guarantees, for a volatile resource pool affect user value and future resource management?

3. Does an information model constrain cloud providers’ flexibility to meet foreground requirements and how to preserve statistical guarantees?

1.3 Approach

To answer these questions, we formulate and explore the question of volatile resource management that creates volatile resource pools, studying volatile resource management (VRM) algorithms and evaluating their impact on interval distributions, variability, and value. VRM algorithms are to grant and revoke volatile resources while foreground workload fluctuates; it takes such fluctuation as a given and respond to them in the form of releasing new volatile resources or revoking volatile resources in use. By choosing different volatile resources upon releasing and revoking, VRM algorithms can create different usable intervals between revocations for volatile resource users and hence change the overall interval distribution. We describe four volatile resource management approaches (Random, FIFO, LIFO, LIFO-pools) and evaluate them using commercial cloud resource traces from 608 Amazon EC2 Spot Instance pools.

We also vary the information model, exploring the tradeoff between enabling accurate user targeting and the desire to shield proprietary data center specs. Information models are to disclose certain statistical information about the volatile resource interval durations
, which are a result from specific VRM algorithm, to volatile resource users as statistical guarantees. Given such guarantees, users try to maximize benefits by adjusting job runtimes or checkpoint strategies. We propose four information model approaches (MTTR, partial percentile statistics, Full distribution, and Oracle), from minimal knowledge to full information, and evaluate them using the same Amazon instance pools. 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.

Finally, a concern of cloud providers about information models is that any such commitment constrains resource management flexibility to meet foreground load. We study two variations of foreground load changes, change in magnitude and frequency, and their impact on volatile resource statistical guarantees. We propose three algorithms which attempt to maintain a guarantee for a simple information model under foreground load changes. We first study an offline algorithm, which delaying the release of each volatile resources to users by a short period of time, to understand if guarantees can be fully preserved at all. Then, we study two online algorithms – AIMD algorithm and Distribution Targeting algorithm that dynamically recalibrate the delay time based on online knowledge. We evaluate and compare these algorithms by analyzing their ability to preserve guarantees, waste of resources, and increase in user value on previous instance pools under different foreground load changes.

1.4 Contributions

Our results show that VRM has dramatic effect on the statistics of volatile resources, and that good choices can significantly increase user value. The choice of information model is critical for effective volatile-resource exploitation. Our study of offline and online algorithms show that even under extreme foreground load change, maintaining a simple guarantee is possible, and doing so can increase user value significantly.

Specific contributions of this thesis include:

1. Study of 608 instance pools from all four Amazon US regions, each of which contains 2-
6 availability zones, characterizing by two dimensions: volatility and periodicity. Full analysis with four VRMs and info models show that results for the two intensively studied *Stable* exemplars are representative of 80% of the instance pools.

2. Using four exemplar instance pools, detailed evaluation of four VRM algorithms (Random, FIFO, LIFO, and LIFO-pools), including impact on resource availability intervals and variability. LIFO and FIFO vs Random can make more than 2x difference. The best VRM algorithm, LIFO-pools, increases user value up to 2-10x across information models.

3. With four exemplars, design and evaluation of information models (MTTR, 10pctile, 90pctile, Full). MTTR and 10pctile produce poor user value because distribution skew, but 90pctile increases achievable value by 10% to as much as 5x. It approaches the value of providing the Full distribution, suggesting that deliberately disclosing simple statistics can achieve close to ideal value.

4. Comparisons of relative user value per resource-hour for the four exemplars. Results show that 90pctile information model is best in all cases, and achieve close to the max possible. This information model is not only realistic, but also requires only minimal information disclosure.

5. 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. These insights suggest that thoughtful optimization of the broader design space is a fruitful area for further research.

6. Study of foreground load change in two variations: magnitude and frequency. Analysis shows that under extreme foreground load changes, statistical guarantees of info models may be violated and user value also largely reduced.

7. Study of an offline VRM algorithm that preserves 90pctile guarantee under extreme foreground load change. The algorithm not only preserves the guarantee, but also increases user value by 5% to 134% at a “waste” of less than 5% of resources.
8. Study of online VRM algorithms: AIMD algorithm and Distribution Targeting algorithm that dynamically preserves a 90pctile guarantee in the face of dramatic foreground load change. This algorithm increases the achieved user value for the Periodic exemplar by up to 82% at 27% resource “waste”, suggesting that further research exploring online algorithms is promising.

1.5 Thesis Outline

The rest of the thesis is organized as follows. Background of public cloud services and volatile resources is covered in Chapter 2. In Chapter 3 we describe the approach which introduces four volatile resource management algorithms and four information models. Our methodology of user value function and targeting strategy and experiment metrics are also described in Chapter 3. Chapter 4 explains how we characterize 608 instance pools from Amazon EC2 Spot Instance market and select exemplars for detailed studies among the large group of instance pools. Chapter 5 presents detailed experiment results on four exemplars and expands to all 608 instance pools. In Chapter 6, we consider the impact of changes in foreground load, and resource management to preserve statistical guarantees. Related work is covered in Chapter 7. Finally, we summarize our work and suggest future directions in Chapter 8.
CHAPTER 2
BACKGROUND

2.1 Public Cloud Services

Cloud computing is growing rapidly as evidenced by a vibrant cloud market[1, 5, 7] and Infrastructure-as-a-Service (IaaS) platforms are expanding rapidly as one of the most used cloud services, offered by a great variety of provider. In IaaS model, cloud providers hold the physical servers and other computing infrastructures, like storage and networking accessibilities in data centers; users can request and access resources whenever they need the resources with a designated amount they desire. Therefore, users can avoid extra expenses and complexities of buying and maintaining a set of physical infrastructures.

Traditional IaaS is offered either as on-demand resource service or reserved resource service. Users may request on-demand resources on-the-go; Once their requests are fulfilled and resources are given to the users, cloud providers cannot unilaterally revoke given resources unless they are relinquished by the users. Users only need to pay for what they use, normally in an hourly fashion. Occasionally, on-demand resource requests maybe rejected. For some users that do not want to risk such rejections in the face of sudden spike in demands, they use reserved resource service. Reserved resource users make a reservation of resources and capacity for a consecutive, long period of time (i.e. three years) with a hourly discounted price comparing to on-demand resources. Unlike on-demand, their can request resources they have reserved whenever they need the resources and their requests will always be fulfilled. These two types of IaaS are both considered as reliable resources as they will never go away once given to the users.
2.2 Volatile Resources in Commercial Clouds

With increasing competition and fluctuating needs, cloud operators must build capacity to meet internal and external demand spikes as much as possible. Large cloud providers, such as Amazon’s Elastic Compute Cloud (EC2), Google Compute Engine (GCE), and Microsoft Azure, are expanding their infrastructures continually.

The demand for reliable and user-responsive cloud services varies widely over both short and long time scales, leaving a large quantity of resources idle. These idle resources consume energy, cooling, maintenance, space, etc., so it is important for cloud providers to extract value from them. Cloud operators have introduced volatile resource services to utilize and garner revenue from idle resources. These volatile resources, are defined by the property that they can be unilaterally revoked, giving cloud providers the flexibility to meet sudden increases in reliable cloud resource demand. Users of volatile resources are rewarded by a significant discounts, as much as 90% lower than the same compute-memory-storage in a reliable, on-demand resource. Thus providers can capture some revenue from excess resources, yet still maintain the management flexibility required to support the primary reliable resource services.

However, the way volatile resources are offered differs greatly [3, 9, 19, 6]. Table 2.1 summarizes several major volatile resource services on the market. Traditional resource managers match resource requests (with a resource requirements descriptions) to a fixed set of resources that are owned by the resource manager [42, 13, 16].

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. Revocation of Spot Instance is also market bid driven. If the market price rises above a current Spot instance’s bid, it will be revoked. Such
revoked instances are given a notification 2 minutes in advance of termination. Significant academic research has explored the Spot markets[9, 19].

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].

<table>
<thead>
<tr>
<th></th>
<th>Duration Info</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Amazon</strong></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</td>
<td>Pay fixed price per hour</td>
</tr>
<tr>
<td>≥ 1, ≤ 6hrs</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Google</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preemptible VM</td>
<td>≤ 24hrs</td>
<td>No guarantee</td>
</tr>
</tbody>
</table>

Table 2.1: Cloud provider volatile resource services

2.3 Applications on Volatile Resources

Since volatile resources can encounter unpredicted preemption at any time, any intermediate results from jobs that are not transferred to a reliable location will be lost. Such preemptible nature limits volatile resources’ ability to serve interactive applications or time sensitive applications. These types of applications cannot afford frequent revocations of unknown durations, such as web services or financial analysis. Volatile resources are suitable for batch-oriented applications that are disruption tolerant. Many batch scientific analytic jobs are chosen to run on volatile resources, because users are willing to trade longer job completion time due to revocations for discounts in services[8]. Some common scientific workloads like biological sequence search, or stochastic simulations like Monte Carlo risk analysis that are divisible and pleasingly parallel are well suited for volatile resources[26]. Even MapReduce is amenable, because of its design for fault tolerance. They are normally running in parallel with multiple machines, and multiple copies of data are store in partitions. Even if a fraction of the group are revoked suddenly, MapReduce can restart the current step by recomputing.
the data from another alive resource. Some recent work further explore efficient use of MapReduce and Spark framework on volatile resources[39].

Applications can always restart from the beginning upon revocations. However, it can be very inefficient if revocations are frequent or if revocations happen close to completions. Therefore, many studies explore efficient and stable application use on volatile resources. One alternative is to migrate to other resources after receiving a revocation warning, which is generally from 30 seconds to 2 minutes[33, 40]. Migrations do not require extra computation and degradation in performance unless there are revocations, and live migrations can avoid or minimize service downtime[32]. Nevertheless, such migrations can be difficult since revocation warning time is normally too short to copy whole memory pages to destination host. Many previous work study methods to predict revocations and migrate before a warning is sent out or to compress memory footprint[21]. Another method is to checkpoint current state or intermediate data to local or reliable resources periodically, so the job can resume from most recent checkpoint whenever resources are preempted. Frequent checkpoints can incur large unnecessary overheads and degrade performance; rare checkpoints might lead to great volume of recomputation after revocations. Therefore, a number of efforts focus on smart checkpointing based on different kinds of threshold[40, 41, 33].
CHAPTER 3
APPROACH AND METHODOLOGY

3.1 Approach

To study 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 first decompose 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.

![Figure 3.1: Volatile resource managers (VRMs) make decisions to allocate and revoke resources, producing volatile resource intervals of varying lengths.](image)

3.1.1 Volatile Resource Management (VRM)

Available volatile resources are determined by the fluctuation of foreground workload (i.e. on-demand and reserved instances) (see Figure 3.1), and the task of the *volatile resource manager* (VRM) algorithm is to grant and revoke volatile resources while encounters such fluctuations. 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, it is released for volatile resource use and the VRM starts a new interval. VRMs produce different interval distributions by choice of which volatile resource to revoke first (see Figure 3.2). 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 interval distributions and hence difference volatile resource properties. We consider four different VRM algorithms:

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

Figure 3.2: Volatile resource managers (VRMs) can generate distinct pools. Information models for each can provide users with statistical characterizations.

Of these algorithms, LIFO-Pools bears explanation. 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; the topmost has the shortest intervals and lowest availability. Further down the stack these properties improve. For example, one could separate a single pool of N resources into k pools each with N/k number of resources with LIFO algorithm, so the first pool has the most total revocations for all the resources within the pool, and the last has the least. Partitioning could enable separating frequently revoked resources from rarely revoked ones and interval distribution within each pool will be narrower than a single big pool of resources. Consequently, the statistics is more precise if the number of pools increase, but that also introduces complications in management and information revealing; Each pool is expected to have different statistics (e.g. median, stdev) based on the resource availability profile, allowing better application targeting, and thereby increased value. In experiments, we use N=5 pools – more pools provided negligible improvement.

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.

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.

When users relinquish a virtual machine from cloud services, their progresses and modifications on that machine will be completely erased, so when they request a new
one, there is no previously saved state unless they have explicitly saved the state elsewhere. Therefore, we assume that when foreground load requires resources, instead of some specific resources, it only requires a quantity of certain type of resources that satisfy its hardware need. This gives VRM the flexibility to choose instances to revoke in order to create distinct volatile resource properties.

### 3.1.2 Information Models

Information about the volatile resource interval durations affects users’ ability to capture value from them. Given an interval length distribution, users maximize value statistically by targeting job runtimes to maximize expected value. Current cloud operators do not disclose statistical information about volatile resources\(^1\), but could disclose information for each volatile resource pool (see Figure 3.2). While one might propose full transparency as a good policy for cloud providers, they are reluctant, fearing the loss of resource management flexibility and leakage of competitive information.

The information models decide what level of sophistication of information and statistics are provided to the users, from the most basic information to full knowledge of 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><strong>10pctile</strong></td>
<td>Median and the 10th percentile (duration for which 10% of the intervals are shorter)</td>
</tr>
<tr>
<td><strong>90pctile</strong></td>
<td>Median and the 90th percentile (duration for which 90% of the interval are shorter)</td>
</tr>
<tr>
<td><strong>Full</strong></td>
<td>Full histogram, equivalent to the probability distribution function of interval durations.</td>
</tr>
<tr>
<td><strong>Oracle</strong></td>
<td>Exact duration length for each interval, in advance (unrealizable).</td>
</tr>
</tbody>
</table>

\(^1\) see Section 7 for a discussion of defined duration instances.
to derive value. Users exploit the available information to target job runtimes, tune checkpointing strategy, etc. Therefore, information model is a critical element of enabling users to maximize value as users see it as a representation of the actual interval distribution to make decisions. If the targeted job runtime is too short, not much value is captured; if it’s too long, the job may not finish before revocation, causing zero value to be captured. Similarly, if checkpoints are too frequent, the increased overhead wastes resources, and if checkpointing is too infrequent, increased recomputation after revocation is incurred. If the model reveals too little information, users can not target accurately, reducing user value; if the distribution of durations is highly skewed, simple information models may lead users to make poor decisions.

Although providing statistical informations to users as guarantees may increase users’ ability to extract benefits from volatile resources, cloud providers might be afraid that providing such guarantees can constrain their flexibility in meeting reliable resources demand during foreground workload changes. We explore statistical guarantees under foreground load changes and methods to preserve such guarantees in details in Chapter 6

### 3.2 Methodology

In this section we outline key elements of our evaluation methodology: user value functions (Section 3.2.1), how users optimize them for each information model (Section 3.2.2) to measure and compare approaches, and the metrics we study to compare the VRM and information models (Section 3.2.4).

#### 3.2.1 User Value

The user value function maps the actual interval duration a job experiences to a value. With the value function, a user can exploit interval statistics to estimate the expected value, given a target computation length. The step value function depends on the target duration as
below:

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

(3.1)

where \( target \) represents the target duration, and \( V_0(target) \) the maximum value (see Figure 3.3(a)). That is, the job only delivers value after runtime \( L \) exceeds a target duration, and further execution accrues no further value. The step function is a good model for batch and workflow computations, where results are produced only when the job is completely finished, and no more results thereafter. As is evident, accurate targeting is critical for applications to maximize value.

We consider two scalings of the maximum achievable value \( v_0(target) \): linear \((target)\) and polynomial \((target^{1.5})\) as shown in Figure 3.3(b). 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 normalize the \( v_0(target) \) value so that they are equal for the linear and \( target^{1.5} \) scalings for \( target = 1 \).
3.2.2 Targeting Strategy

Given a value function, users select target duration to maximize value based on statistical information available about volatile resources as below. Because we have a step value model, if the target overestimates the actual interval duration, no value will be captured. If it underestimates the actual duration significantly, then most of the potential value will be missed. Experiments assume sufficient demand to utilize all volatile resources – as in AWS market clearing [3].

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

2. **10pctile**: Select \( target \) to maximize expected value assuming Gaussian distribution

3. **90pctile**: Select \( target \) to maximize expected value assuming Gaussian distribution

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

5. **Oracle**: Users target exact interval duration to extract maximum value (unrealizable)

3.2.3 Resource Availability Profiles

Volatile resource availability is defined by capacity and foreground load. Both of these are considered to be highly proprietary trade secrets by all major cloud providers. Thus
only limited summary information is available; insufficient for our broad studies [35, 14]. While historical cluster data released by Google were widely studied [35], which describes various traces running on Google compute cells from parts of the Google cluster management software and systems. As an internal cluster usage data, it was never declared to be cloud trace; hence, it is insufficient to be studied as public cloud usage in this study. Instead, we exploit plentiful pricing information available for Amazon Spot Instance markets 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. Unlike most of other popular volatile resources on the current cloud market, EC2 Spot Instance allows users to specify the a bid price that they are willing to pay. Instances are successfully launched and maintained whenever the bid exceeds the current Spot price, which is decided on the supply and demand of available unused capacity. 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). To note that, this choice affects the absolute value of available resources, but does not have any impact on the metrics evaluated in the paper, as all of our results are comparative across approaches; each with the same quantity of available volatile resources. Hence, the resource availability at time \( t \) is:

\[
VR(t) = (\text{maxPrice} - \text{price}(t)) \times \frac{\text{Max volatile resources}}{\text{maxPrice} - \text{minPrice}}
\]  (3.2)
Figure 3.4 illustrates pricing and resource availability for four exemplar instance pools \((r3.4xlarge,i3.4xlarge,r3.xlarge,\text{ and } r4.16xlarge)\) over a 3-month period in one AZ of us-west-2. Prices are shown in red and inferred resource availability in blue at 5 minutes price intervals. The horizontal line in darker red depicts the on-demand price; the left y-scale represents the price scale for each instance pool. The datacenter is assumed to have minimum 50% foreground load use (on-demand and reserved instances), represented by the blue horizontal line, annotated by "reserved resources"; the right y-scale in blue represents the availability scale.

### 3.2.4 Metrics

We use several performance metrics:

1. **Volatile Resource Availability:** Number of instances available, 5-minute periods. Induced by foreground load; the input to the VRM.
2. **Total Available Volatile Resources (resource-hours):** sum of available instance per 5-minute period, over the entire period, used to compare different instance pools.
3. **Interval Statistics:** the VRM induced interval statistics for an instance pool – includes mean (MTTR), %-tiles, distributions, standard deviation.
4. **Target (minutes):** Target duration selected by user given a resource pool information model.
5. **Success rate:** Fraction of computations achieving non-zero value.
6. **Total Value:** Sum of the job values over intervals of an instance pool.
CHAPTER 4

INTO 608 INSTANCE POOLS

Amazon EC2 in the US consists of four regions, each with 2-6 availability zones. We studied volatile resource management and info models on 608 instance types all drawn from these regions. We categorized all 608 instance pools two dimensions statistically - volatility (standard deviation of the resource availability) and periodicity (ratio of fourier coefficients for a period of 1 day vs. average) that capture important characteristics commonly found in cloud resources, such as diurnal periodicity. Note that the fourier coefficients are computed by taking the fast-fourier transform (FFT) of the resource availability profile, and then the ratio of the frequency bands.

Figure 4.1 presents scatterplots showing the distribution of instance pools along the two statistical dimensions, for all four US regions from all availability zones in us-east-1, us-east-2, us-west-1, and us-west-2 accordingly. X-axis denotes the standard deviation of availability, hence the availability volatility, and Y-axis represents the level of periodicity at a diurnal pattern. 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.

Our studies show that vast majority of these instance pools have similar behavior for volatility (see Chapter 5), so we identified four exemplars based on these two statistical dimensions, capturing such key characteristics, for the purpose of exposition. We selected four exemplars of instance pools - 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 4.2, and the selected exemplars are highlighted in red. The two Stable instances reflect the predominance of low volatility pools, each represents a slightly different level of volatility, and Periodic instance represents the ones with more obvious diurnal patterns; Unstable instance represents a small group of highly volatile instance pools. We return to the large set of instance pools in Section 5.6, showing how the two Stable
Figure 4.1: Scatter-plot of Instance pool pricing behavior from all availability zones

((a)) us-east-1
((b)) us-east-2
((c)) us-west-1
((d)) us-west-2
Figure 4.2: Scatter-plot of Instance pool pricing behavior.

<table>
<thead>
<tr>
<th>Pool Description</th>
<th>Dynamic Range</th>
<th>Mean</th>
<th>Std. Dev.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stable 1 (r3.4xlarge)</td>
<td>13.798</td>
<td>0.378</td>
<td>0.674</td>
</tr>
<tr>
<td>Stable 2 (i3.4xlarge)</td>
<td>0.436</td>
<td>0.162</td>
<td>0.038</td>
</tr>
<tr>
<td>Periodic (r3.xlarge)</td>
<td>0.043</td>
<td>0.049</td>
<td>0.005</td>
</tr>
<tr>
<td>Unstable (r4.16xlarge)</td>
<td>42.135</td>
<td>12.802</td>
<td>17.836</td>
</tr>
</tbody>
</table>

Table 4.1: Key statistics for Instance Pools Exemplars.

exemplars are representative of 80% of the Amazon EC2 instance pools.

Key price statistics for each exemplar are shown in Table 4.1. Price data for each exemplar is processed to create a resource availability profile (see Figure 3.4). Statistics for resource availability, and the number of intervals created by each VRM are shown in Table 4.2.
<table>
<thead>
<tr>
<th>Instance Pool</th>
<th>Total Volatile Hours</th>
<th>VRM Algorithm</th>
<th># of Intervals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stable 1 (r3.4xlarge)</td>
<td>$1.05 \times 10^7$</td>
<td>Random FIFO LIFO</td>
<td>366,720 347,529 414,713</td>
</tr>
<tr>
<td>Stable 2 (i3.4xlarge)</td>
<td>$9.74 \times 10^6$</td>
<td>Random FIFO LIFO</td>
<td>387,225 360,889 451,394</td>
</tr>
<tr>
<td>Periodic (r3.xlarge)</td>
<td>$6.38 \times 10^6$</td>
<td>Random FIFO LIFO</td>
<td>416,876 390,007 485,364</td>
</tr>
<tr>
<td>Unstable (r4.16xlarge)</td>
<td>$7.46 \times 10^6$</td>
<td>Random FIFO LIFO</td>
<td>723,939 722,159 732,563</td>
</tr>
</tbody>
</table>

Table 4.2: Resource and Intervals Statistics (Exemplars).
CHAPTER 5
EXPERIMENTS AND EVALUATIONS

5.1 Experiments

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, 37]. We show detailed analysis for 4 exemplars chosen from 608 instance pools explained in Chapter 4 and return to the large set of instance pools in Section 5.6.

5.2 How VRM Shapes Intervals and User Value

VRM and Interval Properties  The VRM algorithm creates availability intervals in a volatile resource pool. In Figure 5.1, 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 – 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

![Figure 5.1: Basic Interval statistics, 4 Instance Pool Exemplars (Stable 1, Stable 2, Periodic, and Unstable) and 4 VRMs](image-url)
can create differentiated statistics for each sub-pool.

On the left side of Figure 5.1 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 5.1 is Periodic (r3.xlarge, a 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 5.1 presents 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.

![Figure 5.2: Comparing four VRMs, MTTR info model.](image-url)

**VRM and User-value** To assess how interval duration properties affect users’ ability to extract value, we compare the four VRM’s, using the metric of derived user value for the Step value function as defined in Section 3.2.1. 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 5.2. 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 5.1 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 How Info Models affect User Value

Because volatile resources vary statistically, information about interval duration distribution is critical to maximize derived value. As mentioned in Section 3.2.2, we assume users do optimized targeting based on the 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 5.3, 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 5.3: 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 5.3 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 5.3), 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 4.2), 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 5.4 we present user value results for polynomial scaling of the step value function as described in Section 3.2.1. Several differences from linear scaling in Figure 5.3 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 5.4 are arbitrarily much higher and should not be compared directly with those in Figure 5.3. 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.

To compare VRM algorithms across information models, we normalize to Random, focused on instance pool *Stable 1* (see Figure 5.5(a)). Good VRM can increase value beyond Random significantly, up to 1.7x and LIFO-pools consistently achieves highest value (1.5x increase), but LIFO is good as well. To compare across information models, we normalize to MTTR (see Figure 5.5(b)). All information models increase value, including the simple information models (10pctile, 90pctile). Combining a good VRM such as LIFO or LIFO-pools with them yields 1.2x to 1.45x value increase. Further while Full consistently achieves highest value, it may not be desirable for cloud providers. After Full, 90pctile is best by a significant margin (up to 45% value increase).

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.

**5.4 Info Models, Targeting, & Success Rate**

We have seen VRM algorithm and information model directly affect user value. To understand how, we explore how they affect targeting and success rate. The Stable 1 and Periodic are chosen because they illustrate the effects most clearly. In Figure 5.6, on the x-axis, major steps are information models and within each cluster the four VRMs distinguished by color. For LIFO-pools, one target is shown for each sub-pool for a total of 5 target durations.
Leftmost in Figure 5.6 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 5.6, 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 5.6, 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.

Longer targets must be combined with comparable or higher success rates to increase user value. With *Stable 1* (see Figure 5.7, 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 5.1). For *Stable 2* (see Figure 5.7, 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 5.7, 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 5.7, 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.
5.5 Exemplars and Realizable Value

How well do our results compare to the best “statistically” possible? We factor in total hours available and the full distribution for comparison. In Figure 5.8, the plain green bars show max value statistically possible, normalized by the total available resource hours for fair comparison across pools. Across the four exemplars we can see 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 5.8 show the same ratio, for the 90pctile information model (best in all cases), and achieves close to the max possible. This information model is realistic, requiring only minimal information disclosure by the cloud provider.

![Figure 5.8: Relative User value per Resource-hour (VRM/Random) for the four exemplars.](image)

5.6 Relating Four Exemplars & 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 5.9 we present the mean and median results across the 475 instance pools closest to *Stable 1* and *Stable 2*.
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.

Then, Figure 5.10 shows the average count of each VRM+Info models exceeding other combinations within the 475 instance pools. For example, the dotted bar in green depicts the average count of LIFO-pool VRM+90pctile info model exceeding all other combinations is close to 475, the total count. This figure shows similar benefits of certain VRM-Info model combinations as in two Stable exemplars. LIFO-pools always perform significantly better

---

1. 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.
than Random and FIFO VRMs. LIFO outperforms Random and FIFO if with slightly richer information. Regarding info models, 90pctile achieve the best results within each VRM besides Full info models.

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.
<table>
<thead>
<tr>
<th>Configuration</th>
<th>Reln</th>
<th>Count</th>
<th>%-age</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIFO-Pools + Full</td>
<td>Top 1</td>
<td>468</td>
<td>98.5%</td>
</tr>
<tr>
<td>LIFO + Full</td>
<td>Top 1</td>
<td>334</td>
<td>70.3%</td>
</tr>
<tr>
<td>FIFO + Full</td>
<td>Top 1</td>
<td>6</td>
<td>1.2%</td>
</tr>
<tr>
<td>Random + Full</td>
<td>Top 1</td>
<td>0</td>
<td>0%</td>
</tr>
</tbody>
</table>

**Full Excluded**

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Reln</th>
<th>Count</th>
<th>%-age</th>
</tr>
</thead>
<tbody>
<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-Pools + 90pctile in best 10%</td>
<td>in best 10%</td>
<td>399</td>
<td>84%</td>
</tr>
<tr>
<td>LIFO + 90pctile in best 10%</td>
<td>in best 10%</td>
<td>403</td>
<td>85%</td>
</tr>
<tr>
<td>FIFO + 90pctile in best 10%</td>
<td>in best 10%</td>
<td>228</td>
<td>48%</td>
</tr>
<tr>
<td>Random + 90pctile in best 10%</td>
<td>in best 10%</td>
<td>223</td>
<td>47%</td>
</tr>
<tr>
<td>LIFO-Pools + 10pctile in best 10%</td>
<td>in best 10%</td>
<td>231</td>
<td>49%</td>
</tr>
<tr>
<td>LIFO + 10pctile in best 10%</td>
<td>in best 10%</td>
<td>306</td>
<td>64%</td>
</tr>
<tr>
<td>FIFO + 10pctile in best 10%</td>
<td>in best 10%</td>
<td>200</td>
<td>42%</td>
</tr>
<tr>
<td>Random + 10pctile in best 10%</td>
<td>in best 10%</td>
<td>223</td>
<td>47%</td>
</tr>
<tr>
<td>LIFO-Pools + MTTR in best 10%</td>
<td>in best 10%</td>
<td>33</td>
<td>6.9%</td>
</tr>
<tr>
<td>LIFO + MTTR in best 10%</td>
<td>in best 10%</td>
<td>11</td>
<td>2.3%</td>
</tr>
<tr>
<td>FIFO + MTTR in best 10%</td>
<td>in best 10%</td>
<td>15</td>
<td>3.2%</td>
</tr>
<tr>
<td>Random + MTTR in best 10%</td>
<td>in best 10%</td>
<td>14</td>
<td>2.9%</td>
</tr>
</tbody>
</table>

Within 10% of best is defined as getting value at least 90% of the best achieved value. These results support that (LIFO-Pools, Full) is the best overall in nearly all instance pools, and (LIFO, Full) achieves the second best results as Full info model enables optimal targeting under LIFO VRM. On the other hand, consistent with Stable 1 and Stable 2, FIFO and Random VRM’s show less benefits in combination with Full info model. 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. FIFO and Random achieve similar level of results across different information models, where Random does poor uniformly. The 10pctile and MTTR information models do not achieve Top 2 performance in the majority of the instance pools. These ordinal results match our detailed analysis of Stable 1 and Stable 2 well.

5.7 Summary

Our results show the importance of VRM to shape interval distributions. Distribution information must be exposed to users in order to enable effective targeting and high success rates. The key is to reduce standard deviation and expose sufficient information. Differentiating pools (e.g. LIFO-pools) is a useful technique to increase value.
CHAPTER 6
PRESERVING STATISTICAL GUARANTEES

Cloud providers may be concerned about sharing statistical characterization of volatile re-
sources (info models) because doing may constrain their flexibility when foreground resource
demand varies. Foreground demand changes may lead to empirical statistical characteri-
ization unmatching current statistics. However, cloud providers must comply with the in-
formation model because it is provided to users as statistical guarantees. This means that
cloud providers may have to sacrifice foreground requests to meet the guarantees, which
goes against their original purpose of volatile resources. In this chapter, we consider several
types of foreground load changes, their impact on volatile resource interval statistics. We
then consider two types of algorithms offline and online to understand the feasibility of
preserving statistical guarantees while still maintaining the flexibility to meet foreground load
requirement.

6.1 Changing Foreground Loads

To explore how changes in foreground load may affect interval statistics, we consider two
types of foreground load changes. For our analysis, we focus on the 90pctile information
model with LIFO VRM because together they produced the best and most robust results in

(a) Stable2 - Interval CDF((b)) Periodic - Interval CDF((c)) Stable2 - 90pctile Info((d)) Periodic - 90pctile Info
 changes changes changes changes

Figure 6.1: Interval CDF and 90pctile Information Model changes for 4x Magnitude Foreground
Load changes
Section 5. We do not consider *Unstable* for the same reasons.

**Magnitude** We vary foreground load so as to reduce the available volatile resources to a fraction (1/2, 1/3rd, 1/4th) of their original quantity (curves annotated as 2,3,4, respectively). Results in Figure 6.1 for *Stable2* and *Periodic* show that the interval length distributions are identical for all of the foreground load magnitudes. While the number of resources (intervals) available is affected, there is no effect on the distribution of interval lengths, and thus no violation of the 90pctile statistical guarantee offered to users.

**Frequency** We increase the frequency of volatility to $F$ times its base rate ($F = 2, 3, 4$) by contracting our availability trace to represent $1/F$th of its original length. This produces much more frequent availability changes, radically altering the interval length distributions as shown in Figure 6.2. As $F$ increases, there are many more short intervals, and the 90pctile of a 2-week sliding window drops dramatically. In short, the 90pctile statistical guarantee is violated.

### 6.2 Offline Preserving VRM Algorithm

To explore the possibility of resource management to preserve info model guarantees under foreground load change, we analyze the system using a simple offline algorithm. Consider a VRM algorithm that delays the release of each volatile resource to users by X time units, shown in Algorithm 1. This shifts the distribution towards a higher 90th pctile by eliminating...
Algorithm 1 Offline Algorithm: Optimal X

1:   $X \leftarrow 0$
2:   procedure Delaying($X$)
3:       $A_i \leftarrow$ volatile resource availability at time $i$
4:       for each time $i$ in total timeline do
5:           if $A_i < A_{i-1}$ then
6:               revoke resources
7:           else if $A_i > A_{i-1}$ then
8:               wait $X$ minutes, then release resources
9:       end if
10:   end for
11:   end procedure
12:   while not every 2-week sliding window satisfies guarantees do
13:       $X = X + 1$
14:       goto Delaying($X$)
15:   end while
16:   return $X$

The shortest intervals ($< X$ time units). It also shortens the rest of the intervals by $X$ units, wasting $X$ time units $\times$ (# of volatile resource intervals) resources. We call this offline algorithm \textit{Optimal X} because the smallest possible value of $X$ that preserves the 90pctile info model is chosen in hindsight. This offline algorithm is an optimistic assessment and can provide a baseline for other online algorithms in performance. The results of offline algorithm will assess whether delaying the release of volatile resources can successfully and completely restore info models and how well this approach works.

The original 90pctile and the changed frequency trace is shown in Figures 6.3 and 6.4 in green and blue, and the offline trace in black is fully stacked on the original trace, showing that our simple offline algorithm can effectively restore the statistical guarantee by eliminating some short intervals. Further, this guarantee can be restored at relatively low resource waste, as shown in in Table 6.1. Each box shows the $X$ value chosen, and the percentage of volatile resource hours wasted in each configuration. With small amount of resource waste, the 90pctile guarantees can be restored. In cases where waste is 0%, of course $X$ is 0. For moderate frequency changes, the algorithm only needs to delay the release of each volatile resource by 5 minutes (when $F = 3$ for Stable 2 and $F = 2$ for Periodic).
to fully preserve the original info model. By doing so, the total waste of resources is only 0.5\% to 1.4\%. Even under extreme frequency change when statistics reduces significantly, for example $F = 4$ for Periodic, the algorithm can restore with 8.9\% waste. Results with offline algorithm show that the approach of delaying the release each volatile resource can efficiently preserve statistical guarantees. It is also encouraging that under significant foreground load change, maintaining 90\% pctile guarantee is possible at modest cost. This suggests that further studies around online algorithms to preserve guarantees is feasible and interesting.

<table>
<thead>
<tr>
<th>Pool</th>
<th>BASE</th>
<th>F=2</th>
<th>F=3</th>
<th>F=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stable1</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Stable2</td>
<td>0%</td>
<td>0%</td>
<td>5min,0.5%</td>
<td>7.5min,1.3%</td>
</tr>
<tr>
<td>Periodic</td>
<td>0%</td>
<td>5min,1.4%</td>
<td>43min,6.5%</td>
<td>54min,8.9%</td>
</tr>
</tbody>
</table>

Table 6.1: X-value, Resource waste for Offline algorithm.
6.3 Online Preserving VRM Algorithms

In Section 6.2, our results show that clever VRM can restore statistical guarantees promised to users under unpredictable foreground load variation. However our offline algorithm requires foreknowledge to choose X. We next explore two online algorithms, AIMD and Distribution Targeting Algorithms, that attempt to dynamically preserve statistical guarantees based on online knowledge.

6.3.1 AIMD Algorithm

Algorithm 2 Online Algorithm: AIMD

\[ X \leftarrow 0 \]

2: procedure AIMD

\[ A_i \leftarrow \text{volatile resource availability at time } i \]

4: for each time \( i \) in total timeline do

\[ \text{if } A_i < A_{i-1} \text{ then} \]

6: revoke resources

\[ \text{else if } A_i > A_{i-1} \text{ then} \]

8: wait \( X \) minutes, then release resources

end if

10: historical window = 2-week sliding window

\[ \text{if previous historical windown does not satisfy guarantee then} \]

12: \( X = \text{min}(X + 1, 12\text{hour}) \)

\[ \text{else if previous 5 historical windows all satisfy guarantee then} \]

14: \( X = \text{max}(X/2, 0) \)

end if

16: end for

end procedure

We first explore AIMD online algorithm inspired by TCP protocol [11, 25] to select a value of \( X \) online by checking the recent (moving 2-week window) distribution compared to the statistical guarantee (info model). Specifically it updates \( X \) at each interval end using Additive Increase Multiplicative Decrease (AIMD) method, shown in Algorithm 2, to gently probe \( X \) value.
6.3.2 Distribution Targeting Algorithm

Algorithm 3 Online Algorithm: Distribution Targeting

\[ X \leftarrow 0 \]

\begin{algorithmic}
  \Procedure{AdjustIncrease}{(adjustment window)}
    \State \textbf{return} optimal \(X_i\) that enables meeting guarantees in adjustment window
  \EndProcedure

  \Procedure{AdjustDecrease}{(adjustment window)}
    \State \textbf{return} optimal \(X_d\) that still meets guarantees in adjustment window
  \EndProcedure

  \Procedure{Targeting}
    \State \(A_i \leftarrow \text{volatile resource availability at time } i\)
    \For{each time }\(i\) \text{ in total timeline}
      \If{\(A_i < A_{i-1}\)}
        \State revoke resources
      \ElseIf{\(A_i > A_{i-1}\)}
        \State wait \(X\) minutes, \textbf{then} release resources
      \EndIf
    \EndFor
    \State historical window = 2-week sliding window
    \State adjustment window = 2-day sliding window
    \If{previous historical window does not satisfy guarantee}
      \State \(X_i = \text{AdjustIncrease(adjustment window)}\)
      \State \(X = \min(X + 3/4(X_i - X), 12\text{hour})\)
    \Else
      \State \(X_d = \text{AdjustDecrease(adjustment window)}\)
      \State \(X = \max(X - 1/4(X - X_d), 0)\)
    \EndIf
  \EndProcedure
\end{algorithmic}

Then we consider another online algorithm that dynamically probe \(X\) value based on previous distribution, Distribution Targeting Online Algorithm. As Algorithm 3 is shown, it looks at the distribution from the past two days, which is defined as adjustment window, and calibrates \(X\) based on optimal delaying value \(X_i\) or \(X_d\), which is calculated from adjustment window, at each 5 minute end. This approach attempts to decide \(X\) on most recent distribution and avoid old distribution aggressively affecting \(X\) value. We also cap \(X\) at 12 hours.
6.3.3 Online Algorithm Results

As before, we evaluate how well our AIMD algorithm and Distribution Targeting Algorithm maintain the 90pctile info model statistical guarantee. Figures 6.3 and 6.4 compare results for the AIMD algorithm to the original system, higher frequency traces, and offline algorithm. Figures 6.5 and 6.6 compare results for Distribution Targeting algorithm to the higher frequency traces. The results show that both online algorithms respond to 90pctile change, and can restore the info model guarantee much of the time. AIMD algorithm works better with extreme foreground changes, and Distribution Targeting algorithm with moderate ones. Then, Table 6.2 documents the volatile resource waste incurred by AIMD algorithm and Table 6.3 by Distribution Targeting algorithm. These wastes from both online algorithms are greater than for the offline algorithm, despite the fact the online algorithm has more freedom – it can use different X values at different points in time. The offline algorithm’s advantage is its knowledge of the entire trace’s statistics. Within online algorithms, wastes for Dis-
tribution Targeting algorithm are less than AIMD algorithm in nearly all cases, saving up to 9% of total resource hours. These results are consistent with Figure 6.7 and 6.8, which shows the delay value of X of two online algorithms for Periodic exemplar. Both algorithms aggressively ramps X when the 90pctile guarantee is at risk. AIMD algorithm respond quick to the status of 90pctile guarantee, so X value fluctuates aggressively. On the other side, since Distribution Targeting algorithm calibrates X by most recent distribution, its X value are more stable for a period of time.

<table>
<thead>
<tr>
<th>Pool</th>
<th>BASE</th>
<th>K=2</th>
<th>K=3</th>
<th>K=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stable1</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Stable2</td>
<td>0%</td>
<td>0%</td>
<td>0.41%</td>
<td>4.77%</td>
</tr>
<tr>
<td>Periodic</td>
<td>0%</td>
<td>7.19%</td>
<td>18.67%</td>
<td>26.80%</td>
</tr>
</tbody>
</table>

Table 6.2: Resource Waste, AIMD algorithm

<table>
<thead>
<tr>
<th>Pool</th>
<th>BASE</th>
<th>K=2</th>
<th>K=3</th>
<th>K=4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stable1</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td>Stable2</td>
<td>0%</td>
<td>0%</td>
<td>0.62%</td>
<td>2.42%</td>
</tr>
<tr>
<td>Periodic</td>
<td>0%</td>
<td>2.8%</td>
<td>9.64%</td>
<td>17.83%</td>
</tr>
</tbody>
</table>

Table 6.3: Resource Waste, Distribution Targeting Online algorithm

The objective of information model preserving resource management is to maintain high user value in the face of frequency change. In Figure 6.9, we present the absolute (left) and relative (right) of total user value versus increasing frequency, and variations of resource management. We show results for Periodic as they illuminate the behavior most clearly. As
Figure 6.8: Distribution Targeting Algorithm for Periodic, Delay Value X

Figure 6.9: Comparing User Value achieved by Offline and Online Statistical-Guarantee Preserving Algorithms
frequency increases, total value decreases drastically (more than 50%). The offline algorithm is effective, restoring total value at all frequencies to the original (F=1) environment, increasing total value by up to 134%. The online algorithm, without the benefit of foreknowledge does not perform quite that well, but achieves value increases of as much as 82%. Overall, both algorithms can improve the drop in achieved user values significantly. These results suggest that online algorithms are a viable technique, and careful design not only can preserve statistical guarantees with mild resource waste but also significantly elevate achieved user values, thus an interesting area for future research.
A broad range of research explores use of volatile resources and cloud resource management, including statistical characterization and duration prediction, efficient volatile resource exploitation, and efficient cloud resource management. In contrast, our work takes the perspective of a cloud operator, assumes foreground resource management for reliable cloud services as given, explores a variety of volatile resource management algorithms and information models on 608 Amazon EC2 instance pools, and proposes several algorithms that preserve such information guarantees to the users.

7.1 Volatile Resource Characterization

None of the commercial volatile resource services provide statistical characterization of resources; numerous academic efforts have sought to do so. They study Amazon’s Spot markets empirically, deriving models from price [30, 19] and revocation behavior [12]. Examples of key derived variables include interval durations (MTTR), variation, and prediction of interval duration. Brevik and Wolski [36, 37] used time-series models to predict the bid price needed for reliable spot interval durations with a given reliability SLO. Our work explores how different information models affect achievable user value, complementing these efforts. Amazon’s recent defined duration 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.

7.2 Engineering Reliable Resources

Studies have employed mechanisms such as checkpointing [41, 18, 20, 23], replication [33, 38], and migration [40, 21] to shield applications from resource volatility. Some work use pricing threshold to predict incoming revocations in EC2 Spot Markets; this enables complete
live migration to new resources before current resources get preempted and thus protects application from such volatility. Historical empirical characterization of volatility is used to reduce the overheads of reliability mechanisms. For example, Carvalho et al. [10] 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 volatile resource properties and explores viable information model design for intelligent user-management.

### 7.3 Value of Information

Perhaps closest to our work, 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”). Using a superlinear value model, they showed significant resource value increases. Our work goes further in several key dimensions. First, we highlight the deficiency of MTTR as an information model, and analyze four information models (four different “transient guarantees”), showing how the choice affects user value and that two (90pctile, Full) yield much greater value. 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. Finally, we explore guarantee-preserving VRM that maintains providers’ flexibility in meeting foreground load changes at the same time statistical guarantees to volatile resource users.
7.4 Job Scheduling and Resource Management Algorithms

A great number of previous studies explore different resource management or job scheduling methods.

Among those, a group of work focus on scheduling and resource management with service level objectives (SLOs) or deadlines. For example, [27] proposed Availability Knob (AK) that allows users to specify their desire for availability to cloud providers in IaaS cloud. AK considers the risk of failing user-specified SLOs and incurred penalty during VM assignment by using previous history to determine the failure probability. It also periodically migrate over-served VMs to cheapter machines to have more flexible availability. Another study, [22] formalizes implicit user SLOs from historical data and enforces such SLOs through eliminating sharing-induced unpredictability and mitigating inherent unpredictability. The former is done by isolating periodic production jobs from the noisness of sharing to minimize resource over-provisioning; the latter is done by dynamically re-provisioning the current instance of a reservation, in response to job resource consumption. Our work proposes information models as statistical guarantees, which are induced by empirical statistics of resource behaviors. We treat each instance pool as a single group that does not transfer between tiers and try to preserve guarantees within each instance pool.

Some other work explore excess resources in traditional private data centers. Borg[31], a unified container-management system, manages both batch job, which represents low priority job, and latency-sensitive jobs, high priority job. Low priority jobs get schedule whenever there are leftover resources unused by other jobs, and Borg allows evictions on them whenever high priority jobs request. Mesos, borrows idle resources from latency-critical jobs and use them for new containers to achieve both utilization and isolation[16]. Our work, on the other hand, view foreground load demands as a given and try to maximize volatile resource value through VRM algorithms; We also assumes that foreground load request only specifies an instance type, not a particular machine.
7.5 VRM for Preserving Statistical Guarantees

We are not aware of any efforts to build statistical guarantee preserving resource management algorithms – one of the key new ideas in this paper. Our adaptive control approach is inspired by Additive-Increase-Multiplicative-Decrease (AIMD) algorithm widely adapted in TCP Protocol’s congestion control (i.e. TCP Reno Protocol)[11]. AIMD in TCP Protocol is aimed to gently probe the network for spare cpacity; we adapts AIMD algorithm to probe X value for preserving statistical guarantees while trying to maintain a reasonable waste of resource hours. The insight that preserving statistical guarantees requires wasting resources also comes across in user-side resource management through conservative bidding [36], and workload placement in clusters for latency guarantees, e.g., [17].

7.6 Value Functions

Prior work [10, 28] that quantifies the value obtainable from volatile resources models computations as long-lived. Carvalho et al. [10] 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% (mimicing Amazon EC2 charging scheme). Shastri et al. [28] 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 ($v_0(target)$), for which we pick two representative cases: linear growth ($v_0(target) \propto target$), and super-linear polynomial growth ($v_0(target) \propto target^{1.5}$). In [28], the authors use an exponential value function. In [24], 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 [34]). 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.
CHAPTER 8
SUMMARY AND FUTURE WORK

8.1 Summary

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 manage-
ment approaches and four information models that can be provided to the users, and show
that the choice of resource management algorithms can have a large impact on the achieved
value, up to $\sim 2X$. 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. Of course, instance pools with extremely volatile fluctuation behaviors make captur-
ing value difficult, all VRM algorithms achieve similar results; the best info model also tend
to be hard to harness value. 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, and Unstable cases are rather rare. Information models with a highly
skewed distribution might show misleading information causes poor targeting, sharply re-
ducing users’ ability to derive value; however, in combination with VRM, information model
with simple statistics that captures the actual distribution can largely increase achievable
value, close to ideal results. Thus, a cloud operator has to deliberately choose the volatile
resource management approach and information model for users in order to maximize value.

We study offline and online algorithms which attempt to maintain the 90pctile infor-
mation model. These studies show even under extreme foreground load change, 90pctile guarantees can be preserved with offline algorithm with trivial waste in resources (0.5% to 8.9%) and significant increase in user value (up to 134%). With offline algorithm as a baseline, studies of online algorithms show that not only can 90pctile guarantees be restored with moderate waste but doing so increases user value by as much 82% (online). While our results are a promising first step, they suggest design of statistical guarantee preserving resource management as an interesting research direction.

### 8.2 Future Directions

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. In this thesis, we use a simple step value function to represent most of the analytics and scientific workloads; to understand more fine-grained user behaviors and resulting impacts, one can explore different value models.

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. In this thesis, we use empirical statistics as a guideline to provide statistical guarantees. 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.

In this thesis, we studied characterization of statistics as guarantees, one could think about different types of guarantees cloud providers may want to serve the users and the methods to preserve such guarantees. For example, 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


