SpotCheck: Designing a Derivative IaaS Cloud on the Spot Market

Prateek Sharma, Stephen Lee, Tian Guo, David Irwin, Prashant Shenoy

UMASS CS
School of Computer Science

EuroSys 2015
Infrastructure Clouds

Cloud computing is popular and offers many benefits:

- Ease of deployment, scalability, pay-as-you-go
- Infrastructure, Platform, Software as a Service
- Infrastructure clouds offer computing and storage resources
Cost vs. Availability Tradeoffs

Cloud servers have different cost vs. availability tradeoffs:

**On-demand servers:**
- Fixed price per unit time
- Non-revocable
Cost vs. Availability Tradeoffs

Cloud servers have different cost vs. availability tradeoffs:

**On-demand servers:**
- Fixed price per unit time
- Non-revocable

**Spot servers:**
- Variable prices based on market conditions
- Revocable $\implies$ lower availability
- Surplus capacity sold at lower price

![Graph showing cost comparison between on-demand and spot servers](image-url)
Exploiting Spot Servers

Spot servers ideal for batch jobs

- Batch jobs are disruption tolerant
- Checkpoint-Restart to tolerate spot revocations
Exploiting Spot Servers

Spot servers ideal for batch jobs
- Batch jobs are disruption tolerant
- Checkpoint-Restart to tolerate spot revocations

Spot servers for interactive applications?
- Revocable $\implies$ Downtime
  - Mitigate impact of revocation?
  - Potentially lower costs
Running Interactive Applications on Spot Servers
Running Interactive Applications on Spot Servers

Spot

Application

Naive solution:

- Spot server revoked $\implies$ migrate to on-demand
Running Interactive Applications on Spot Servers

Naive solution:
- Spot server revoked \( \Rightarrow \) migrate to on-demand
Running Interactive Applications on Spot Servers

Naive solution:
- Spot server revoked $\implies$ migrate to on-demand
- Revocation $\implies$ risk of losing application state
- Users cannot manage complexity & risk of migration

**Problem Statement:**
Design a system to **transparently** use a mix of spot & on-demand servers to run interactive applications
Talk Outline

1. Motivation
2. System Design
3. Evaluation
4. Related Work
5. Conclusion
**Derivative Cloud**

- Cloud middleware derived from infrastructure clouds
- Buy servers from IaaS and resell to users
- Use servers of different types $\implies$ different SLA from IaaS
- Example: spot & on-demand pool to lower costs and increase availability
SpotCheck

- SpotCheck: Derivative cloud using spot & on-demand servers
- Provide low-cost, non-revocable servers to run unmodified applications
- Key idea: Run on spot. When revoked, migrate to on-demand
- Requirement: Transparently migrate VMs without losing state
VM Migration

Key idea:
Run on spot. When revoked, migrate to on-demand

Naive approach: Virtual Machine Live Migration

- Migration may not complete
- Small termination warning (~2 minutes on EC2)
- Incomplete migration \( \Rightarrow \) loss of state
Bounded-time VM Migration

- Guaranteed bound on VM migration time
- Independent of memory size, application behaviour
Bounded-time VM Migration

- Guaranteed bound on VM migration time
- Independent of memory size, application behaviour

Spot

VM

Continuously checkpoint memory

Backup Server

- Residual dirty pages sent in bounded time
Bounded-time VM Migration

- Guaranteed bound on VM migration time
- Independent of memory size, application behaviour

Spot

- Continuously checkpoint memory

Backup Server

On-demand

- Lazily restore memory

- Residual dirty pages sent in bounded time
- Lazy restore: quickly restore VM
**Nested Virtualization**

**Problem:**
- Bounded-time VM migration needs hypervisor modification
- IaaS clouds don’t allow hypervisor modifications

![Diagram of nested virtualization with cloud server and IaaS hypervisor]
**Nested Virtualization**

**Problem:**
- Bounded-time VM migration needs hypervisor modification
- IaaS clouds don’t allow hypervisor modifications

**Solution:** Nested Virtualization
- User VMs run on a nested hypervisor (XenBlanket)
- Bounded-time migration, VM lazy restore implemented inside nested hypervisor
Spreading Revocation Risk

Problem: Revocation storms
Spreading Revocation Risk

**Problem: Revocation storms**
- Spot pool revoked $\Rightarrow$ migrate all VMs to on-demand
Spreading Revocation Risk

**Problem: Revocation storms**

- Spot pool revoked $\implies$ migrate all VMs to on-demand
Spreading Revocation Risk

Problem: Revocation storms

- Spot pool revoked $\implies$ migrate all VMs to on-demand

Solution: Multiple independent pools

- Revocations from one pool don't affect other pools
Pool Management Policies

- Distribute VMs among multiple pools
- Example: availability zones, server sizes (small, large)
- Spot prices in different pools uncorrelated
- Pool distribution policies:
  1. Equally
  2. Weighted by pool cost
  3. Weighted by pool availability
- Centralized *controller* dynamically selects pool for new VMs.
Evaluation

SpotCheck prototype implemented on Amazon EC2

- Application Performance
- Cost of SpotCheck VMs
- Availability

- Benchmarks: SpecJBB, TPC-W
- Experiments on EC2 m3.medium instances
- Spot prices from April to Oct 2014
Impact on Application Performance

<table>
<thead>
<tr>
<th></th>
<th>Spec-JBB</th>
<th>TPC-W</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.0015%</td>
<td>16.7%</td>
</tr>
</tbody>
</table>

Performance degradation due to continuous checkpointing
Impact on Application Performance

<table>
<thead>
<tr>
<th>Spec-JBB</th>
<th>0.0015%</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPC-W</td>
<td>16.7%</td>
</tr>
</tbody>
</table>

Performance degradation due to continuous checkpointing

- Backup server can support up to 40 VMs
- *Each VM shares* $\frac{1}{40}$ *the cost of a backup server*
Cost of Running VMs on SpotCheck

- Cost with different pool management policies is \(~$0.014/hr\)
- Cost saving of \(80\%\) compared to on-demand
Availability

- Availability of **99.998%** relative to on-demand servers
- Downtime (~20 seconds) during migrations
Related Work

**Spot instances:** [Javadi et al., 2011, Ben-Yehuda et al., 2011]

**Derivative Clouds:** PiCloud, Heroku

**Nested Virtualization:** XenBlanket [Williams et al., 2012], Turtles [Ben-Yehuda et al., 2010]

**Bounded time Migration:** Yank [Singh et al., 2013], Remus [Cully et al., 2008]

**Lazy VM restoration:** Post-Copy migration [Hines et al., 2009], SnowFlock [Lagar-Cavilla et al., 2009]
Conclusion

- Different cost vs. availability tradeoffs in Infrastructure clouds
- Spot servers: Cheap but volatile
- Derivative Cloud: abstraction layer between user and cloud
- SpotCheck: Derivative cloud with spot & on-demand servers
- Prototyped on Amazon EC2
- Cost savings of 80%, 5 9’s availability
Thank You
Spot price spikes

The graph shows the price per hour ($/hr) over time. The blue line represents the spot price, and the red dashed line represents the on-demand price. The graph indicates several instances of spike prices, with the highest spikes occurring around the time points of 50 and 150.
Spot Prices

![Spot Price Graph](image)

- Spot Price
- Ondemand Price
- Availability CDF
- m3.medium
- m3.large
- m3.xlarge
- m3.2xlarge

Availability CDF

- Spot-price/Ondemand-price ratio
- m3.medium
- m3.large
- m3.xlarge
- m3.2xlarge

Graph showing availability CDF for different instance types.
Time line

![Graph showing price changes over time with labels Run on Spot Server, On-demand, and Spot. The x-axis represents time, and the y-axis represents price ($/hr). The graph includes a blue line and a red dashed line representing Spot price and bid price, respectively. There is a shaded area indicating cost.](image-url)
Price Correlation
What if everybody starts using this?

- Public cloud server market is large
- More incentive to expand spot markets
- Bidding is truth revealing
- Benefits both users and cloud providers
Cost of running VMs on SpotCheck

Factors which affect cost:

\[ \begin{align*}
  p & : \text{Probability of revocation} \\
  E(S) & : \text{Expected Spot price based on price traces} \\
  D & : \text{On-demand price} \\
  B & : \text{Backup server cost} = 1 \text{ large server} \\
  N & : \# \text{ VMs sharing a backup server} = 40
\end{align*} \]

Expected Cost of running a VM on SpotCheck:

\[
E(\text{Cost}) = (1 - p) \cdot E(S) + p \cdot D + \frac{B}{N}
\]

Expected Cost = 0.2 \times \text{On-Demand} = $0.014 / \text{hour}
Availability

<table>
<thead>
<tr>
<th>Max. num. of concurrent revocations</th>
<th>N/4</th>
<th>N/2</th>
<th>3N/4</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Pool</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$1.74 \times 10^{-4}$</td>
</tr>
<tr>
<td>2-Pool</td>
<td>0</td>
<td>3.75 $\times 10^{-3}$</td>
<td>0</td>
<td>2.25 $\times 10^{-5}$</td>
</tr>
<tr>
<td>4-Pool</td>
<td>$7.4 \times 10^{-3}$</td>
<td>$7.71 \times 10^{-5}$</td>
<td>1.92 $\times 10^{-5}$</td>
<td>0</td>
</tr>
</tbody>
</table>

Probability of the maximum number of concurrent revocations for different pools. N is the number of VMs.
Nested Virtualization overhead

- Nested Virtualization overhead is dependent on application characteristics
- Can use containers.
- Migration mechanisms in containers need to be implemented
Storage & Networking

- Direct access to EBS and S3
- Bridged or NAT networking inside nested hypervisor
- VPC to isolate user VMs
Restore overhead

![Graph showing the relationship between the number of VMs being concurrently lazily restored and TPC-W response time. The x-axis represents the number of VMs, ranging from 0 to 10, and the y-axis represents the response time in milliseconds, ranging from 0 to 70. The graph shows a steady increase in response time as the number of VMs increases.]
1. Spot pricing, CDF, avail.
2. What if everyone starts using this. Market large. More incentive for IaaS.
3. Storage and networking. EBS/S3. Bridge or NAT nw interfaces.
4. Inter cloud operation. exploring practical challenges with inter-cloud networking
5. Nested overhead. Indeed significant for certain workloads. 1) Implementations will mature. 2) Only use this because IaaS doesnt expose migration functionality 3) Looking at using OS-level virtualization and containers.
References

(2014).
Heroku.

(2014).
PiCloud.

