Transient Guarantees:

Maximizing the Value of Idle Cloud Capacity

Supreeth Shastri, Amr Rizk, David Irwin

UMassAmherst

Idle Cloud Capacity

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.*

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.*

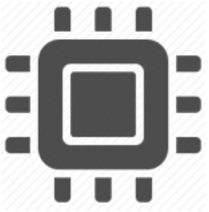
has its limitations

Selling Idle Cloud Capacity

amazon web services™ | EC2

Users bid in a
2nd price auction

EC2 continually evaluates supply-demand to price spot servers



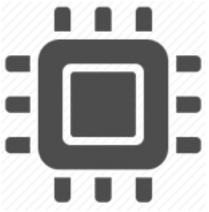
Allocate: bid price ≥ spot price
Revoke: bid price < spot price

Selling Idle Cloud Capacity

amazon web services™ | EC2

Users bid in a
2nd price auction

EC2 continually evaluates supply-demand to price spot servers



Allocate: bid price ≥ spot price
Revoke: bid price < spot price

> *On average, AWS customers are using **more** compute capacity on **spot instances** than across all of EC2 in **2012***

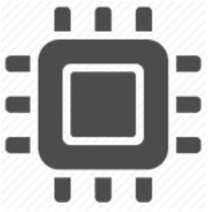
https://aws.amazon.com/10year/

Selling Idle Cloud Capacity

amazon web services™ | EC2

Users bid in a
2nd price auction

EC2 continually evaluates supply-demand to price spot servers



Allocate: bid price ≥ spot price
Revoke: bid price < spot price

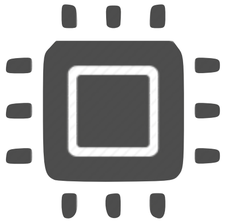
*❝ On average, AWS customers are using **more** compute capacity on **spot instances** than across all of EC2 in **2012** ❞*

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

*❝ Experiment that discovered the **Higgs Boson** used AWS spot instances ❞*

Fermilab Scientific Computing Division

Compute Time vs. Commodity Markets

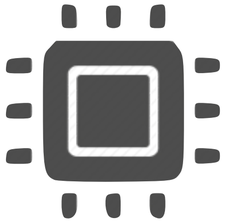


vs.

. . .

Commodity markets are inherently *Volatile* and *Unpredictable*

Compute Time vs. Commodity Markets



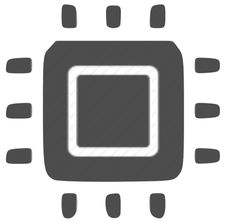
vs.

. . .

Commodity markets are inherently *Volatile* and *Unpredictable*

Compute time is
Stateful

Compute Time vs. Commodity Markets



vs.

Commodity markets are inherently *Volatile* and *Unpredictable*

Compute time is *Stateful*

1. *Losing a server unpredictably incurs an* **overhead**
2. *This overhead* **decreases** *the useful compute time of the server*

∴ *Market volatility* ***reduces*** *the value of* ***compute time*** *purchased*

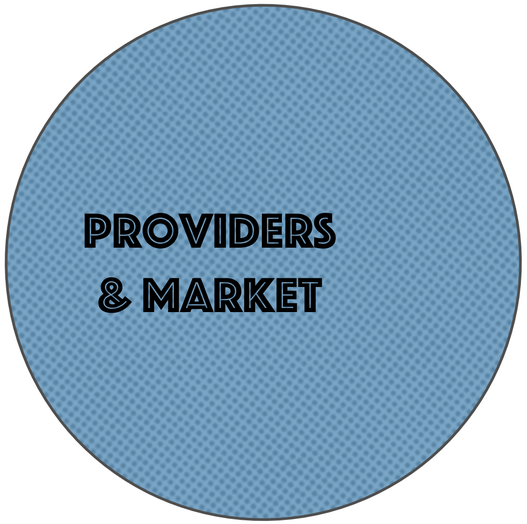
How to Maximize the Value of Idle Cloud Capacity?

How to Maximize the Value of Idle Cloud Capacity?

Google economy class [**SoCC '14**]

Amazon Spot-blocks [**Amazon '15**]

spot market dynamics [**HotCloud '16, ICDCS '16, ToEC '13**]

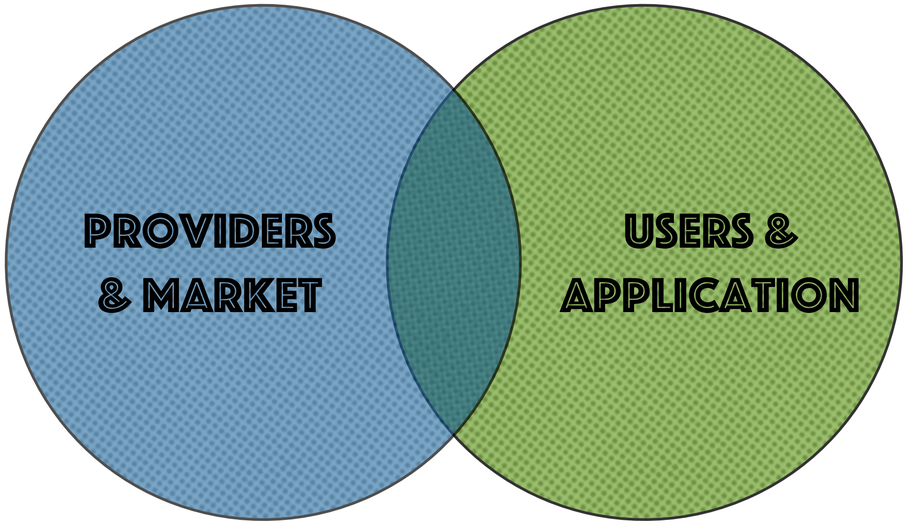


How to Maximize the Value of Idle Cloud Capacity?

Google economy class [**SoCC '14**]

Amazon Spot-blocks [**Amazon '15**]

spot market dynamics [**HotCloud '16, ICDCS '16, ToEC '13**]



Modify applications for transiency [**HPDC '15, HPDC '14**]

Handle transiency at system-level [**SoCC '15, EuroSys '15**]

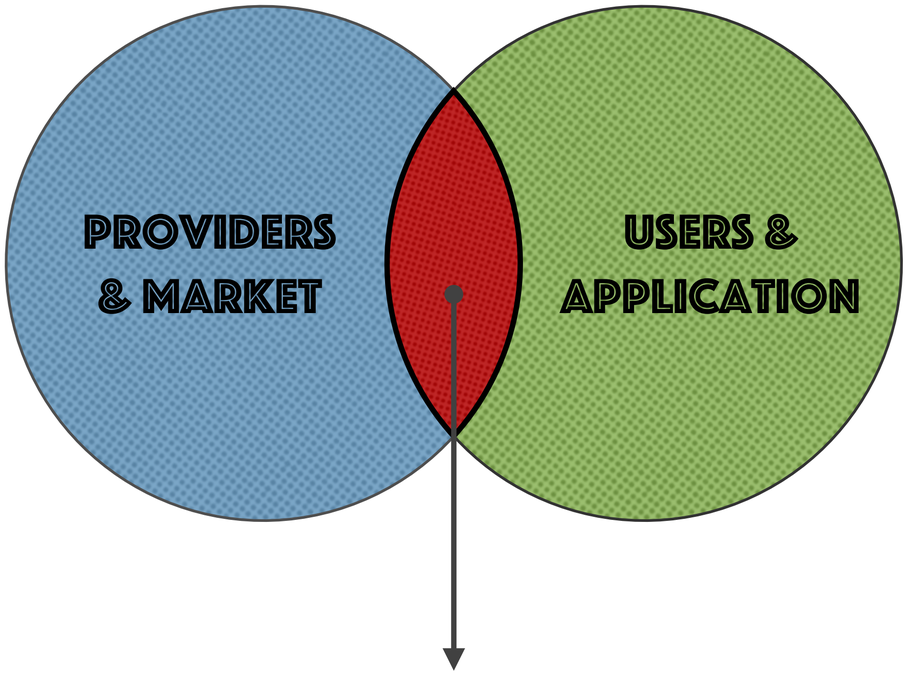
Design optimal bidding policies [**SIGCOMM '15, CLOUD '12**]

How to Maximize the Value of Idle Cloud Capacity?

Google economy class
[**SoCC '14**]

Amazon Spot-blocks
[**Amazon '15**]

spot market dynamics
[**HotCloud '16, ICDCS '16, ToEC '13**]



Modify applications for transiency
[**HPDC '15, HPDC '14**]

Handle transiency at system-level
[**SoCC '15, EuroSys '15**]

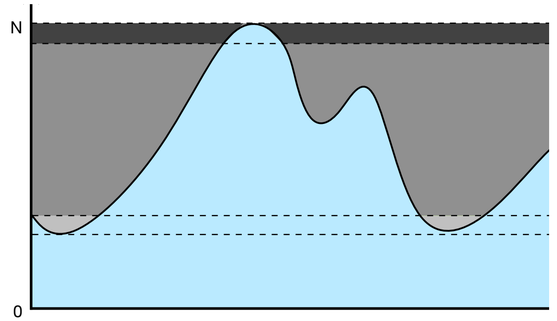
Design optimal bidding policies
[**SIGCOMM '15, CLOUD '12**]

Rest of the Talk . . .

Transient Server Characteristics

Transient Guarantees

Evaluations



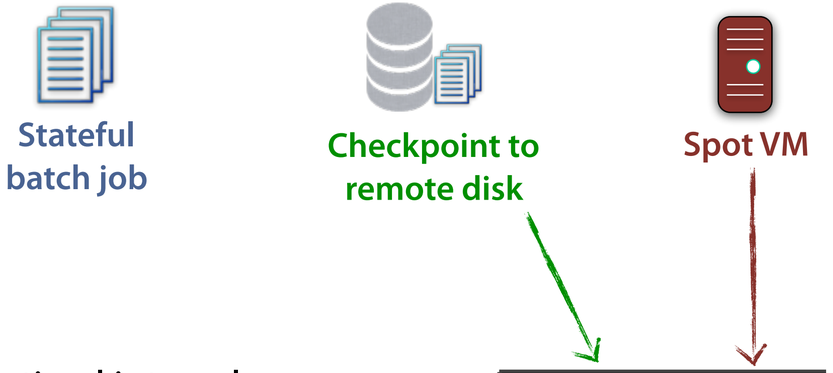
Transient Server Characteristics

Transient Servers are Intrinsically Less Valuable!

Batch job on a spot VM using checkpointing

Transient Servers are Intrinsically Less Valuable!

Batch job on a spot VM using checkpointing

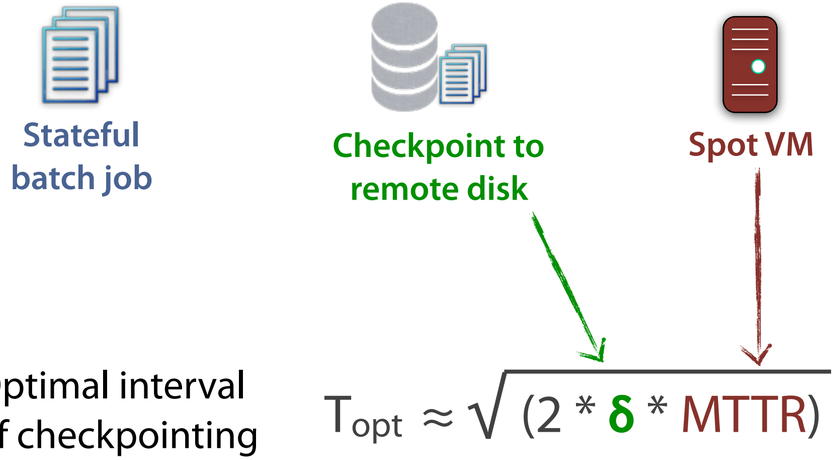


Optimal interval
of checkpointing

$$T_{opt} \approx \sqrt{(2 * \delta * MTTR)}$$

Transient Servers are Intrinsically Less Valuable!

Batch job on a spot VM using checkpointing



Optimal interval of checkpointing

$$T_{opt} \approx \sqrt{(2 * \delta * MTTR)}$$

Expected runtime

$$E[T_{spot}] = T + \left(\frac{T}{T_{opt}} * \delta\right) + \left(\frac{T}{MTTR} * \frac{T_{opt}}{2}\right)$$

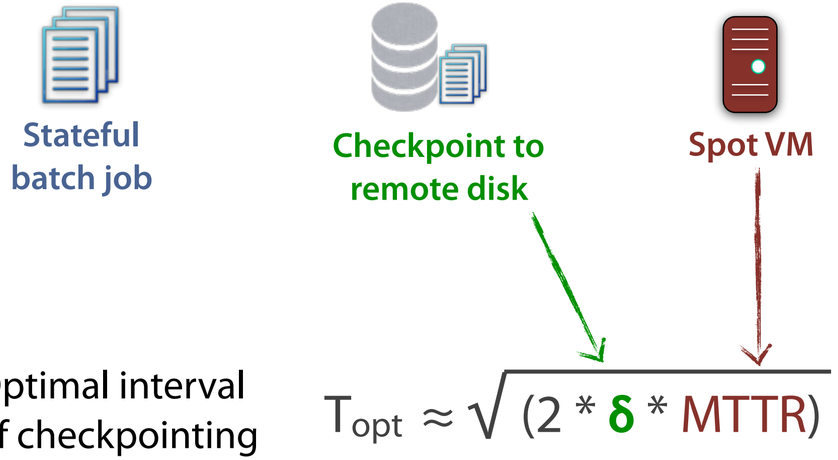
Actual Runtime

Checkpointing Overhead

Recomputation

Transient Servers are Intrinsically Less Valuable!

Batch job on a spot VM using checkpointing



Optimal interval of checkpointing

$$T_{opt} \approx \sqrt{(2 * \delta * MTTR)}$$

Expected runtime

$$E[T_{spot}] = T + \left(\frac{T}{T_{opt}} * \delta\right) + \left(\frac{T}{MTTR} * \frac{T_{opt}}{2}\right)$$

Actual Runtime

Checkpointing Overhead

Recomputation

❝ On average, transient servers get less work done per unit of time compared to an equivalent on-demand server ❞

Transient Servers are Intrinsically Less Valuable!

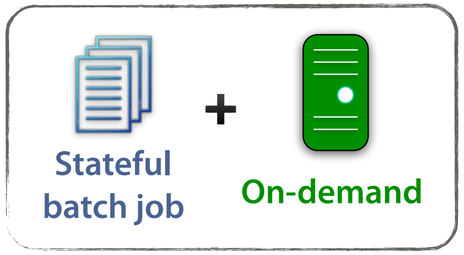
Equilibrium Price of Transient Server
(or *price when spot stops being cheap*)

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

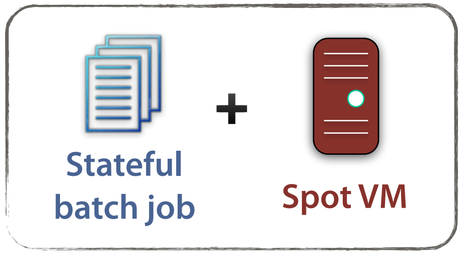
Transient Servers are Intrinsically Less Valuable!

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

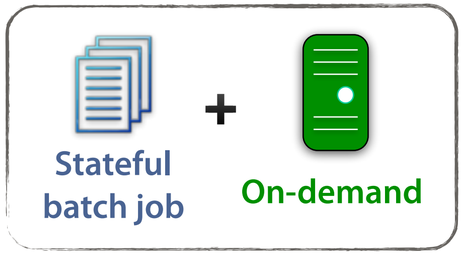


Completion time
20 hours

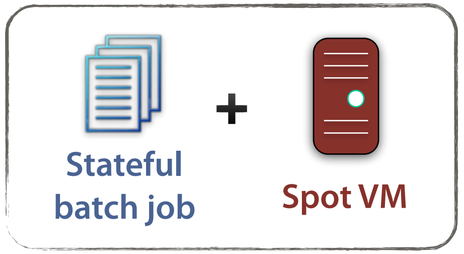
Transient Servers are Intrinsically Less Valuable!

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

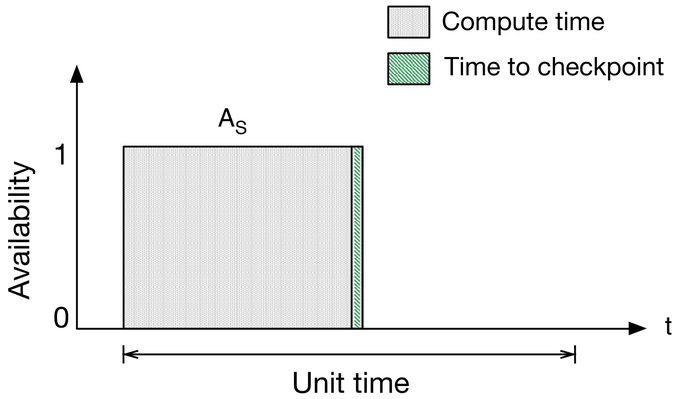
❝ For this application, a spot server with **40%** *discount, provides* **no savings at all** *❞*

Distilling the Transient Server Characteristics

We identify **three** key metrics: **Availability**, **VOLATILITY**, *Predictability*

Distilling the Transient Server Characteristics

We identify **three** key metrics: **Availability**, VOLATILITY, *Predictability*

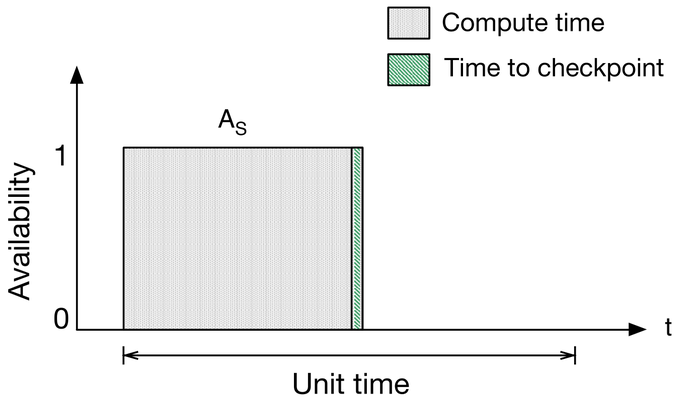


Limited Availability

Checkpoints = 1

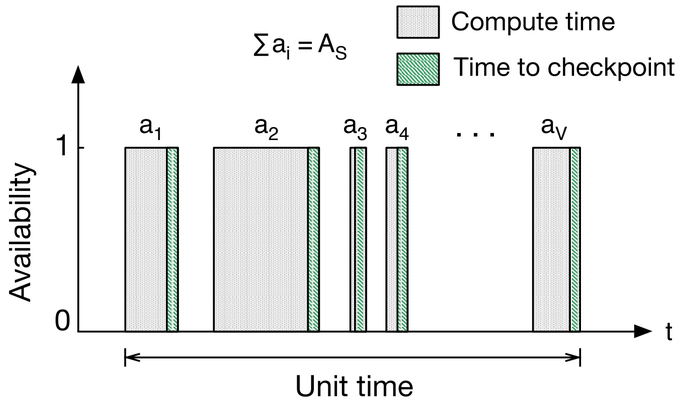
Distilling the Transient Server Characteristics

We identify **three** key metrics: **Availability**, VOLATILITY, *Predictability*



Limited Availability

Checkpoints = 1

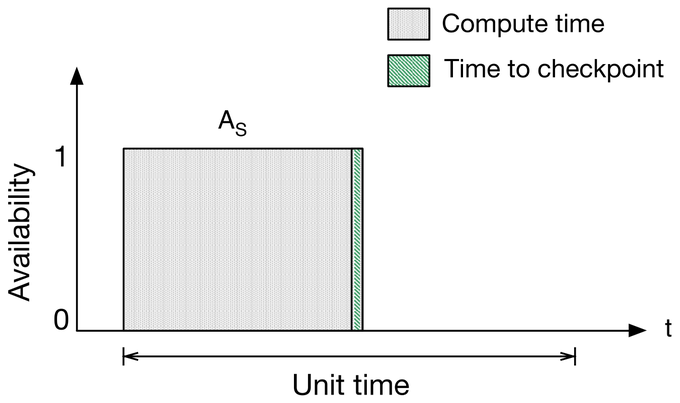


Limited Availability
+ **Volatility**

Checkpoints = no. of revocations

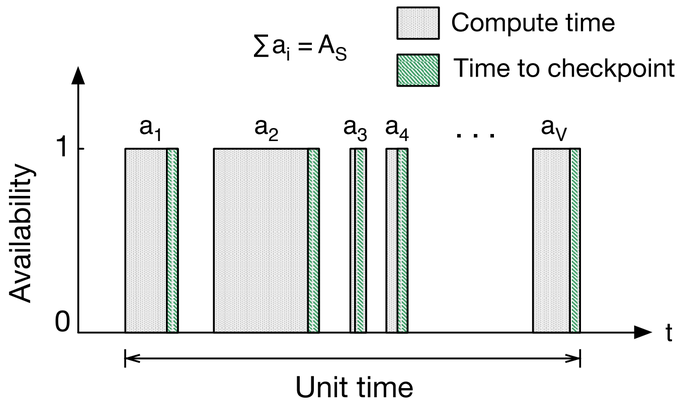
Distilling the Transient Server Characteristics

We identify **three** key metrics: **Availability**, VOLATILITY, *Predictability*



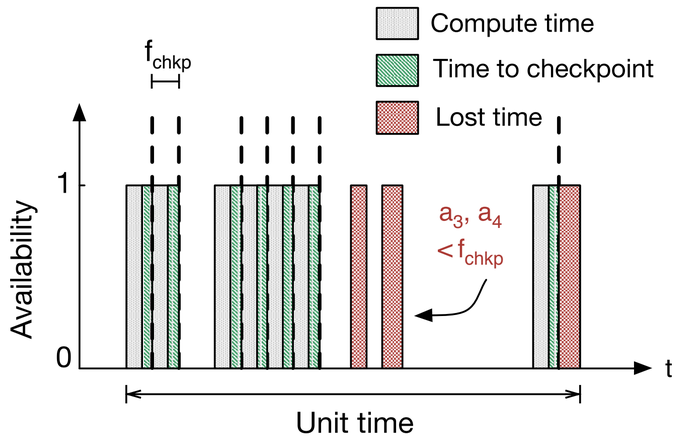
Limited Availability

Checkpoints = 1



Limited Availability
+ **Volatility**

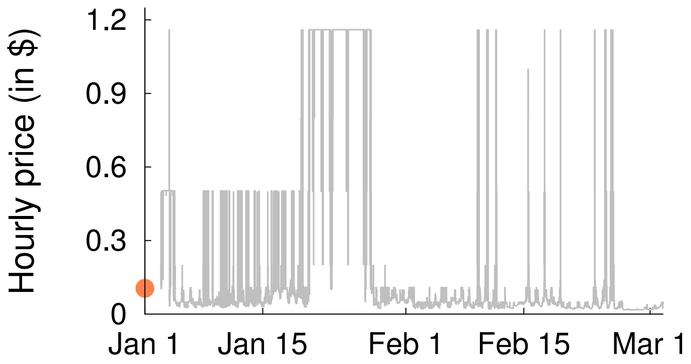
Checkpoints = no. of revocations



Limited Availability + Volatility
+ **Unpredictability**

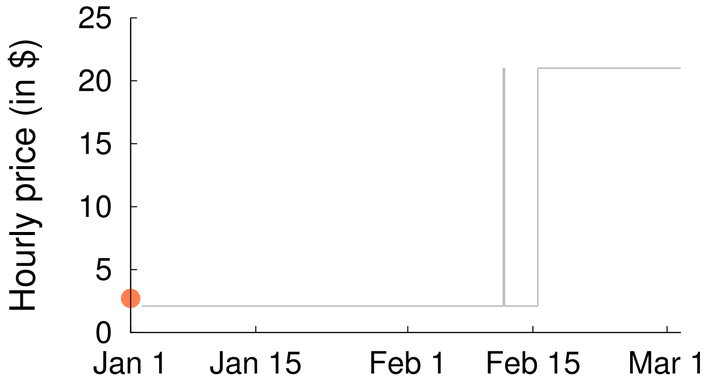
Periodic checkpointing

Back To Amazon Spot Markets . . .



c4.large (Linux) us-east-1

Mature markets are
more volatile and less predictable



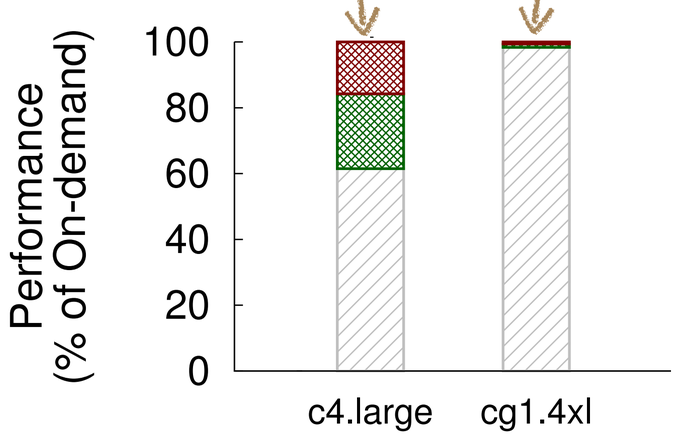
cg1.4xlarge (Linux) us-east-1

Deprecated/rarely used markets are
less volatile and more predictable

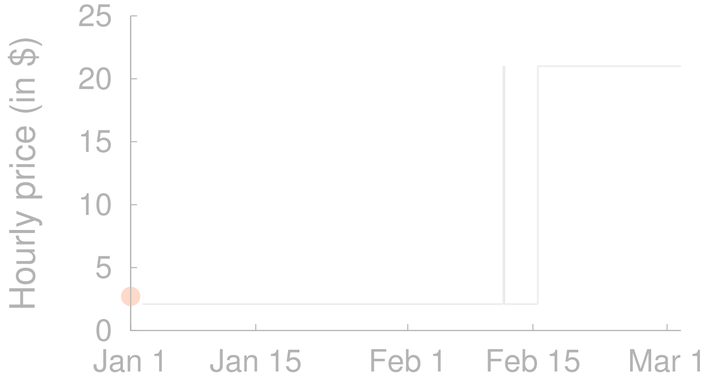
Back To Amazon Spot Markets . . .

c4.large (Linux) us-east-1

Mature markets are
more volatile and less predictable



Performance of markets



cg1.4xlarge (Linux) us-east-1

Deprecated/rarely used markets are
less volatile and more predictable

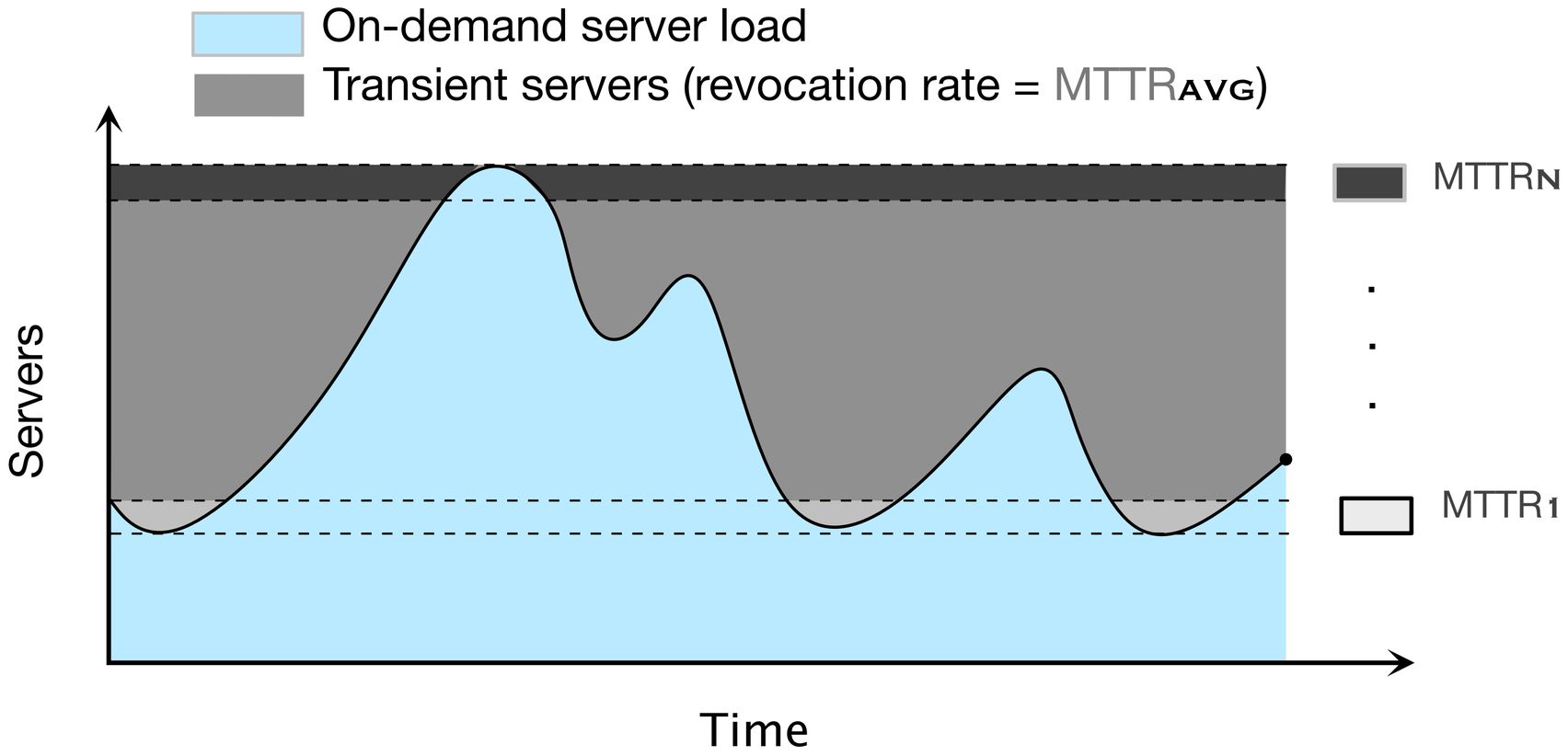
Transient Guarantees

❝ 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)

Characterizing Idle Capacity



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

Highly Volatile nodes ← → **Highly Available nodes**

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

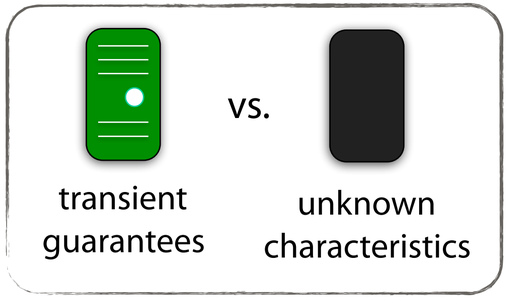
Providing probabilistic assurances on *availability, volatility and predictability* of transient servers

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***



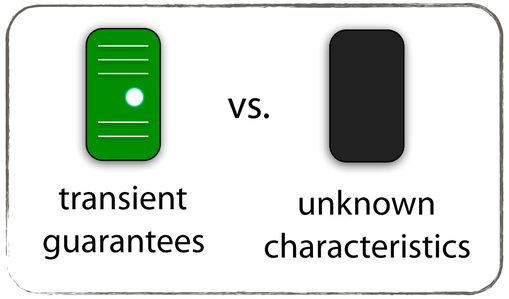
Advertised characteristics

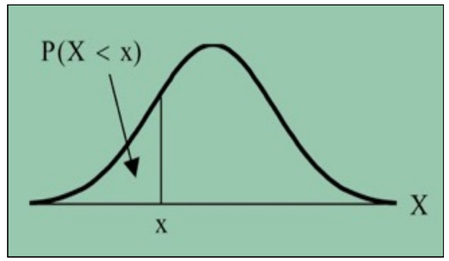
Value the servers correctly (users and providers)

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

Value the servers correctly (users and providers)

Probabilistic assurance

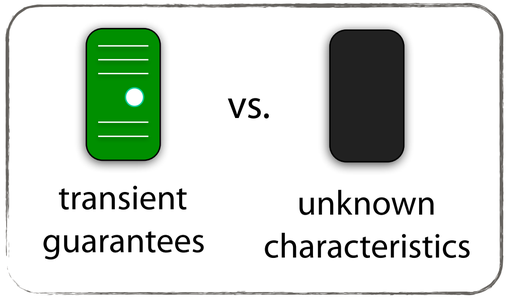
Revoke any server anytime (providers)

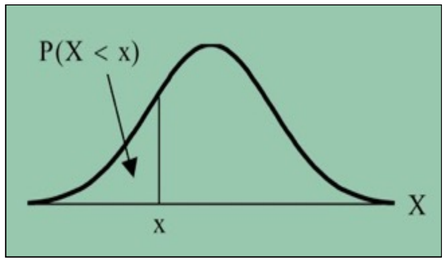
Fine-tune fault-tolerance (users)

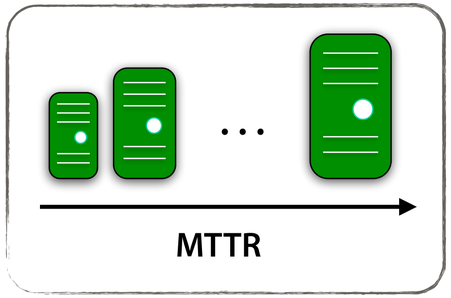
Transient Guarantees

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

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







Advertised characteristics

Value 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)

Constructing Transient Classes

Partition N servers into K classes, with class c containing n_c servers

$$N = \sum_{c=1}^{K} n_c$$

Constructing Transient Classes

Partition N servers into K classes, with class c containing n_c servers

$$N = \sum_{c=1}^{K} n_c$$

Aggregate Performance

$$\sum_{c=1}^{K} \left(\sum_{i=n_c}^{n_{c+1}} \text{perf}(S_i \text{ with } \text{MTTR}_c) \right)$$

Constructing Transient Classes

Partition N servers into K classes, with class c containing n_c servers

$$N = \sum_{c=1}^{K} n_c$$

Aggregate Performance

$$\sum_{c=1}^{K} \left(\sum_{i=n_c}^{n_{c+1}} \text{perf}(S_i \text{ with } \text{MTTR}_c) \right)$$

K = 1
Worst

.
.
.

K = N
Best

Constructing Transient Classes

Partition N servers into K classes, with class c containing n_c servers

$$N = \sum_{c=1}^{K} n_c$$

Aggregate Performance

$$\sum_{c=1}^{K} \left(\sum_{i=n_c}^{n_{c+1}} \text{perf}(S_i \text{ with } \text{MTTR}_c) \right)$$

K = 1
Worst

.
.
.

K = N
Best

$$O(N^K)$$

intractable for large N and K

Policies for partitioning:
equal-split, greedy-split

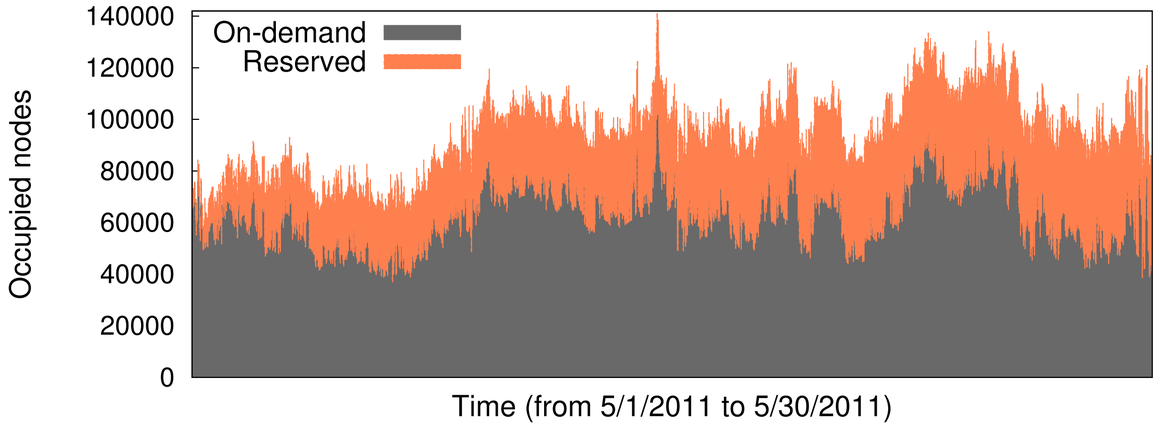
EValuations

How do transient characteristics look in a **real-world cluster**?

Do **transient classes** improve server performance?

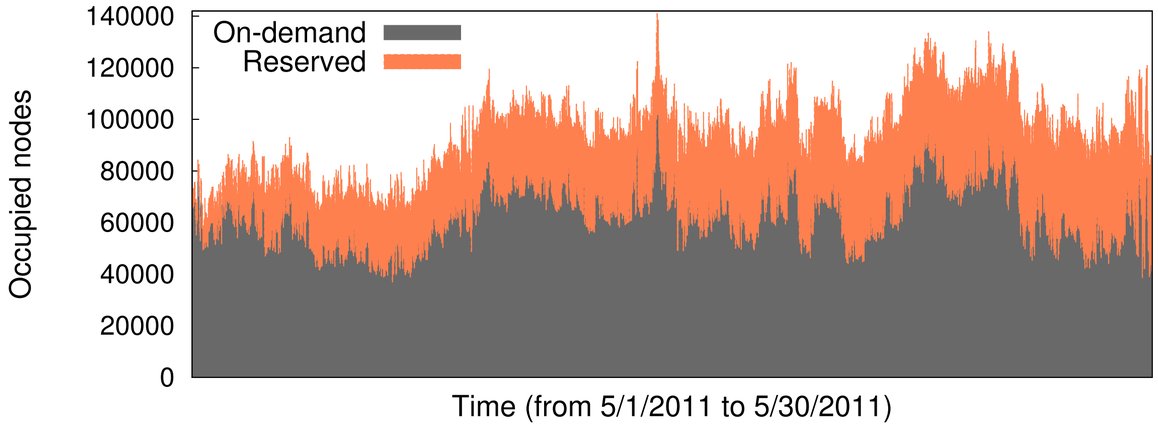
How close can we get to the **maximum** value?

Idle Capacity ⇢ Transient Servers



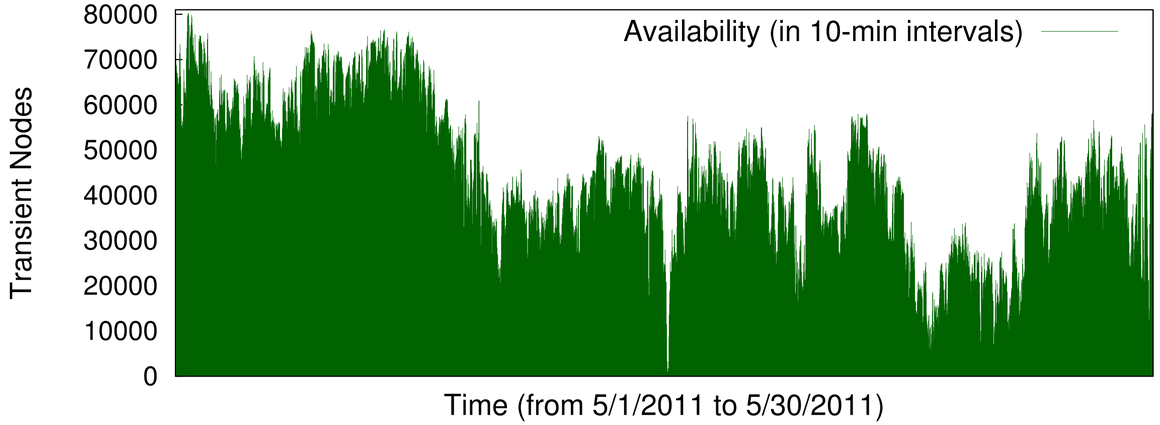
Server allocation from a production cluster at ***Google*** and the resulting idle capacity

Idle Capacity ⇢ Transient Servers

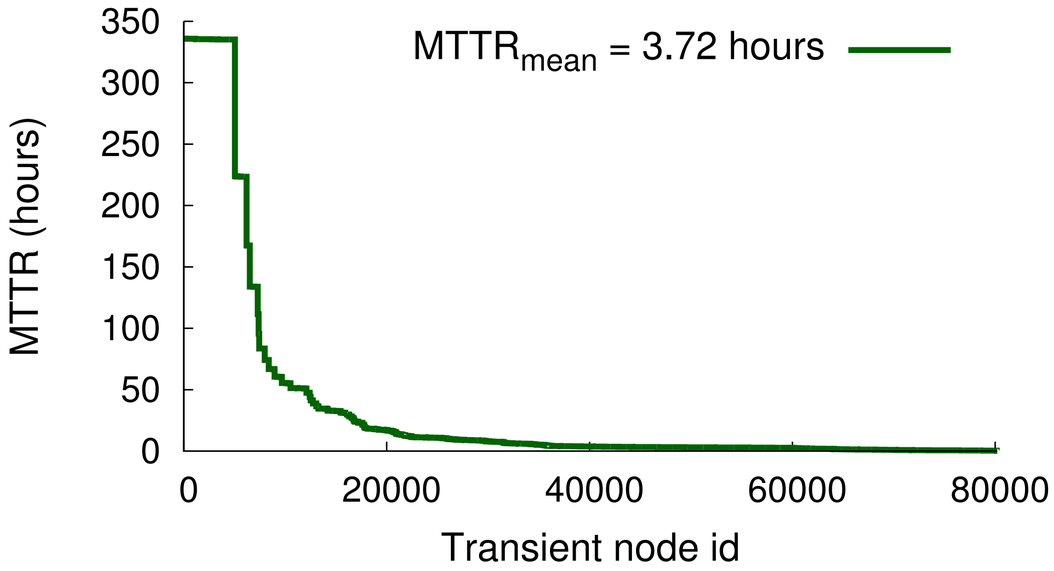


Server allocation from a production cluster at ***Google*** and the resulting idle capacity

Availability of transient servers varies considerably over time

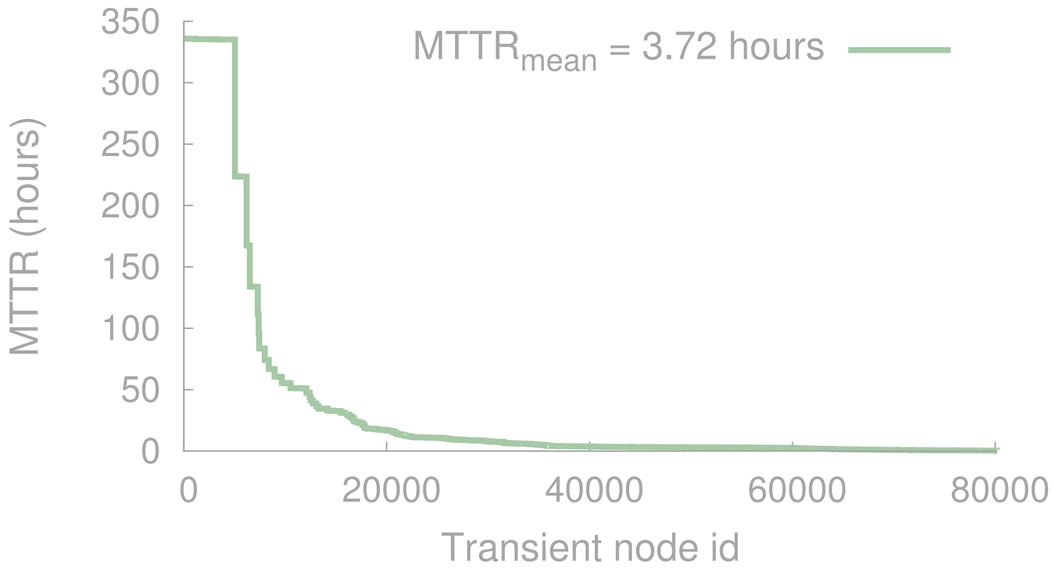


Transient Server Characteristics

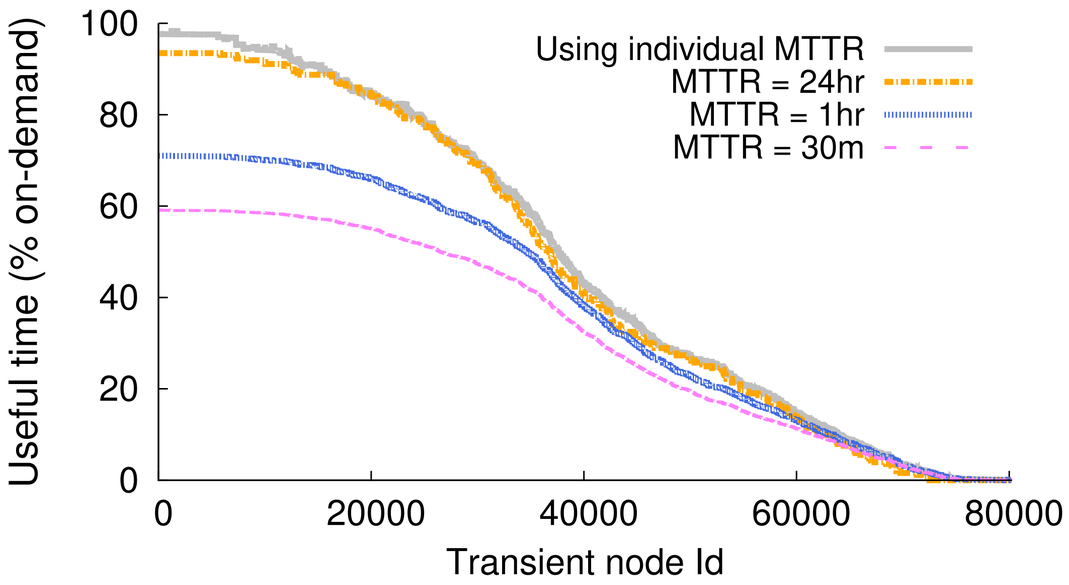


Transient Servers exhibit
a ***wide range*** of characteristics

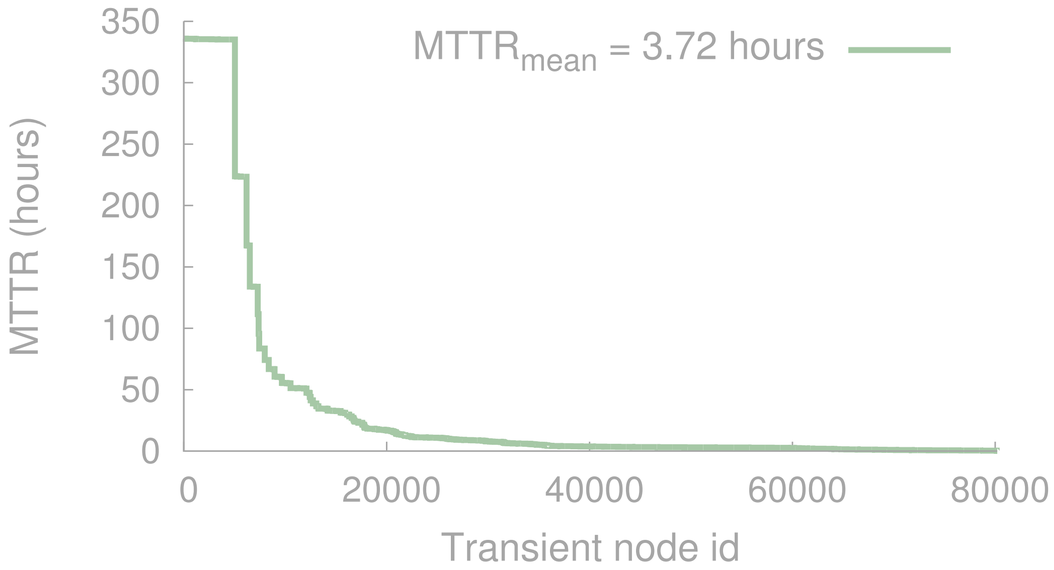
Transient Server Characteristics



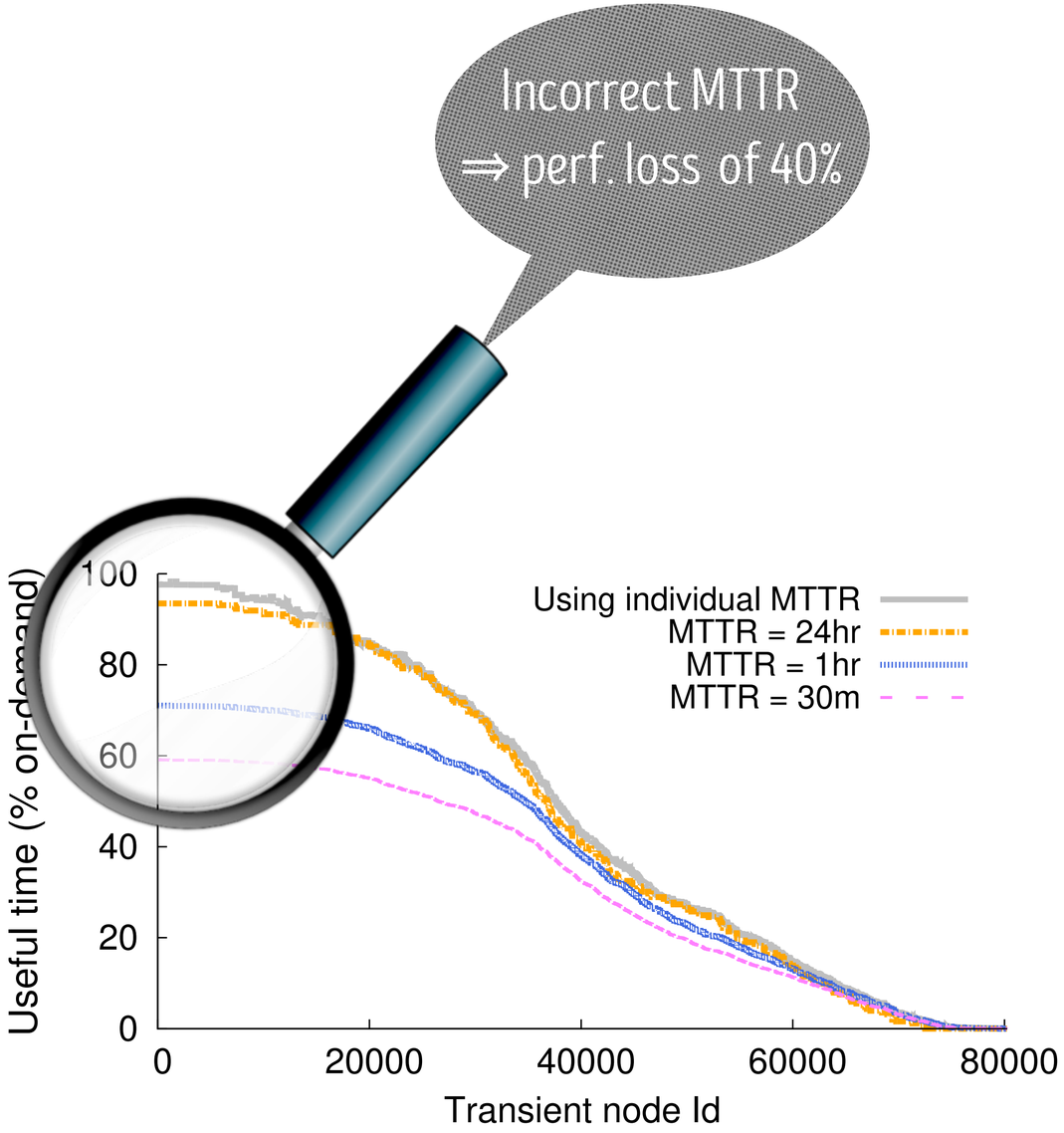
Transient Servers exhibit
a ***wide range*** of characteristics



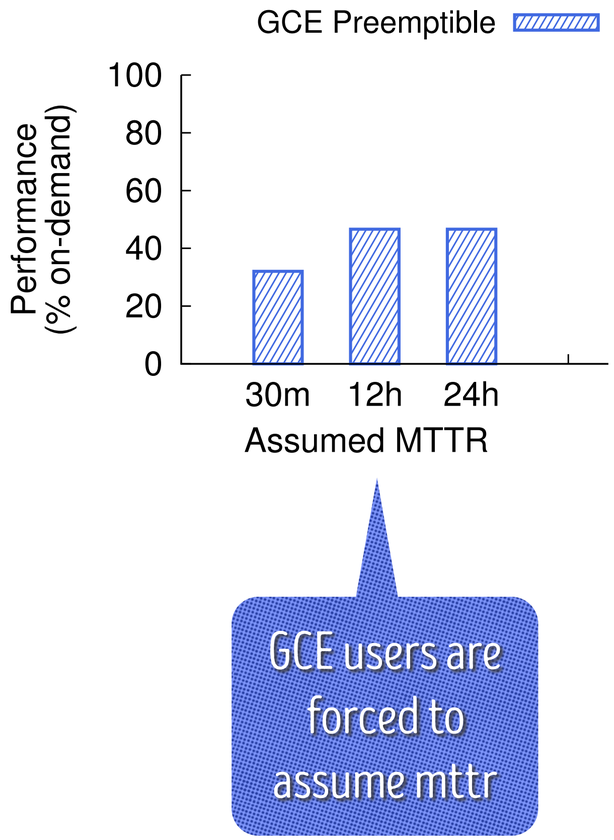
Transient Server Characteristics



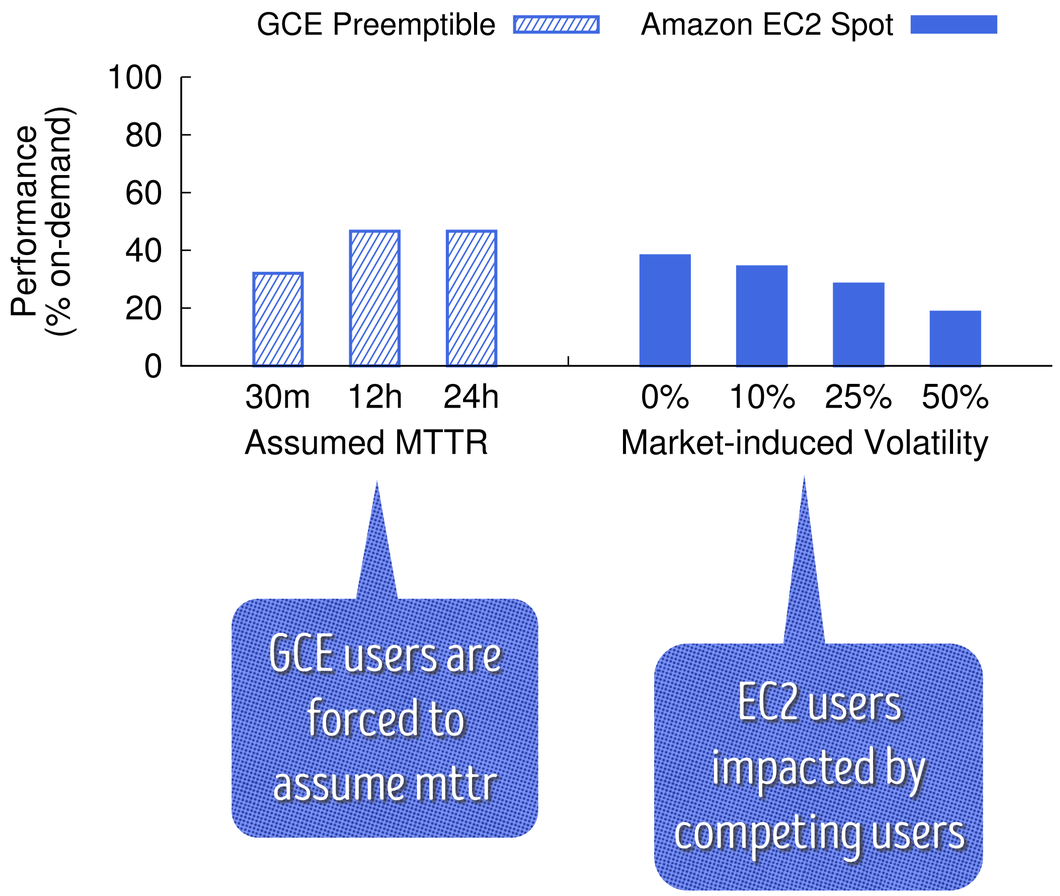
Transient Servers exhibit
a ***wide range*** of characteristics



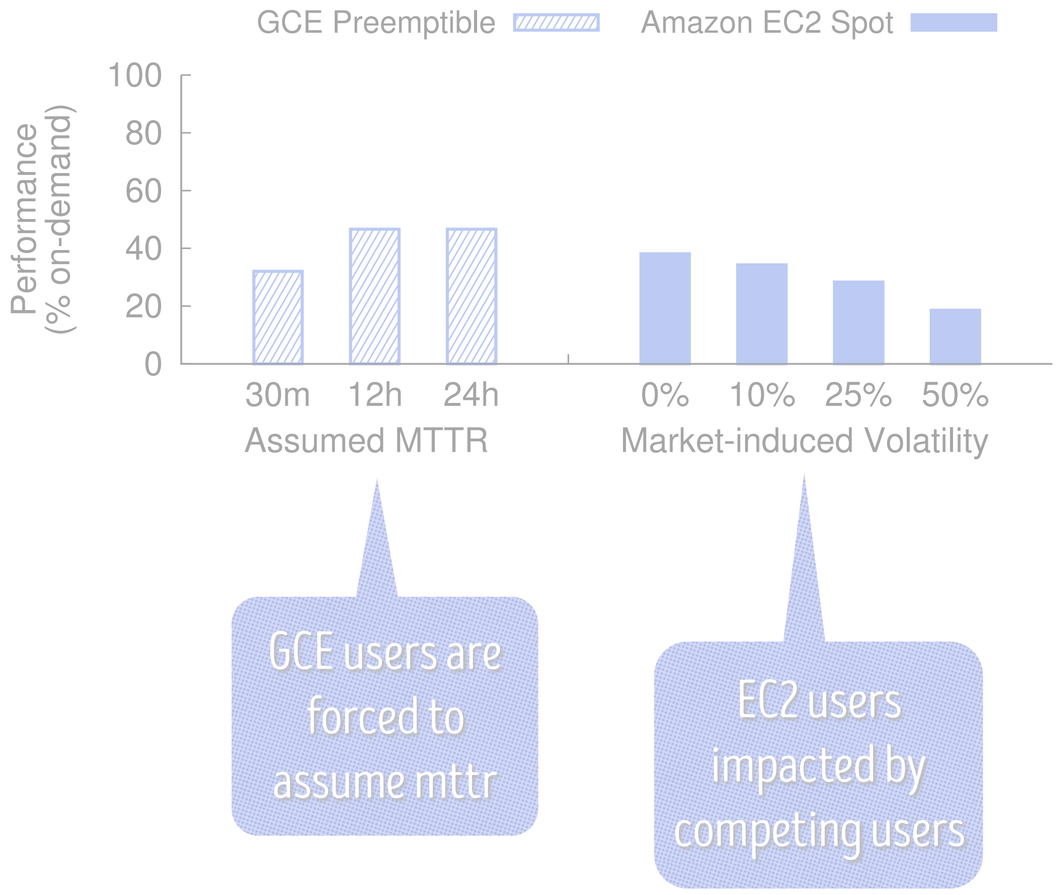
Performance of Transient Servers



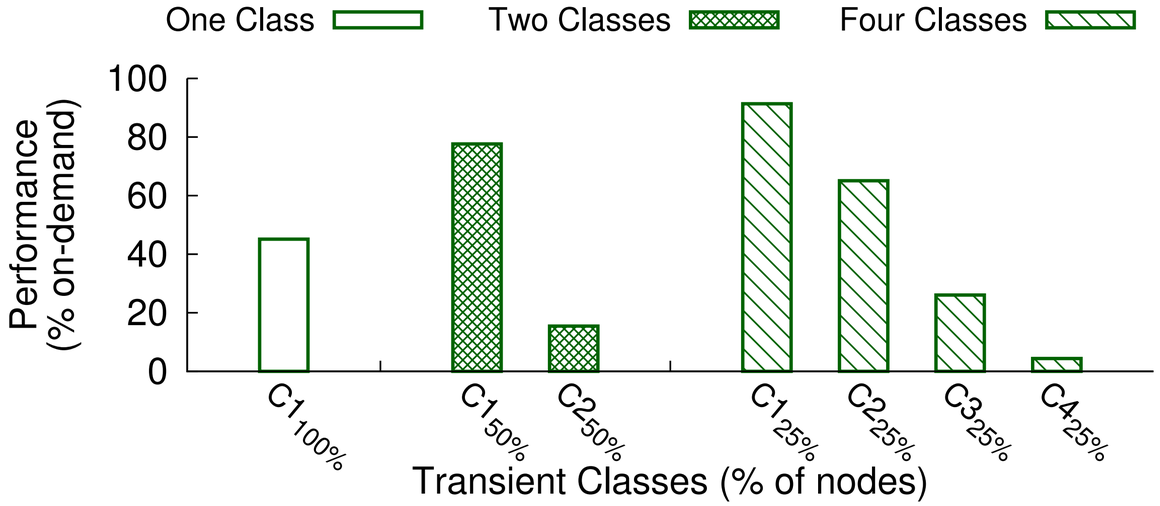
Performance of Transient Servers



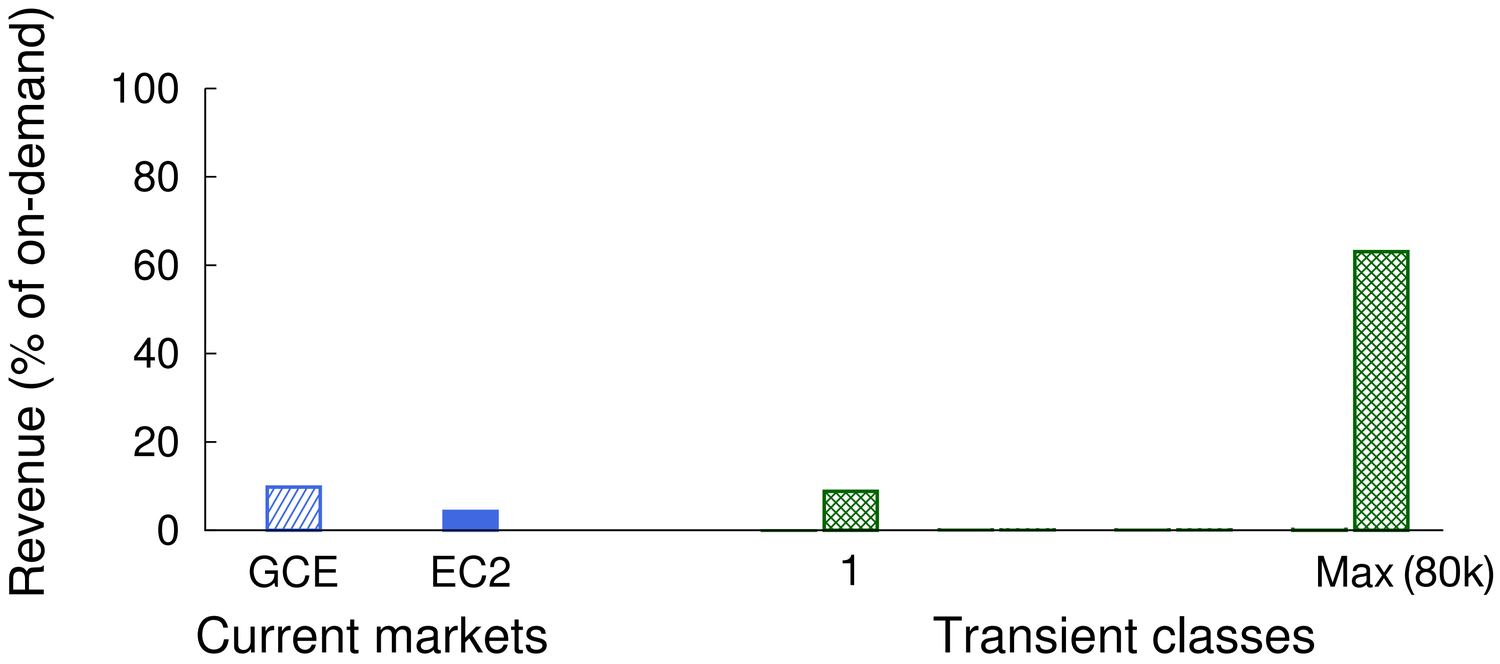
Performance of Transient Servers



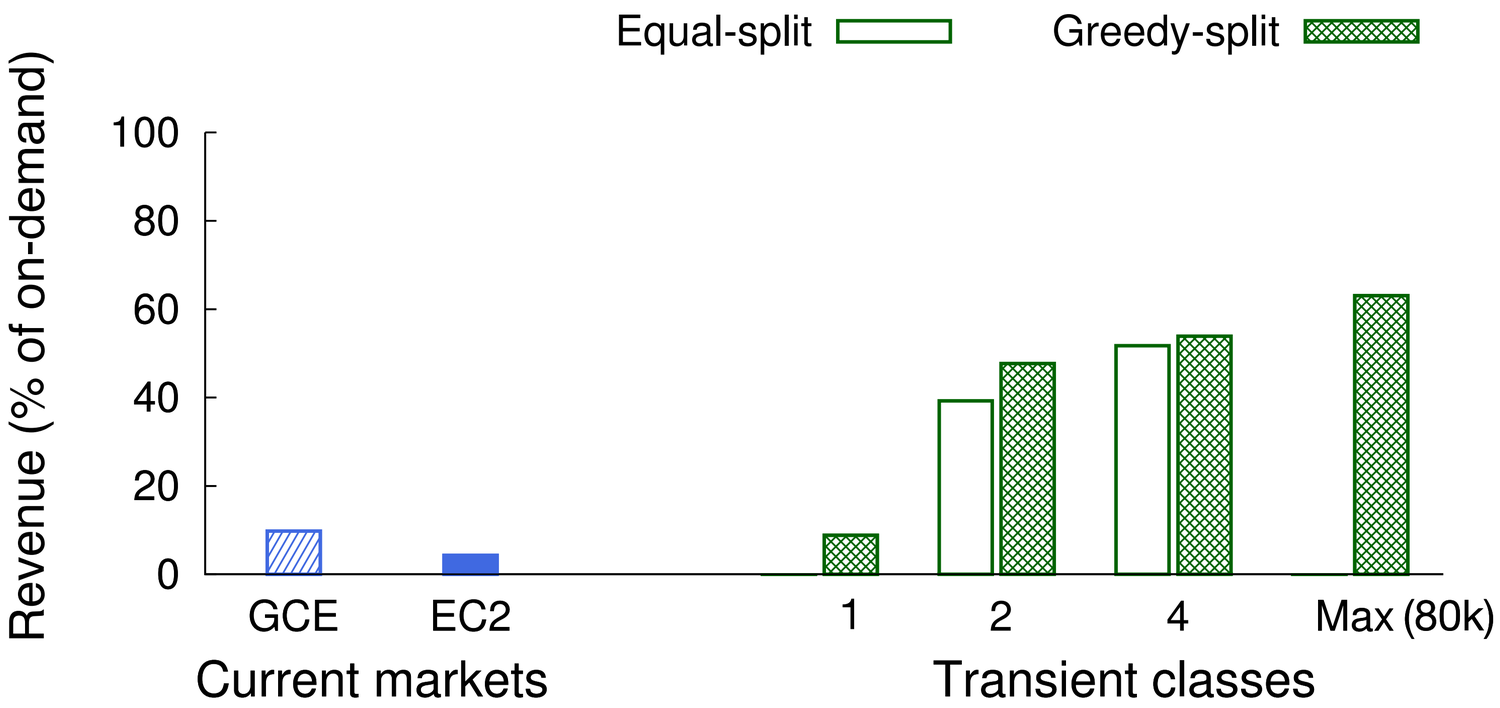
Transient Classes create
differentiated server pools



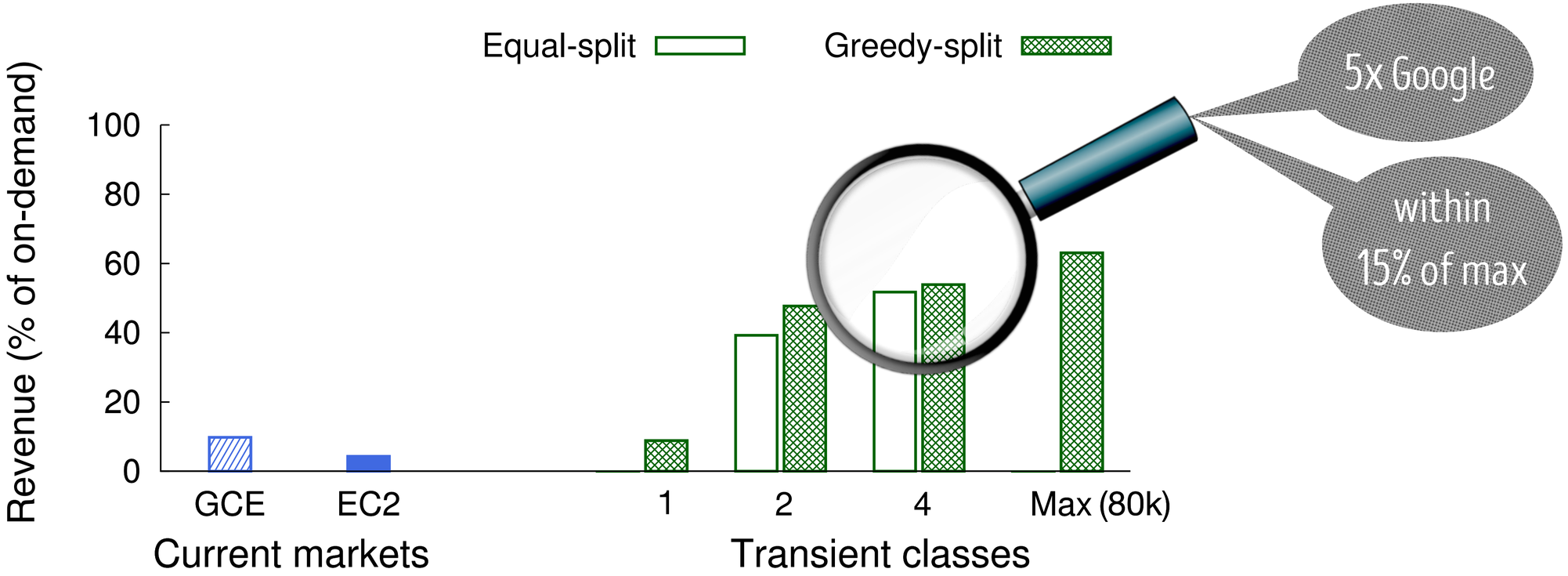
Provider's Revenue from Transient Offering



Provider's Revenue from Transient Offering



Provider's Revenue from Transient Offering



Conclusion

Transient servers are a new concept, and current markets do not maximize their value

Conclusion

Transient servers are a new concept, and current markets do not maximize their value

Transient Server Characteristics

Availability
Volatility
Predictability

Transient Guarantees

Transient Classes
Policies for Server Partitioning
Server Pricing

Evaluations

Prevent performance loss up to 40%

Revenue ⬆ 5x

Thank you!

Supreeth Shastri

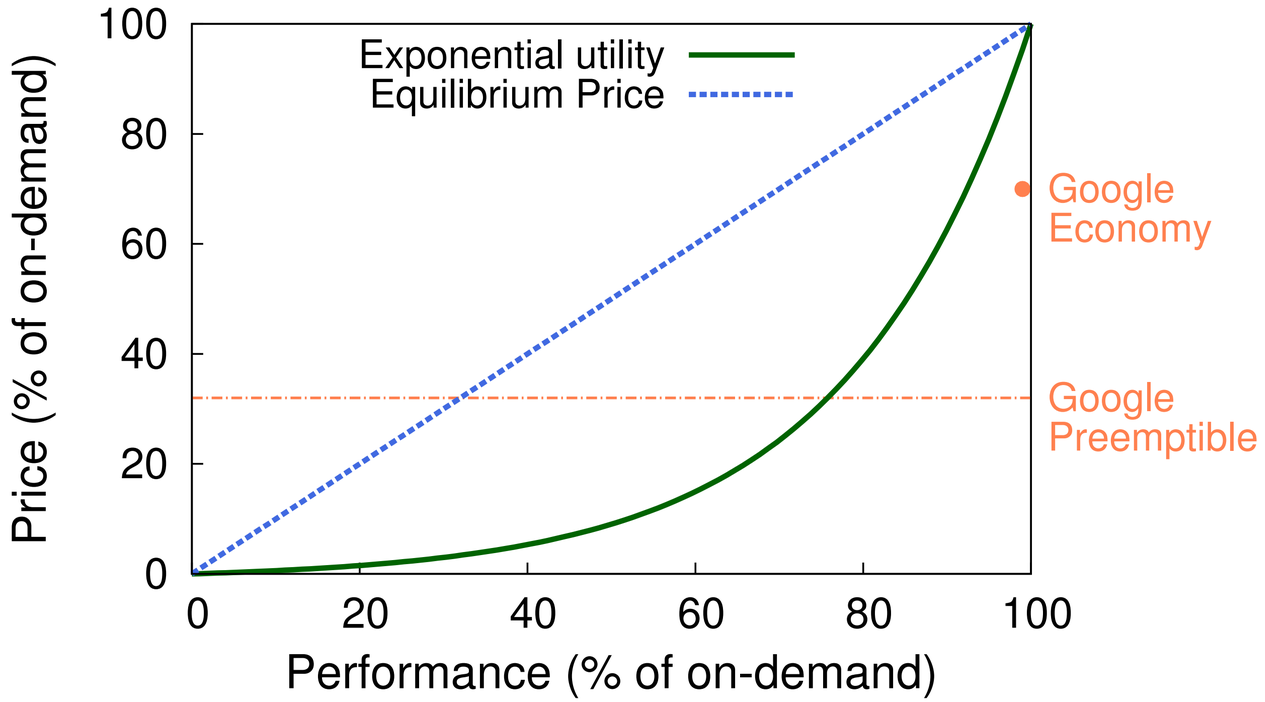
http://people.umass.edu/shastri/

UMassAmherst

Comparing Current Transient Markets

Pricing	Real-time, supply-demand based	Fixed, discounted
Allocation	Users compete in 2nd price auction	Unknown
Revocation	When market prices goes above the user bid	Unknown (max 24 hours)
Warning time	120 seconds	30 seconds
Spot markets	~2000	~250

Pricing Transient Servers

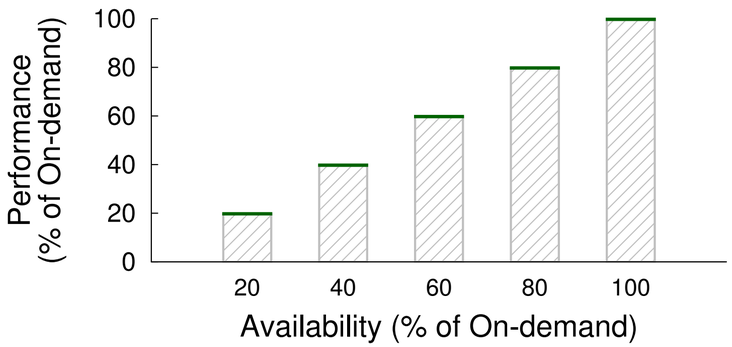


Distilling the Transient Server Characteristics

Availability, **VOLATILITY**, *Predictability* affect the performance differently

Distilling the Transient Server Characteristics

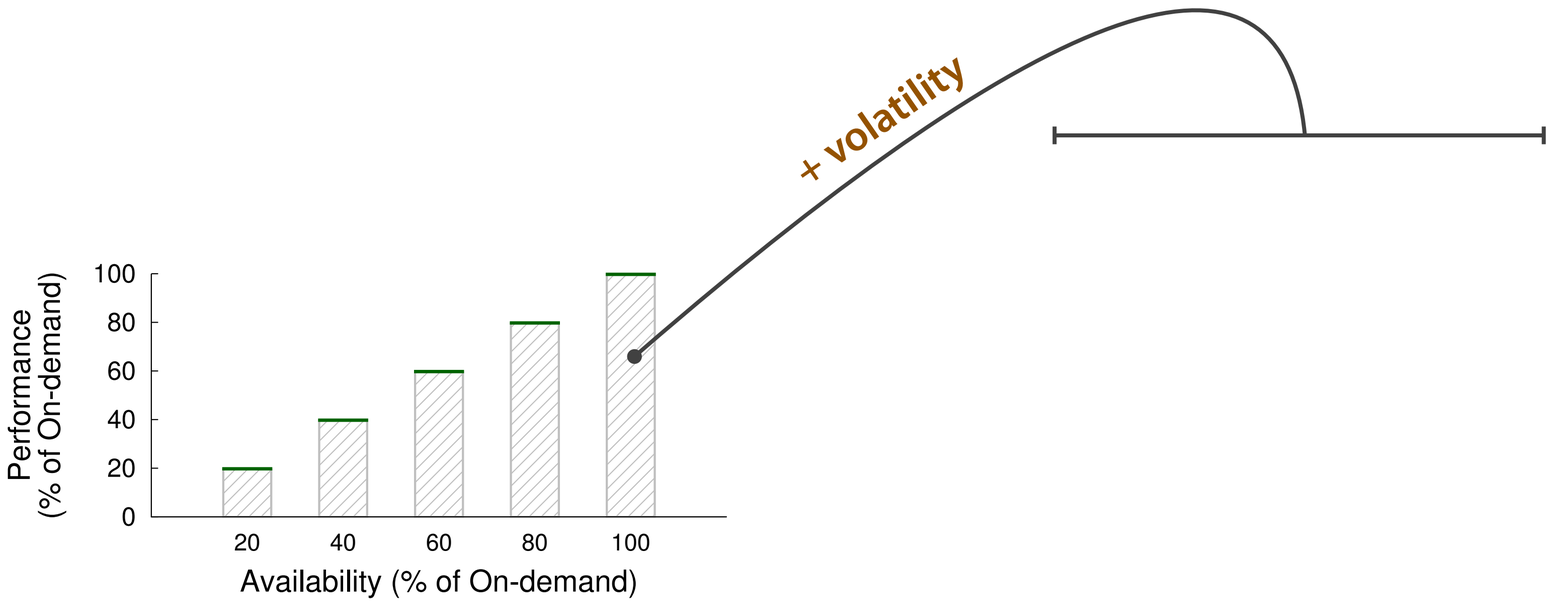
Availability, **VOLATILITY**, *Predictability* affect the performance differently



Availability affects the server performance **linearly**

Distilling the Transient Server Characteristics

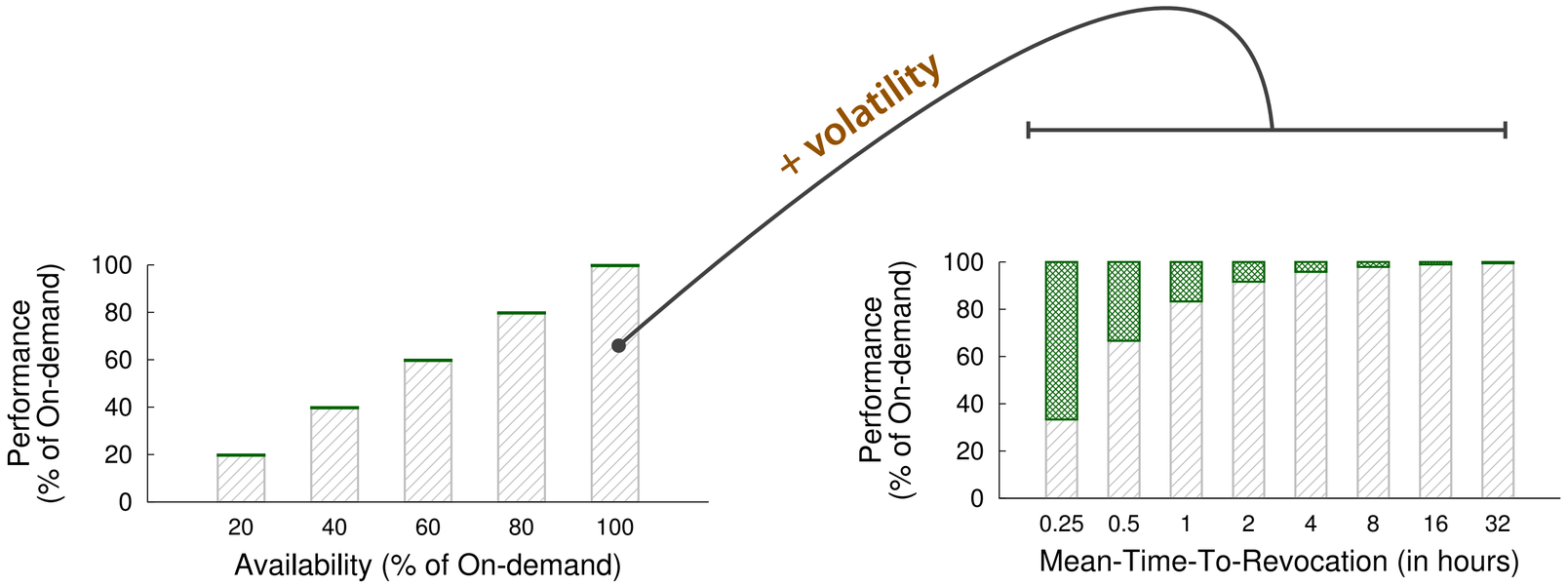
Availability, **VOLATILITY**, *Predictability* affect the performance differently



Availability affects the server performance **linearly**

Distilling the Transient Server Characteristics

Availability, **VOLATILITY**, *Predictability* affect the performance differently

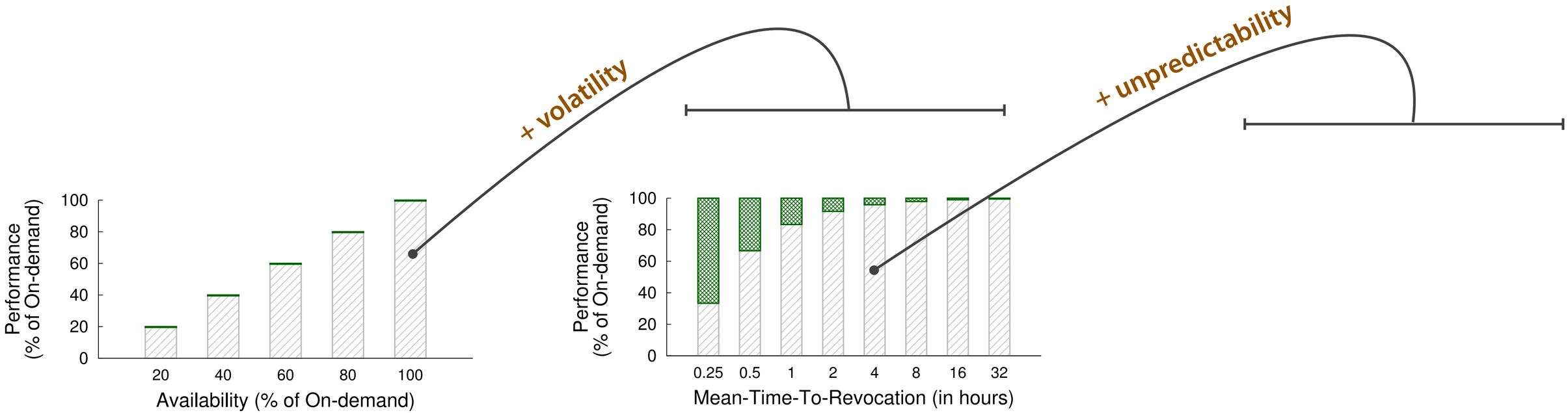


Availability affects the server performance **linearly**

High volatility (~2 rev/hr) can induce overheads of **~35%**

Distilling the Transient Server Characteristics

Availability, **VOLATILITY**, *Predictability* affect the performance differently

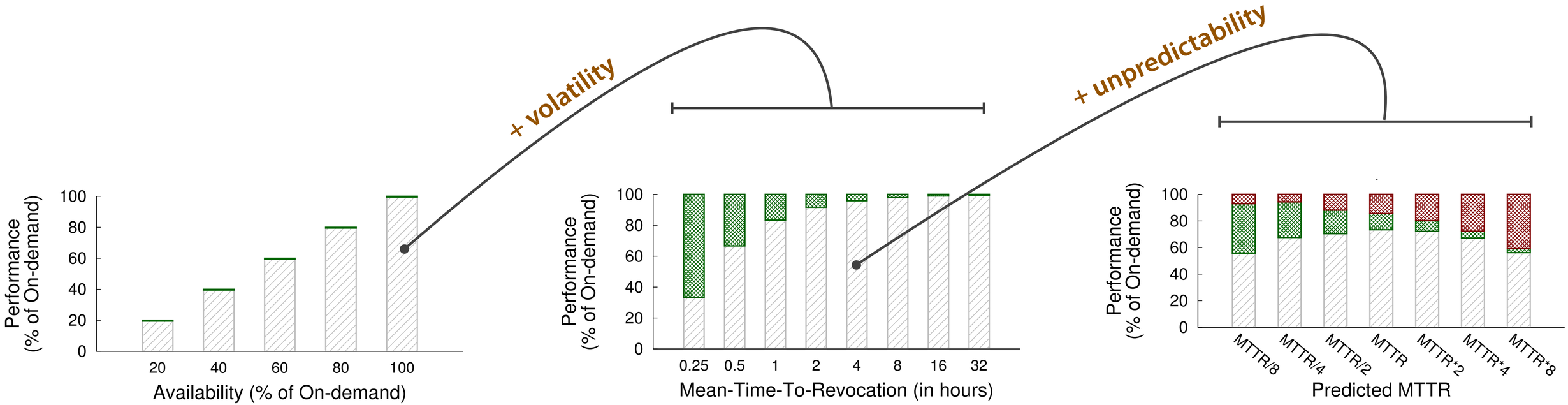


Availability affects the server performance **linearly**

High volatility (~2 rev/hr) can induce overheads of **~35%**

Distilling the Transient Server Characteristics

Availability, **VOLATILITY**, *Predictability* affect the performance differently



Availability affects the server performance **linearly**

High volatility (~2 rev/hr) can induce overheads of **~35%**

Unpredictability incurs overheads of **25-40%**