System Support for **Managing Risk** in Cloud Computing Platforms

PhD oral defense

Supreeth Shastri

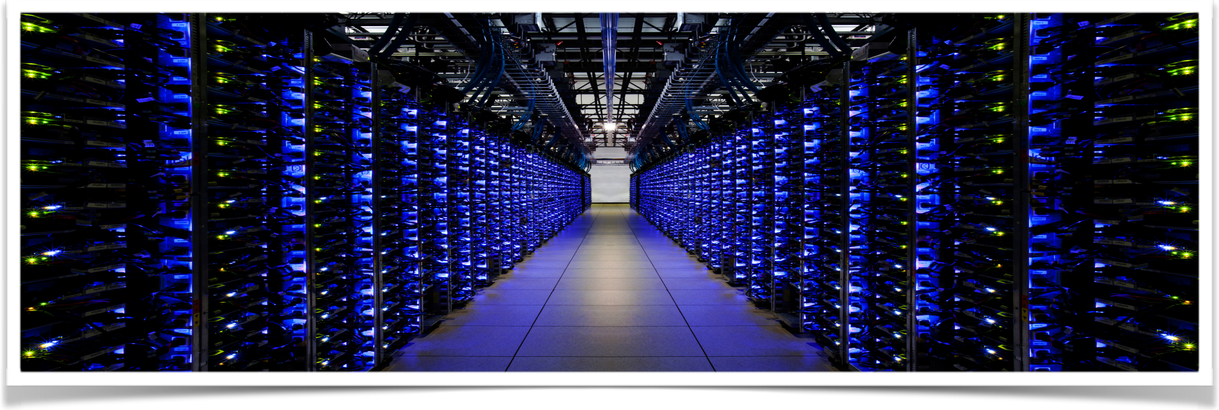
University of Massachusetts Amherst

Compute server

Static characteristics that do not change in the lifetime of an application

“A rose by any other name would smell as sweet ”

Compute server

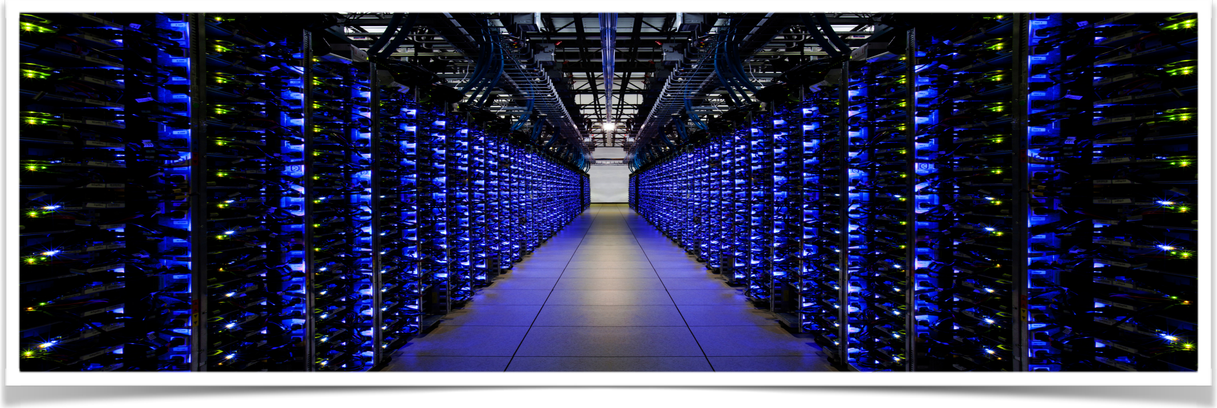


Compute Servers in the Cloud

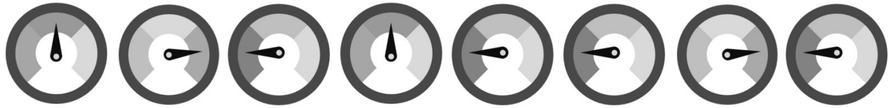
Static characteristics that do not change in the lifetime of an application

"A rose by any other name ~~would~~ *may not* smell as sweet"

Compute server



Compute Servers in the Cloud

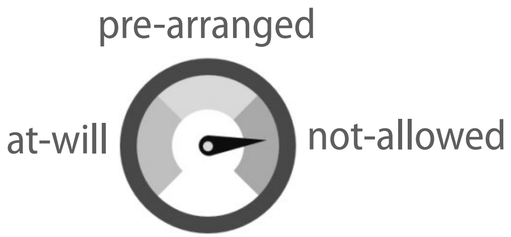


Static characteristics that do not change in the lifetime of an application

All service-level characteristics are **controlled** in fine granularity

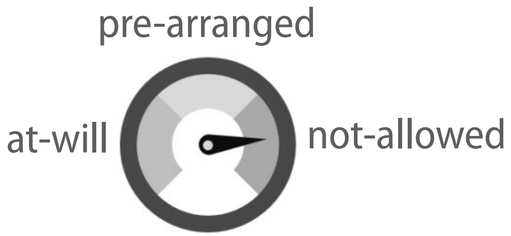
What has the Cloud Done to Our Good Old Servers?

What has the Cloud Done to Our Good Old Servers?

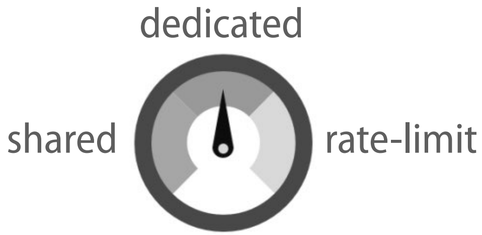


Revocability

What has the Cloud Done to Our Good Old Servers?

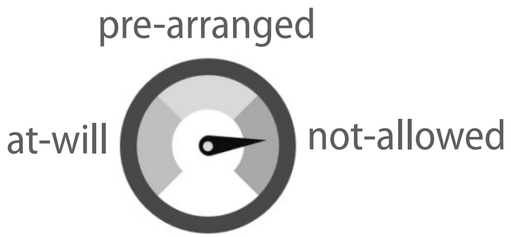


Revocability



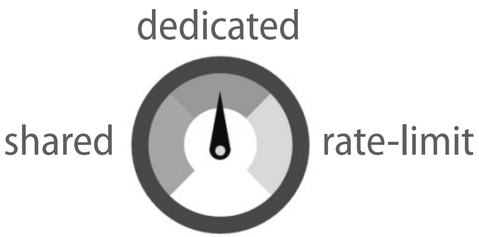
Performance Guarantees

What has the Cloud Done to Our Good Old Servers?



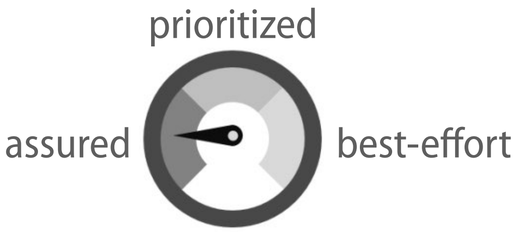
Revocability

Pricing Model

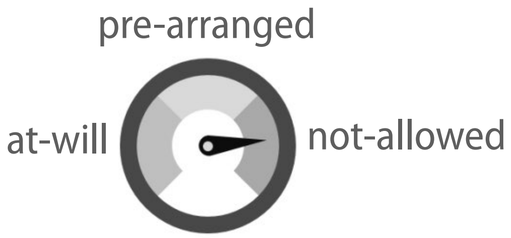


Performance Guarantees

What has the Cloud Done to Our Good Old Servers?

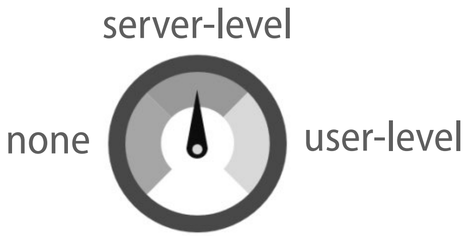


Availability

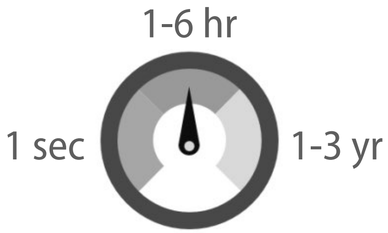


Revocability

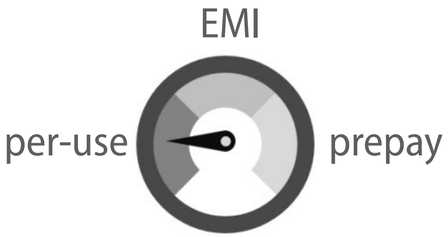
Pricing Model



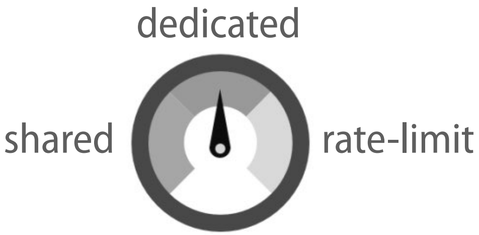
Isolation



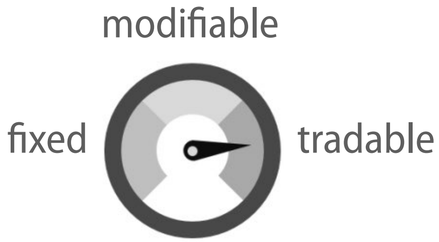
Time Commitment



Billing Frequency

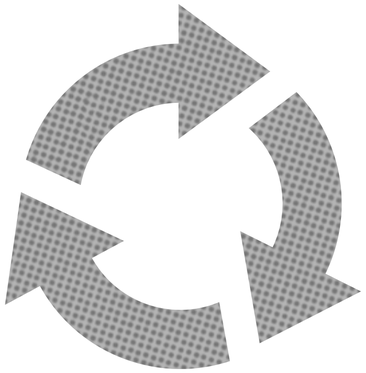


Performance Guarantees



Terms of Contract

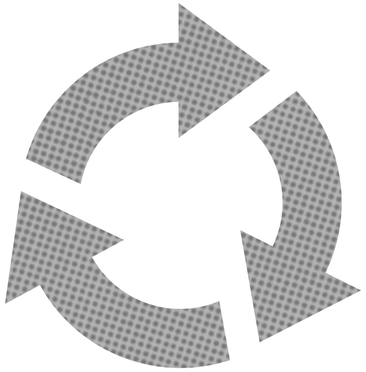
Cloud risks are exposed throughout the computing lifecycle



Cloud risks are exposed throughout the computing lifecycle

Rejection Risk

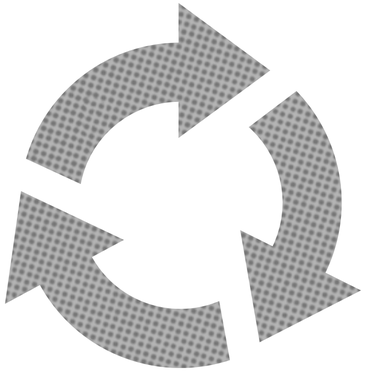
When users request for resources,
it may be denied by the provider



Cloud risks are exposed throughout the computing lifecycle

Rejection Risk

When users request for resources,
it may be denied by the provider



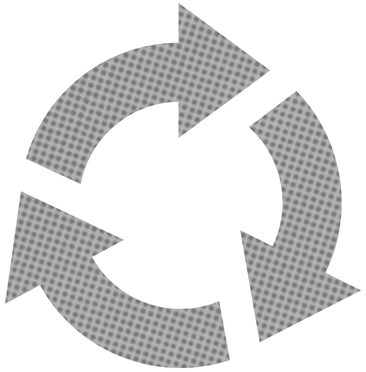
Revocation Risk

Allocated resources may become
unavailable or withdrawn

Cloud risks are exposed throughout the computing lifecycle

Rejection Risk

When users request for resources,
it may be denied by the provider



Revocation Risk

Allocated resources may become
unavailable or withdrawn

Price Risk

Acquired server's price
may rise relative to others

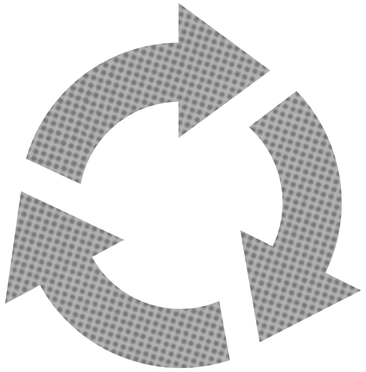
Cloud risks are exposed throughout the computing lifecycle

Rejection Risk

When users request for resources,
it may be denied by the provider

Valuation Risk

Users may not be able to value a
server's utility for their application



Revocation Risk

Allocated resources may become
unavailable or withdrawn

Workload Risk

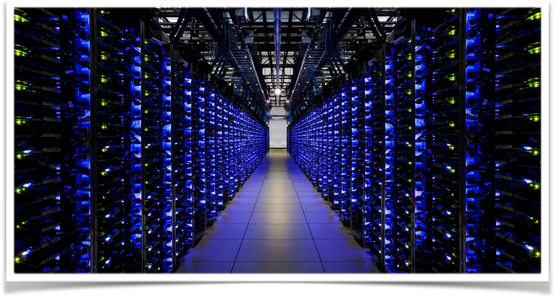
Workload requirements do not
match the acquired servers

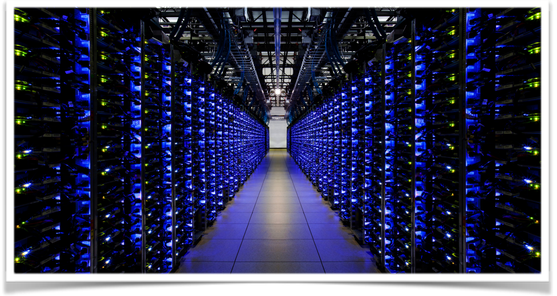
Price Risk

Acquired server's price
may rise relative to others

Uncertainty Risk

Price or performance variability
that makes planning difficult

Find in  = **Cloud Nirvana?**

Find in  = Cloud Nirvana?

There is No Such Thing as a Risk-free Cloud Contract

	Rejection	Revocation	Price	Uncertainty	Workload	Valuation
EC2/GCE On-demand	Yes	No	No	Yes	No	No
EC2/GCE Reserved	No	No	Yes	Yes	Yes	No
EC2 Spot	Yes	Yes	Yes	Yes	No	Yes
GCE Preemptible	Yes	Yes	No	Yes	No	Yes
EC2 Defined-Duration	Yes	No	Yes	Yes	Yes	Yes
EC2/GCE Burstable	Yes	No	No	Yes	No	Yes

Image source: www.freepik.com

Image source: www.freepik.com

Short-term rental	Assisted living	Long-term room share	Lease-to-own

Image source: www.freepik.com

Contract diversity is omnipresent in the real world

Short-term rental	Assisted living	Long-term room share	Lease-to-own

Image source: www.freepik.com

Limitations of Current Systems

“ Risk comes from not knowing what you are doing ” – Warren Buffett

Limitations of Current Systems

“ Risk comes from not knowing what you are doing ” – Warren Buffett

Applications
Batch, Interactive, Iterative, Real-time

System Frameworks
Apache Spark and Hadoop

Cluster Managers
Kubernetes, Apache Mesos

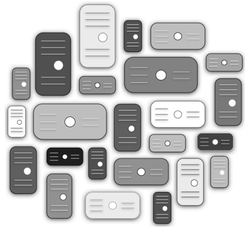
Limitations of Current Systems

" Risk comes from not knowing what you are doing " – Warren Buffett

Applications
Batch, Interactive, Iterative, Real-time

System Frameworks
Apache Spark and Hadoop

Cluster Managers
Kubernetes, Apache Mesos



+

On-Demand BURSTABLE
RESERVED SPOT MARKET
Preemptible Dedicated

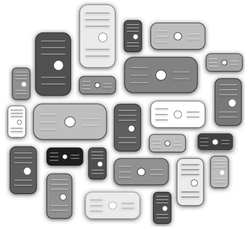
Limitations of Current Systems

" Risk comes from not knowing what you are doing " – Warren Buffett

Applications
Batch, Interactive, Iterative, Real-time

System Frameworks
Apache Spark and Hadoop

Cluster Managers
Kubernetes, Apache Mesos



+ **On-Demand** BURSTABLE
RESERVED SPOT MARKET
Preemptible Dedicated

1. Current system software **ignore the risks** exposed by the cloud contracts
2. In turn, **pass the burden** of risk management to the applications
3. Unprepared applications incur **potential losses** due to unmanaged cloud risks

We **observe** that the cloud has evolved into an ***advanced marketplace*** but customers operating there ***lack the knowhow*** and toolset to manage their risks.

We **observe** that the cloud has evolved into an ***advanced marketplace*** but customers operating there ***lack the knowhow*** and toolset to manage their risks.

Our work elevates **risk management** to a first-class **system design** principle

---

We **observe** that the cloud has evolved into an ***advanced marketplace*** but customers operating there ***lack the knowhow*** and toolset to manage their risks.

Our work elevates **risk management** to a first-class **system design** principle

---

Then, we **hypothesize** that by ***adapting and extending*** techniques from ***economics and finance***, we can effectively manage the cloud risks.

Financializing Cloud Computing

Financializing Cloud Computing

1

Translate cloud server's characteristics and system software's actions into *financial value for the application*

Financializing Cloud Computing

1

Translate cloud server's characteristics and system software's actions into *financial value for the application*

2

Analyze risk-reward tradeoffs, and **rationalize** decision making

Financializing Cloud Computing

1. **Translate** cloud server's characteristics and system software's actions into *financial value for the application*

2. **Analyze** risk-reward tradeoffs, and **rationalize** decision making

3. Build system software that manages the risks **autonomously**

Structure for the rest of the talk…

Structure for the rest of the talk…

Mitigating Revocation Risk with Insurance	Minimizing Valuation Risk via Asset Pricing	Reducing Price Risk thru' Active Trading	Eliminating Uncertainty Risk by Index Tracking
SpotOn [SoCC 2015]	Transient Guarantee [SC 2016]	HotSpot [SoCC 2017]	Cloud Index [In submission]

Transient server **revocations** are intentional, frequent and come with advanced warning. How to manage this **new threat model**?

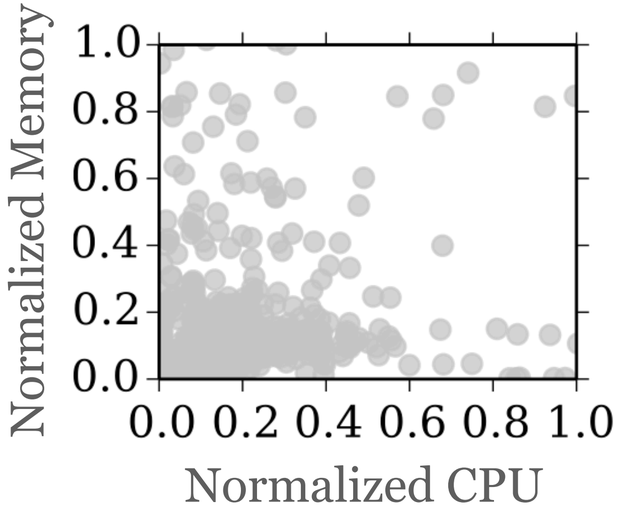
Transient server **revocations** are intentional, frequent and come with advanced warning. How to manage this **new threat model**?

Mitigating Revocation Risk with Insurance	Minimizing Valuation Risk via Asset Pricing	Reducing Price Risk thru' Active Trading	Eliminating Uncertainty Risk by Index Tracking
SpotOn [SoCC 2015]	Transient Guarantee [SC 2016]	HotSpot [SoCC 2017]	Cloud Index [In submission]

Run **batch jobs** using **fault-tolerance** on **spot markets**

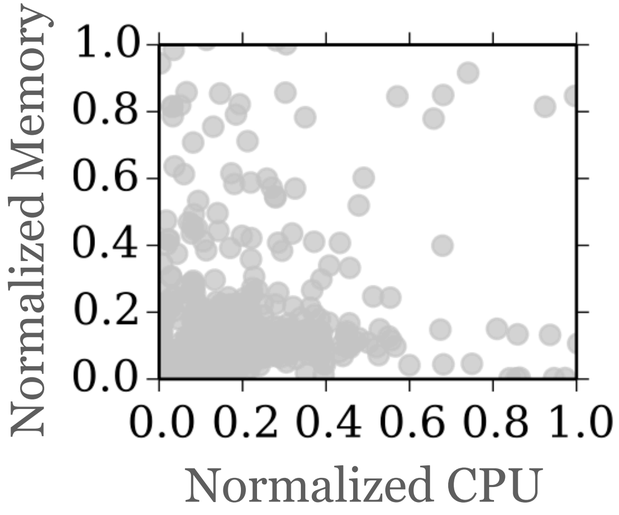
Run **batch jobs** using **fault-tolerance** on **spot markets**

Jobs from Google cluster traces

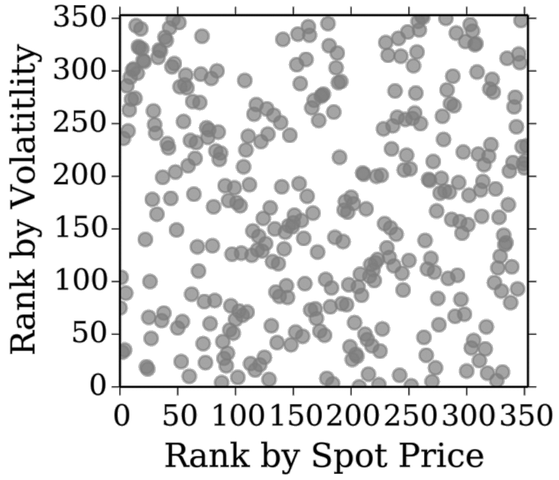


Run **batch jobs** using **fault-tolerance** on **spot markets**

Jobs from Google cluster traces

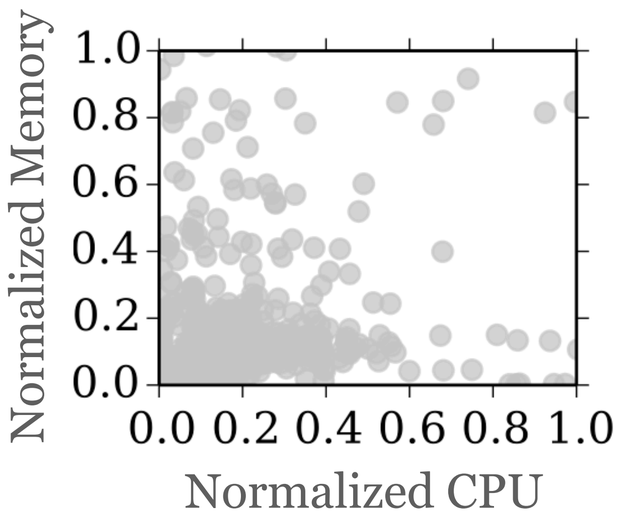


Linux spot markets in US-East-1



Run **batch jobs** using **fault-tolerance** on **spot markets**

Jobs from Google cluster traces

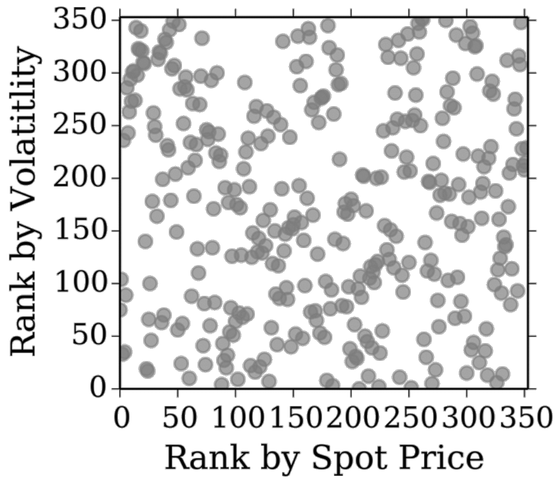


Proactive Checkpointing

Proactive Replication

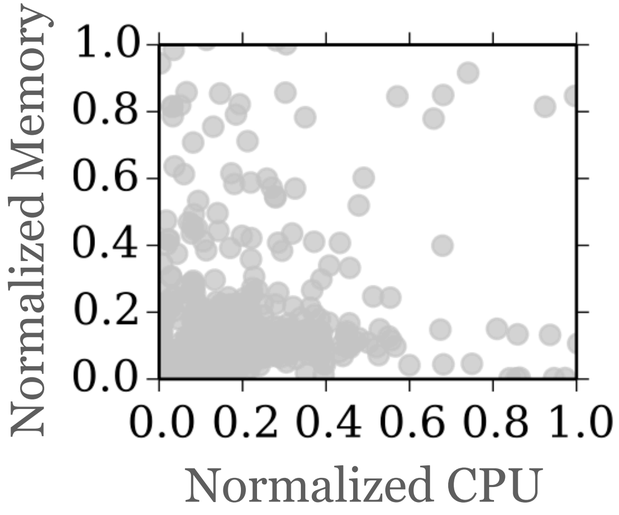
Reactive Migration

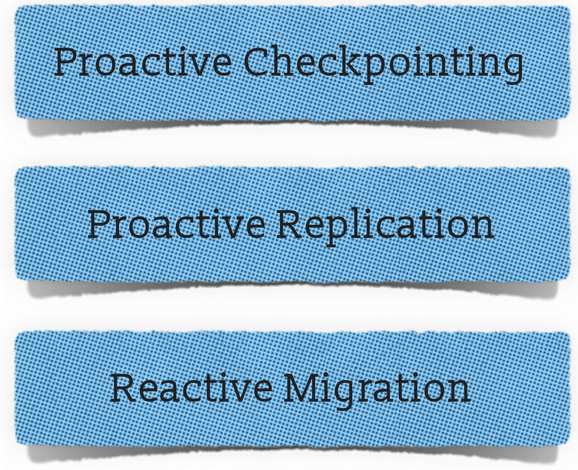
Linux spot markets in US-East-1



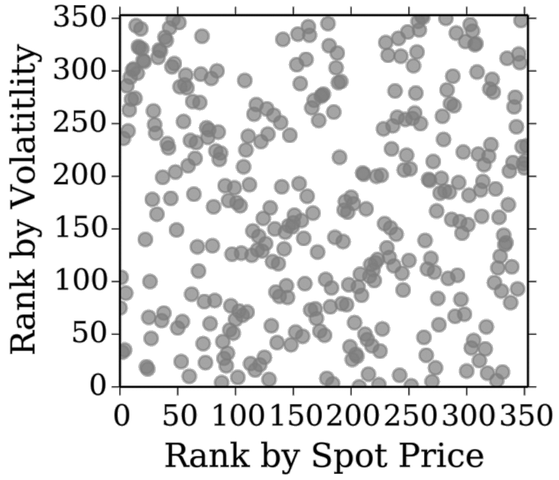
Run **batch jobs** using **fault-tolerance** on **spot markets**

Jobs from Google cluster traces





Linux spot markets in US-East-1



Applications, fault-tolerance mechanisms and spot servers exhibit **diverse characteristics**

Modeling Fault-tolerance Overhead

Modeling Fault-tolerance Overhead

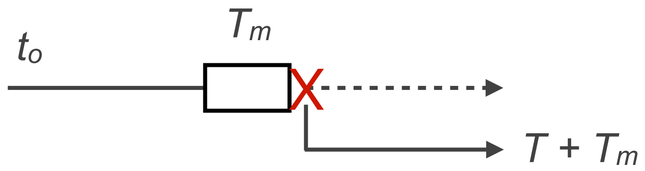
Reactive **Migration**

Proactive **Checkpointing**

Replication

Modeling Fault-tolerance Overhead

Reactive **Migration**



Overhead

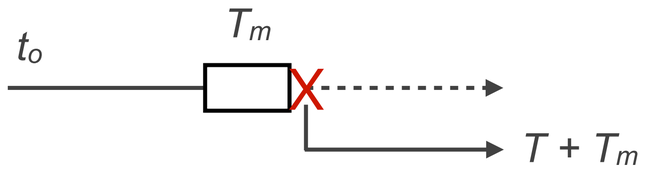
f (job's resource usage, market characteristics)

Proactive **Checkpointing**

Replication

Modeling Fault-tolerance Overhead

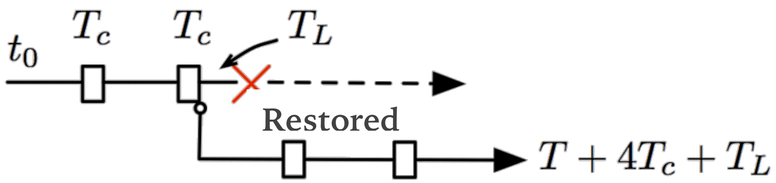
Reactive **Migration**



Overhead

f (job's resource usage, market characteristics)

Proactive **Checkpointing**



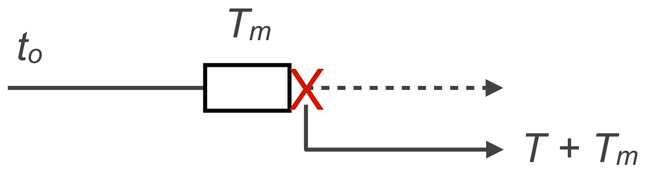
Overhead

f (job's resource usage)

Replication

Modeling Fault-tolerance Overhead

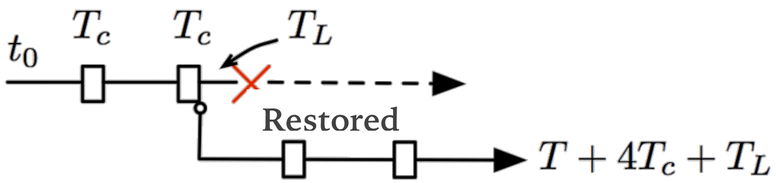
Reactive **Migration**



Overhead

f (job's resource usage, market characteristics)

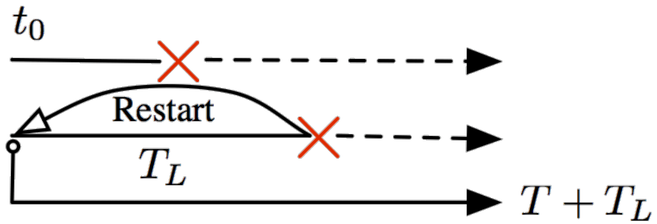
Proactive **Checkpointing**



Overhead

f (job's resource usage)

Replication

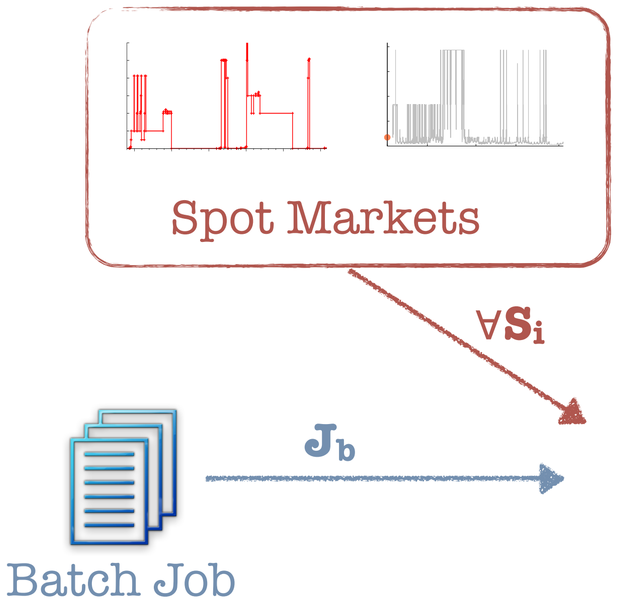


Overhead

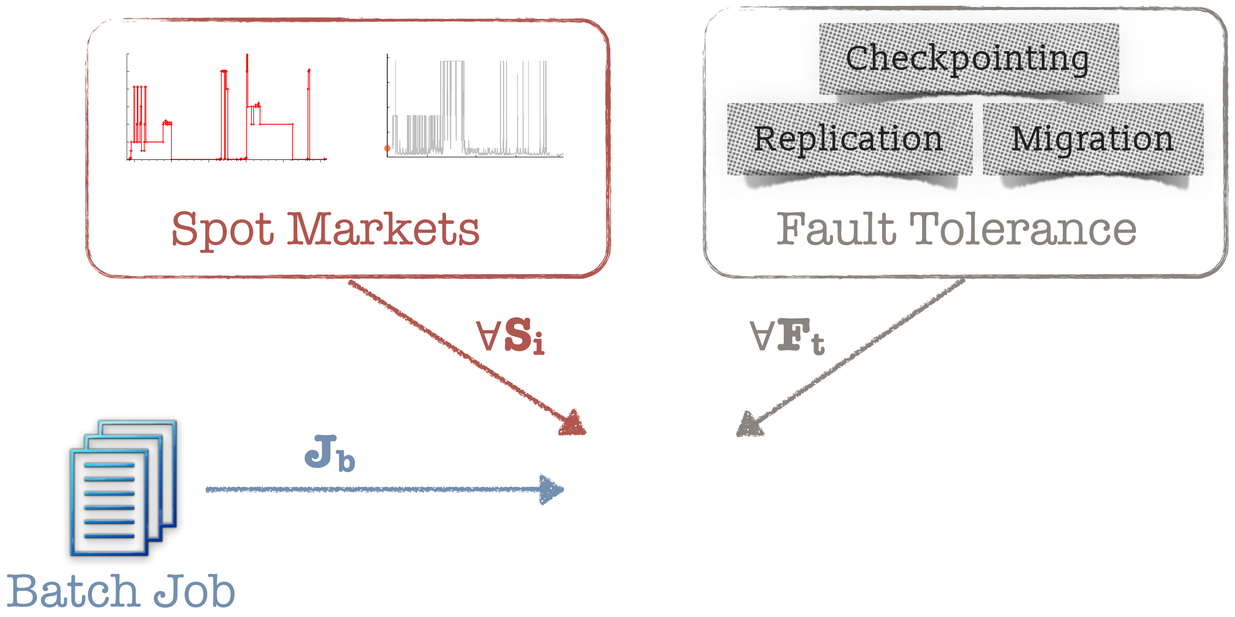
f (market characteristics, replication factor)

Cost-efficient Insurance via Greedy Selection

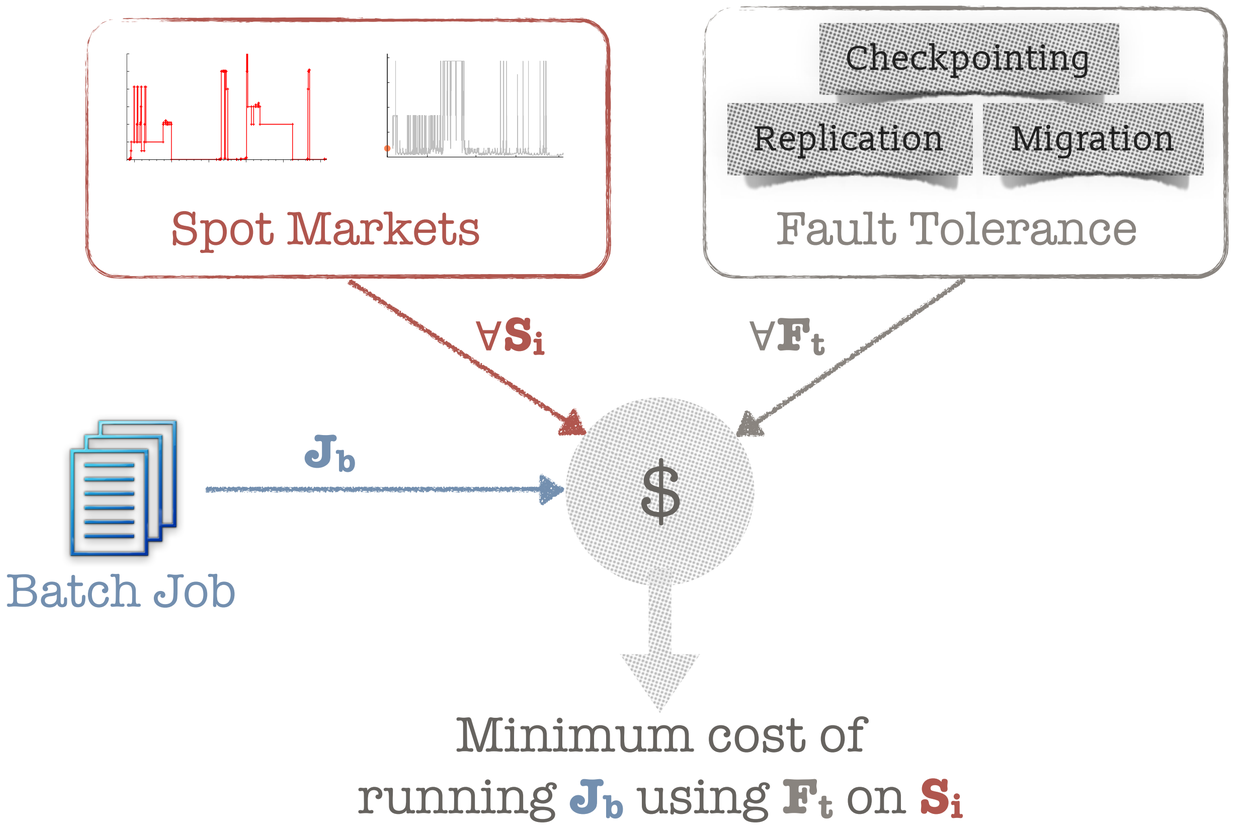
Cost-efficient Insurance via Greedy Selection



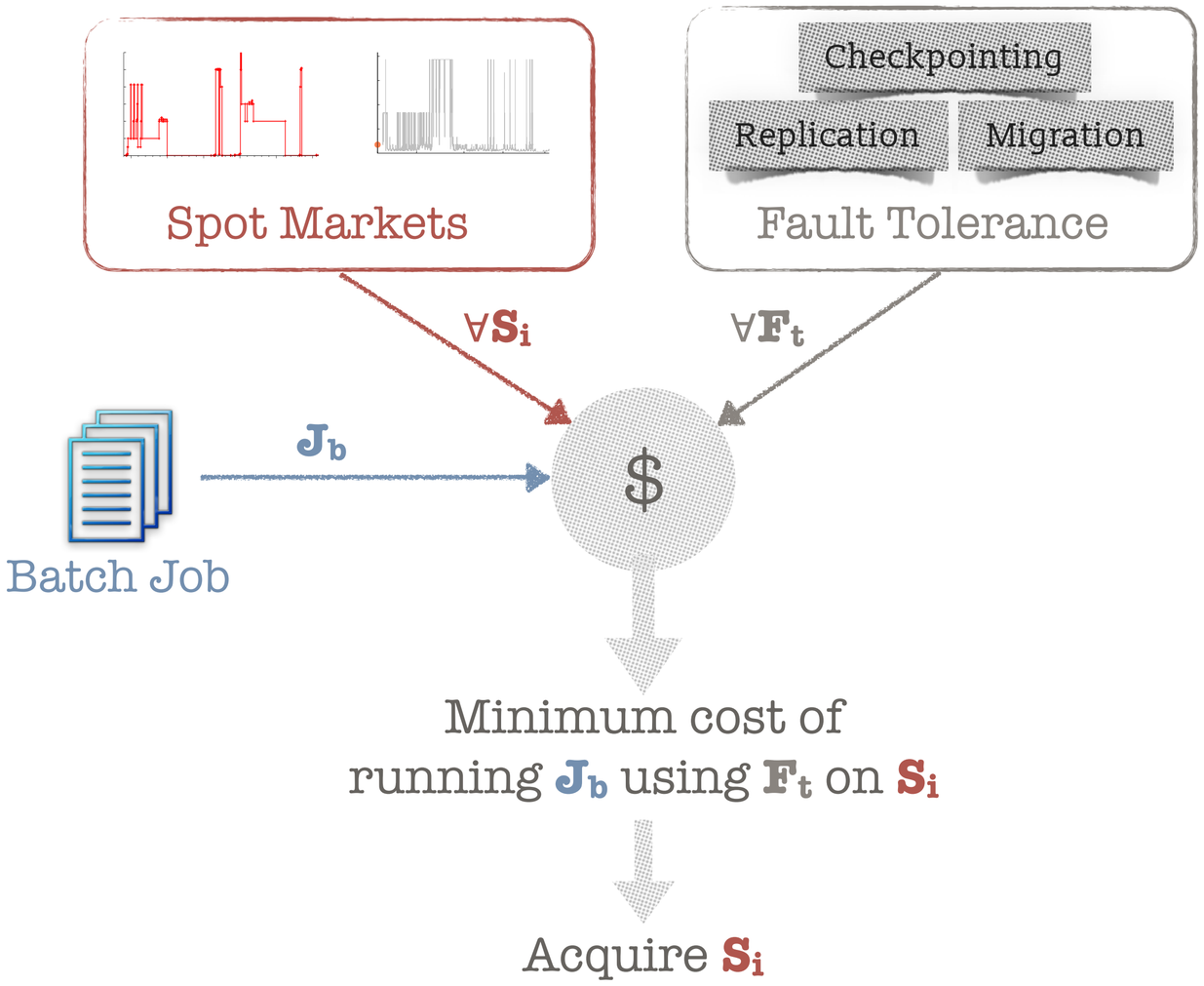
Cost-efficient Insurance via Greedy Selection



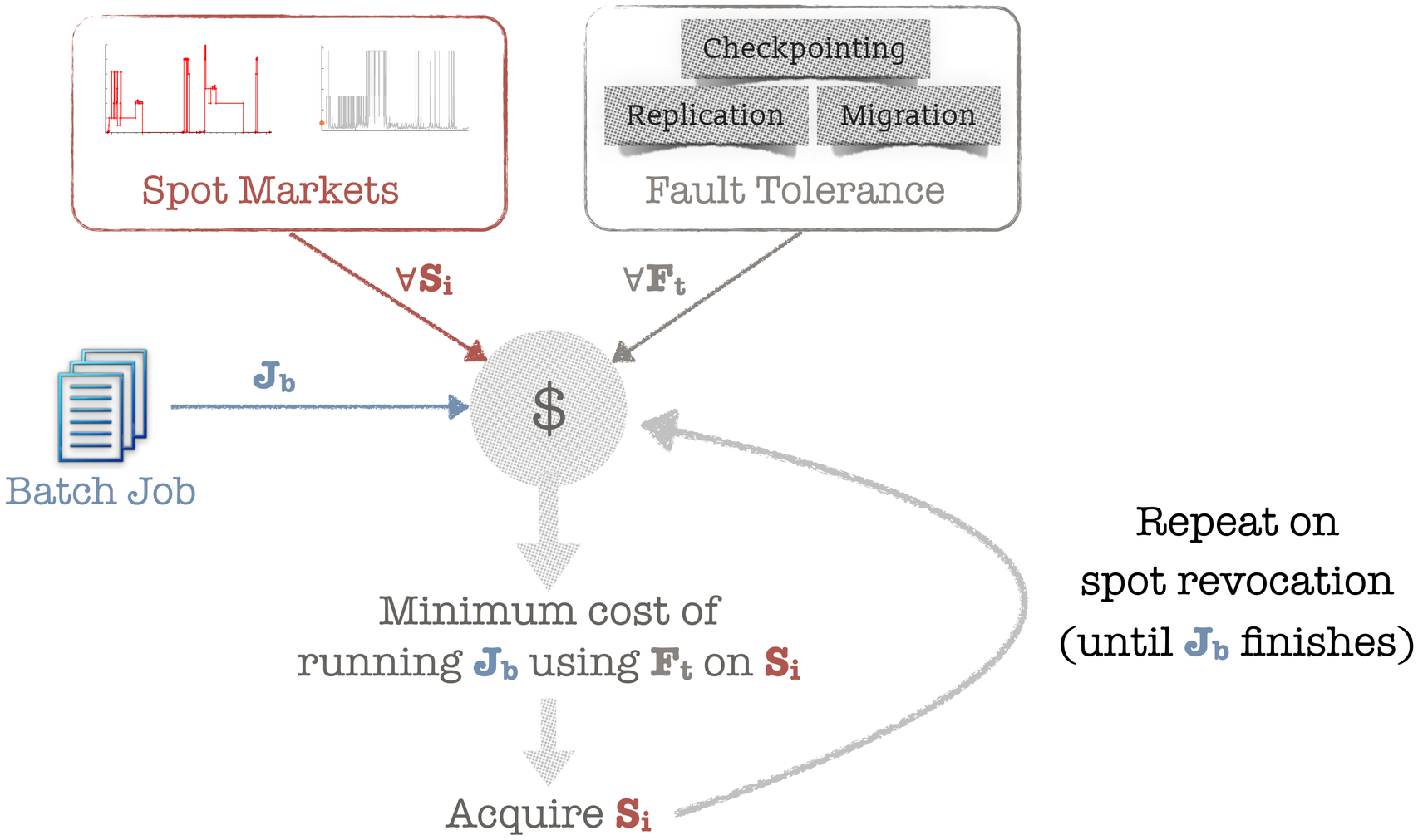
Cost-efficient Insurance via Greedy Selection



Cost-efficient Insurance via Greedy Selection



Cost-efficient Insurance via Greedy Selection

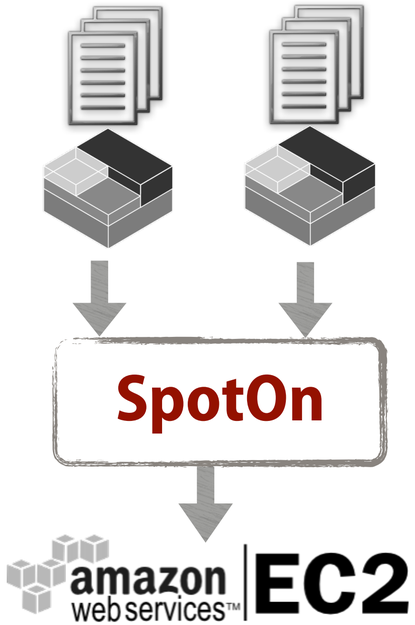


SpotOn: Design and Implementation

- Achieve on-demand-level **performance** but paying spot-level **price**
- Build a **self-optimizing** insurance policy at the system-level

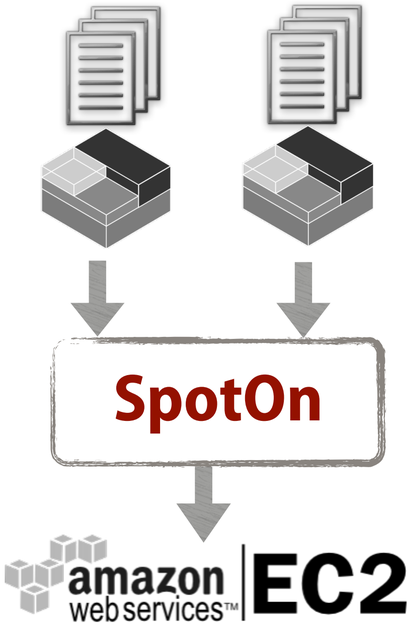
SpotOn: Design and Implementation

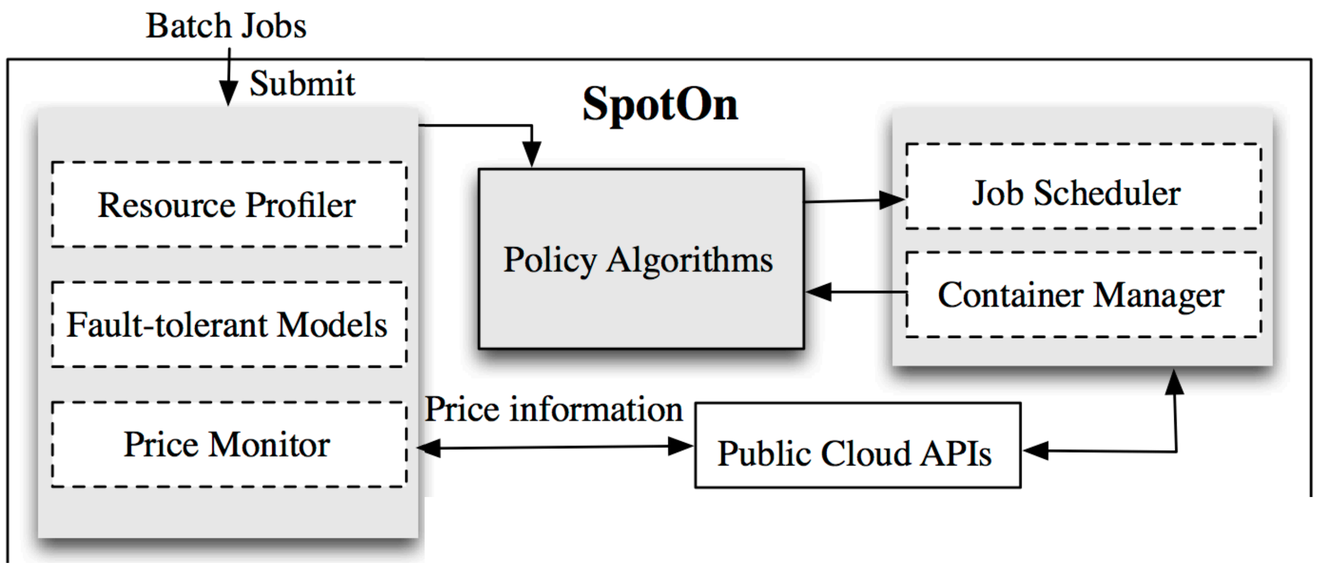
- Achieve on-demand-level **performance** but paying spot-level **price**
- Build a **self-optimizing** insurance policy at the system-level



SpotOn: Design and Implementation

- Achieve on-demand-level **performance** but paying spot-level **price**
- Build a **self-optimizing** insurance policy at the system-level

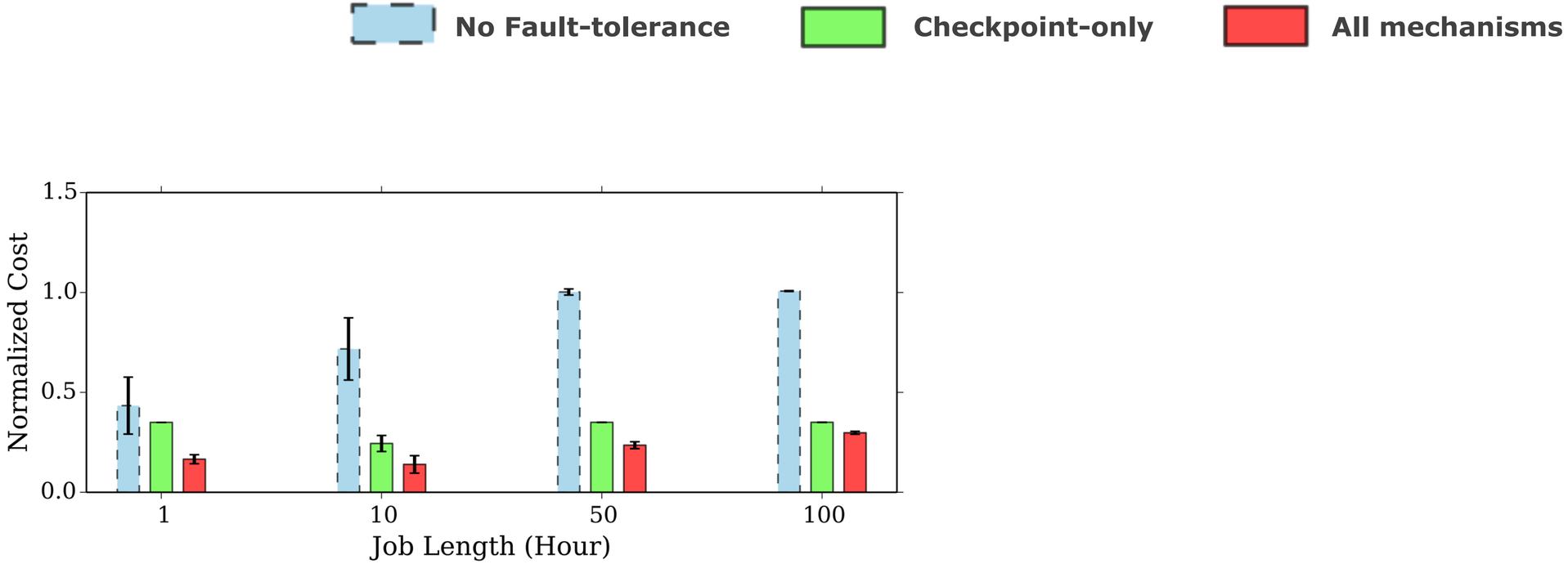




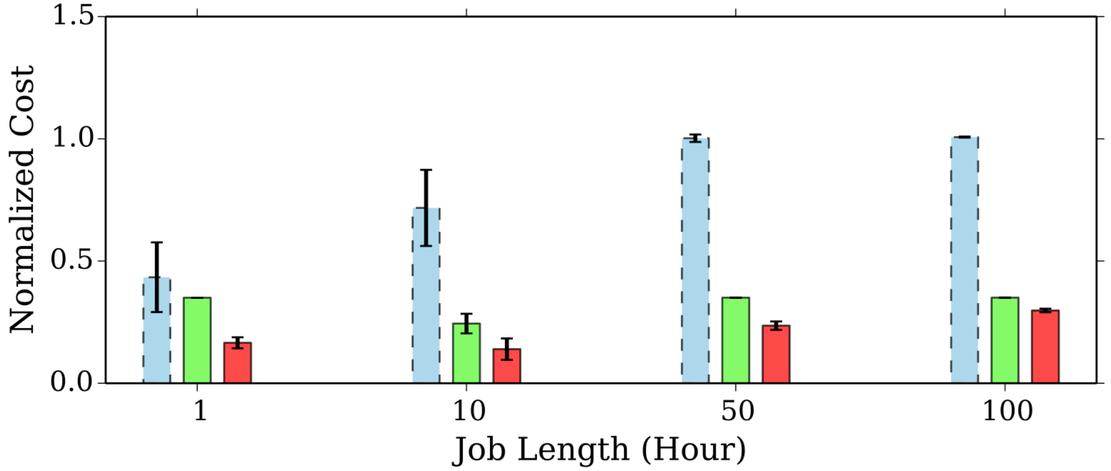
SpotOn's architectural components

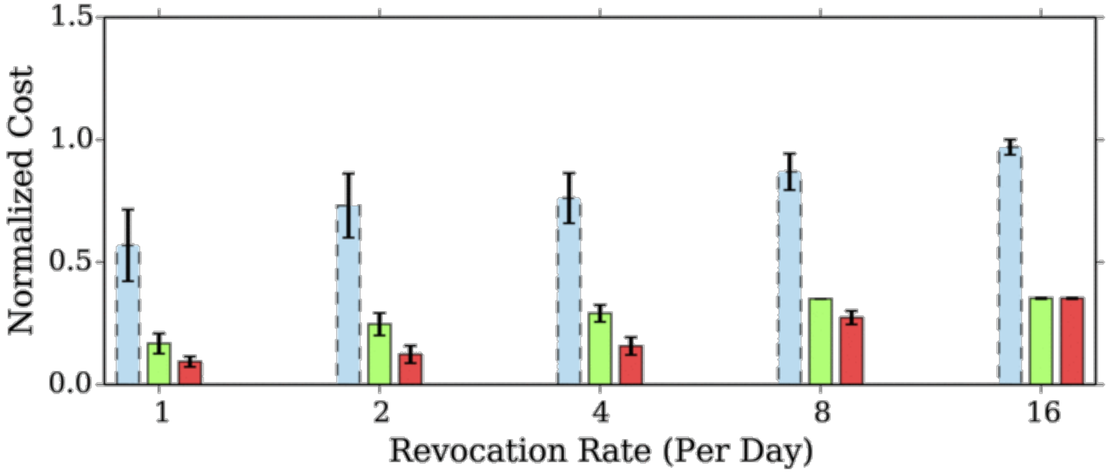
Evaluation: Cost-efficient Insurance Policy

Evaluation: Cost-efficient Insurance Policy

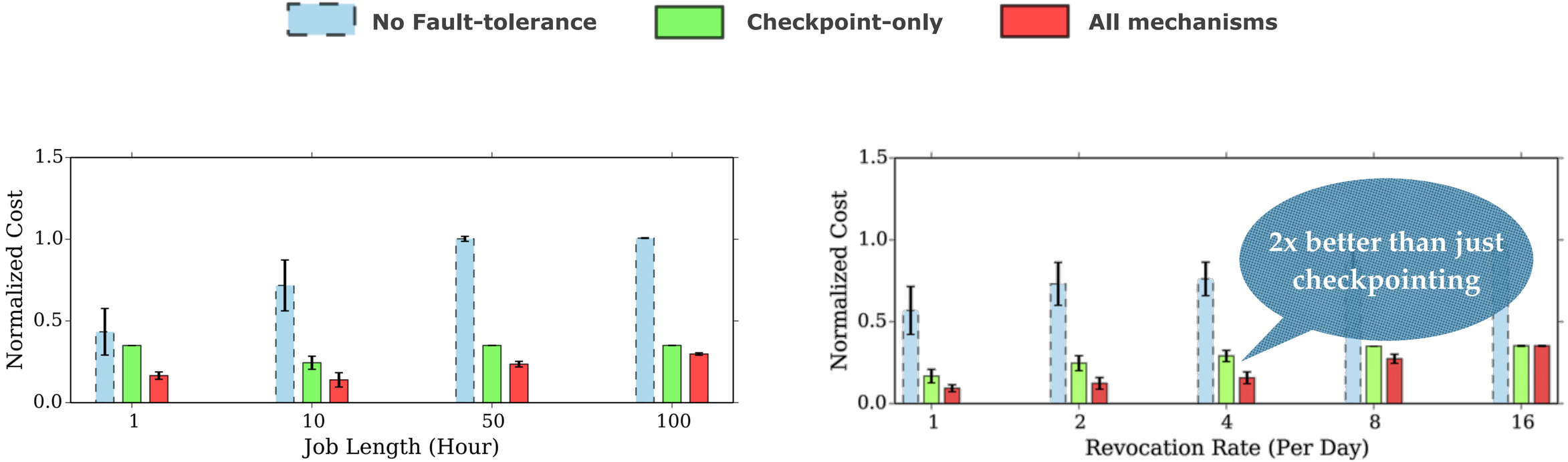


Evaluation: Cost-efficient Insurance Policy

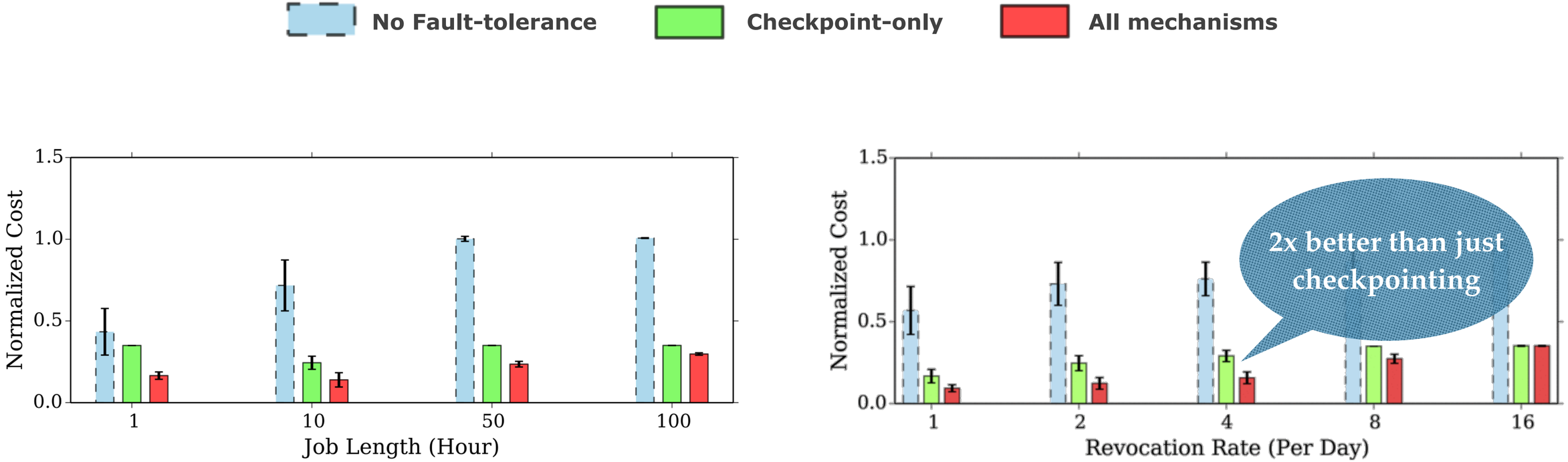




Evaluation: Cost-efficient Insurance Policy



Evaluation: Cost-efficient Insurance Policy



On Google cluster traces, SpotOn achieved **91.9% savings** compared to on-demand with little impact on performance

Transient server contracts do not reveal precise transiency information.

This opacity makes it difficult to **gauge** their **true value**.

Transient server contracts do not reveal precise transiency information.

This opacity makes it difficult to **gauge** their **true value**.

Mitigating
Revocation Risk
with Insurance

SpotOn
[SoCC 2015]

Minimizing
Valuation Risk
via Asset Pricing

Transient Guarantee
[SC 2016]

Reducing
Price Risk thru'
Active Trading

HotSpot
[SoCC 2017]

Eliminating
Uncertainty Risk
by Index Tracking

Cloud Index
[In submission]

Transient Servers are Intrinsically Less Valuable!

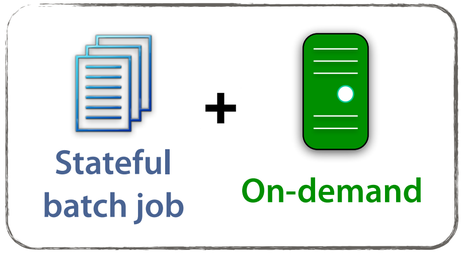
Equilibrium Price of Transient Server
(or *price when spot stops being cheap*)

$$P_{eq} = P_{\text{on-demand}} * \frac{T_{\text{on-demand}}}{E[T_{\text{spot}}]}$$

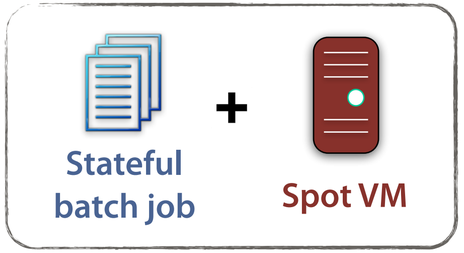
Transient Servers are Intrinsically Less Valuable!

Equilibrium Price of Transient Server
(or *price when spot stops being cheap*)

$$P_{eq} = P_{on\text{-}demand} * \frac{T_{on\text{-}demand}}{E[T_{spot}]}$$



Completion time
12 hours

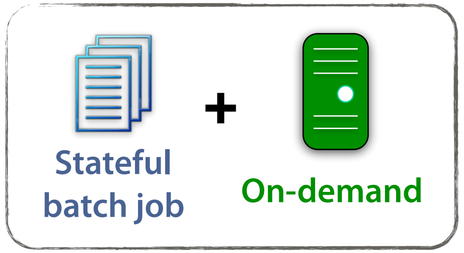


Completion time
20 hours

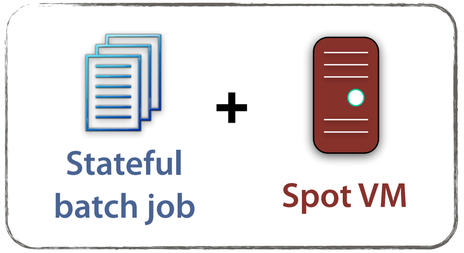
Transient Servers are Intrinsically Less Valuable!

Equilibrium Price of Transient Server
(or *price when spot stops being cheap*)

$$P_{eq} = P_{on\text{-}demand} * \frac{T_{on\text{-}demand}}{E[T_{spot}]}$$



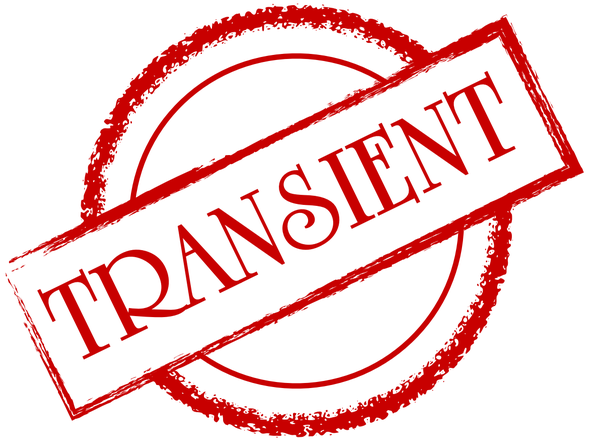
Completion time
12 hours



Completion time
20 hours

❝ For this application, a spot server with **40%** *discount, provides* **no savings at all** *❞*

Transient Guarantees



“ Uncertainty is more stressful than knowing for sure something bad will happen ”

nature COMMUNICATIONS

de Berker, Archy O., et al. *“Computations of uncertainty mediate acute stress responses in humans.”* Nature communications 7 (2016)

Not All Transient Servers are Alike

. . . and there are many ways to sell them

Idle Cloud Capacity

Not All Transient Servers are Alike

. . . and there are many ways to sell them

Idle Cloud Capacity

EC2 Spot and
GCE Preemptible

No explicit information on availability and volatility

Not All Transient Servers are Alike

. . . and there are many ways to sell them

Idle Cloud Capacity

EC2 Spot and
GCE Preemptible

No explicit information on availability and volatility

Max Performance
Offering

Every server sold with its unique revocability characteristics

Transient Guarantees

Providing probabilistic assurances on *availability, volatility and predictability* of transient servers

Transient Guarantees

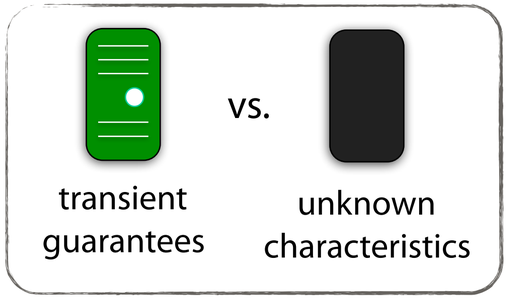
Providing probabilistic assurances on *availability, volatility and predictability* of transient servers

E.g., ***Transient Classes with advertised MTTR for each class***

Transient Guarantees

Providing probabilistic assurances on *availability, volatility and predictability* of transient servers

*E.g., **Transient Classes with advertised MTTR for each class***



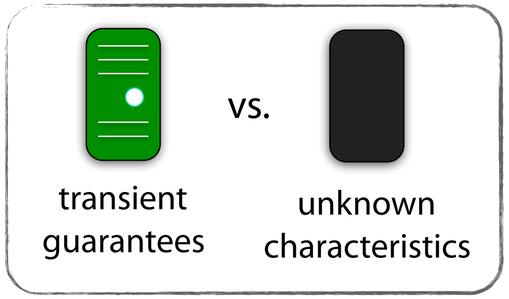
Advertised characteristics

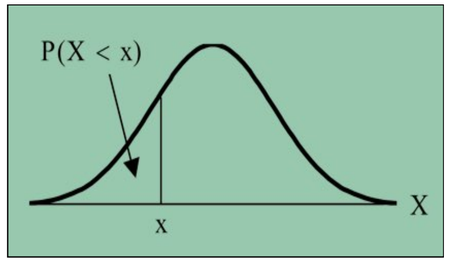
Price the servers correctly
(users and providers)

Transient Guarantees

Providing probabilistic assurances on *availability, volatility and predictability* of transient servers

*E.g., **Transient Classes with advertised MTTR for each class***





Advertised characteristics

Price the servers correctly (users and providers)

Probabilistic assurance

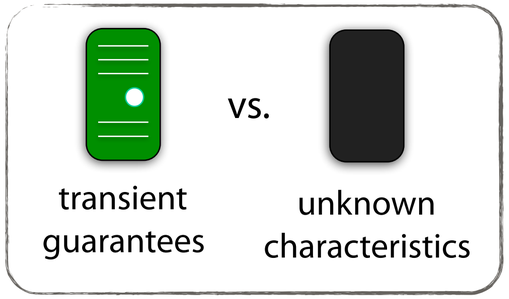
Revoke any server anytime (providers)

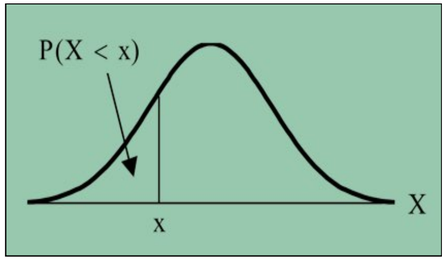
Fine-tune fault-tolerance (users)

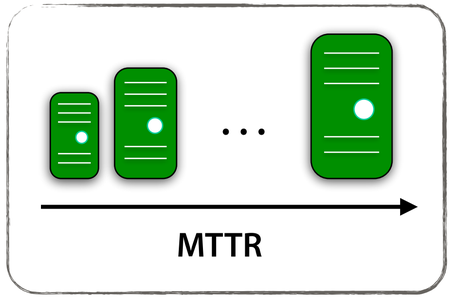
Transient Guarantees

Providing probabilistic assurances on *availability, volatility and predictability* of transient servers

*E.g., **Transient Classes with advertised MTTR for each class***







Advertised characteristics

Price the servers correctly
(users and providers)

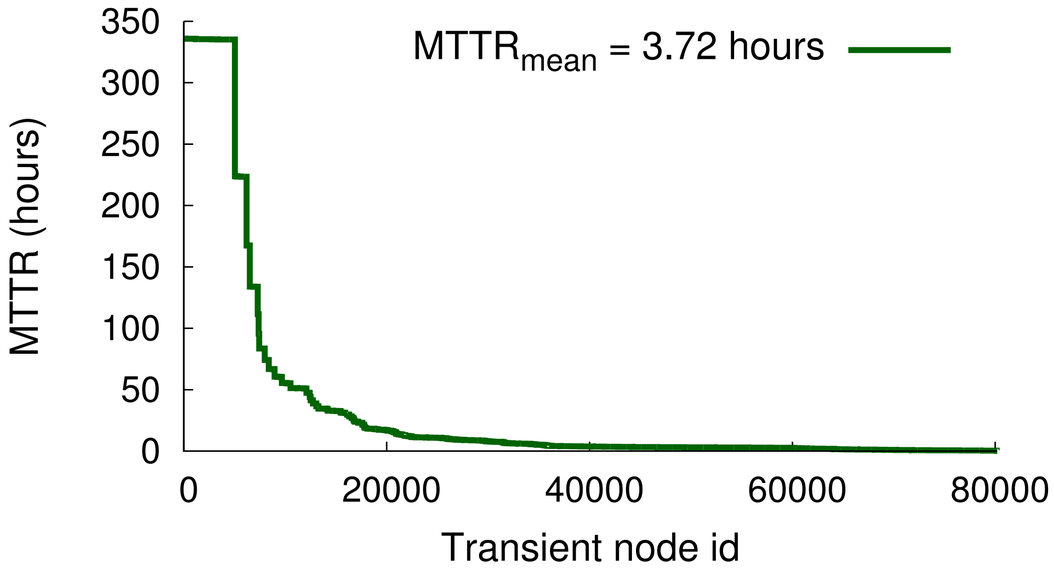
Probabilistic assurance

Revoke any server anytime (providers)
Fine-tune fault-tolerance (users)

Differentiated offering

Higher revenue (providers)
Better choice (users)

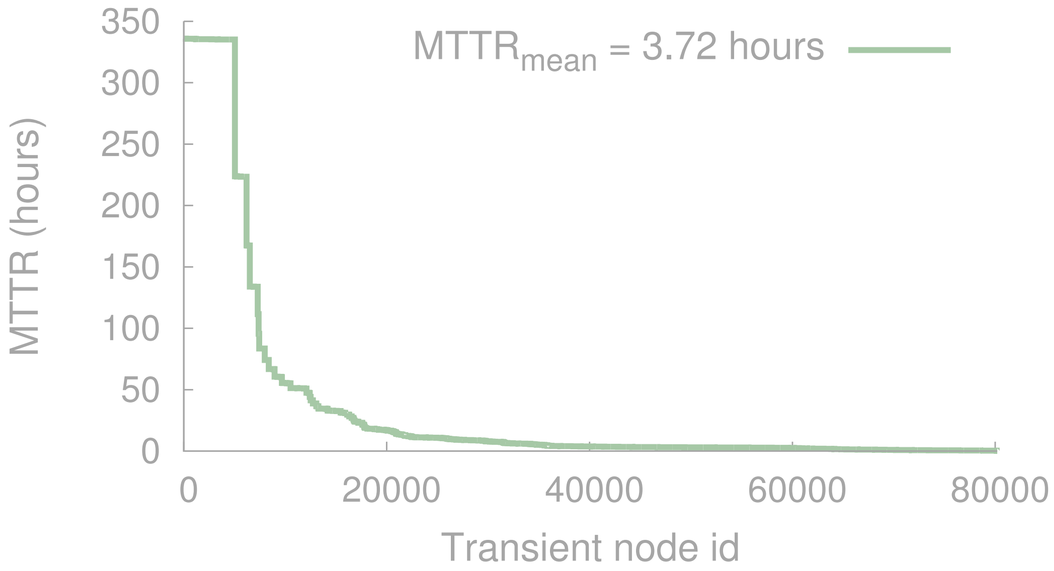
Evaluation: Transient Server Characteristics



Transient Servers exhibit
a ***wide range*** of characteristics

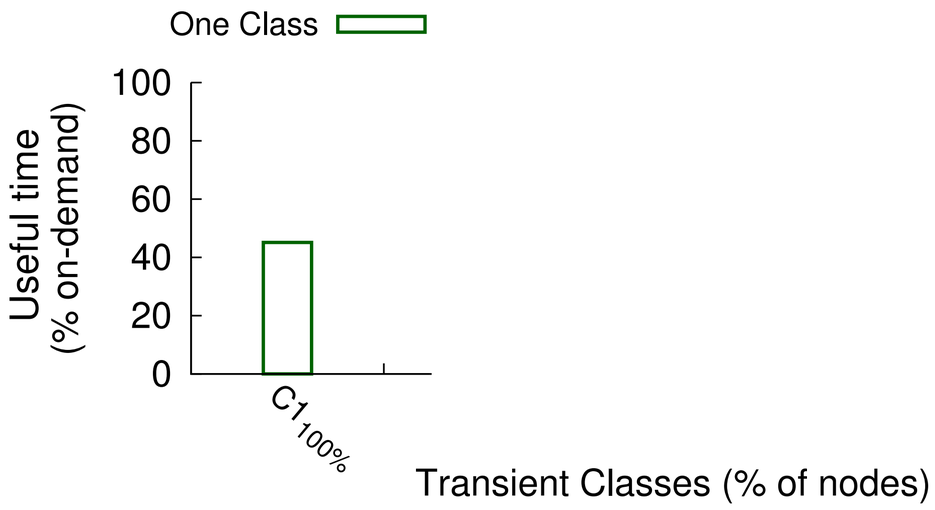
Server characteristics are modeled from a production cluster at Google

Evaluation: Transient Server Characteristics



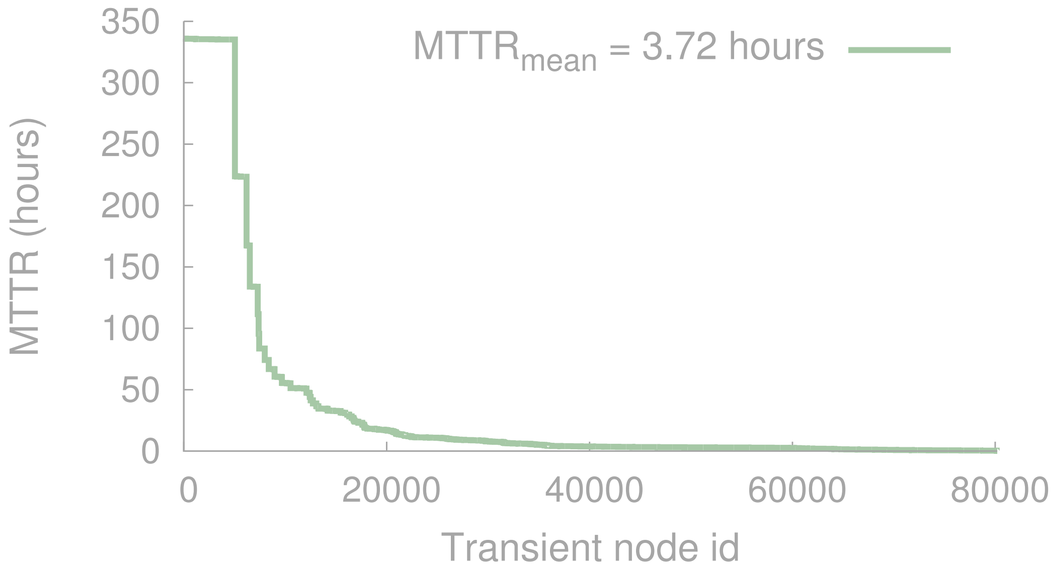
Transient Servers exhibit
a ***wide range*** of characteristics

Transient Classes create
differentiated server pools



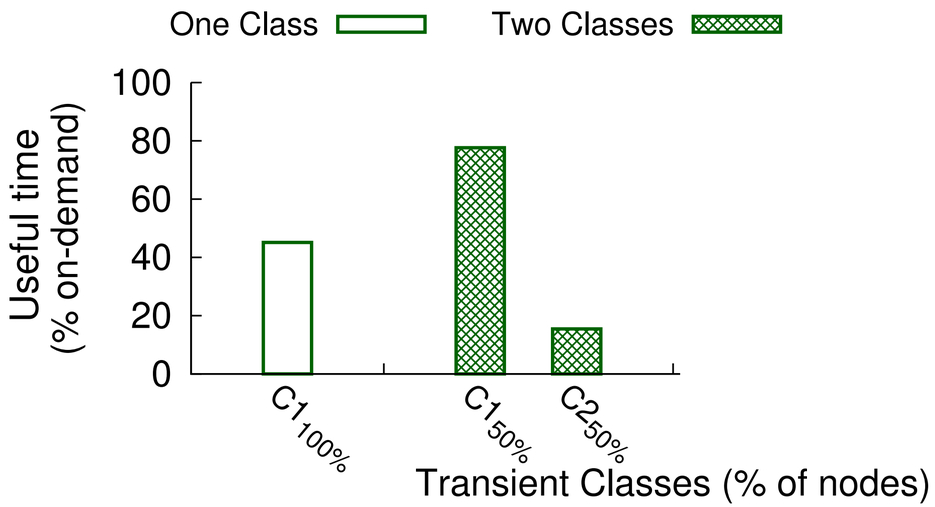
Server characteristics are modeled from a production cluster at Google

Evaluation: Transient Server Characteristics



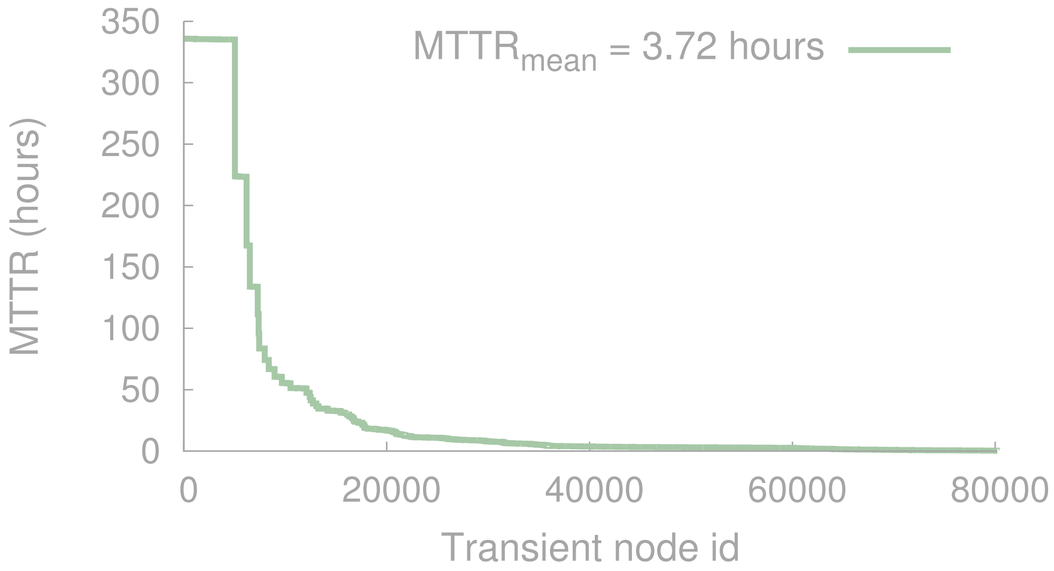
Transient Servers exhibit
a ***wide range*** of characteristics

Transient Classes create
differentiated server pools



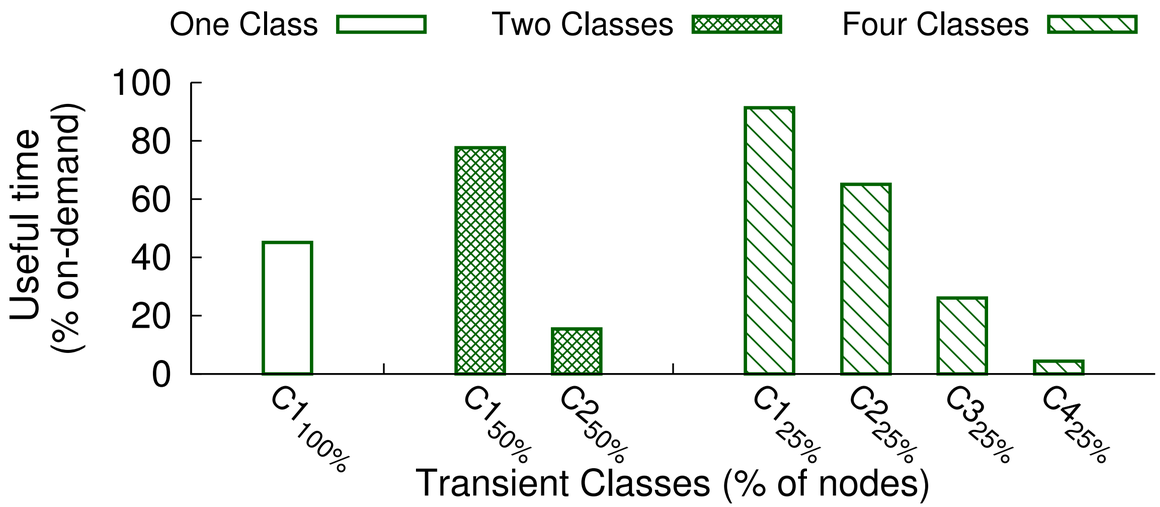
Server characteristics are modeled from a production cluster at Google

Evaluation: Transient Server Characteristics



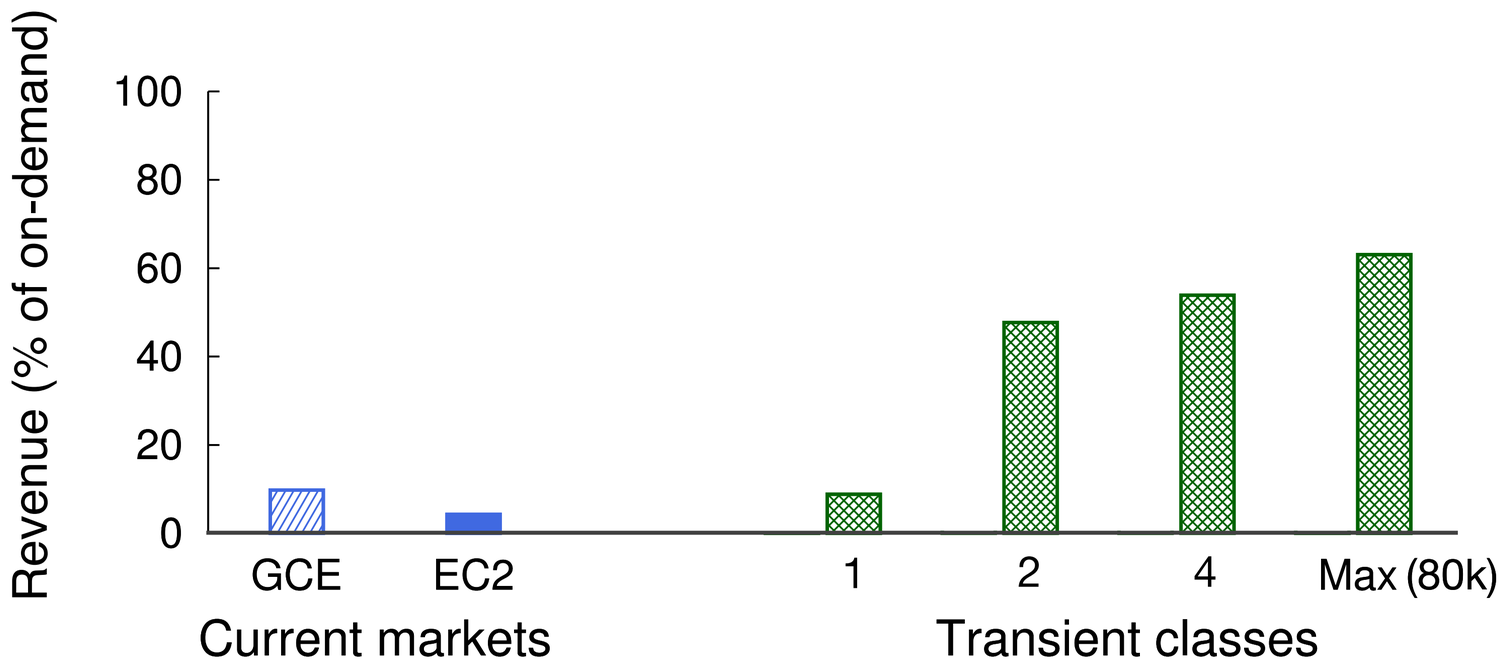
Transient Servers exhibit
a ***wide range*** of characteristics

Transient Classes create
differentiated server pools

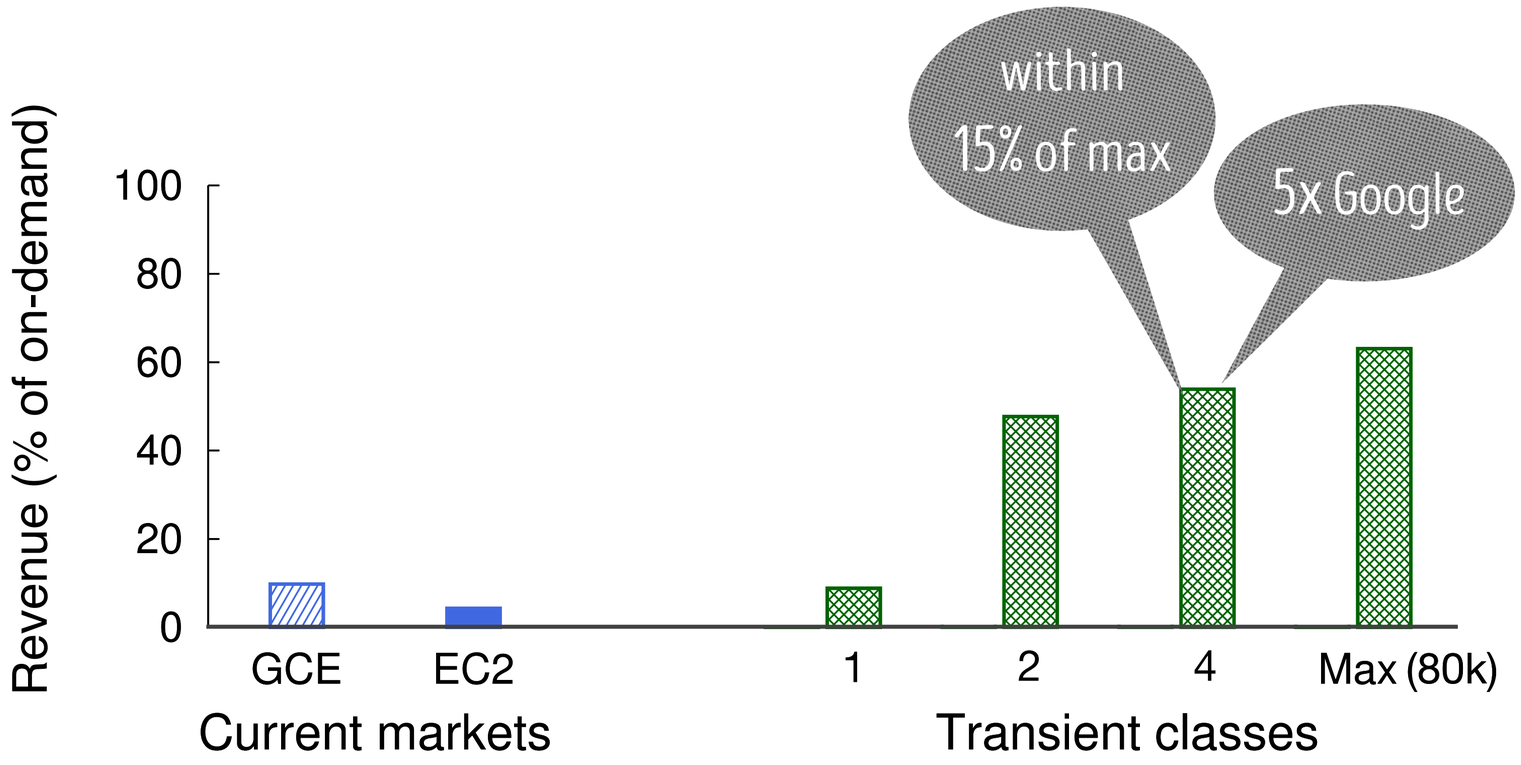


Server characteristics are modeled from a production cluster at Google

Revenue from Transient Server Offering



Revenue from Transient Server Offering



Transient servers with market-based pricing exhibit **price variations**, **inversions** and **arbitrages**. How to reduce the price risk for applications?

Transient servers with market-based pricing exhibit **price variations**, **inversions** and **arbitrages**. How to reduce the price risk for applications?

Mitigating
Revocation Risk
with Insurance

SpotOn
[SoCC 2015]

Minimizing
Valuation Risk
via Asset Pricing

Transient Guarantee
[SC 2016]

Reducing
Price Risk thru'
Active Trading

HotSpot
[SoCC 2017]

Eliminating
Uncertainty Risk
by Index Tracking

Cloud Index
[In submission]

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**]

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**]

… but
insurance-like approaches ignore

Price Risk

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

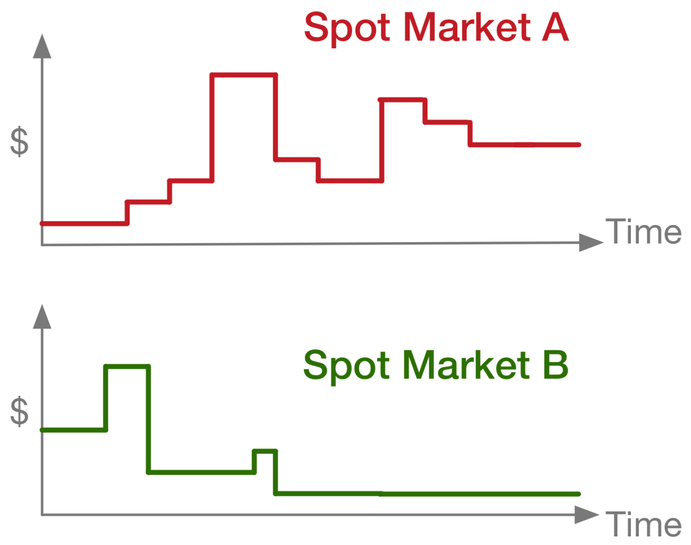
Prior work treats ***revocations as failures***,
and employs fault-tolerance to reduce its impact

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

… but
insurance-like approaches ignore

Price Risk

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



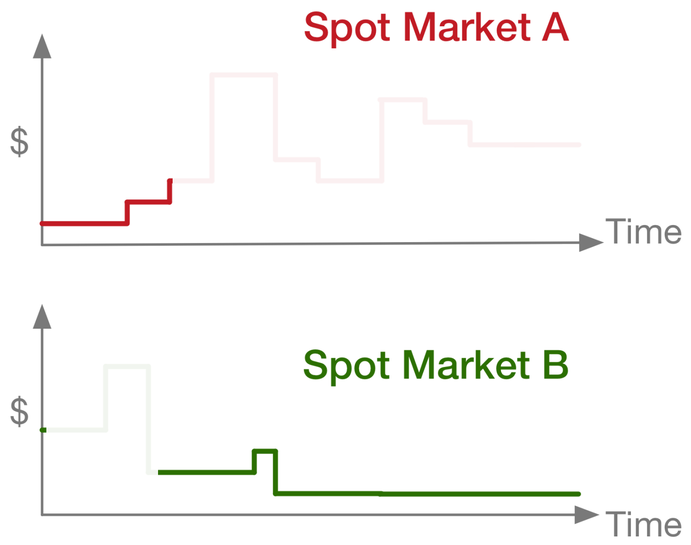
Prior work treats *revocations as failures*,
and employs fault-tolerance to reduce its impact

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

… but
insurance-like approaches ignore

Price Risk

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

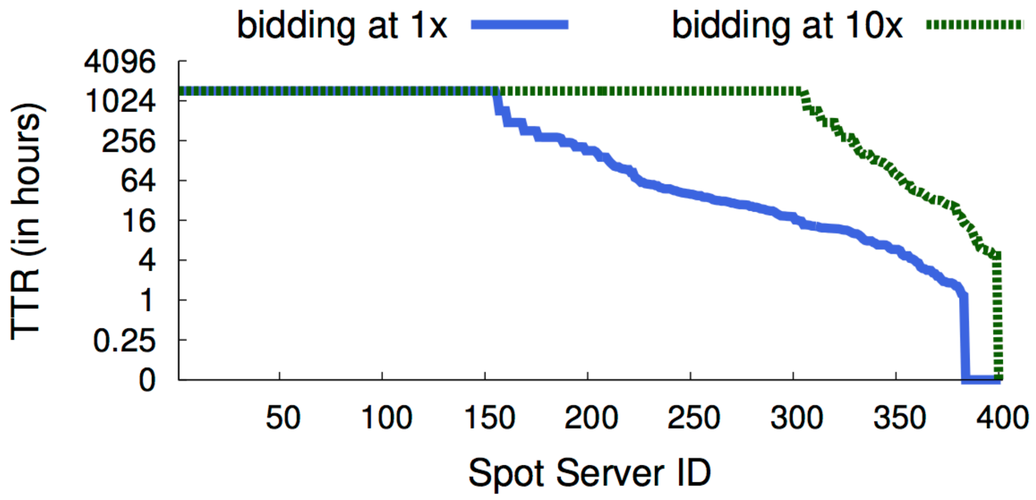


Price Risk >> Revocation Risk

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

Price Risk >> Revocation Risk

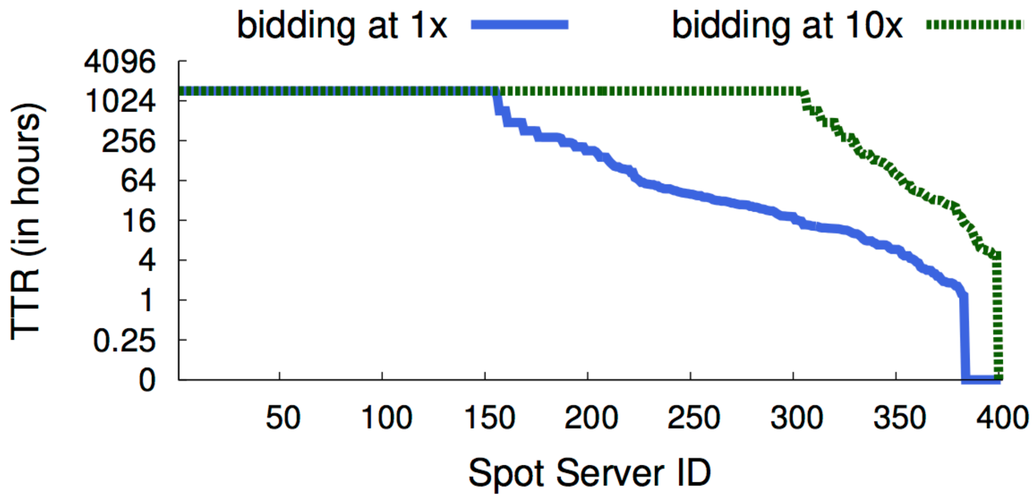
Data from all 402 spot servers in US-East-1 over 3/1/2017 to 5/1/2017



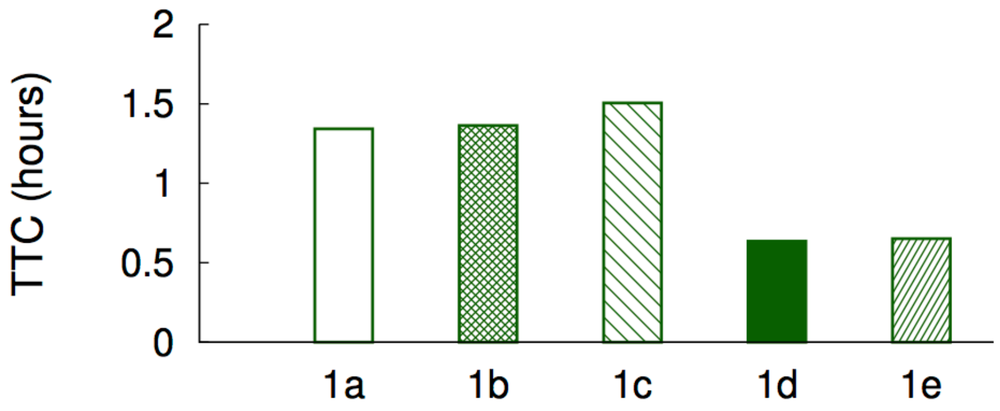
Mean Time-to-Revocation (TTR) when bidding 1x is **~25 days** and 10x is **~47 days**

Price Risk >> Revocation Risk

Data from all 402 spot servers 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

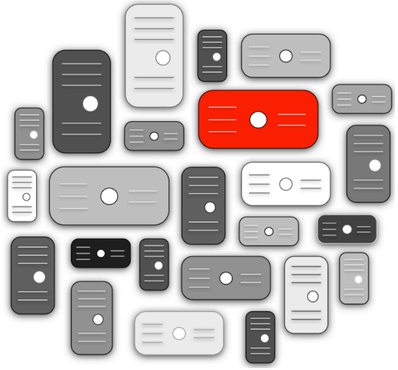
*Does mitigating the price risk affect **revocation risk**?*

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

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

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

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



HotSpot: Automated Server Hopping

Automated Server Hopping

“ Change, before you have to ” – Jack Welch

Quote from Jack Welch, former CEO of General Electric

Automated Server Hopping

“ Change, before you have to ” – Jack Welch

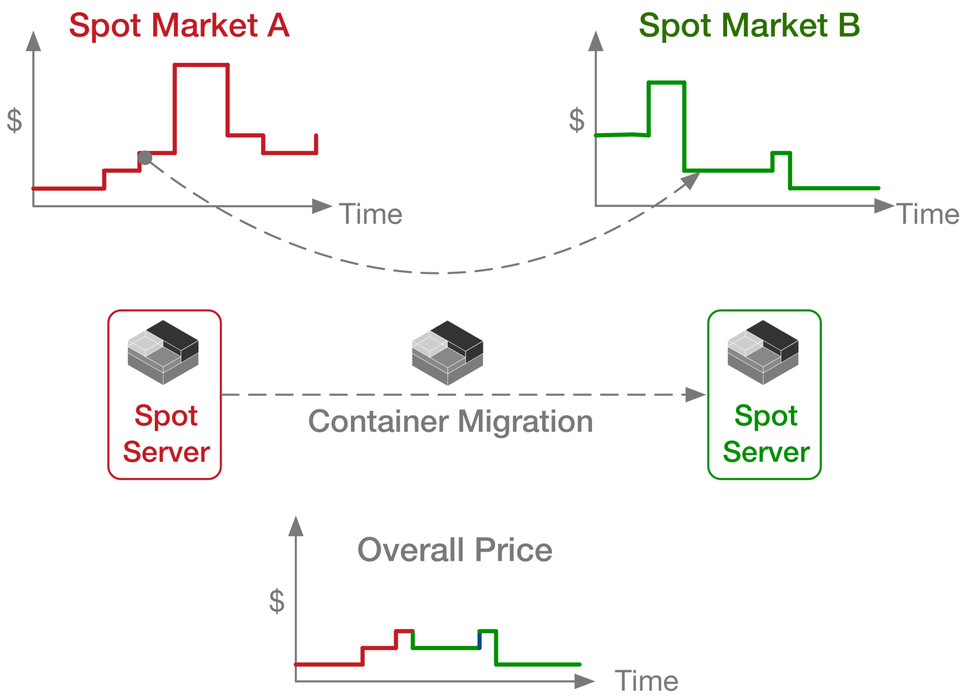
A resource container that automatically hops spot servers as market conditions change

Quote from Jack Welch, former CEO of General Electric

Automated Server Hopping

“ Change, before you have to ” – Jack Welch

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

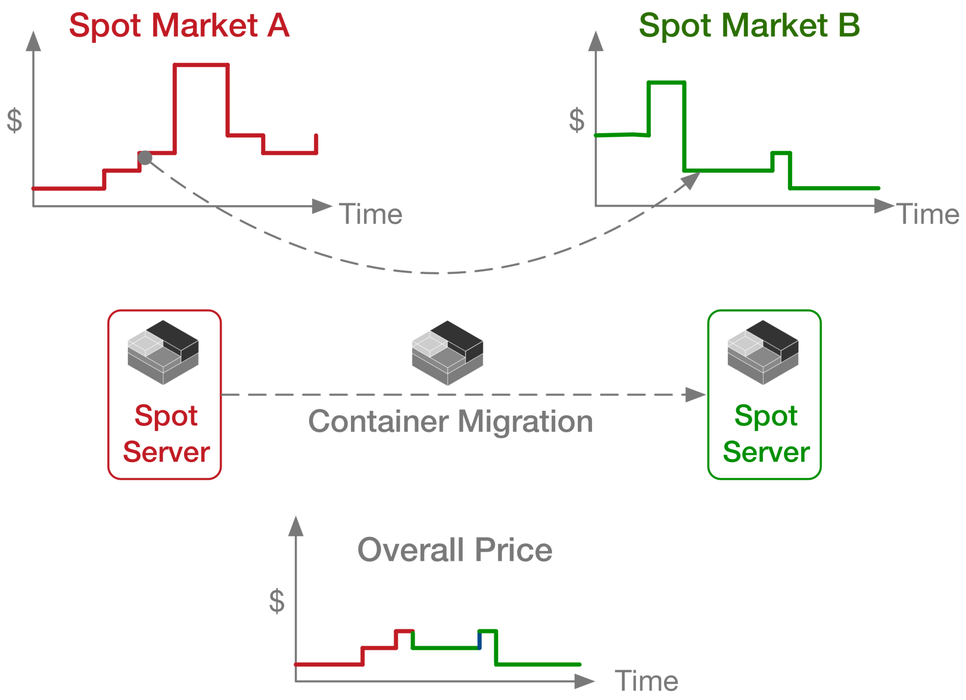


Quote from Jack Welch, former CEO of General Electric

Automated Server Hopping

“ Change, before you have to ” – Jack Welch

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



Benefits of Server Hopping

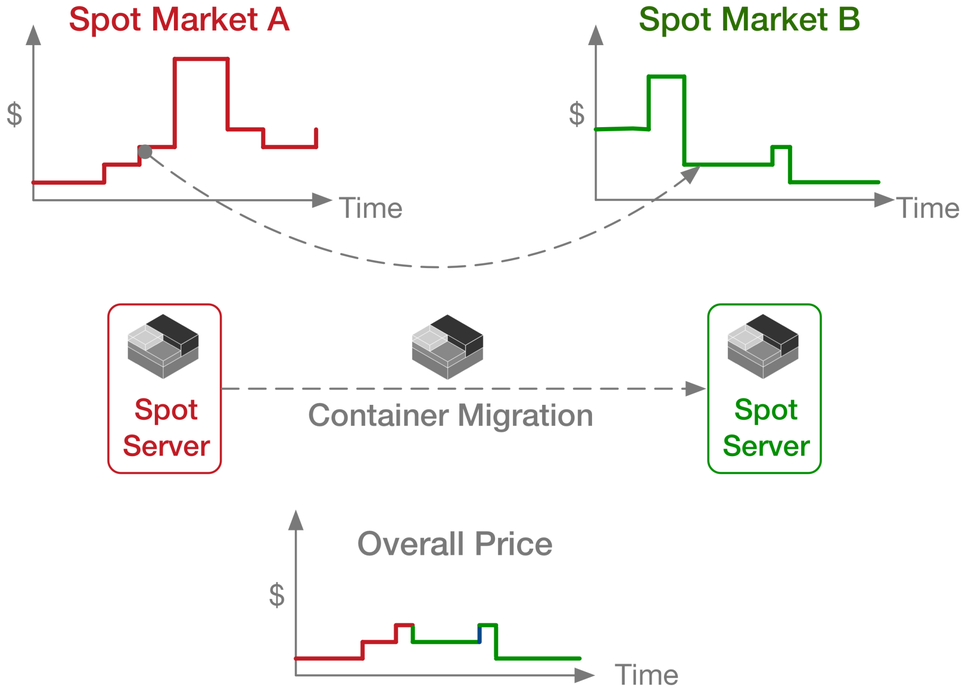
US-East-1 markets (3/1/2017 - 5/1/2017)

Quote from Jack Welch, former CEO of General Electric

Automated Server Hopping

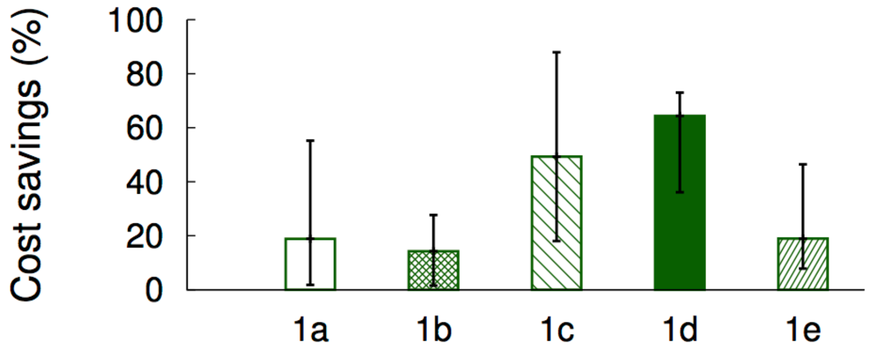
“ Change, before you have to ” – Jack Welch

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



Benefits of Server Hopping

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

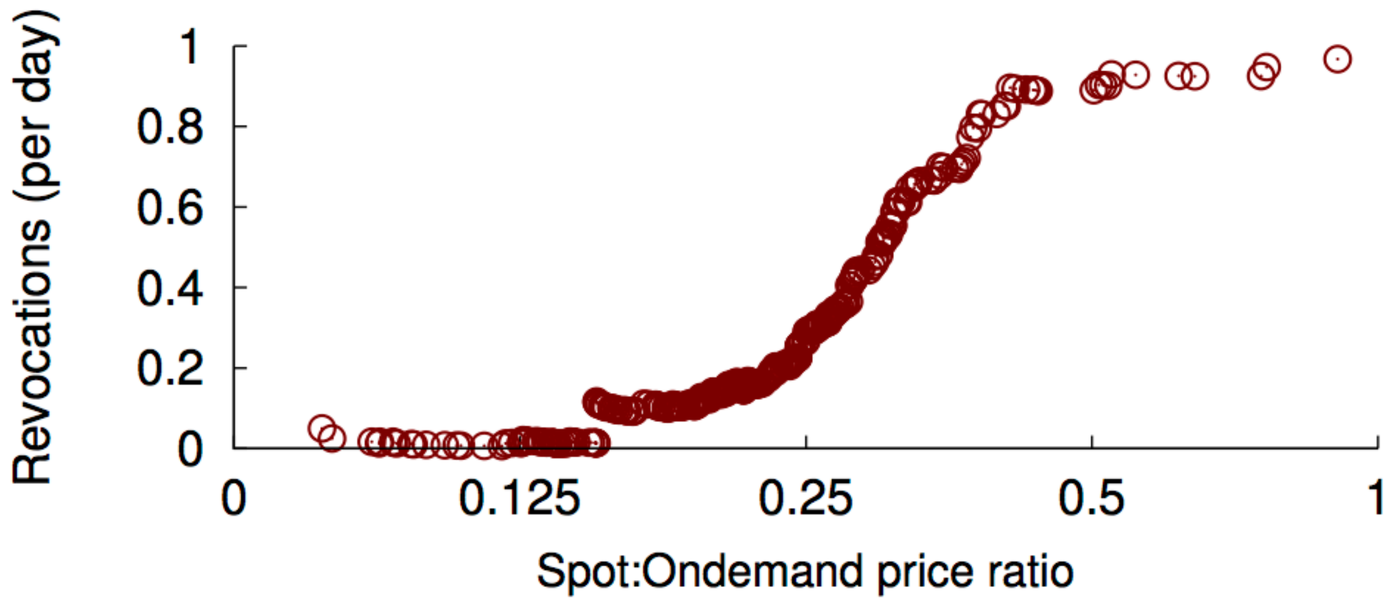
Effect on Revocation Risk

Effect on Revocation Risk

Highly discounted servers tend to have lower revocation risk

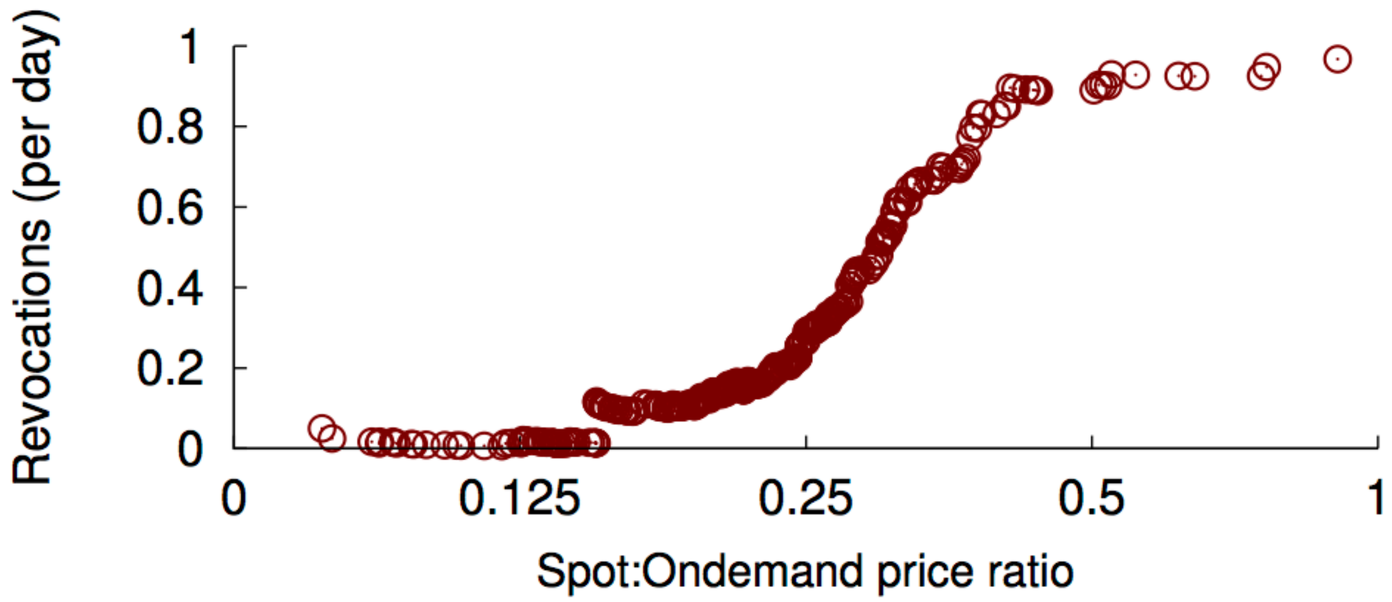
Effect on Revocation Risk

Highly discounted servers tend to have lower revocation risk



Effect on Revocation Risk

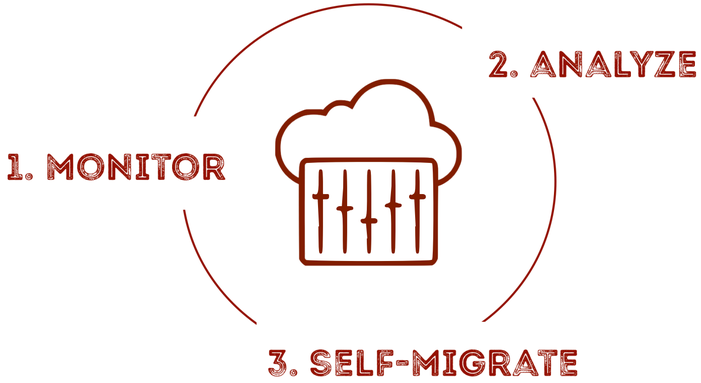
Highly discounted servers tend to have lower revocation risk



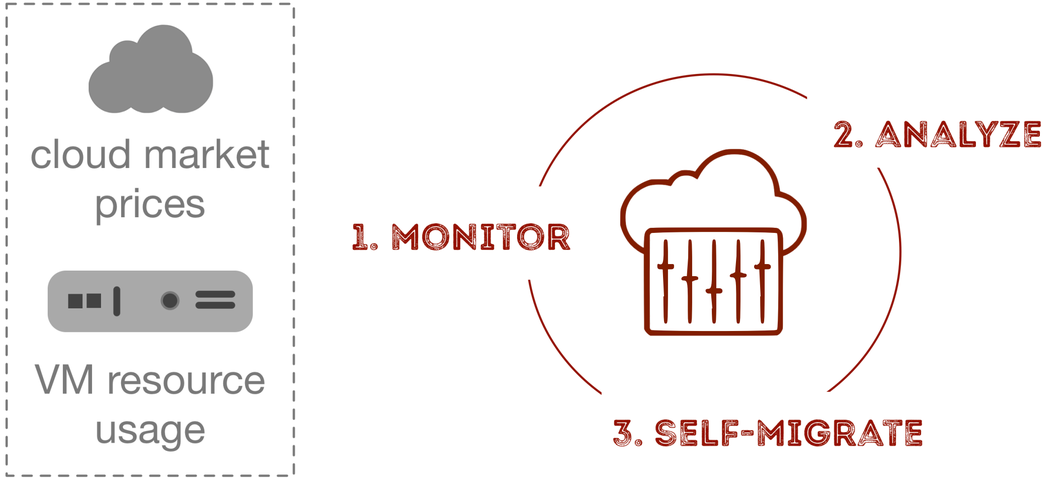
Server hopping lowers revocations along with improving cost-efficiency

HotSpot: Design and Implementation

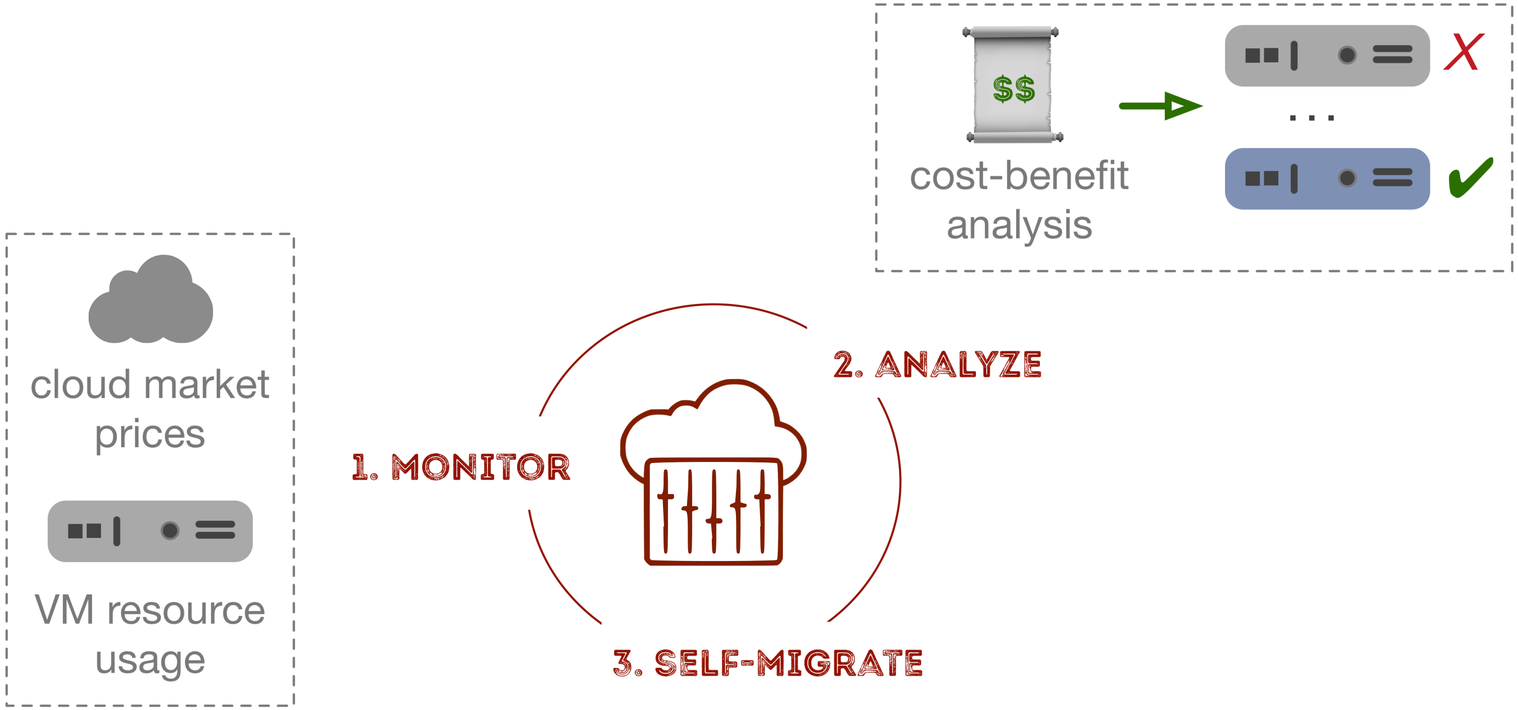
HotSpot: Design and Implementation



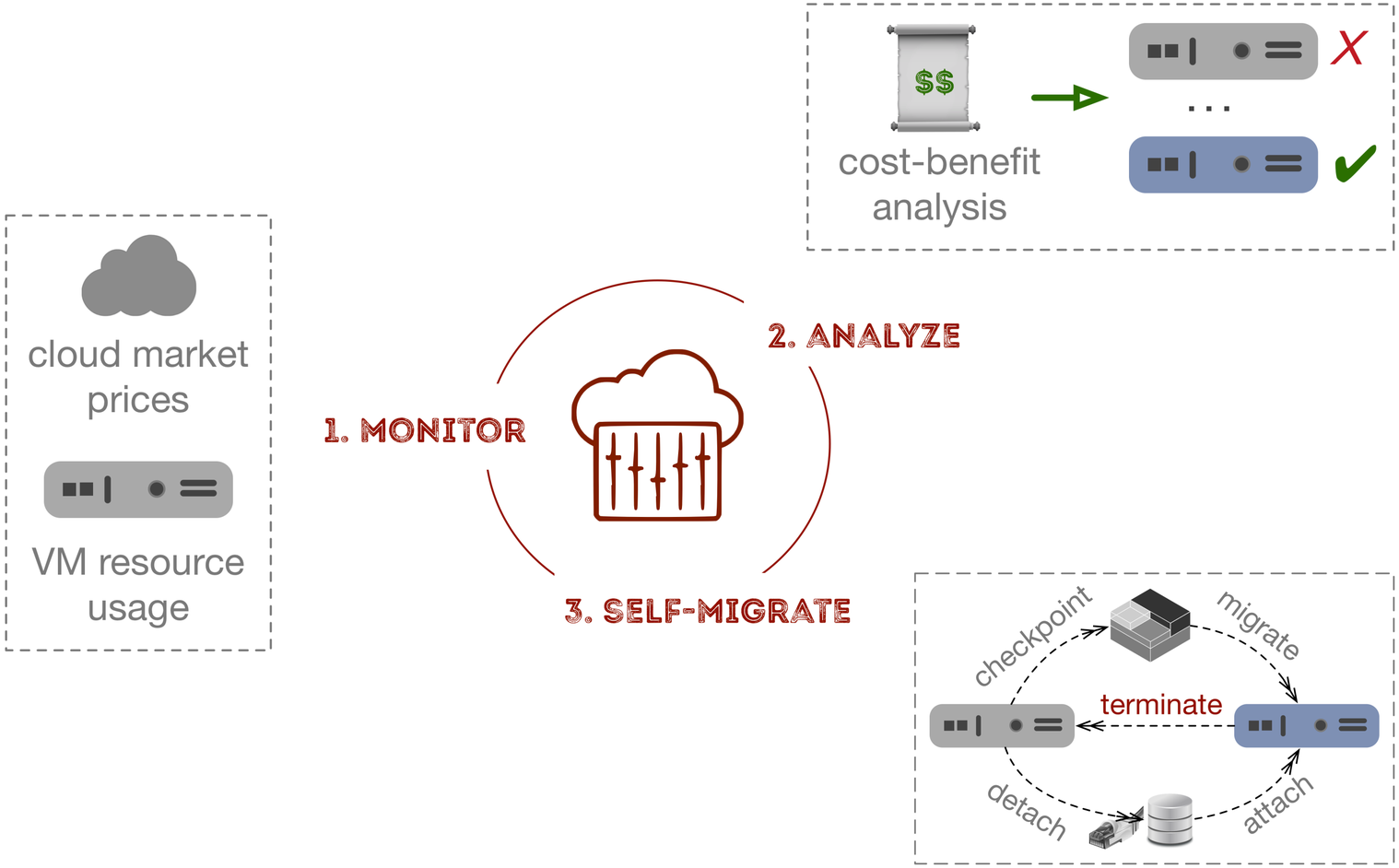
HotSpot: Design and Implementation



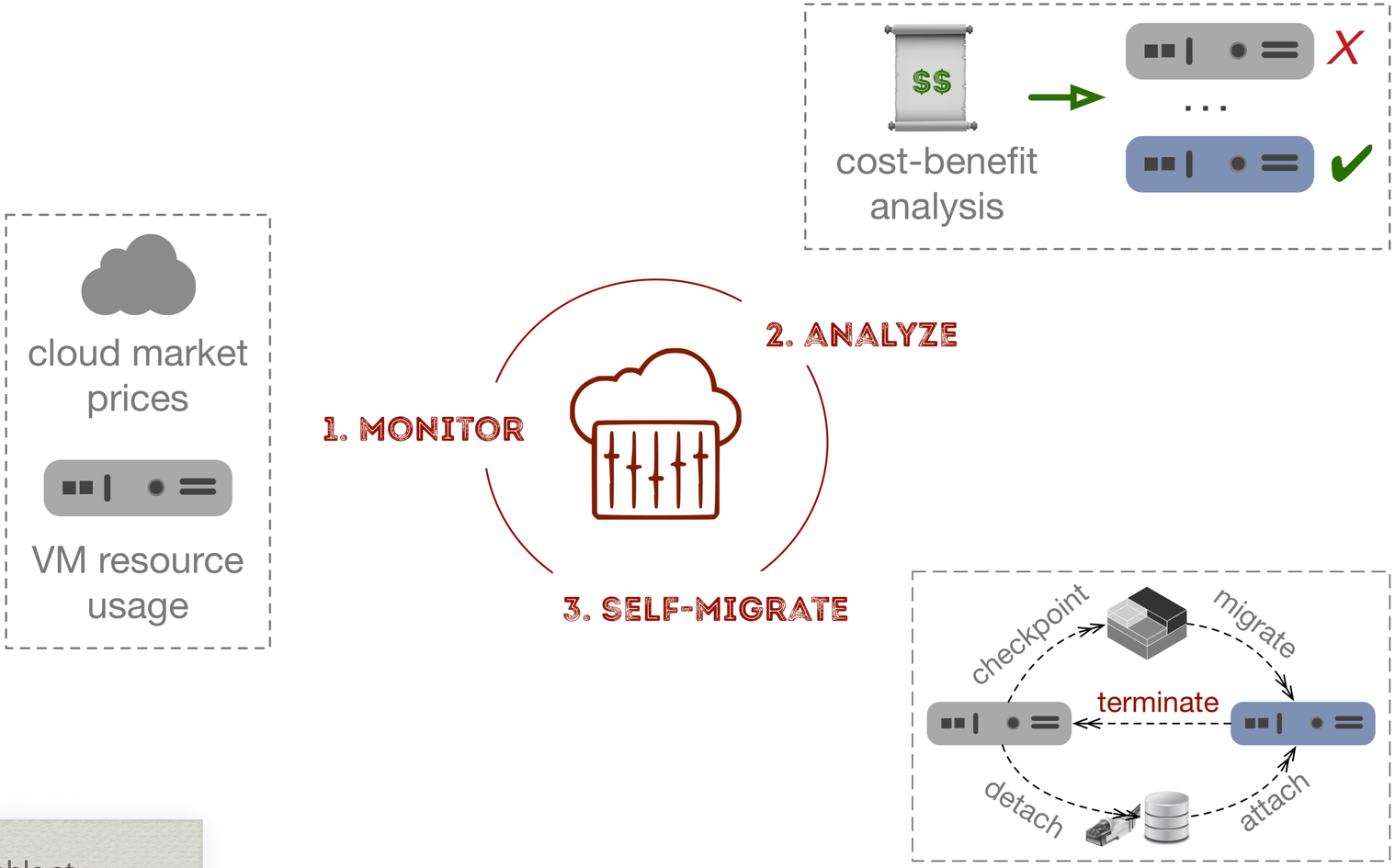
HotSpot: Design and Implementation



HotSpot: Design and Implementation



HotSpot: Design and Implementation



Fully functional prototype available at:

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

Design of Server Hopping Logic

Design of Server Hopping Logic

Hopping Policy Invariant

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

Design of Server Hopping Logic

Hopping Policy Invariant

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

Cost-Benefit Analysis

$$Cost = (P_{current} + P_{new}) * T_{migration}$$

- Double-paying for servers
- f(application footprint)

Design of Server Hopping Logic

Hopping Policy Invariant

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

Cost-Benefit Analysis

$$\text{Cost} = (P_{current} + P_{new}) * T_{migration}$$

- Double-paying for servers
- f(application footprint)

$$\text{Benefit} = (P_{current} - P_{new}) * E[T_{new}]$$

- Gain in cost-efficiency
- f(market characteristics)

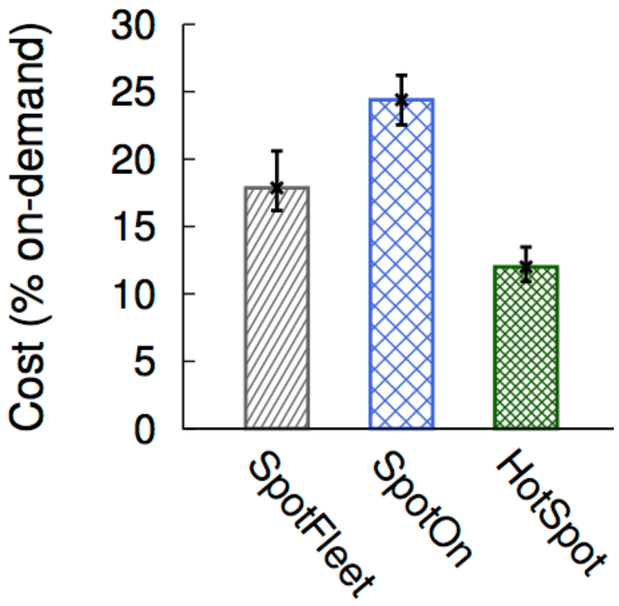
Evaluation

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

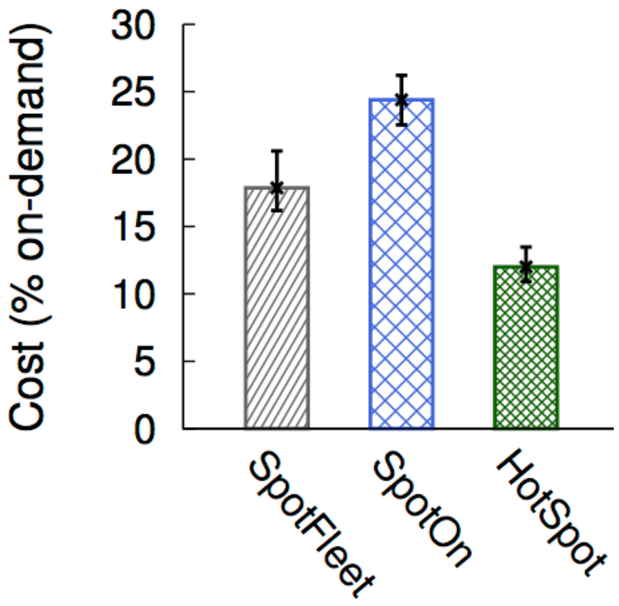
Spot server with **no protection** (SpotFleet)	vs.	Spot server with **fault-tolerance** (SpotOn)	vs.	Spot server with **server hopping** (HotSpot)

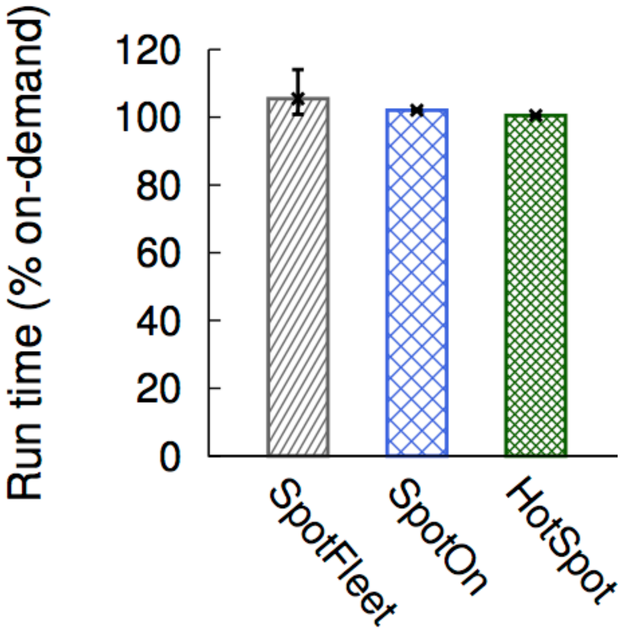
Google Cluster Jobs on EC2 Spot Markets

Google Cluster Jobs on EC2 Spot Markets

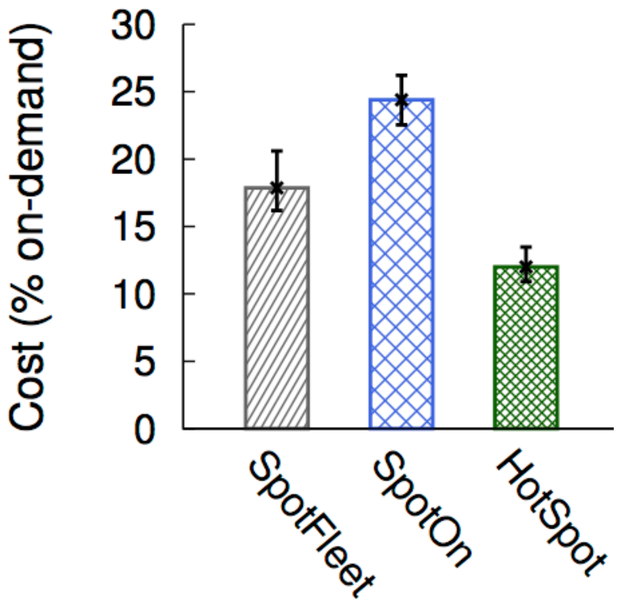


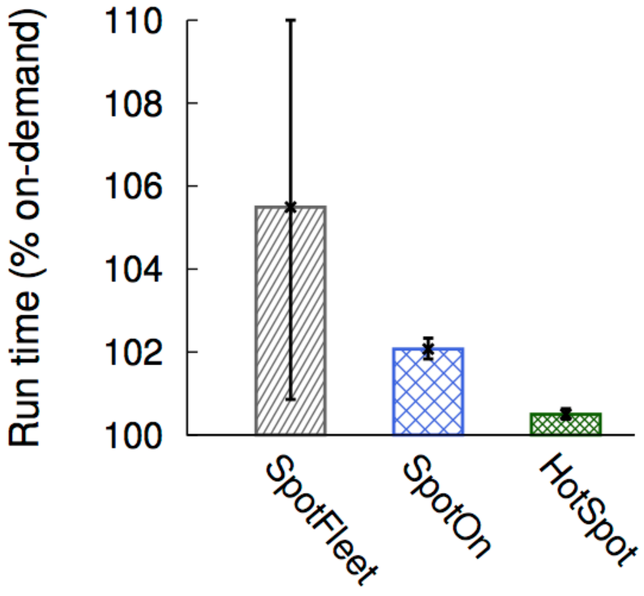
Google Cluster Jobs on EC2 Spot Markets



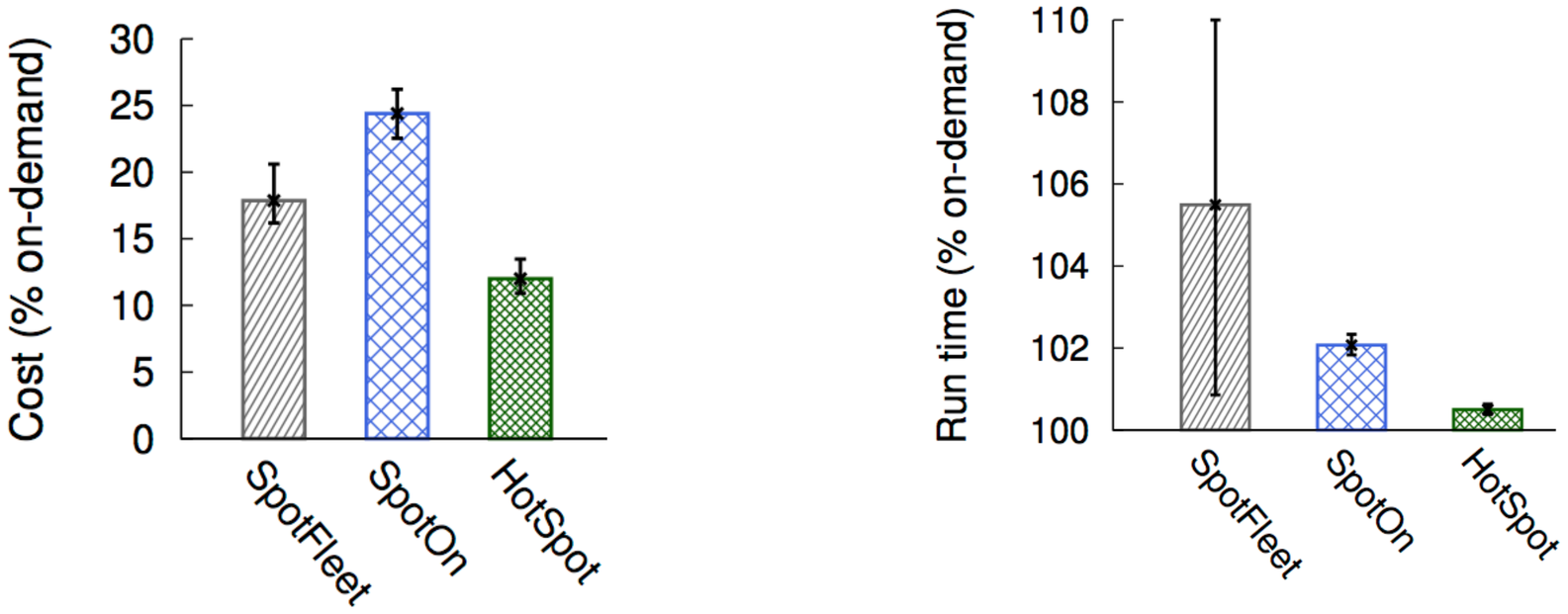


Google Cluster Jobs on EC2 Spot Markets





Google Cluster Jobs 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

Applications using variable-priced servers suffer from cost **uncertainty**.

This makes it hard for customers to **plan** their IT expenses.

Applications using variable-priced servers suffer from cost **uncertainty**.
This makes it hard for customers to **plan** their IT expenses.

Mitigating Revocation Risk with Insurance	Minimizing Valuation Risk via Asset Pricing	Reducing Price Risk thru' Active Trading	Eliminating Uncertainty Risk by Index Tracking
SpotOn [SoCC 2015]	Transient Guarantee [SC 2016]	HotSpot [SoCC 2017]	Cloud Index [In submission]

Predicting Spot Prices is Important

Predicting Spot Prices is ~~Important~~ Accurately Difficult

68 • 2 • 2-5 • 14 • 2 = 7600+

Hardware config | OS types | Zones (datacenters) | Regions (country, state) | Time commitments | worldwide markets

Predicting Spot Prices is ~~Important~~ Accurately Difficult

68 • 2 • 2-5 • 14 • 2 = 7600+

Hardware config • OS types • Zones (datacenters) • Regions (country, state) • Time commitments = worldwide markets

One size fits all model is unlikely

No visibility into market internals

Limited correlation with external variables

Market Properties from First Principles

Market Properties from First Principles

PROPERTY-1

Spot markets originating from the same physical machine family are not free from mutual interference

Market Properties from First Principles

PROPERTY-1

Spot markets originating from the same physical machine family are not free from mutual interference

Not all spot markets can be **individually** modeled and predicted

Market Properties from First Principles

PROPERTY-1

Spot markets originating from the same physical machine family are not free from mutual interference

Not all spot markets can be **individually** modeled and predicted

PROPERTY-2

For public cloud providers, the datacenter's aggregate idle capacity tends to be stable

Market Properties from First Principles

PROPERTY-1

Spot markets originating from the same physical machine family are not free from mutual interference

Not all spot markets can be **individually** modeled and predicted

PROPERTY-2

For public cloud providers, the datacenter's aggregate idle capacity tends to be stable

If idle capacity were priced like commodity, its **clearing price** will be **stable** and **predictable**

Market Properties from First Principles

PROPERTY-1

Spot markets originating from the same physical machine family are not free from mutual interference

Not all spot markets can be **individually** modeled and predicted

PROPERTY-2

For public cloud providers, the datacenter's aggregate idle capacity tends to be stable

If idle capacity were priced like commodity, its **clearing price** will be **stable** and **predictable**

SUMMARY: we should model the behavior of **groups of spot markets** at family- and datacenter-levels

A Market-based Index

Image credit: www.cnbc.com/mad-money/

A Market-based Index

Image credit: www.cnbc.com/mad-money/

A Market-based Index *for CLOUD*

Rather than focusing **exclusively** on predicting **individual servers**, cloud users should make decisions based on **broader market indices**

Image credit: www.cnbc.com/mad-money/

A Market-based Index *for CLOUD*

vs.

Rather than focusing **exclusively** on predicting **individual servers,** cloud users should make decisions based on **broader market indices**

Image credit: www.cnbc.com/mad-money/

A Market-based Index *for CLOUD*

Characterizing an individual server i

Price = P_i , *Memory* = M_i *GB*

Compute = C_i *ECUs*

$$P_i^{norm} = \frac{P_i}{\sqrt{(C_i * M_i)}}$$

A Market-based Index *for CLOUD*

Characterizing an individual server i

Price = P_i , *Memory* = M_i *GB*

Compute = C_i *ECUs*

$$P_i^{norm} = \frac{P_i}{\sqrt{(C_i * M_i)}}$$

Characterizing a group of servers

Average of normalized prices

$$\text{Index-level} = \frac{\sum_{i=1}^{N} P_i^{norm}}{N}$$

A Market-based Index *for CLOUD*

Characterizing an individual server i

Price = P_i , *Memory* = M_i *GB*

Compute = C_i *ECUs*

$$P_i^{norm} = \frac{P_i}{\sqrt{(C_i * M_i)}}$$

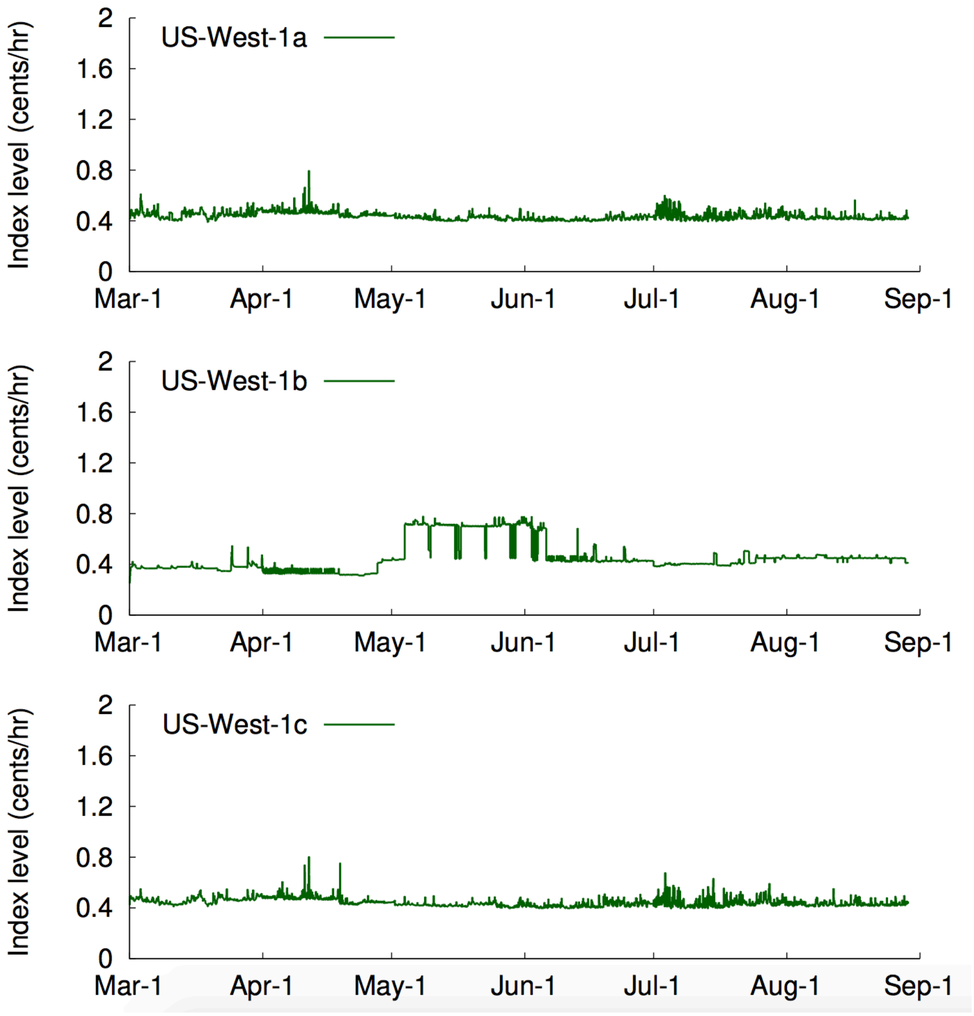
Characterizing a group of servers

Average of normalized prices

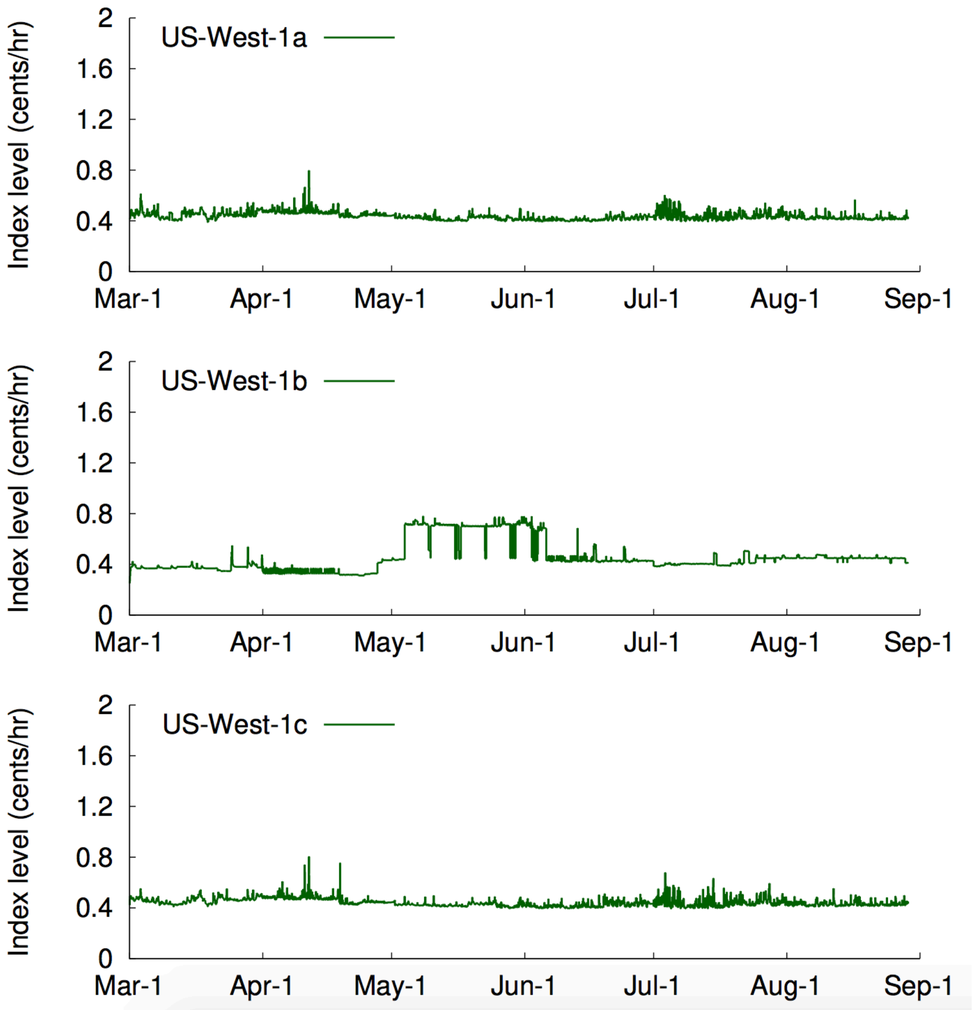
$$\text{Index-level} = \frac{\sum_{i=1}^{N} P_i^{norm}}{N}$$

Cloud index value represents the **average price per unit of compute time** for the selected group of servers

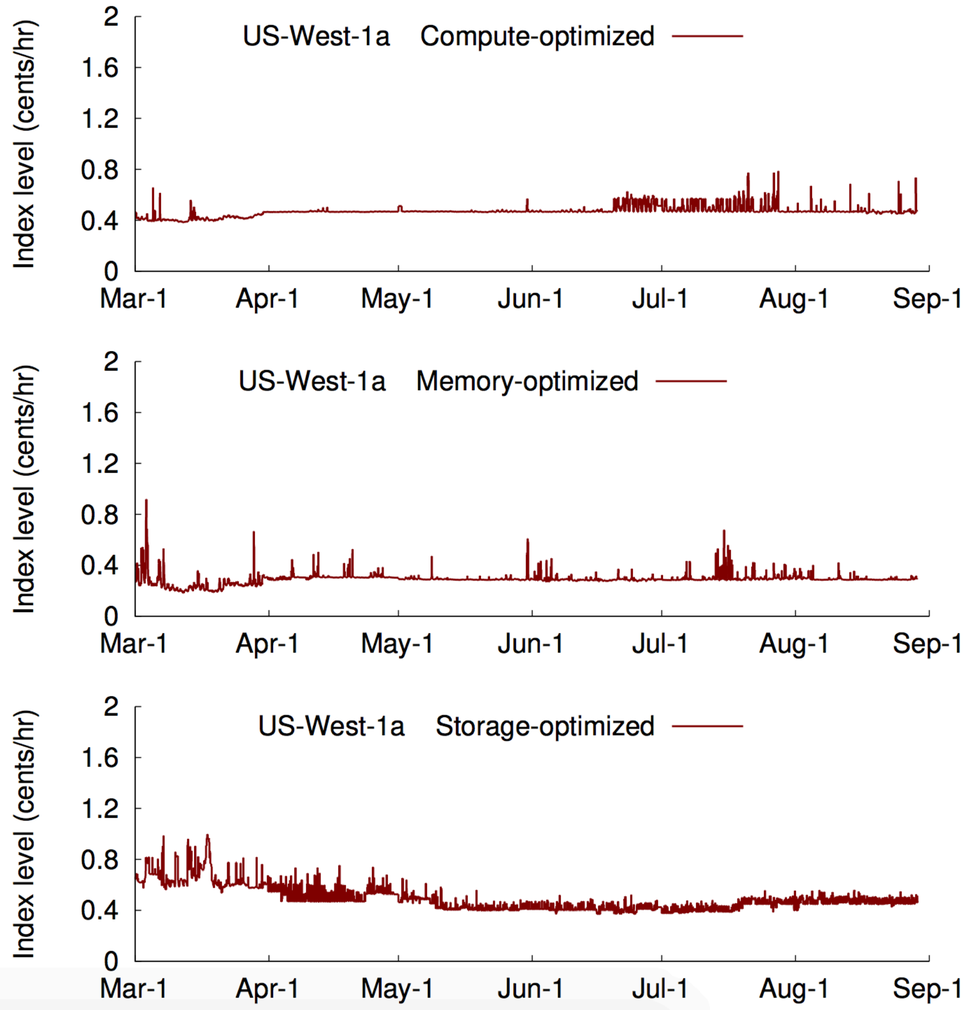
Datacenter Level (US-West-1a/b/c)



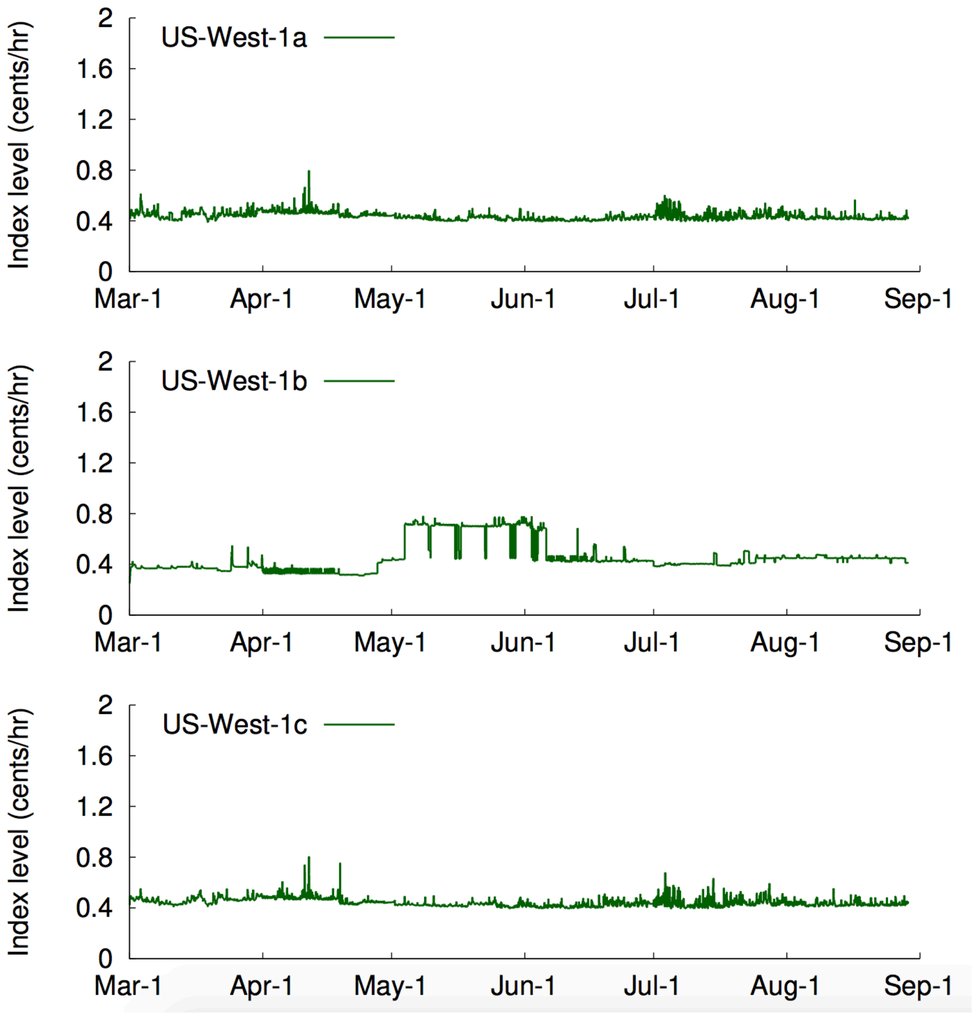
Datacenter Level (US-West-1a/b/c)



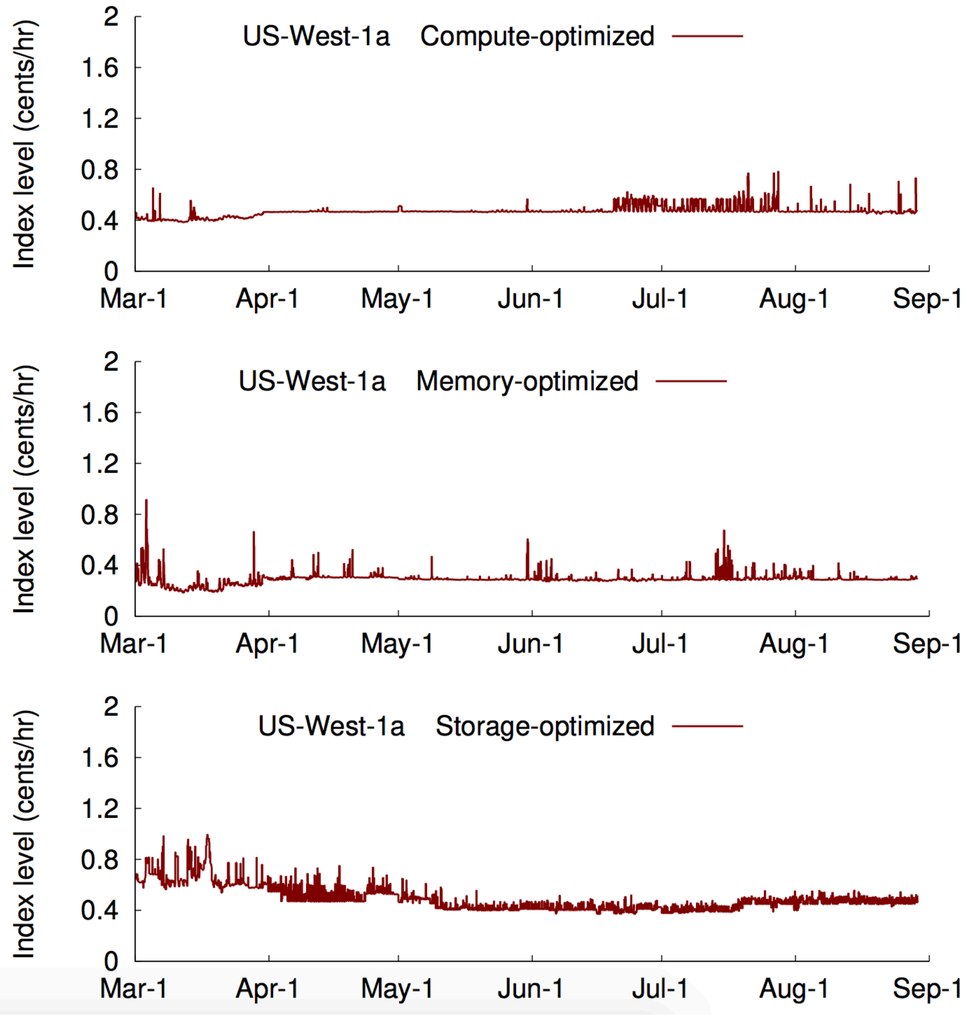
Server Family Level (US-West-1a)



Datacenter Level (US-West-1a/b/c)



Server Family Level (US-West-1a)



Price prediction is more accurate and stable at **datacenter-** and **server family level** than **individual** level

Cost-Predictive Server Hosting

(Host a flexible application on variable-priced servers such that it incurs predictable cost-efficiency)

Cost-Predictive Server Hosting (Host a flexible application on variable-priced servers such that it incurs predictable cost-efficiency)

We adapt the risk management technique of **Index Tracking** to server hosting

Goal is to achieve **index-level cost-efficiency** despite the market volatility

Cost-Predictive Server Hosting

(Host a flexible application on variable-priced servers such that it incurs predictable cost-efficiency)

We adapt the risk management technique of **Index Tracking** to server hosting

Goal is to achieve **index-level cost-efficiency** despite the market volatility

1

Determine a broad set of candidate markets, and then compute its market index

Cost-Predictive Server Hosting

(Host a flexible application on variable-priced servers such that it incurs predictable cost-efficiency)

We adapt the risk management technique of **Index Tracking** to server hosting

Goal is to achieve **index-level cost-efficiency** despite the market volatility

1. Determine a broad set of candidate markets, and then compute its market index

2. Host the application on a server that meets the index-level cost-efficiency

Cost-Predictive Server Hosting

(Host a flexible application on variable-priced servers such that it incurs predictable cost-efficiency)

We adapt the risk management technique of **Index Tracking** to server hosting

Goal is to achieve **index-level cost-efficiency** despite the market volatility

1. Determine a broad set of candidate markets, and then compute its market index

2. Host the application on a server that meets the index-level cost-efficiency

3. If application or market conditions violate index invariant, then transparently hop to a better server

Properties of Index-Tracking-by-Server-Hopping

Properties of Index-Tracking-by-Server-Hopping

Scales well with increased adaptation

If everyone targets index-level efficiency ==> the market becomes less volatile ==> reduces the number of hopping needed ==> stability and cost-efficiency increases for all

Properties of Index-Tracking-by-Server-Hopping

Scales well with increased adaptation

If everyone targets index-level efficiency ==> the market becomes less volatile ==> reduces the number of hopping needed ==> stability and cost-efficiency increases for all

Better than fully-predictive and purely-reactive approaches

Index-based predictions are stable and more resilient to market changes

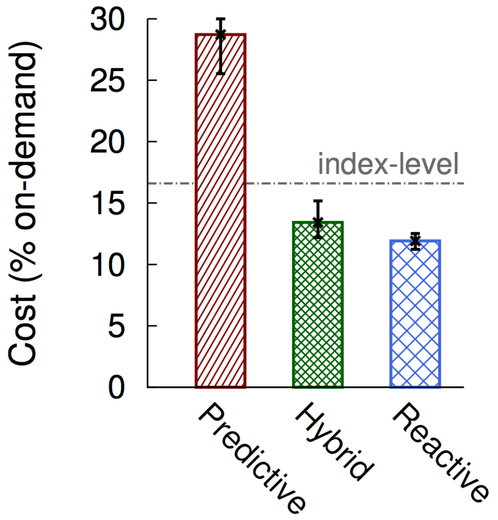
Hybrid technique naturally rate-limits the frequent hopping of reactive approaches

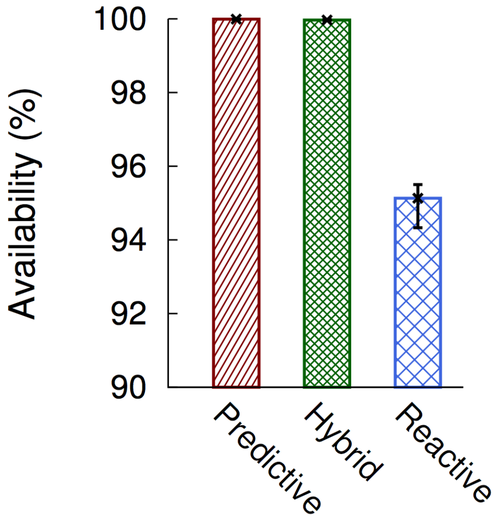
Evaluation

Compare **cost-predictability** and **cost-availability tradeoff** of running a flexible batch application on

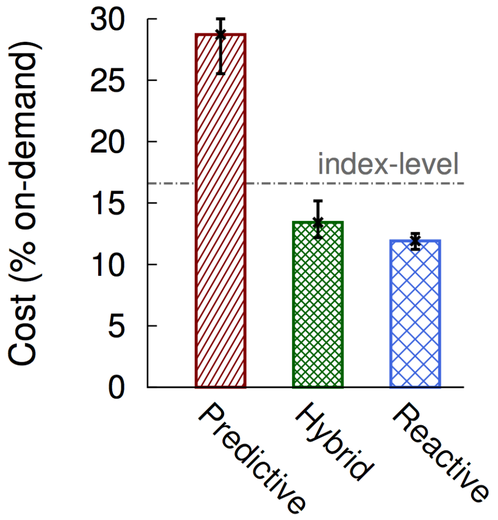
Spot server with **predictive technique** (SpotFleet)	vs.	Spot server with **reactive technique** (HotSpot)	vs.	Spot server with **hybrid technique** (Index Tracking)

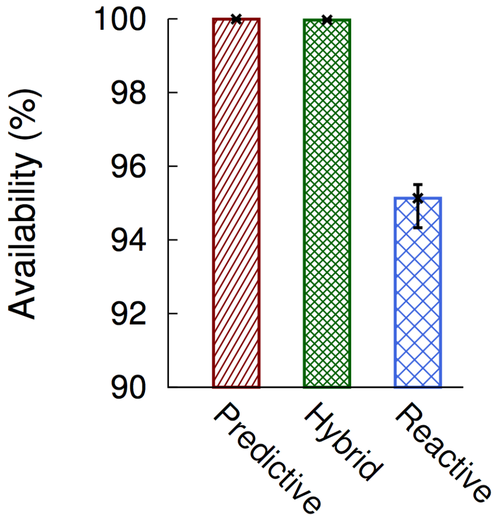
Long-running Application



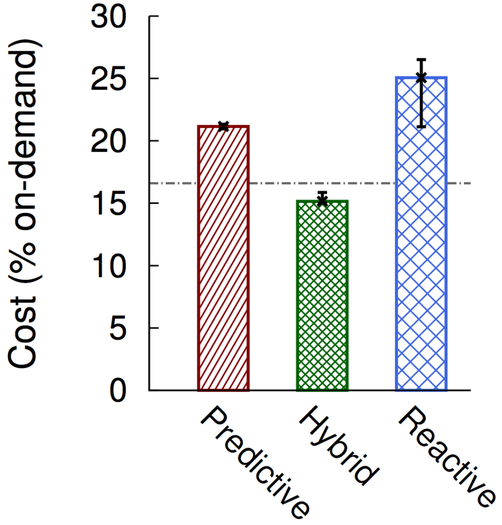


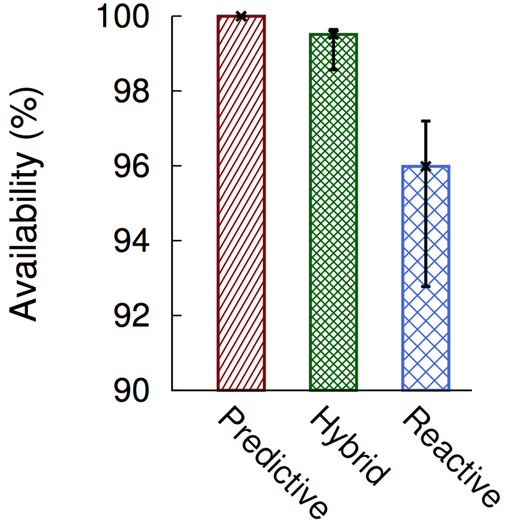
Long-running Application



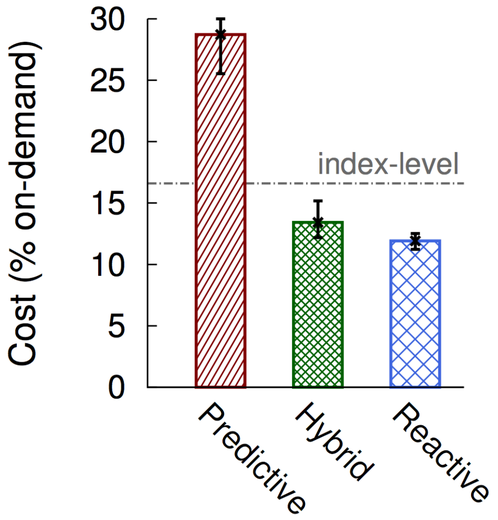


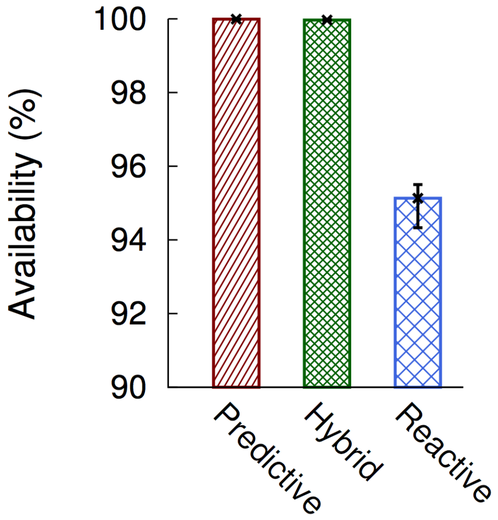
Parallel Applications



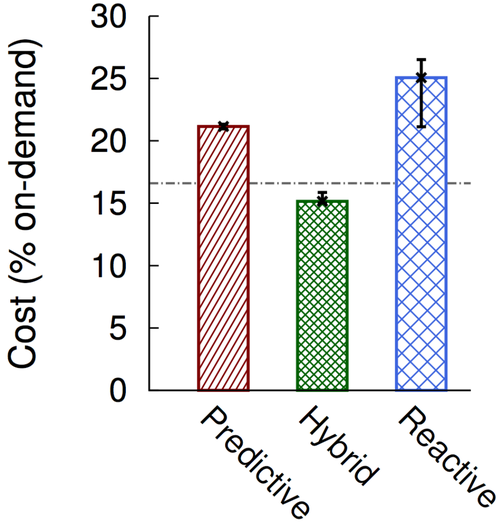


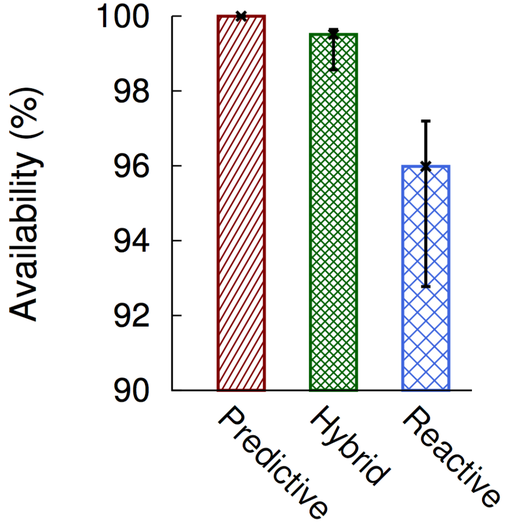
Long-running Application





Parallel Applications





The hybrid approach of Index-Tracking not only meets the **predicted cost-efficiency** but also achieves the **best cost-availability tradeoff** compared to other approaches.

Thesis Contributions

Risks	Problems	Proposed Solution	Result
Revocation	Revocations pose a new fault-model that undermines the benefit of using transient servers	Cost-efficient insurance via *SpotOn*	91% cost-savings over on-demand
Price	Applications may pay higher price for less efficient servers in variable-priced cloud markets	Active server trading via *HotSpot*	50% cost reduction over insurance schemes
Uncertainty	Applications deployed on variable-priced cloud markets suffer from cost uncertainty	Server hosting that tracks *Cloud Indices*	Predictable costs that match the index-level
Valuation	Transient servers are difficult to price (for providers) and to value (for consumers)	Asset pricing via *Transient Guarantees*	5x increase in the overall value

Thesis Contributions

Risks	Problems	Proposed Solution	Result
Revocation	Revocations pose a new fault-model that undermines the benefit of using transient servers	Cost-efficient insurance via *SpotOn*	91% cost-savings over on-demand
Price	Applications may pay higher price for less efficient servers in variable-priced cloud markets	Active server trading via *HotSpot*	50% cost reduction over insurance schemes
Uncertainty	Applications deployed on variable-priced cloud markets suffer from cost uncertainty	Server hosting that tracks *Cloud Indices*	Predictable costs that match the index-level
Valuation	Transient servers are difficult to price (for providers) and to value (for consumers)	Asset pricing via *Transient Guarantees*	5x increase in the overall value

- Evaluating **Market Sustainability Risk** with Asset Pricing [HotCloud 2016]

Thesis Contributions

Risks	Problems	Proposed Solution	Result
Revocation	Revocations pose a new fault-model that undermines the benefit of using transient servers	Cost-efficient insurance via *SpotOn*	91% cost-savings over on-demand
Price	Applications may pay higher price for less efficient servers in variable-priced cloud markets	Active server trading via *HotSpot*	50% cost reduction over insurance schemes
Uncertainty	Applications deployed on variable-priced cloud markets suffer from cost uncertainty	Server hosting that tracks *Cloud Indices*	Predictable costs that match the index-level
Valuation	Transient servers are difficult to price (for providers) and to value (for consumers)	Asset pricing via *Transient Guarantees*	5x increase in the overall value

- Evaluating **Market Sustainability Risk** with Asset Pricing [HotCloud 2016]
- Reducing **Spatial Price Risk** with Cloud Index [HotCloud 2017]

Thesis Contributions

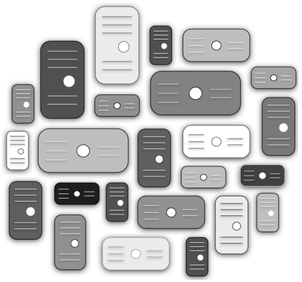
Risks	Problems	Proposed Solution	Result
Revocation	Revocations pose a new fault-model that undermines the benefit of using transient servers	Cost-efficient insurance via *SpotOn*	91% cost-savings over on-demand
Price	Applications may pay higher price for less efficient servers in variable-priced cloud markets	Active server trading via *HotSpot*	50% cost reduction over insurance schemes
Uncertainty	Applications deployed on variable-priced cloud markets suffer from cost uncertainty	Server hosting that tracks *Cloud Indices*	Predictable costs that match the index-level
Valuation	Transient servers are difficult to price (for providers) and to value (for consumers)	Asset pricing via *Transient Guarantees*	5x increase in the overall value

- Evaluating **Market Sustainability Risk** with Asset Pricing [HotCloud 2016]
- Reducing **Spatial Price Risk** with Cloud Index [HotCloud 2017]
- Managing **Energy Risk** of datacenters and applications [ICPE 2016]

Future Directions: Managing Risks in Emerging Computing Landscape

Future Directions: Managing Risks in Emerging Computing Landscape

Not all compute is
created equal

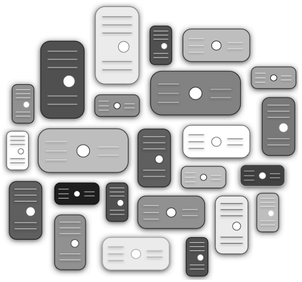


+

On-Demand BURSTABLE
RESERVED **SPOT MARKET**
Preemptible Dedicated

Future Directions: Managing Risks in Emerging Computing Landscape

Not all compute is
created equal



+

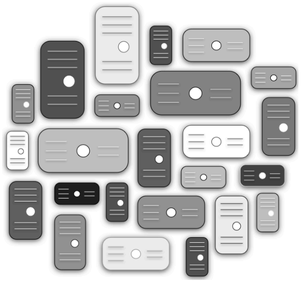
On-Demand BURSTABLE
RESERVED **SPOT MARKET**
Preemptible Dedicated

Not all compute is
consumed equal

Batch | Iterative | Interactive | Real-time

Future Directions: Managing Risks in Emerging Computing Landscape

Not all compute is **created** equal



+

On-Demand BURSTABLE
RESERVED **SPOT MARKET**
Preemptible Dedicated

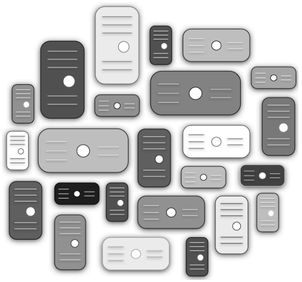
Financialized system software

Not all compute is **consumed** equal

Batch | Iterative | Interactive | Real-time

Future Directions: Managing Risks in Emerging Computing Landscape

Not all compute is **created** equal



+

On-Demand BURSTABLE
RESERVED **SPOT MARKET**
Preemptible Dedicated

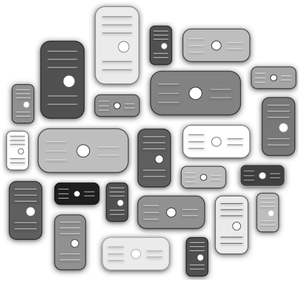
Financialized system software

Not all compute is **consumed** equal

Batch | Iterative | Interactive | Real-time

Future Directions: Managing Risks in Emerging Computing Landscape

Not all compute is **created** equal



+

On-Demand BURSTABLE
RESERVED SPOT MA
Preemptible Ded

NEW contracts

Financialized system software

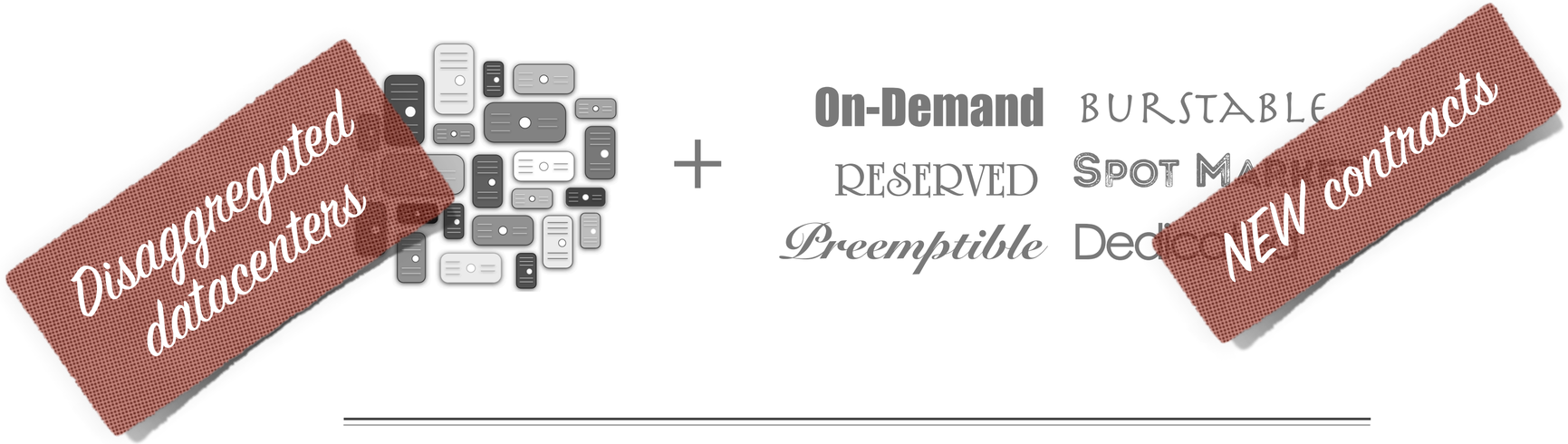
Not all compute is **consumed** equal

Batch | Iterative | Interactive | Rea

NEW categories

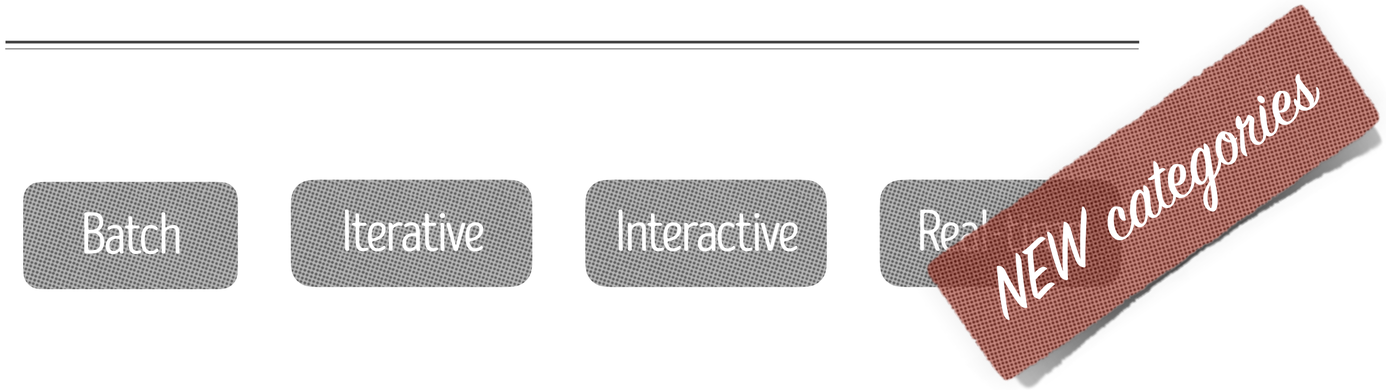
Future Directions: Managing Risks in Emerging Computing Landscape

Not all compute is **created** equal



Financialized system software

Not all compute is **consumed** equal



List of Publications

1. ***Supreeth Subramanya***, Zain Mustafa, David Irwin, and Prashant Shenoy. Energy-agility: A new grid-centric metric for evaluating system performance. In ***LIMITS***, June 2015.
2. ***Supreeth Subramanya***, Tian Guo, Prateek Sharma, David Irwin, and Prashant Shenoy. SpotOn: A Batch Computing Service for the Spot Market. In ***SoCC***, August 2015.
3. ***Supreeth Subramanya***, Zain Mustafa, David Irwin, and Prashant Shenoy. Beyond Energy-Efficiency: Evaluating Green Datacenter Applications for Energy-Agility. In ***ICPE***, March 2016.
4. ***Supreeth Subramanya***, Amr Rizk, and David Irwin. Cloud Spot Markets are Not Sustainable: The Case for Transient Guarantees. In ***HotCloud***, June 2016.
5. ***Supreeth Shastri***, Amr Rizk, and David Irwin. Transient Guarantees: Maximizing the Value of Idle Cloud Capacity. In ***SC***, November 2016.
6. David Irwin, Prateek Sharma, ***Supreeth Shastri***, and Prashant Shenoy. Financialization of Cloud Computing: Opportunities and Challenges. In ***ICCCN***, July 2017.
7. ***Supreeth Shastri*** and David Irwin. Towards Index-based Global Trading in Cloud Spot Markets. In ***HotCloud***, July 2017.
8. ***Supreeth Shastri*** and David Irwin. Automaton: Active Server Trading in the Cloud. In ***SoCC***, September 2017.
9. ***Supreeth Shastri*** and David Irwin. IndeX Marks the Spot: Cost-Predictive Server Hosting in Spot Markets. In ***Submission***.

Backup Slides

Backup Slides

Case Study for Cloud Risks: *Transient Cloud Servers*

Idle Cloud Capacity

Case Study for Cloud Risks: *Transient Cloud Servers*

Idle Cloud Capacity

*“ Shared warehouse scale machines tend to have **10-50%** utilization ”*

[**2013**] *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines.*

Case Study for Cloud Risks: *Transient Cloud Servers*

Idle Cloud Capacity

*“ Shared warehouse scale machines tend to have **10-50%** utilization ”*

[**2013**] *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines.*

has limitations

Case Study for Cloud Risks: *Transient Cloud Servers*

*❝ Shared warehouse scale machines tend to have **10-50%** utilization ❞*

[**2013**] *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines.*

has limitations

amazon web services™ | EC2

Case Study for Cloud Risks: *Transient Cloud Servers*

*❝ Shared warehouse scale machines tend to have **10-50%** utilization ❞*

[**2013**] *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines.*

has limitations

amazon web services™ | **EC2**

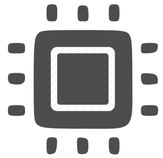
Spot servers in a nutshell

1

The users bid for VMs in a second price auction

2

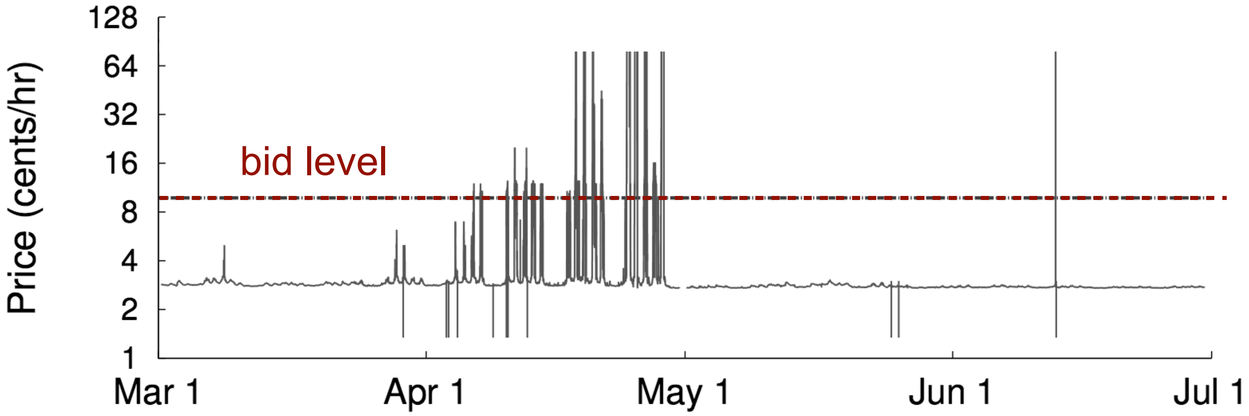
EC2 evaluates supply-demand dynamic to price spot servers



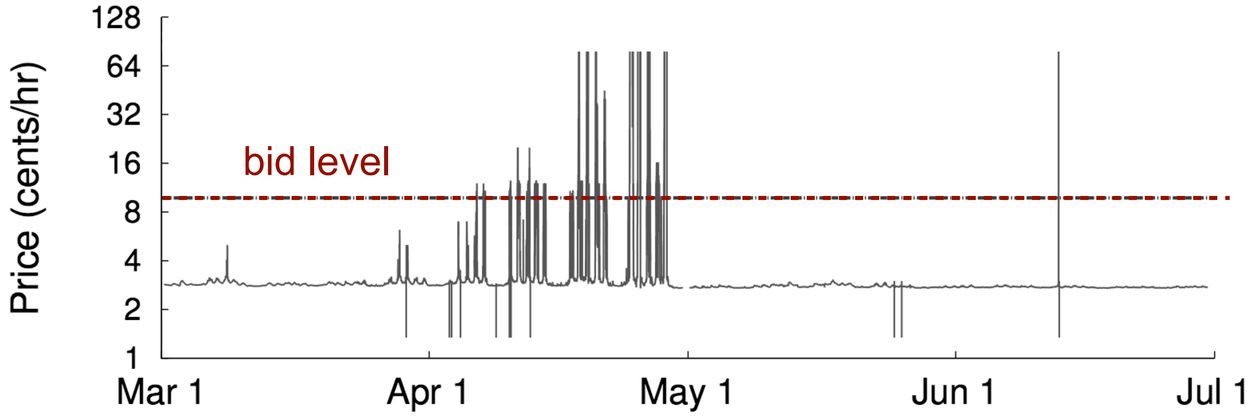
3

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

Applications Hate Spot Servers… but Can't Live Without Them



Applications Hate Spot Servers… but Can't Live Without Them



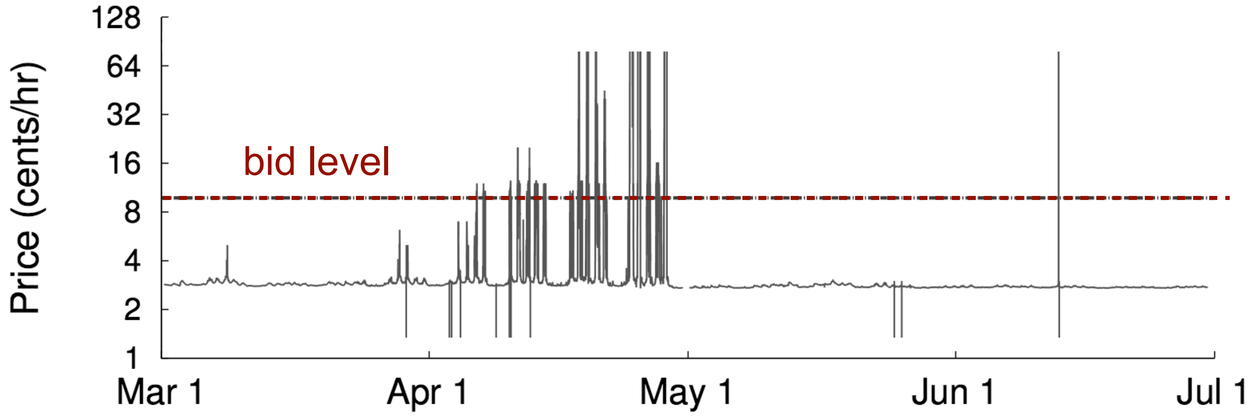
Low average price

50-90% discount over on-demand price

Deliberate revocations

when spot price rises above the bid level

Applications Hate Spot Servers… but Can't Live Without Them



Low average price

50-90% discount over on-demand price

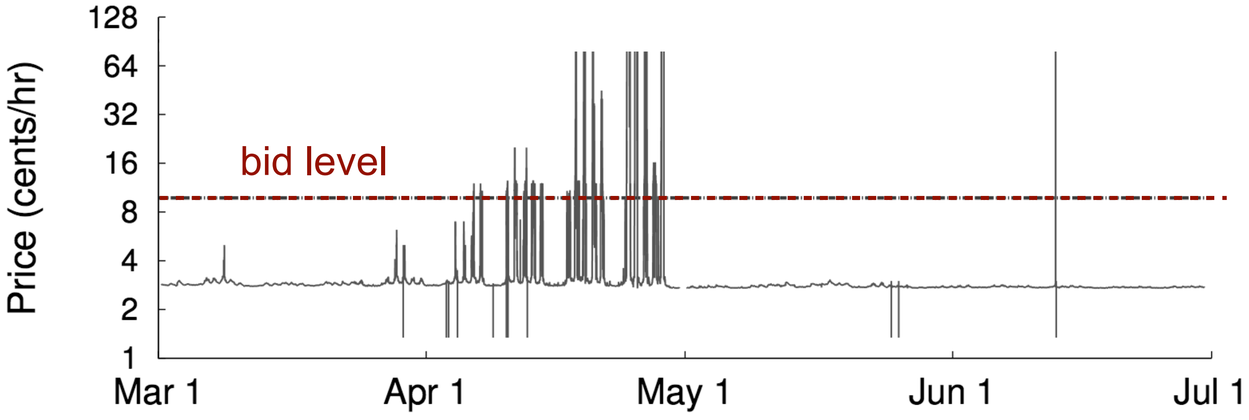
Deliberate revocations

when spot price rises above the bid level

Fermilab

"Spot instances helped scale our clusters up by **4X** during the discovery of the Higgs Boson"

Applications Hate Spot Servers… but Can't Live Without Them



Low average price

50-90% discount over on-demand price

Deliberate revocations

when spot price rises above the bid level

Fermilab

"Spot instances helped scale our clusters up by **4X** during the discovery of the Higgs Boson"

Spot compute consumed in 2016 > All EC2 compute consumed in 2012

https://aws.amazon.com/10year/

When "*Just Throw in Some Fault-tolerance*" Does Not Help!

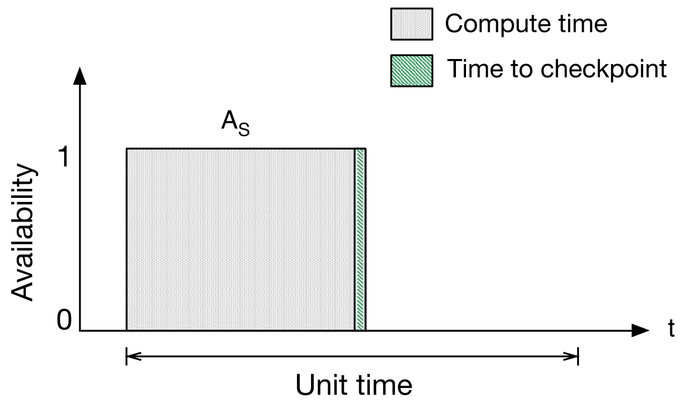
Spot server behavior comprises of **three** key characteristics:

Availability, **VOLATILITY**, *Predictability*

When "*Just Throw in Some Fault-tolerance*" Does Not Help!

Spot server behavior comprises of **three** key characteristics:

Availability, **VOLATILITY**, *Predictability*



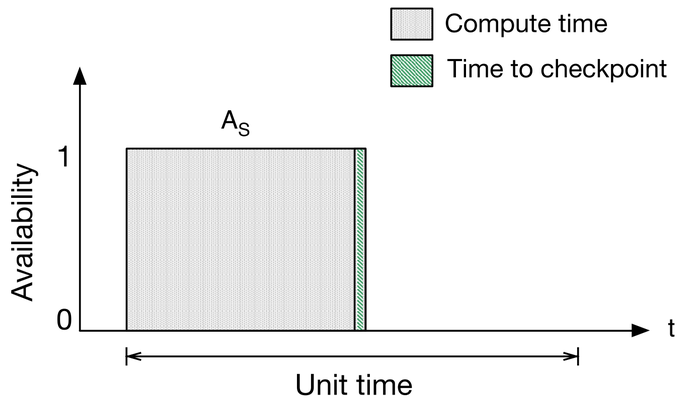
Limited Availability

Checkpoints = 1

When "*Just Throw in Some Fault-tolerance*" Does Not Help!

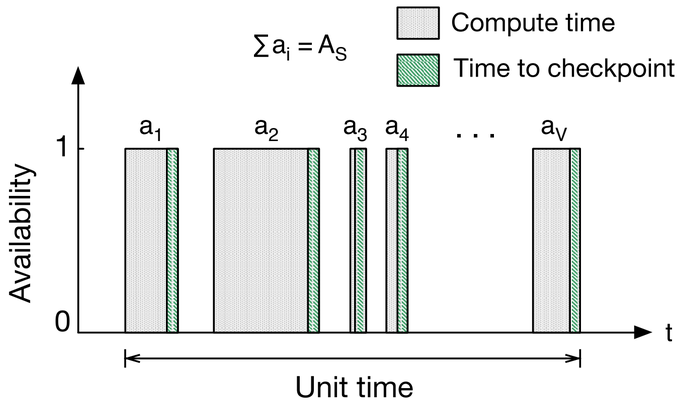
Spot server behavior comprises of **three** key characteristics:

Availability, **VOLATILITY**, *Predictability*



Limited Availability

Checkpoints = 1



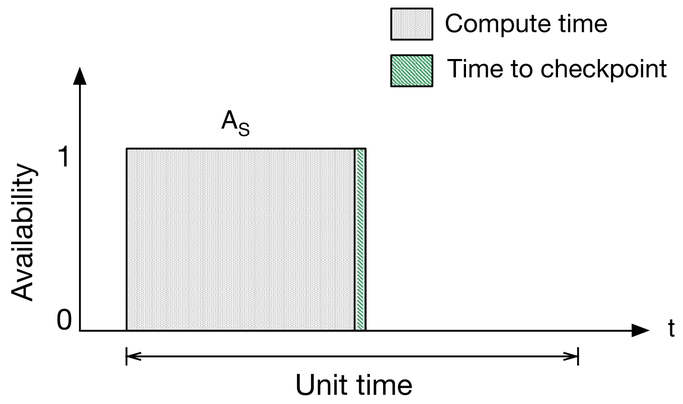
Limited Availability
+ **Volatility**

Checkpoints = no. of revocations

When "*Just Throw in Some Fault-tolerance*" Does Not Help!

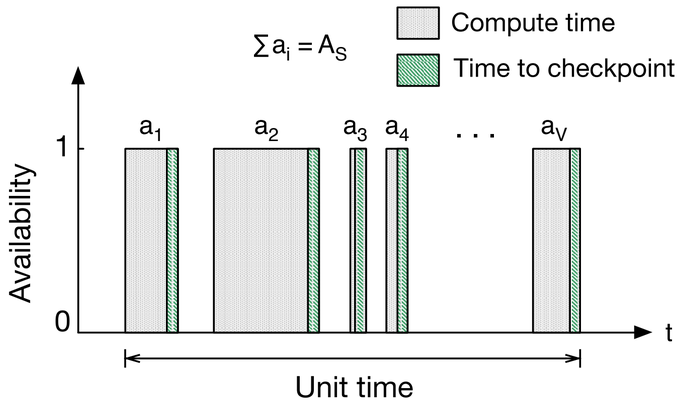
Spot server behavior comprises of **three** key characteristics:

Availability, **VOLATILITY**, *Predictability*



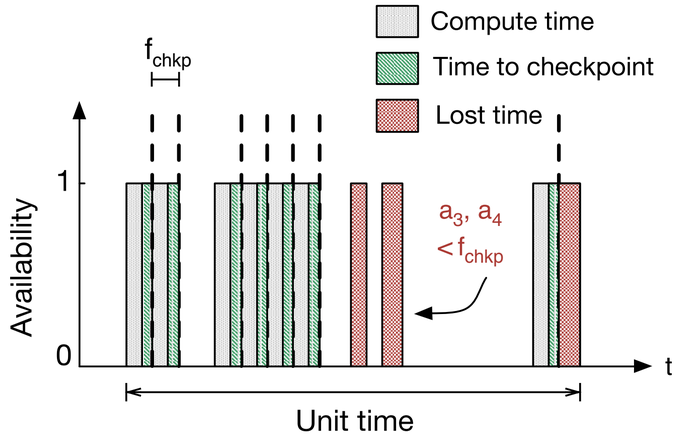
Limited Availability

Checkpoints = 1



Limited Availability
+ **Volatility**

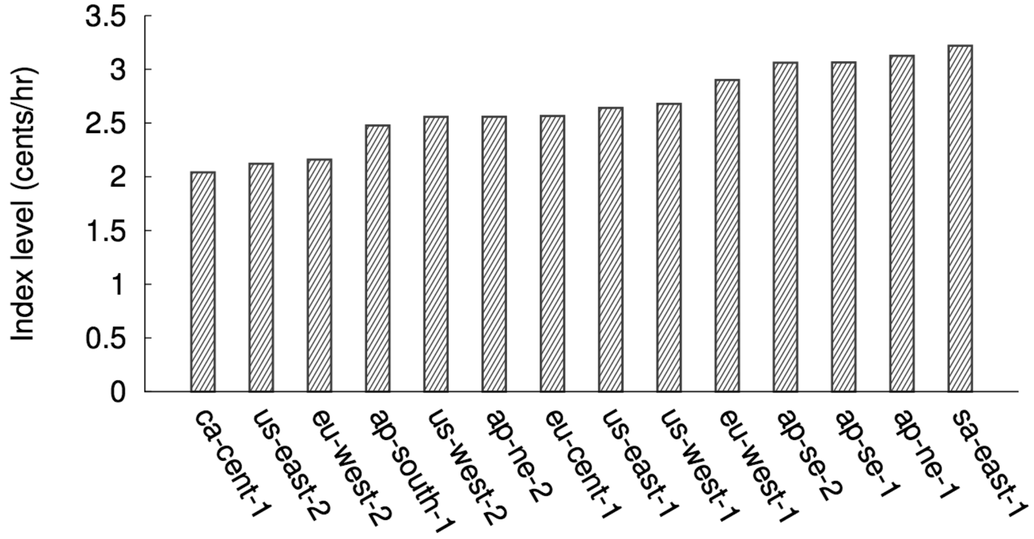
Checkpoints = no. of revocations



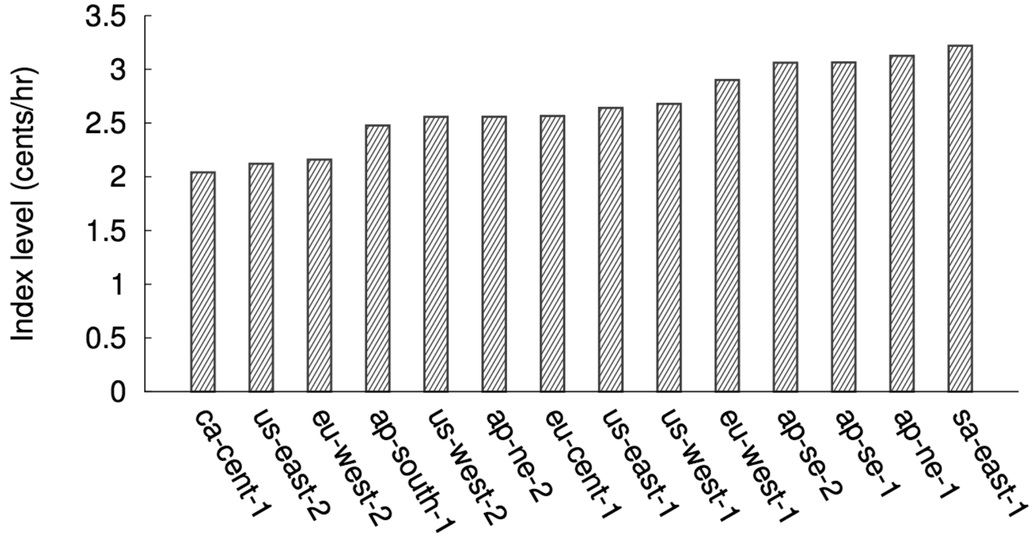
Limited Availability + Volatility
+ **Unpredictability**

Periodic checkpointing

Worldwide On-demand (Linux)



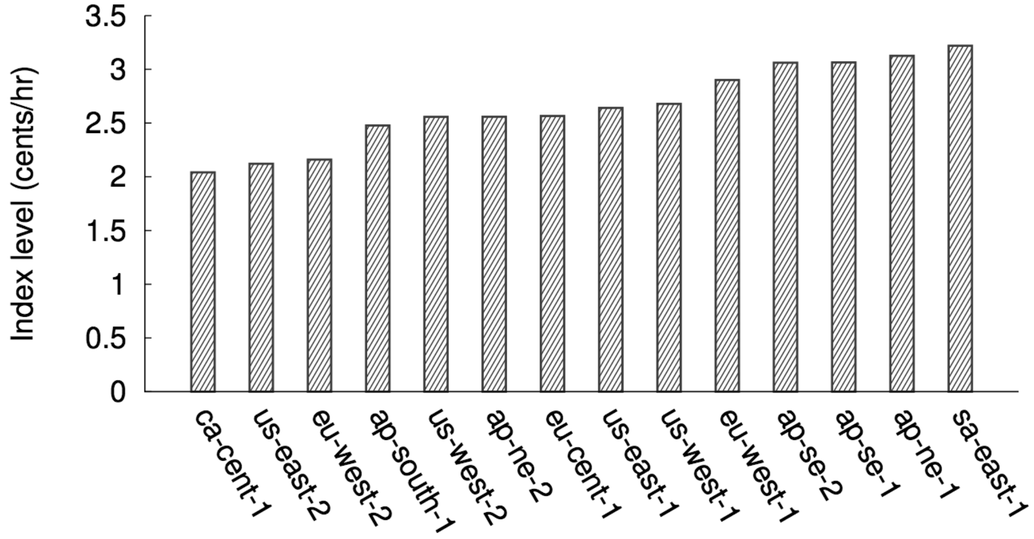
Worldwide On-demand (Linux)



Compute-time is 50% more expensive

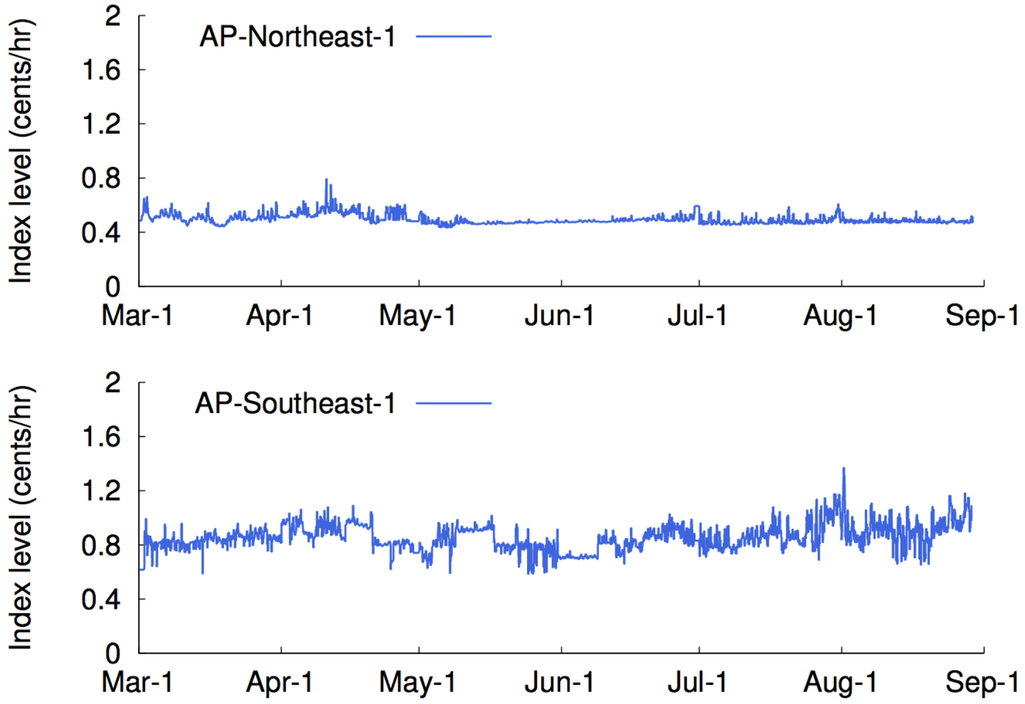
in Brazil than Canada

Worldwide On-demand (Linux)

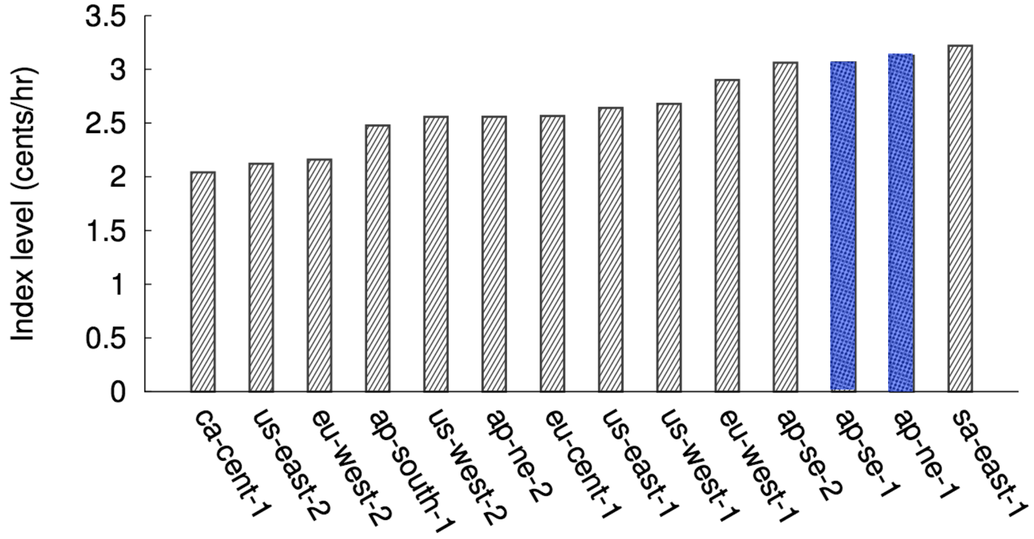


Compute-time is 50% more expensive in Brazil than Canada

Identifying Price Inversions

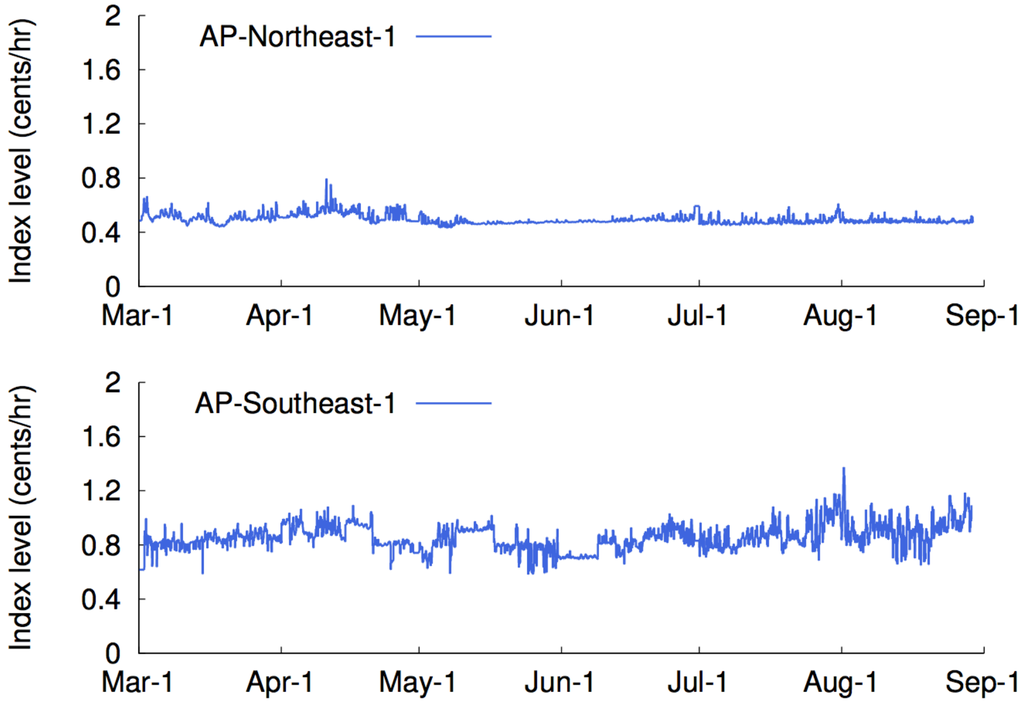


Worldwide On-demand (Linux)



Compute-time is 50% more expensive in Brazil than Canada

Identifying Price Inversions



Indices allow us to readily identify systematic price differentials and arbitrages