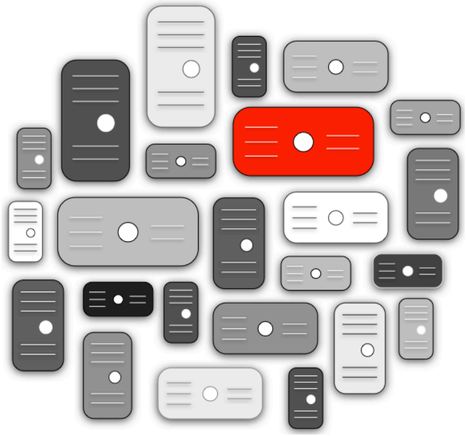
HotSpot: Automated Server Hopping in Cloud Spot Markets



Supreeth Shastri and David Irwin

UMassAmherst

Transient Servers are Ubiquitous in the Cloud

Servers that may **terminate anytime** after an **advance warning** period

Spot Instances: variable-priced transient VMs offered via second price auction

Preemptible VM: short-lived VMs offered at fixed but discounted prices

Internal Use: Resource harvesting in datacenters

[SoCC 2016. OSDI 2016, ATC 2017]

EC2 Spot Markets in a Nutshell

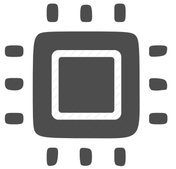
7600+ spot markets worldwide

1

The users bid for VMs in a second price auction

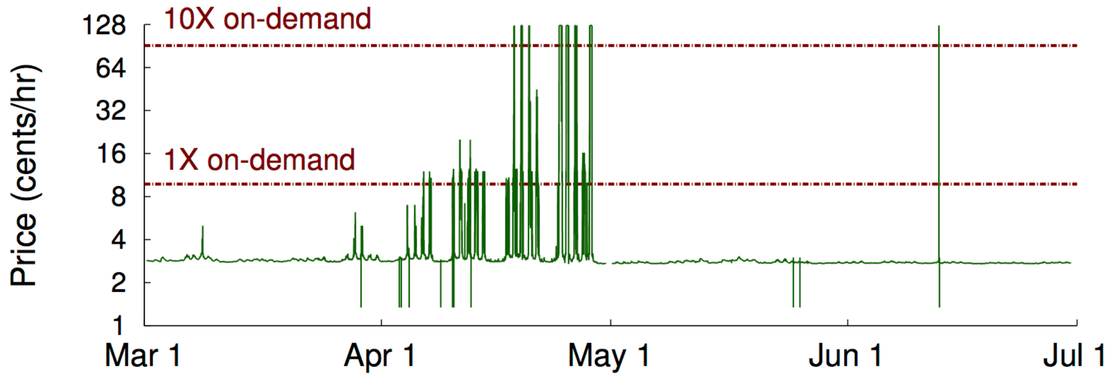
2

EC2 evaluates supply-demand dynamic to price spot servers



3

EC2 allocates if bid price ≥ spot price; Revokes when not.



The defining characteristics of spot VMs are

low average price **and** ***unexpected revocations***

Applications and frameworks do not perform well when the underlying servers are frequently revoked

Prior work treats *revocations as failures*,
and employs fault-tolerance to reduce its impact

2015	2016	2017
SpotOn [**SoCC**]	TR-Spark [**SoCC**]	Proteus [**EuroSys**]
SpotCheck [**EuroSys**]	Flint [**EuroSys**]	Pado [**EuroSys**]
Cumulon [**VLDB**]	BOSS [**Infocom**]	Exosphere [**Sigmetrics**]

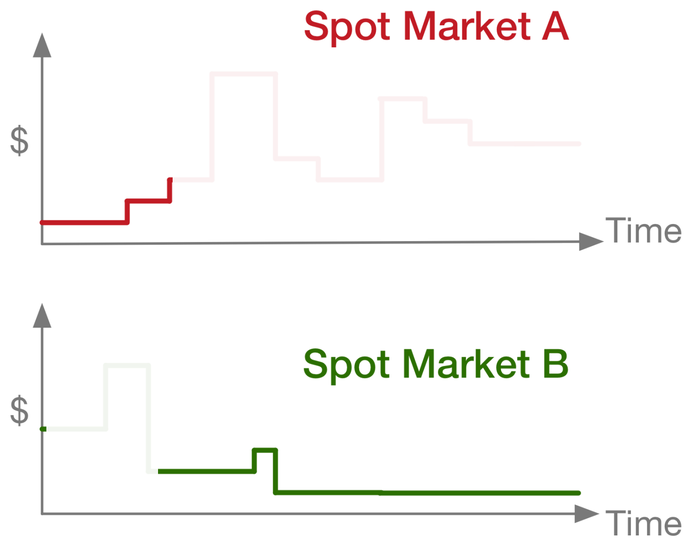
Fault-tolerance ≅ insurance

users pay upfront premiums (i.e., fault-tolerance overhead)
and expect a payout later (i.e., ability to limit the loss of work)

… but
insurance-like approaches ignore

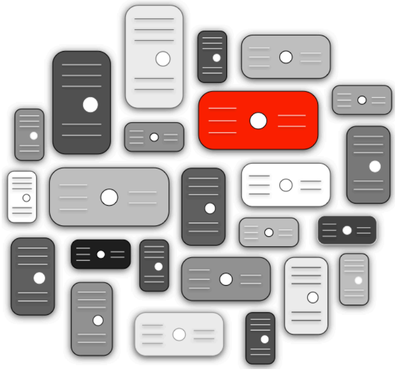
Price Risk

i.e the risk that a VM's price
will increase relative to others



*Does mitigating the price risk affect **performance** and **revocation risk**?*

*How to enable flexible cloud applications to mitigate the **price risk** transparently?*

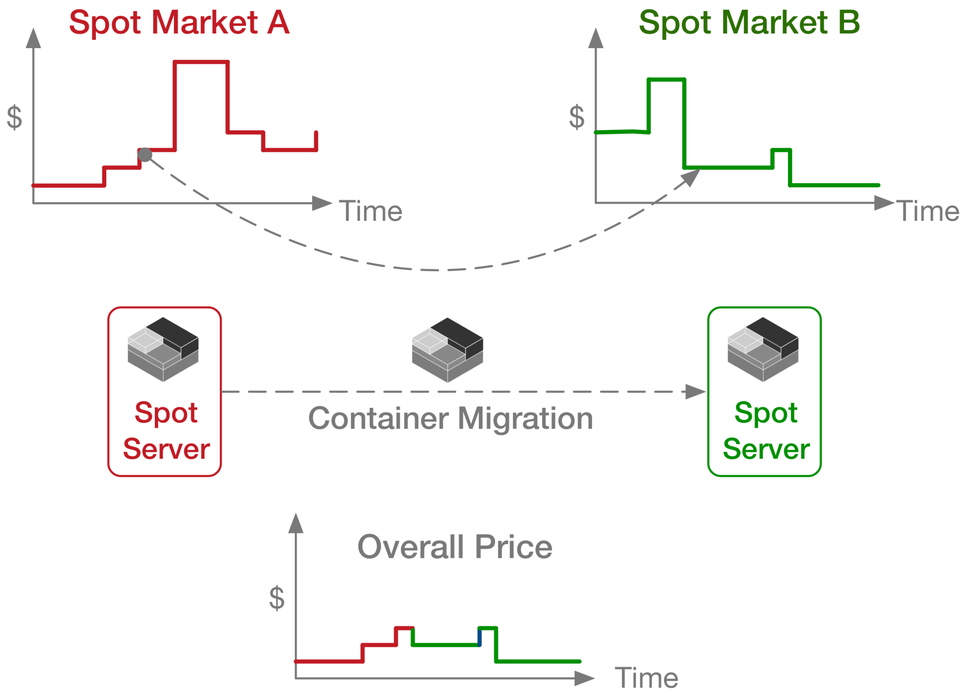


HotSpot: Automated Server Hopping

Automated Server Hopping

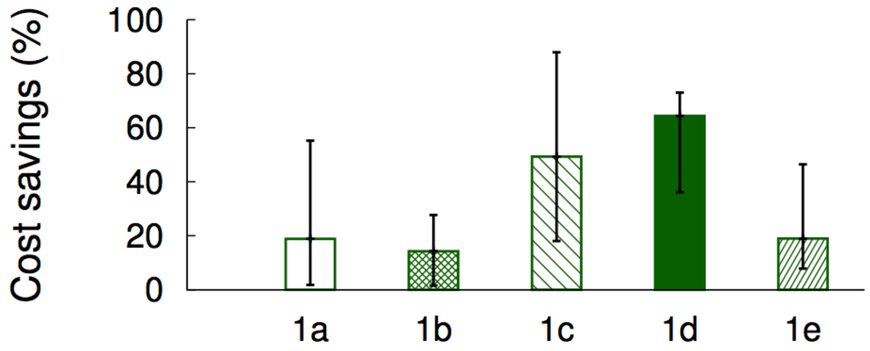
> “ Change, before you have to ”

A resource container that automatically hops spot VMs as market conditions change



Results from the EC2 spot market

US-East-1 markets (3/1/2017 - 5/1/2017)



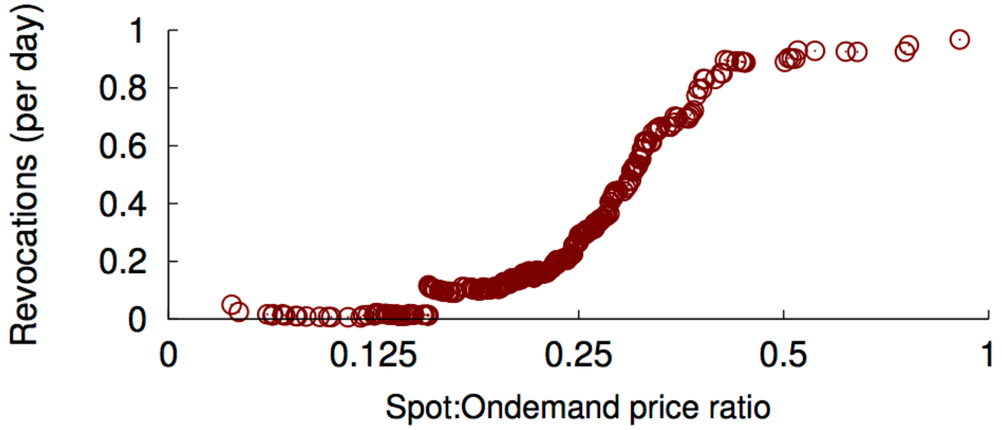
Ideal savings from **hopping** vs. **staying** for a long-running job (30 days)

Quote from Jack Welch, former CEO of General Electric

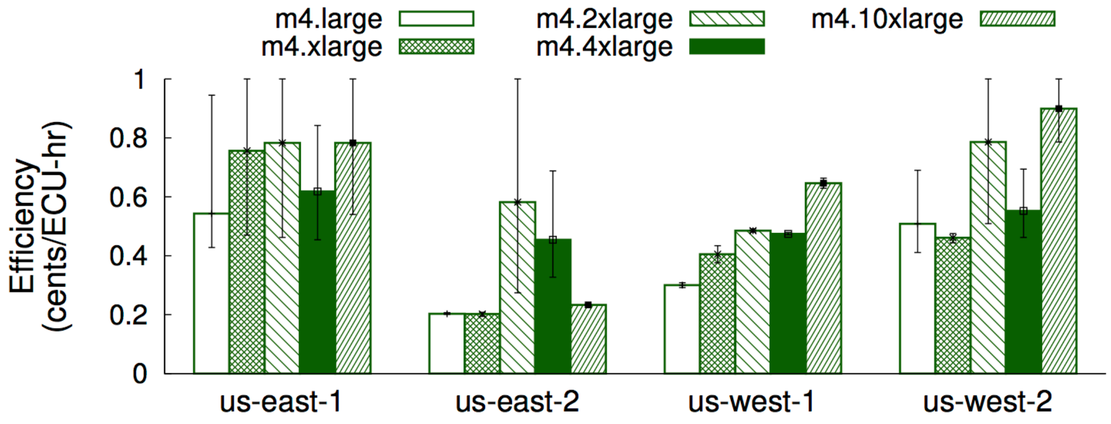
Effect on Revocation Risk and Performance

Insights from spot market analysis

1 Highly discounted servers tend to have lower revocation risk



2 Cost efficiency is uncorrelated with VM capacity (and thus performance)



Server hopping lowers revocations without necessarily degrading performance

Design of Server Hopping Logic

Migration policy

(Run on a VM that has the best cost-efficiency in $/utilized-resource without hindering the performance)

Policy invariant

Trigger a check whenever

- VM utilization changes
- spot market prices change

Cost-benefit analysis

Migrate to the spot vm that gives the highest cost-benefit gain

Expected benefit

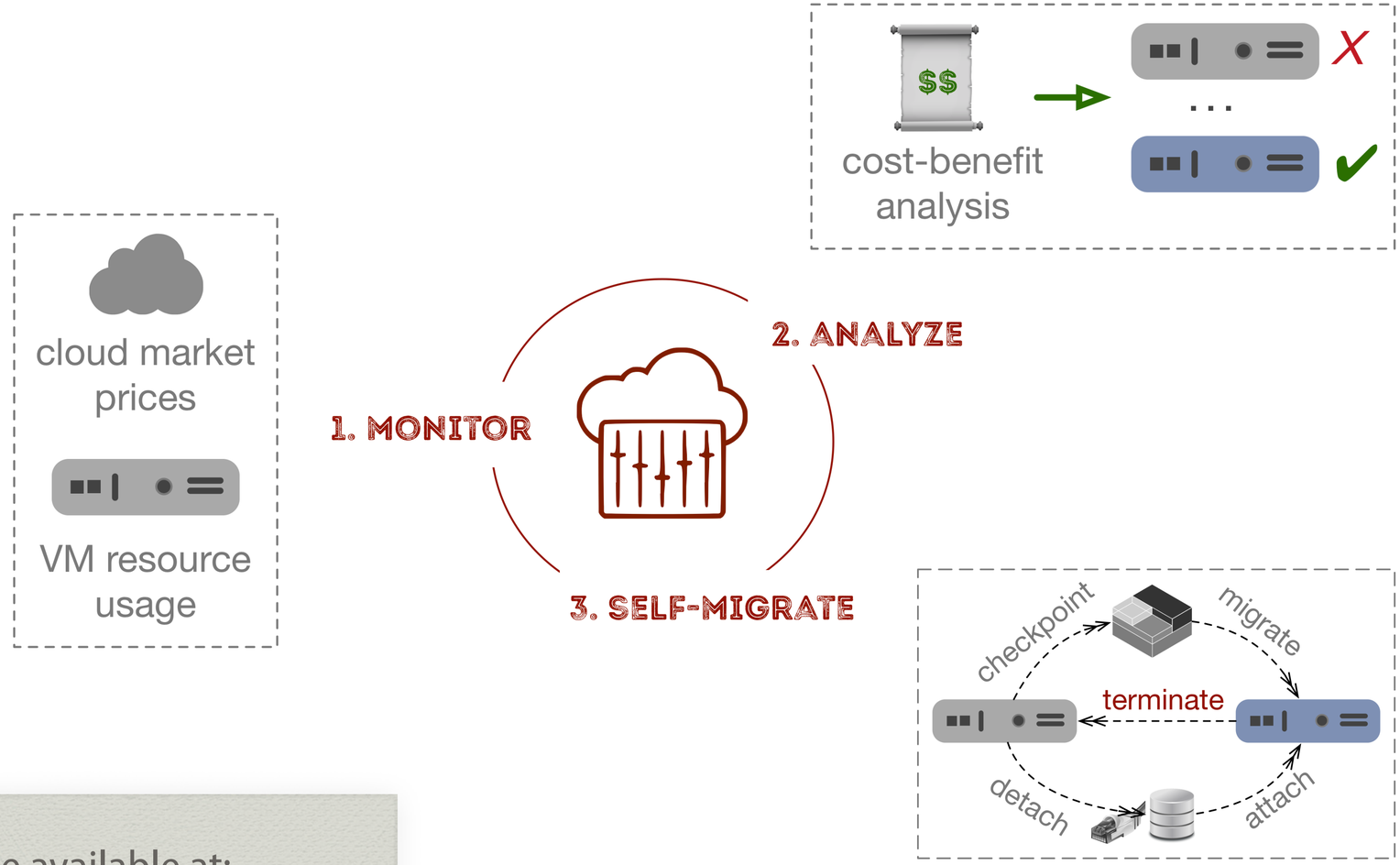
- Gain in cost-efficiency for the duration of expected stay
- f(market characteristics)

>

Migration cost

- Double-paying for VMs + min. VM holding time
- f(application footprint)

HotSpot: Design and Implementation



Fully functional prototype available at:

https://sustainablecomputinglab.github.io/hotspot/

Evaluation

Compare **cost, performance, revocations** of running a flexible batch application on

Spot VM with no protection (SpotFleet) vs. **Spot VM with fault-tolerance (SpotOn [SoCC 2015])** vs. **Spot VM with server hopping (HotSpot)**

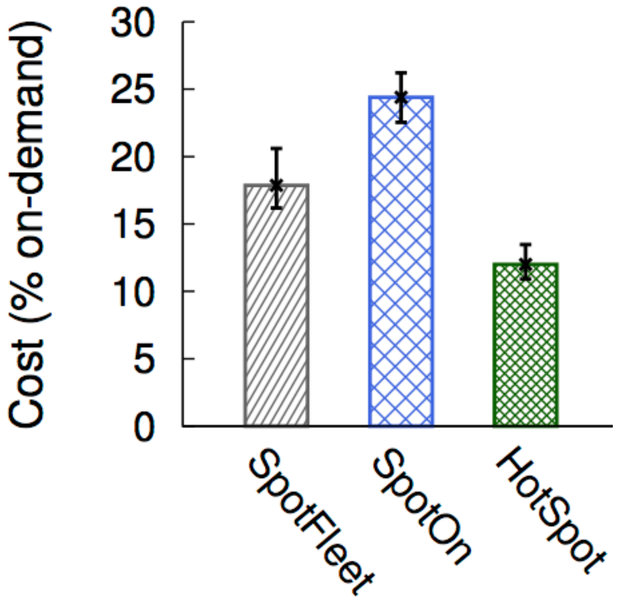
1. How do changes in job and market characteristics affect each approach?

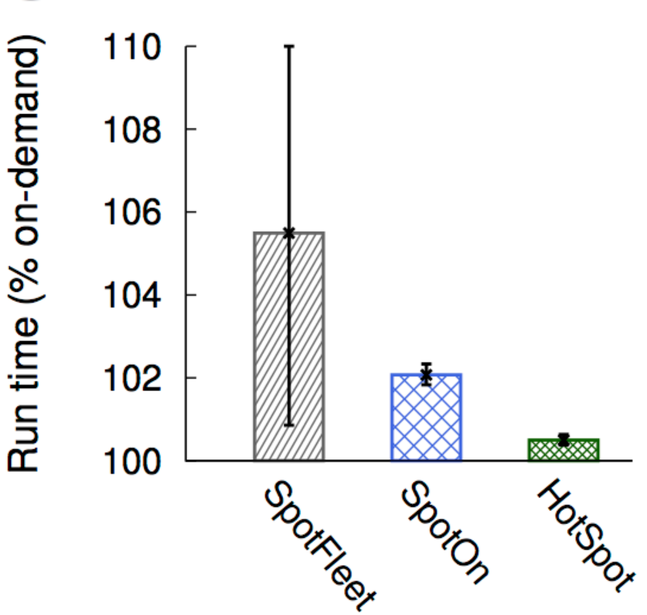
Run the prototypes on EC2 (but control job and market conditions using emulators)

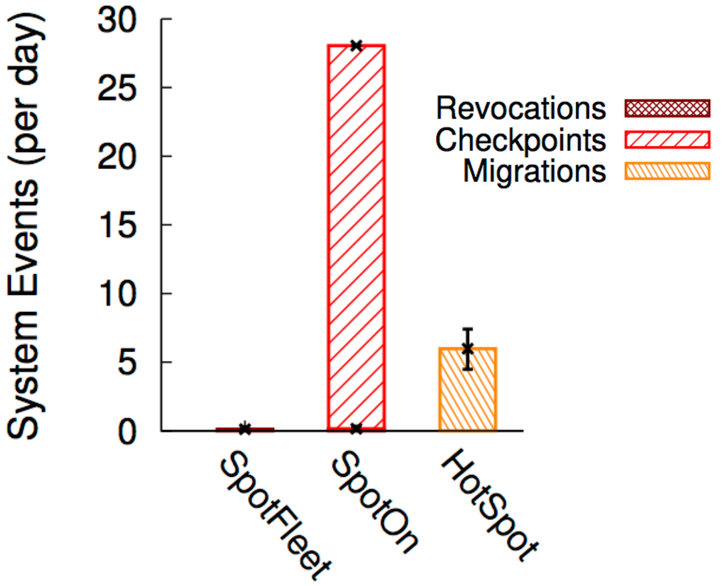
2. How do different approaches perform on the real market for real jobs?

Simulate running Google cluster trace jobs on Amazon spot price traces (03/2017 to 05/2017)

Google Cluster Traces on EC2 Spot Markets







Even in the current EC2 spot markets (with low revocation rates),

optimizing for price-risk results in ***30-50% additional savings*** without degrading performance

Conclusion

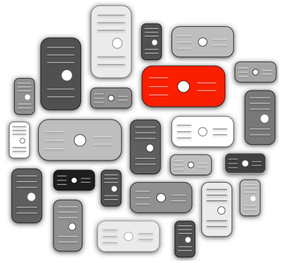
Transient server markets are an emerging area and offer many opportunities for cost savings

Price Risk

Price risk is significant in current spot markets

Mitigating price risk also reduces revocations

HotSpot



Proposed the technique of **automated server hopping**

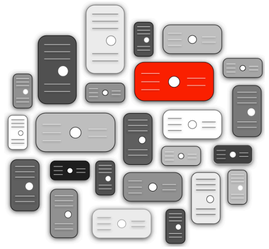
Designed and implemented HotSpot for EC2 spot markets

Evaluations

30-50% COST REDUCTION

vs. other techniques

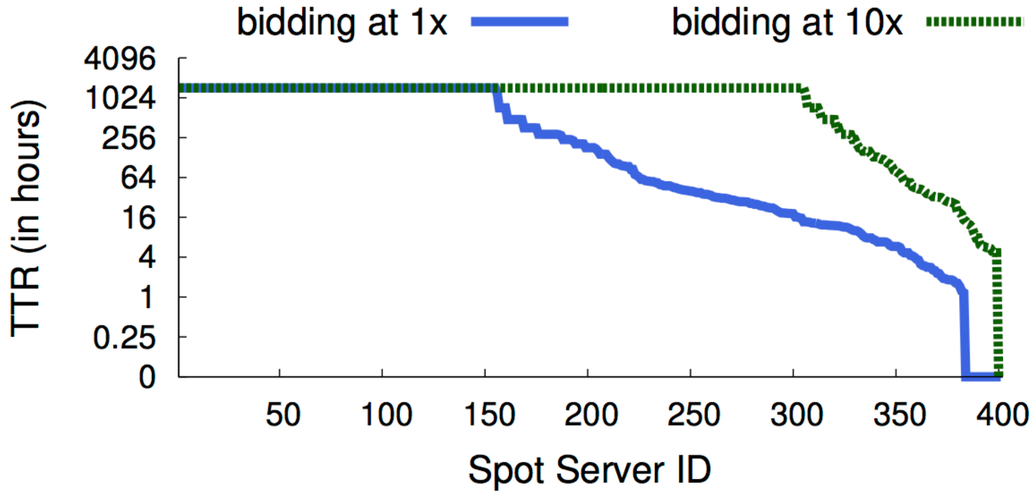
- Lower Overhead
- Lower Revocations
- More Deterministic



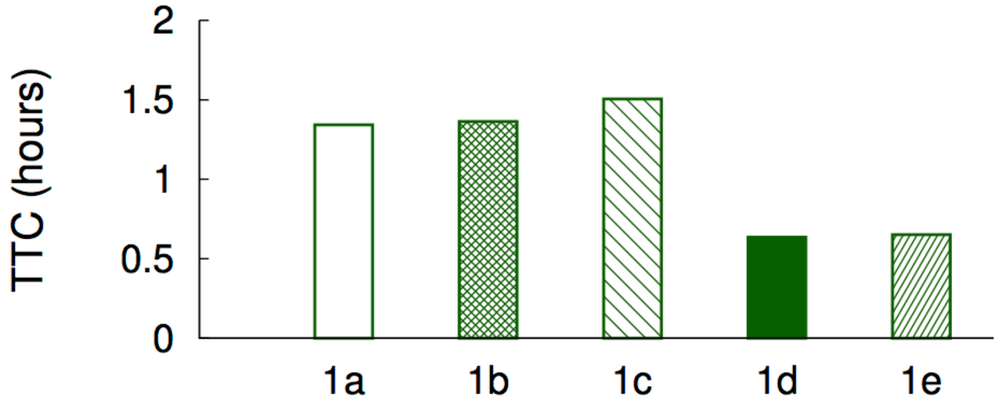
Backup Slides

Price Risk >> Revocation Risk

Data from all 402 spot VMs in US-East-1 over 3/1/2017 to 5/1/2017



Mean Time-to-Revocation (TTR) when bidding 1x is **~25 days** and 10x is **~47 days**

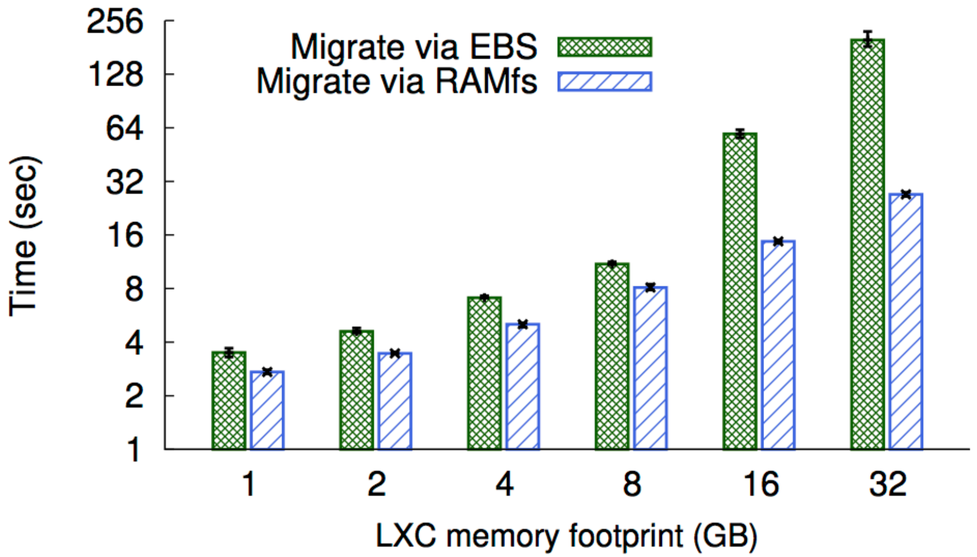


Time-to-Change (TTC) for the cheapest VM is **1.1 hours**

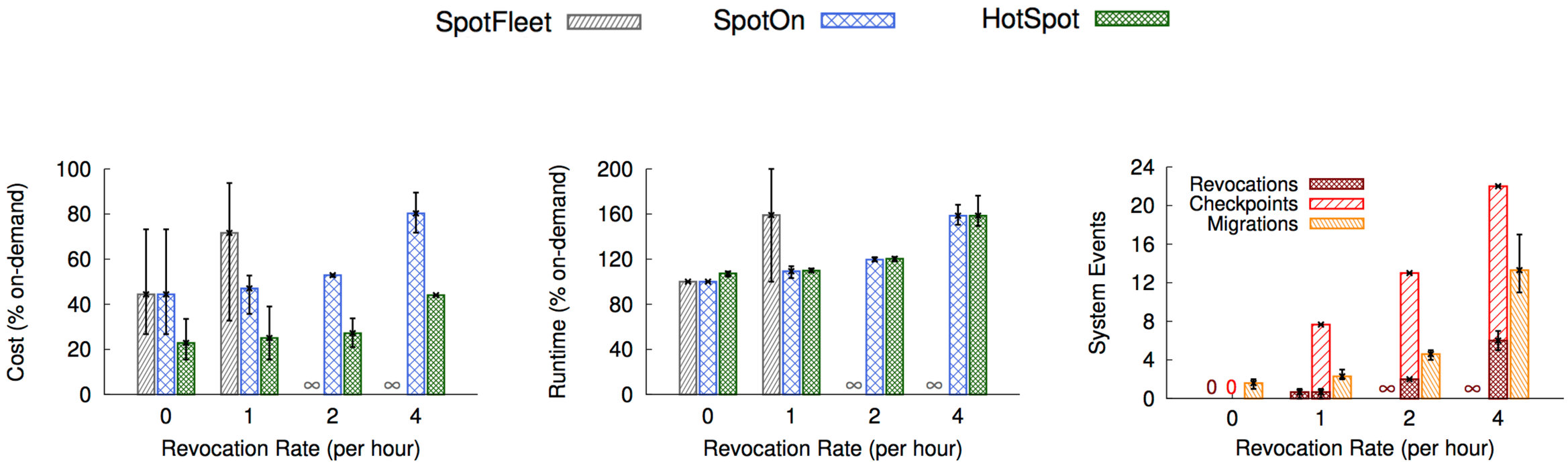
Migration Latencies in EC2

Operation	Min (sec)	Mean (sec)	Max (sec)
Price and Resource Monitoring	<1	<1	<1
Acquire On-demand VM	16	28	31
Acquire Spot VM	31	67	167
Transferring Disk & Network	18	28	48
Terminate Source VM	31	44	46
Total	~64-80	~101-140	~126-262

Platform's API operations



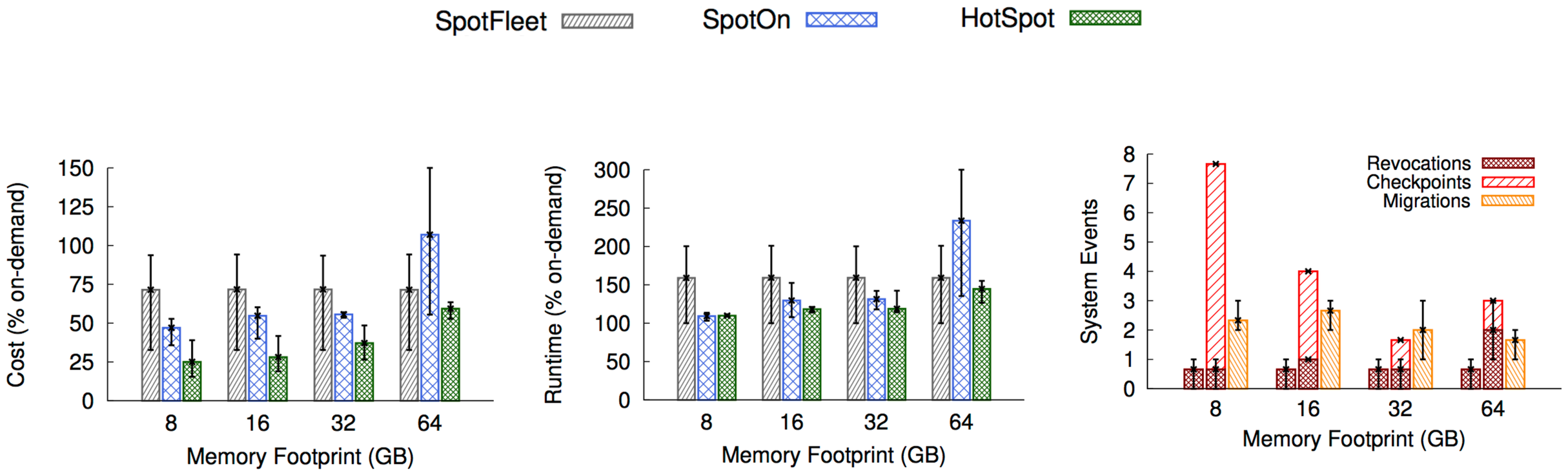
Effect of Changes in Market Volatility



As markets become more volatile,

HotSpot's savings will improve relative to SpotFleet and SpotOn

Effect of Changes in App Footprint



HotSpot outperforms both SpotFleet and SpotOn at all levels, though it's gains reduce with increase in the memory footprint.