SpotOn: A Batch Computing Service for the Spot Market

Supreeth Subramanya, Tian Guo, Prateek Sharma, David Irwin, Prashant Shenoy

University of Massachusetts Amherst
INFRASTRUCTURE CLOUD

On-demand
Infrastructure Cloud

Cost vs. Availability Tradeoff in IaaS Clouds

Cost (per hour)

Cheap

Expensive

Reserved

On-demand

Not guaranteed, Non-revocable

Not guaranteed, Non-revocable

Revocable

Spot

Guaranteed, Non-revocable

Guaranteed, Non-revocable

Not guaranteed, Revocable
INFRASTRUCTURE CLOUD

Cost vs. Availability Tradeoff in IaaS Clouds

<table>
<thead>
<tr>
<th>Cost (per hour)</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cheap</td>
<td>Guaranteed, Non-revocable</td>
</tr>
<tr>
<td></td>
<td>Not guaranteed, Non-revocable</td>
</tr>
<tr>
<td></td>
<td>Not guaranteed, Revocable</td>
</tr>
</tbody>
</table>

Reserved

On-demand

Spot

Spot Instances

Guaranteed, Non-revocable

Not guaranteed, Non-revocable

Not guaranteed, Revocable

Google Cloud Platform

Preemptible VM
**SPOT MARKETS**

Amazon EC2

- **Bid** in a 2nd price auction
- **Acquire** when bid $>$ spot price
- **Terminate** when spot price $>$ bid
Amazon EC2

- **Bid** in a 2nd price auction
- **Acquire** when bid > spot price
- **Terminate** when spot price > bid

How do we mitigate the impact of revocation?

1. Raise the bid
2. Employ fault-tolerance mechanisms
Amazon EC2

- **Bid** in a 2nd price auction
- **Acquire** when bid > spot price
- **Terminate** when spot price > bid

How do we mitigate the impact of revocation?

1. Raise the bid
2. **Employ fault-tolerance mechanisms**
Amazon EC2 operates ~4000 spot markets
**CHALLENGES — SPOT MARKET COMPLEXITY**

Amazon EC2 operates \(~4000\) spot markets

Scatterplot of ranks for EC2 spot markets
CHALLENGES — SPOT MARKET COMPLEXITY

Amazon EC2 operates \(~4000\) spot markets

Selecting an instance that yields lowest cost per unit of computation while also considering the probability of revocation is complex

Scatterplot of ranks for EC2 spot markets
**CHALLENGES — APPLICATION COMPLEXITY**

<table>
<thead>
<tr>
<th>Resource Vector</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CPU : IO</th>
<th>Working Set</th>
<th>Disk Type</th>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:1</td>
<td>8GB</td>
<td>Remote</td>
<td>1 hour</td>
</tr>
</tbody>
</table>
### Challenging Application Complexity

<table>
<thead>
<tr>
<th>CPU : IO</th>
<th>Working Set</th>
<th>Disk Type</th>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:1</td>
<td>8GB</td>
<td>Remote</td>
<td>1 hour</td>
</tr>
</tbody>
</table>

**Resource Vector**

**Spot VM**

<table>
<thead>
<tr>
<th>Cost</th>
<th>Revocation Rate</th>
<th>Fault-tolerance Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>20% On-demand</td>
<td>2.4 per day</td>
<td>Checkpoint (every 900s)</td>
</tr>
</tbody>
</table>
CHALLENGES — APPLICATION COMPLEXITY

Resource Vector

SDB: 1:1  Working Set: 8GB  Disk Type: Remote  Running Time: 1 hour

Cost  Revocation Rate  Fault-tolerance Mechanism
20% On-demand  2.4 per day  Checkpoint (every 900s)

Spot VM

Cost ($)  Completion time (s)

On-demand  Checkpoint
0  0
25  2500
50  5000
75  7500
100  10000

Working Set: 8GB  CPU: IO 1:1  Working Set: 8GB  Running Time: 1 hour  Fault-tolerance Mechanism: Checkpoint (every 900s)
CHALLENGES — APPLICATION COMPLEXITY

Resource Vector

CPU : IO  Working Set  Disk Type  Running Time
1:1       8GB          Remote       1 hour

Cost            Revocation Rate          Fault-tolerance Mechanism
20% On-demand   2.4 per day            Checkpoint (every 900s)

Spot VM

Disk Type          Fault-tolerance Mechanism
Remote             Checkpoint (every 900s)

Replicate (deg=2)
**CHALLENGES — APPLICATION COMPLEXITY**

**Resource Vector**
- CPU : IO 1:1
- Working Set 8GB
- Disk Type
- Running Time 1 hour

**Spot VM**
- Cost 20% On-demand
- Revocation Rate 2.4 per day
- Fault-tolerance Mechanism
  - Checkpoint (every 900s)
  - Replicate (deg=2)

**Graph**
- X-axis: On-demand, Checkpoint, Replicate
- Y-axis: Cost ($), Completion time (s)
  - On-demand: $50, Completion time 2000s
  - Checkpoint: $75, Completion time 6000s
  - Replicate: $50, Completion time 4000s
CHALLENGES — APPLICATION COMPLEXITY

Resource Vector

Spot VM

CPU : IO  |  Working Set  |  Disk Type  |  Running Time
----------|--------------|-------------|---------------
1:1       | 8GB          | Local       | 1 hour        

Cost | Revocation Rate | Fault-tolerance Mechanism
20% On-demand | >24 per day | Checkpoint (every 900s)
CHALLENGES — APPLICATION COMPLEXITY

Resource Vector

CPU : IO  Working Set  Disk Type  Running Time
1:1       8GB       Local    1 hour

Cost  Revocation Rate  Fault-tolerance Mechanism
20% On-demand  >24 per day  Checkpoint (every 900s)

Spot VM

Cost (¢)  Completion time (s)

On-demand  Checkpoint  Replicate  Replicate (Revoked)
0  25  50  75
2000  4000  6000  8000

challenges — application complexity

>24 per day
SpotOn: a batch computing service
**SpotOn: a batch computing service**

Service that accepts batch jobs from users and runs them on spot instances

Manages application and spot market complexity transparently
**SpotOn: a batch computing service**

Service that accepts batch jobs from users and runs them on spot instances.

Manages application and spot market complexity transparently.

Run batch jobs at *on-demand performance* but paying *spot market price*.
GREEDY SELECTION ALGORITHM

Selecting the best spot market and fault-tolerance mechanism
Selecting the best spot market and fault-tolerance mechanism
Selecting the best spot market and fault-tolerance mechanism
GREEDY SELECTION ALGORITHM

Selecting the best spot market and fault-tolerance mechanism

Spot Markets

∀Si

Fault Tolerance

Migrate  Dup  Chkp

Batch Job

Jb

Minimum cost of running Jb using Ft on Si
**GREEDY SELECTION ALGORITHM**

Selecting the best spot market and fault-tolerance mechanism

---

*Spot Markets*

\[\forall S_i\]

*Batch Job*

\[J_b\]

*Fault Tolerance*

\[\forall F_t\]

Minimum cost of running \(J_b\) using \(F_t\) on \(S_i\)

Acquire \(S_i\)
Greedy Selection Algorithm

Selecting the best spot market and fault-tolerance mechanism

Spot Markets

∀S_i

Batch Job

J_b

Fault Tolerance

∀F_t

Migrate
Dup
Chkp

Minimum cost of running J_b using F_t on S_i

Acquire S_i

Repeat on spot revocation (until J_b finishes)
Reactive Migration

$\begin{align*}
t_0 & \quad T_m \\
Migrated & \quad T + T_m
\end{align*}$
Reactive Migration

\[ T_M \propto \frac{\text{size of memory + local disk}}{\text{remote disk bandwidth}} \]
Reactive Migration

$Z_k \rightarrow$ Random variable measuring time to revocation

$P_k \rightarrow$ Probability job gets revoked before completion
Reactive Migration

$Z_k \rightarrow$ Random variable measuring time to revocation

$P_k \rightarrow$ Probability job gets revoked before completion

$$E[\text{Price}_k] = \frac{[(1 - P_k) \cdot T + P_k \cdot (E[Z_k] + T_M)] \cdot \text{spot-price}}{E[\text{Time}_k]}$$

$$T_M \propto \frac{\text{size of memory} + \text{local disk}}{\text{remote disk bandwidth}}$$
Proactive Checkpoint

$t_0 \xrightarrow{T_c} T_c \xrightarrow{T_L} T + 4T_c + T_L$

Restored
Proactive Checkpoint

\[ T_C \propto \frac{\text{size of memory} + \text{local disk}}{\text{remote disk bandwidth}} \]

\[ T_L \propto (\tau / 2) \]

\( \tau \): checkpoint frequency
Proactive Checkpoint

\[ \text{Total Overhead} = \left( \frac{T}{\tau} \right) \times T_C + T_L \]

- \( T_C \propto \frac{\text{size of memory + local disk}}{\text{remote disk bandwidth}} \)
- \( T_L \propto (\tau/2) \)

\( \tau \): checkpoint frequency
Proactive Checkpoint

\[ T_c \propto \frac{\text{size of memory + local disk}}{\text{remote disk bandwidth}} \]

\[ T_L \propto \left(\frac{\tau}{2}\right) \]

\( \tau \): checkpoint frequency

Total Overhead = \((T/\tau) \times T_c + T_L\)

Cost overhead is primarily a function of job’s resource usage
Spot Replication

\[ t_0 \]

\[ \text{Restart} \]

\[ T_L \]

\[ T + T_L \]

\[ T_L \propto \text{market volatility} \]
Spot Replication

\[ t_0 \]

\[ T_L \]

\[ T + T_L \]

\[ T_L \propto \text{market volatility} \]

Total Overhead \( \propto (\text{Replication factor}, T_L) \)

Cost overhead is primarily a function of market characteristics
SpotOn Prototype

Built on *Linux Containers* for efficient checkpointing / migration

*App Emulator* to create synthetic jobs with varying resource usage

**SpotOn** job manager
Effect of Application and Spot Market Characteristics
Effect of Application and Spot Market Characteristics

![Graph showing the effect of spot revocation rate on completion time and cost. The x-axis represents the spot revocation rate (per day), and the y-axis represents completion time (sec) and cost (cent). The graph compares replication and checkpointing methods.](image-url)
Effect of Application and Spot Market Characteristics

Best choice of fault-tolerance mechanism is a function of the spot market and job characteristics.
Effect of Cost-aware Selection
Effect of Cost-aware Selection

![Chart showing the effect of cost-aware selection with different revocation rates. The chart compares Basic, Cost-aware (CKP), and Cost-aware (ALL) methods, showing the normalized cost for varying revocation rates per day.]
Effect of Cost-aware Selection

2x better than just checkpointing
On Google cluster trace, Cost-aware selection achieved 91.9% savings with little impact on performance.
CONCLUSION

• Spot markets offer **arbitrage** opportunities
• SpotOn manages application and market complexities
• We model **fault tolerance** and propose a **selection algorithm**
• **Prototype** on Amazon EC2
• Achieves $\sim90\%$ cost savings
CONCLUSION

- Spot markets offer **arbitrage** opportunities
- SpotOn manages application and market complexities
- We model **fault tolerance** and propose a **selection algorithm**
- **Prototype** on Amazon EC2
- Achieves $\sim 90\%$ cost savings

Thank you!
backup slides
Spot price distribution has long tail

Spot price changes are peaky
Bidding does affect volatility but *not drastically*.

SpotOn always bids at the on-demand price. If spot price goes above on-demand, SpotOn would choose on-demand.

**Spot price distribution has long tail**

**Spot price changes are peaky**