Cloud Economics

Ian Kash

THE UNIVERSITY OF ILLINOIS AT CHICAGO

Outline of the Tutorial

9:00-10:30

- Economics of a giant pile of compute resources
- Spot markets and reservations

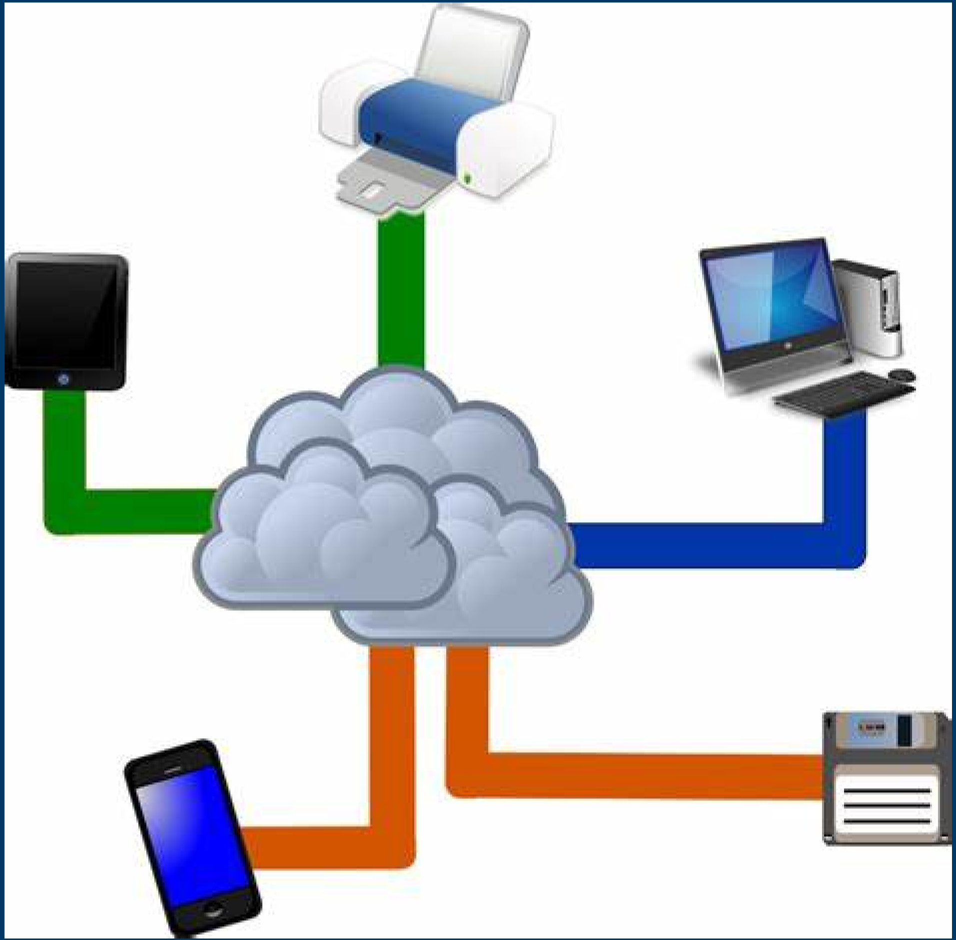
10:30-11:00

- Coffee Break

11-12:30

- Concrete resources & beyond compute
- Future of cloud economics

The Cloud



The promise of the cloud

- Infinite resources!
- Pay only for what you need!

Public Cloud DC

Economics of a Datacenter

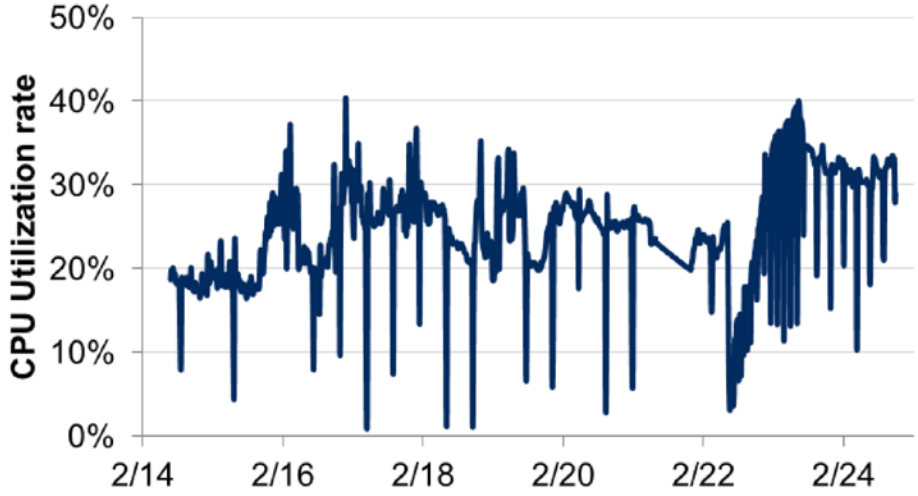
Why have a cloud?

Utilization

- On prem:
  - 5-10% – IDC / VMWare 2009
  - 12-18% - NRDC 2014
  - <20 percent – AWS Blog 2015

Reasons for low utilization

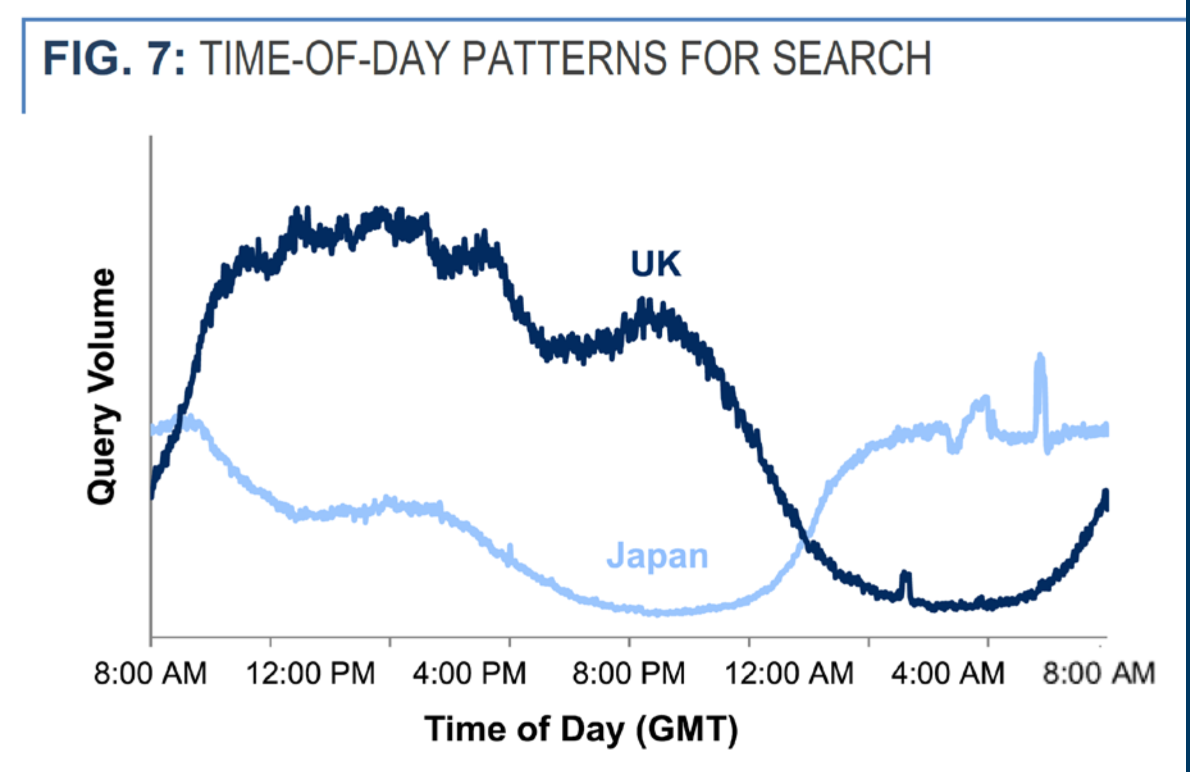
FIG. 6: RANDOM VARIABILITY (EXCHANGE SERVER)



Source: Microsoft.

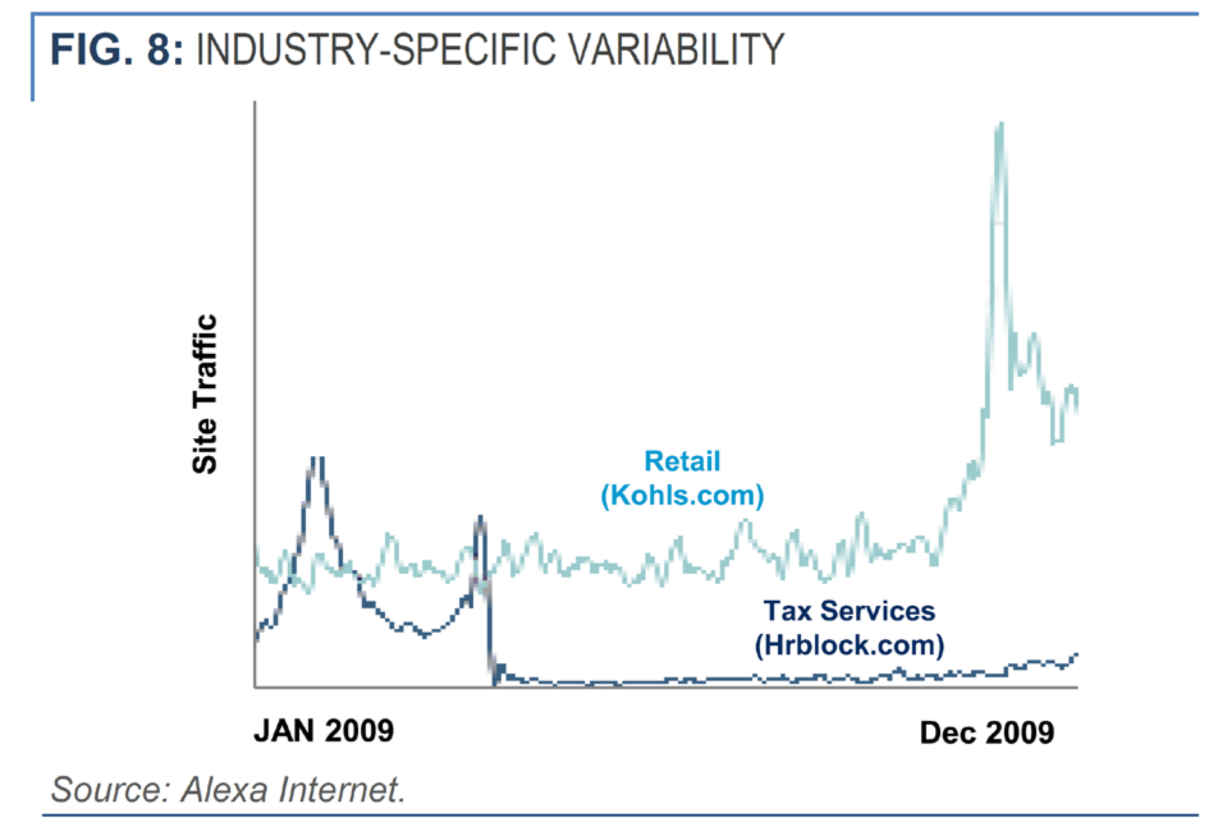
"The Economics of the Cloud" Harms and Yamartino 2010

Reasons for low utilization



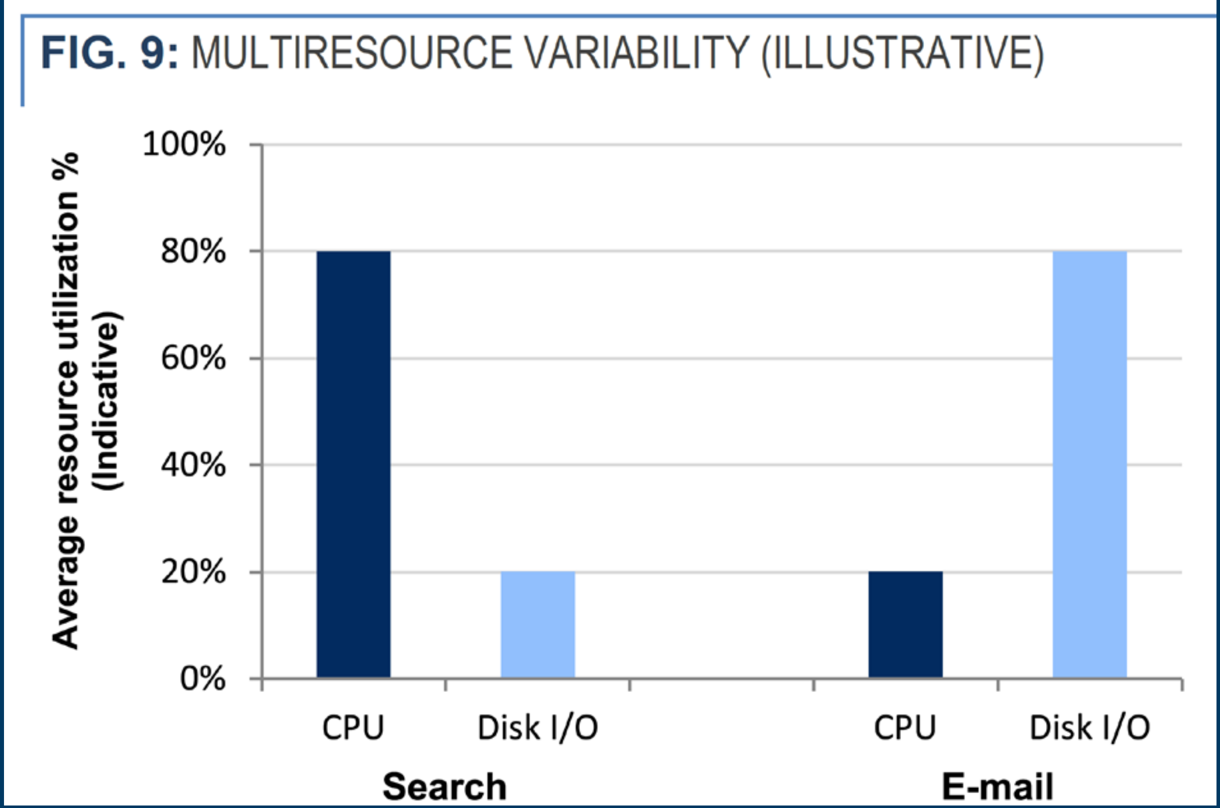
"The Economics of the Cloud" Harms and Yamartino 2010

Reasons for low utilization



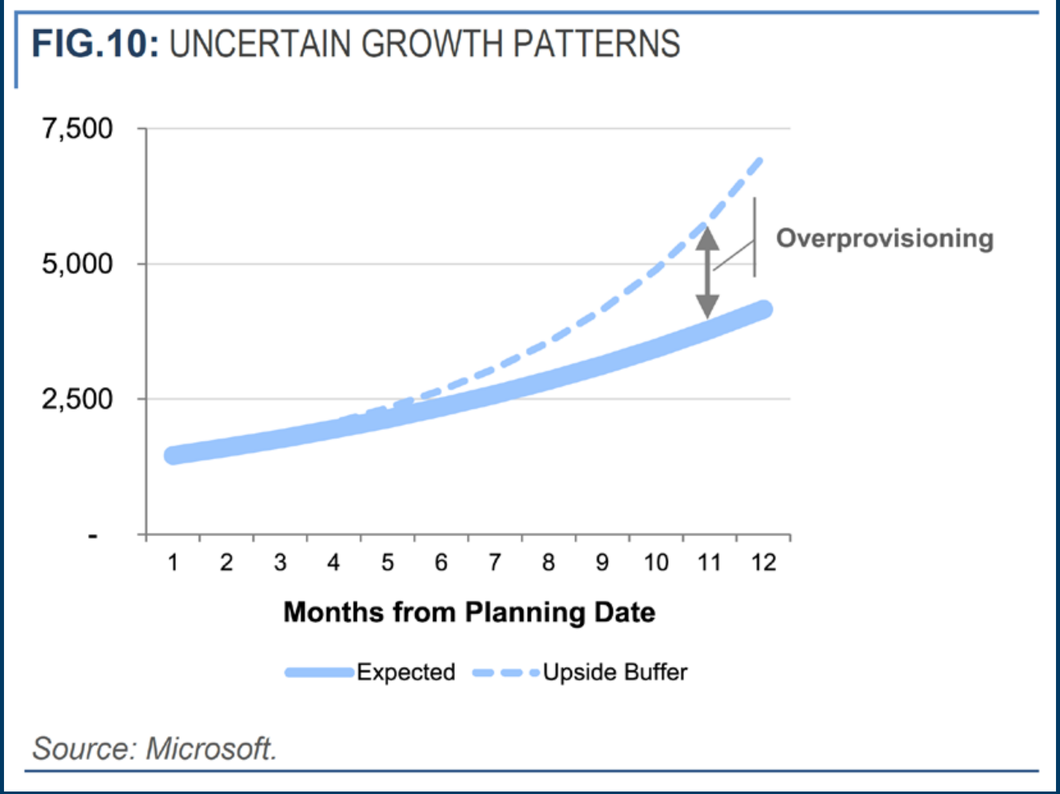
"The Economics of the Cloud" Harms and Yamartino 2010

Reasons for low utilization



"The Economics of the Cloud" Harms and Yamartino 2010

Reasons for low utilization



"The Economics of the Cloud" Harms and Yamartino 2010

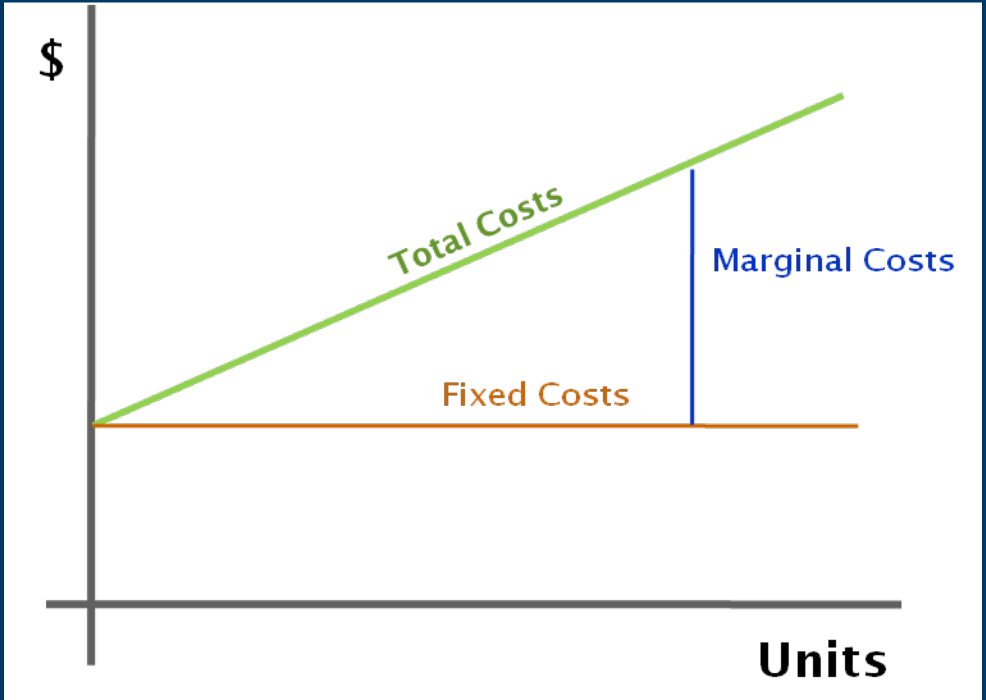
(New) reasons for low utilization

- Not yet in steady state
- Capacity is discrete

The cloud solves utilization?

- Opportunities are real
- But exploiting them requires solving hard problems
  - Coordination
  - Information
  - Re-engineering
  - Pricing
- Better, but still low: AWS claims 65%, but is that billed or real?

Fixed Costs vs Marginal Costs



Cloud Fixed Costs?

- Servers
- Infrastructure: Racks, Cabling, Cooling
- Building
- Land
- Software
- Labor

Cloud Marginal Costs

- Power
- Cooling
- Software Licensing

Economics of Competition

- Bertrand Competition – decide on price
  - Software
  - Cell Phones
  - Restaurants
  - Airlines?
- Cournot Competition – decide on quantity
  - Agriculture
  - Oil
  - Hotel Rooms

Public cloud is profitable!

SEGMENT REVENUE AND OPERATING INCOME

(In millions)(Unaudited)

	Three Months Ended March 31,		Nine Months Ended March 31,	
	2019	2018	2019	2018
Revenue				
Productivity and Business Processes	**$10,242**	$9,006	**$30,113**	$26,197
Intelligent Cloud	**9,649**	7,896	**27,594**	22,613
More Personal Computing	**10,680**	9,917	**34,419**	31,465
Total	**$30,571**	$26,819	**$92,126**	$80,275
Operating Income				
Productivity and Business Processes	**$3,979**	$3,115	**$11,875**	$9,458
Intelligent Cloud	**3,208**	2,654	**9,418**	7,623
More Personal Computing	**3,154**	2,523	**9,261**	7,598
Total	**$10,341**	$8,292	**$30,554**	$24,679

Price Matching

The Power of 'And'

Posted on April 16, 2013

Microsoft Azure

Announcing Infrastructure Services GA and New Price Commitment

Today is an exciting day for Microsoft, Windows Azure and all of our customers around the world. I am very pleased to announce the general availability of Windows Azure Infrastructure Services. This new service now makes it possible for customers to move applications into the cloud. Our announcement today is a significant step in our cloud computing strategy, which has been influenced directly by our discussions with customers and partners around the world. Throughout these conversations, one thing holds true in every discussion - enterprises know that success with the cloud lies in the power of "and." Customers don't want to rip and replace their current infrastructure to benefit from the cloud; they want the strengths of their on-premises investments *and* the flexibility of the cloud. It's not only about Infrastructure as a Service (IaaS) or Platform as a Service (PaaS), it's about Infrastructure Services *and* Platform Services *and* hybrid scenarios. The cloud should be an enabler for innovation, and an extension of your organization's IT fabric, not just a fancier way to describe cheap infrastructure and application hosting Customers have also told me that they don't want to have to choose *either* a low price or good performance; they want a low price *and* good performance. That's why today we are also announcing **a commitment to match Amazon Web Services prices for commodity services such as compute, storage and bandwidth**. This starts with reducing our GA prices on Virtual

Other benefits?

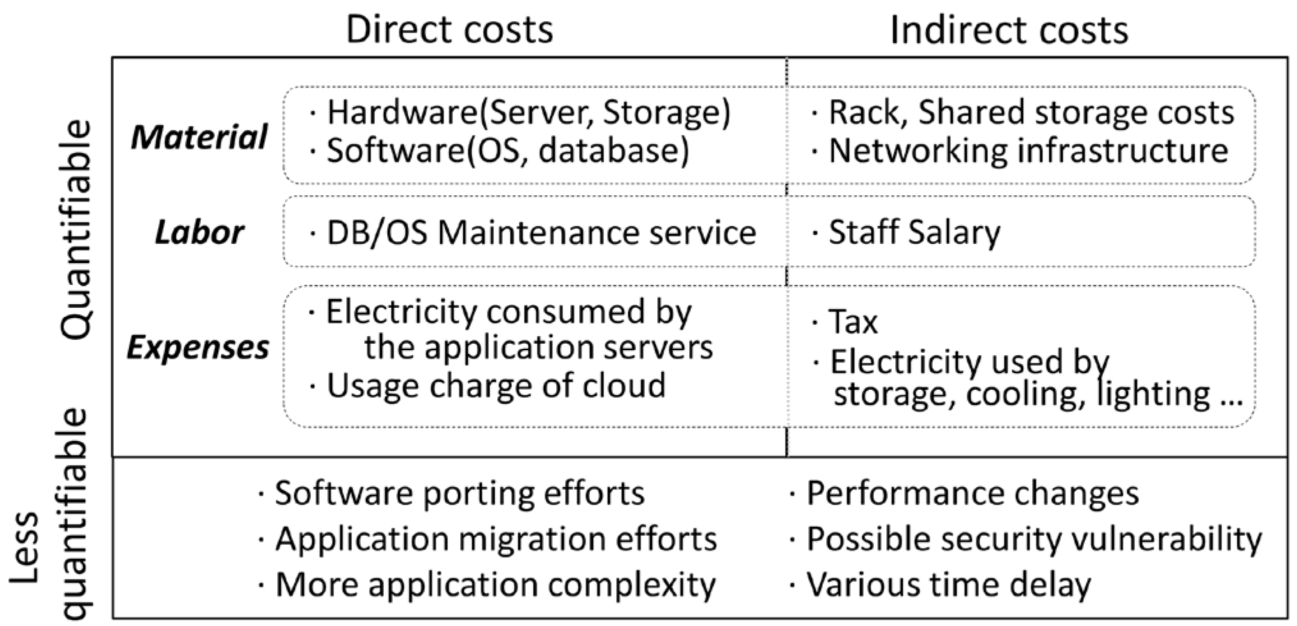


Figure 1: Classification of costs related to migration.

"To Move or Not to Move" Tak, Urgaonkar, Sivasubramaniam 2011

Economies of scale

- Cheaper power / cooling – locate where it is cheap
- Buying power – power, hardware, software, capital, …
- Automation

Renting a VM

MySQL Second Generation pricing

Second Generation pricing is composed of the following charges:

- Instance pricing
- Storage pricing
- Network pricing

Instance Pricing

Instance pricing for Second Generation is charged for every minute that the instance is running (the activation policy is set to `ALWAYS`). The charge depends on the machine type you choose for the instance, and the region where the instance is located. Select your region from the dropdown on the pricing table.

Read replicas and failover replicas are charged at the same rate as stand-alone instances.

Iowa (us-central1) Monthly Hourly

Machine Type	Virtual CPUs	RAM (GB)	Maximum Storage Capacity	Maximum Connections	Price (USD)	Sustained Use Price (USD)
db-f1-micro*	Shared	0.6	3,062 GB	250	$0.0150	$0.0105
db-g1-small*	Shared	1.7	3,062 GB	1,000	$0.0500	$0.0350
db-n1-standard-1	1	3.75	10,230 GB	4,000	$0.0965	$0.0676
db-n1-standard-2	2	7.5	10,230 GB	4,000	$0.1930	$0.1351

Contents

Overview

BigQuery offers scalable, flexible pricing options to meet your technical needs and your budget.

Storage costs are based on the amount of data stored in BigQuery. Storage charges can be:

- Active — A monthly charge for data stored in tables or in partitions that have been modified in the last 90 days.
- Long-term — A lower monthly charge for data stored in tables or in partitions that have not been modified in the last 90 days.

For query costs, you can choose between two pricing models:

- On-demand — This is the most flexible option. On-demand pricing is based on the amount of data processed by each query you run.
- Flat-rate — This predictable pricing option is best for customers with fixed budgets. Flat-rate customers purchase dedicated resources for query processing and are not charged for individual queries.

For more information on storage and query pricing, see Google Cloud Platform SKUs. Note that on-demand query pricing is referred to as analysis pricing on the SKUs page.

Pricing summary

The following table summarizes BigQuery pricing. BigQuery's Quotas and limits apply to these operations.

US (multi-region) Monthly

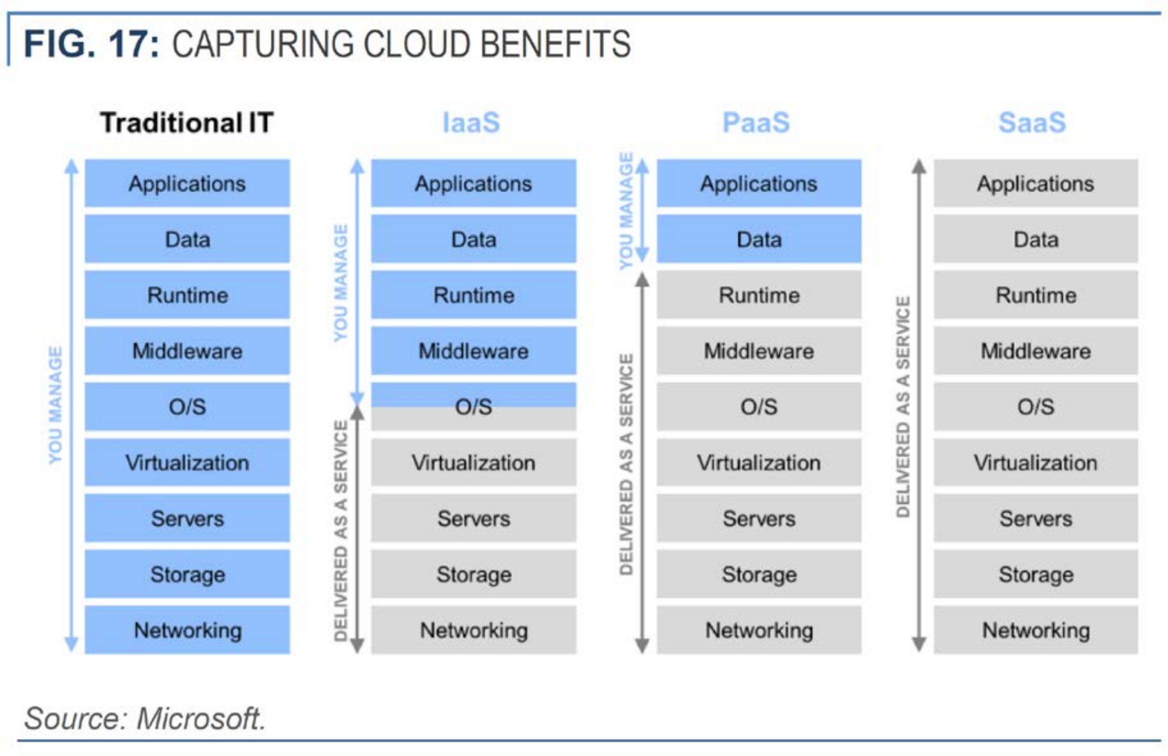
Operation	Pricing	Details

Contents

Hadoop

- IaaS: Get a bunch of VMs and install Hadoop
- PaaS: Amazon EMR
- SaaS?: Cloudera

IaaS vs PaaS vs SaaS



"The Economics of the Cloud" Harms and Yamartino 2010

aws

Contact Sales | Support | English | My Account | Create an AWS Account

Products | Solutions | Pricing | Documentation | Learn | Partner Network | AWS Marketplace | Explore More

Amazon EC2 | Overview | Features | Pricing | Instance Types | FAQs | Getting Started | Resources

On-Demand

With On-Demand instances, you pay for compute capacity by per hour or per second depending on which instances you run. No longer-term commitments or upfront payments are needed. You can increase or decrease your compute capacity depending on the demands of your application and only pay the specified per hourly rates for the instance you use.

On-Demand instances are recommended for:

- Users that prefer the low cost and flexibility of Amazon EC2 without any up-front payment or long-term commitment
- Applications with short-term, spiky, or unpredictable workloads that cannot be interrupted
- Applications being developed or tested on Amazon EC2 for the first time

See On-Demand pricing »

Spot instances

Amazon EC2 Spot instances allow you to request spare Amazon EC2 computing capacity for up to 90% off the On-Demand price. Learn More.

Spot instances are recommended for:

- Applications that have flexible start and end times
- Applications that are only feasible at very low compute prices
- Users with urgent computing needs for large amounts of additional capacity

See Spot pricing »

Reserved Instances

Reserved Instances provide you with a significant discount (up to 75%) compared to On-Demand instance pricing. In addition, when Reserved Instances are assigned to a specific Availability Zone, they provide a capacity reservation, giving you additional confidence in your ability to launch instances when you need them.

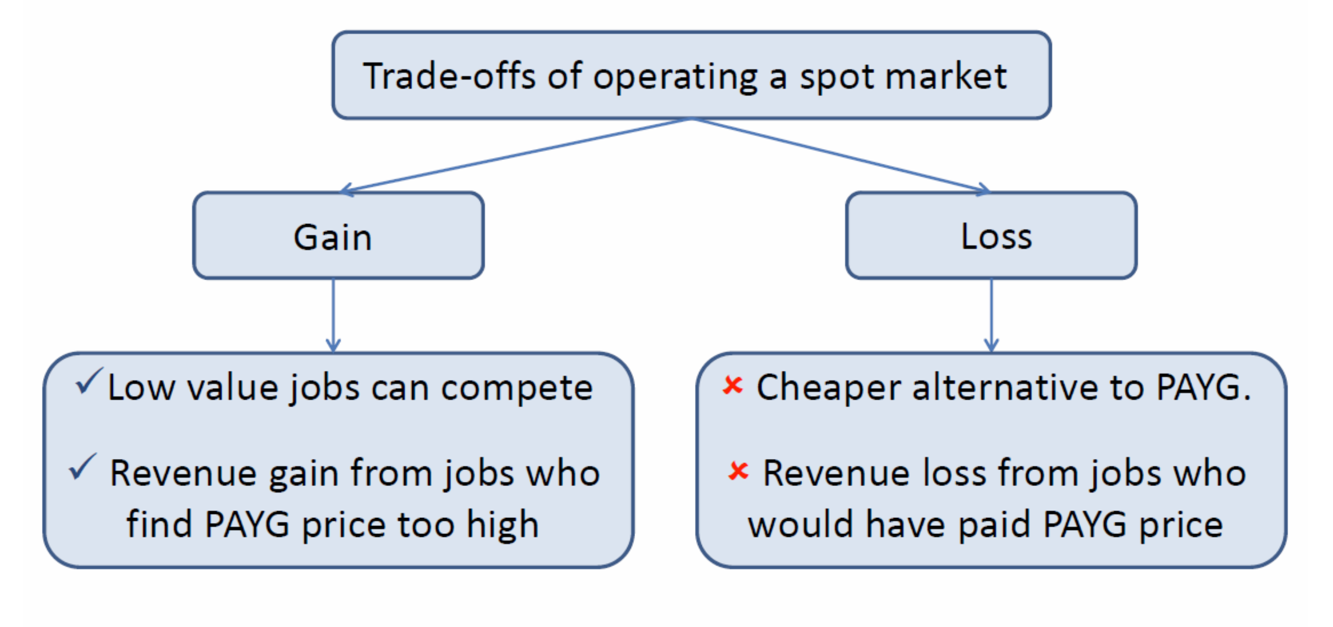
Dedicated Hosts

A Dedicated Host is a physical EC2 server dedicated for your use. Dedicated Hosts can help you reduce costs by allowing you to use your existing server-bound software licenses, including Windows Server, SQL Server, and SUSE Linux Enterprise Server (subject to your license terms), and can also help you meet compliance requirements. Learn more.

Spot Markets

Easy 100% Utilization

Should there be a spot market?



Low prices!

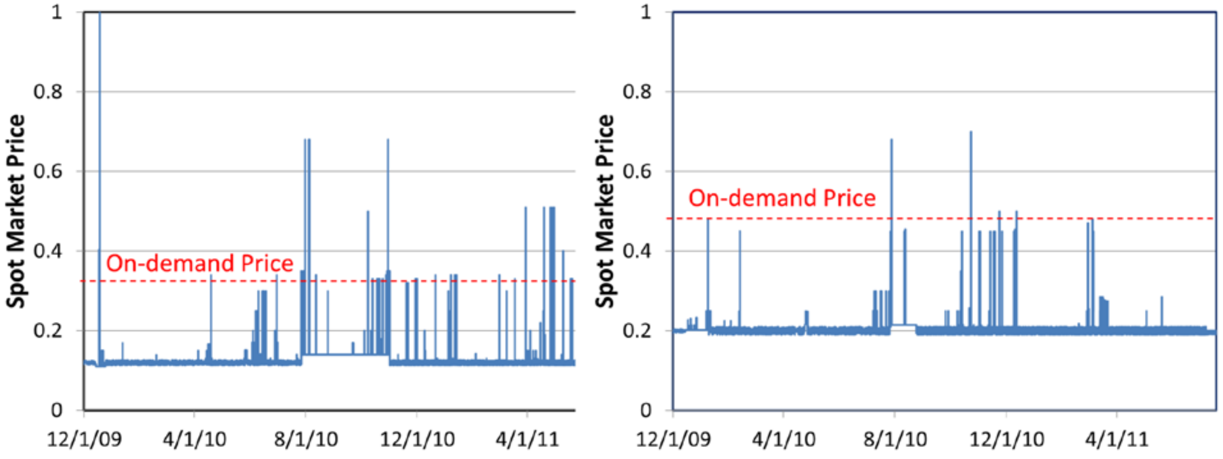


Figure 1: The variation in Amazon EC2 spot market prices for 'large' computing instances in the US East-coast region: Linux (left) and Windows (right). The fixed on-demand price for Linux and Windows instances is 0.34 and 0.48, respectively.

"On-demand, Spot, or Both" Menache, Shamir, Jain 2014

People are Rational

- "On-demand, Spot, or Both" Menache, Shamir, Jain 2014
- "Bidding Strategies for Spot Instances" Karunakaran and Sundarraj 2015
- "Supercloud" van Renesse, Weatherspoon, Shen, Song 2018

spotinst

EXOTANIUM

A Cautionary Tale…

10. EPILOGUE

Amazon's EC2 spot instance pricing mechanism underwent a radical change between the first submission of this paper and its first acceptance. Several days after its acceptance, the spot instance prices underwent another extreme change, and the pricing band disappeared from the traces altogether. For example, in the trace shown in Fig. 14, the spot price is constant throughout October 2011, except for a change in the minimal price. While these radical qualitative changes are further evidence of the former prices being artificially set, the October prices are consistent with a constant minimal price auction, and are no longer consistent with an $AR(1)$ hidden reserve price.

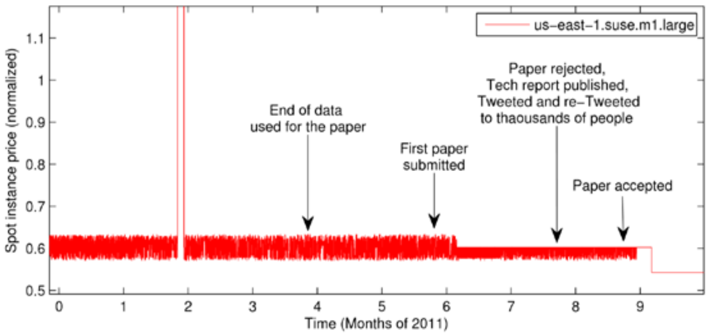


Fig. 14. The history of this paper and the price trace of suse.m1.large on us-east during 2011

"Deconstructing Amazon EC2 Spot Instance Pricing" Ben-Yehuda et al. 2012

Another Cautionary Tale?

Spot markets as price discrimination

1. Model and equilibrium characterization for system with PAYG + Spot

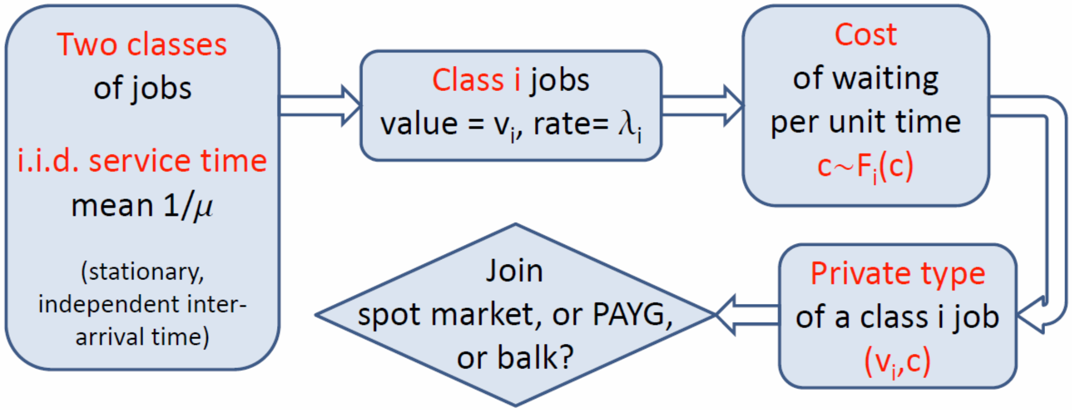
2. Analysis of restricted case showing adding Spot hurts revenue

3. Numerical evidence that suggests this is typically true

“Fixed and Market Pricing for Cloud Services.” Abhishek, Kash, and Key 2012

Model

- Jobs: unit demand, associated with a unique user.



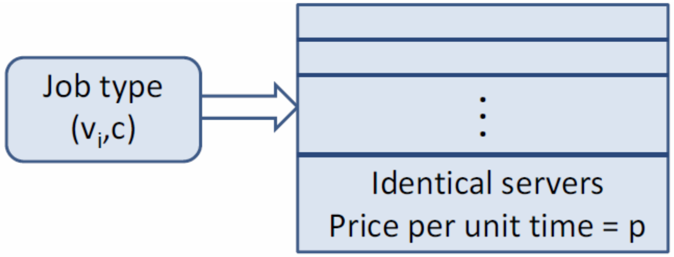
- Payoff = $v_i - cw - m$.

  (Type (v_i, c), waiting time w, payment m) .

"Fixed and Market Pricing for Cloud Services." Abhishek, Kash, and Key 2012

Modeling PAYG

- $GI/GI/\infty$ system, service rate μ.

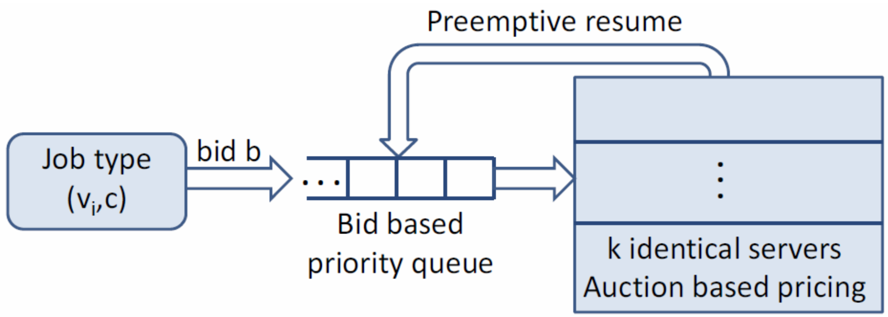


- Waiting time = service time.
- $\mathbb{E}\,[\text{waiting time}] = 1/\mu$, $\mathbb{E}\,[\text{payment}] = p/\mu$.

"Fixed and Market Pricing for Cloud Services." Abhishek, Kash, and Key 2012

Modeling Spot

- $GI/GI/k$ system, service rate μ.



- Waiting time = queuing delay + service time.
- Assume unobservable queue state.

"Fixed and Market Pricing for Cloud Services." Abhishek, Kash, and Key 2012

Equilibrium

Thm: There is a unique equilibrium* where

- All jobs truthfully report their type and cost
- Each type i has a cost cutoff $\bar{c}_i$ s.t.
  - Joins Spot if $c < \bar{c}_i$
  - Joins PAYG or balks otherwise.

*See details in paper

"Fixed and Market Pricing for Cloud Services." Abhishek, Kash, and Key 2012

Impossibly General?

- GI/GI/k
- No specified auction design
  - Assume reserve price is 0
  - Assume priorities are not randomized

“Fixed and Market Pricing for Cloud Services.” Abhishek, Kash, and Key 2012

Impossibly General?

- GI/GI/k
- No specified auction design
  - Assume reserve price is 0
  - Assume priorities are not randomized

Insights from auction theory:

- Can assume bidders just report c
- Waiting time will be decreasing in c
- All that matters is the (expected) delay

"Fixed and Market Pricing for Cloud Services." Abhishek, Kash, and Key 2012

Approach to Theorem

- Given cutoffs $\overline{\mathbf{c}} \triangleq (\overline{c}_1, \overline{c}_2)$:
  - $w(c; \overline{\mathbf{c}}) \triangleq \mathbb{E}$[waiting time in spot market if cost is c].
  - $m(c; \overline{\mathbf{c}}) \triangleq \mathbb{E}$[payment in spot market for if cost is c].
  - Defined for any $\overline{\mathbf{c}}$ for which the queue is stable.
  - $w(c; \overline{\mathbf{c}})$ is decreasing in c, increasing in $\overline{\mathbf{c}}$.
  - $m(c; \overline{\mathbf{c}}) = \int_0^c w(t; \overline{\mathbf{c}})dt - cw(c; \overline{\mathbf{c}})$.

"Fixed and Market Pricing for Cloud Services." Abhishek, Kash, and Key 2012

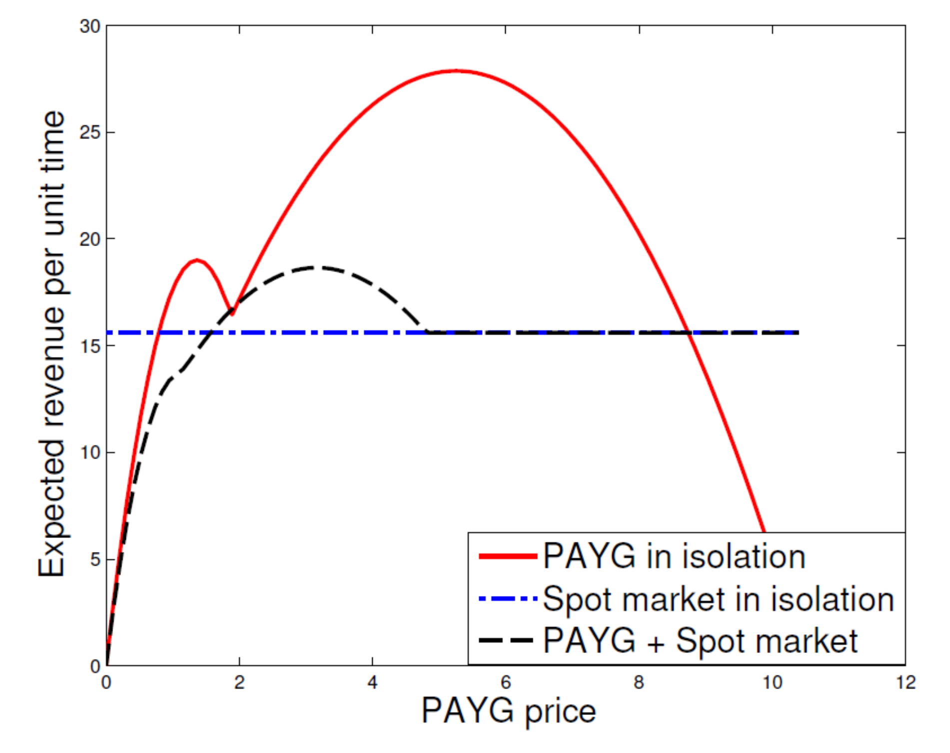
Main Revenue Theorem

Thm: If the revenue maximizing price for PAYG + Spot is low enough that both types participate in PAYG, then:

Revenue(PAYG + Spot) < Revenue(PAYG)

“Fixed and Market Pricing for Cloud Services.” Abhishek, Kash, and Key 2012

Mostly holds in other case too…



“Fixed and Market Pricing for Cloud Services.” Abhishek, Kash, and Key 2012

Related Models

- "To Queue or Not to Queue" Hassin and Haviv 2003
- "Optimal Price and Delay Differentiation in Queueing Systems" Maglaras, Yao, Zeevi 2013
- "On-demand or Spot? Selling the cloud to risk-averse customers" Hoy, Immorlica, Lucier 2016
- "Pricing and bidding strategies for cloud computing spot instances" Song and Guerin 2017
- "The Spot Market Strikes Back" Dierks and Seuken 2018

A Cautionary Tale...

A Cautionary Tale…

- “Paris Metro Pricing for the Internet” Odlyzko 1999
  - Use this style of pricing for network QoS
- “Internet Service Classes Under Competition” Gibbens, Mason, Steinberg 2000
  - Breaks down under competition

Ways to add spot instances

Features/ Cloud	AWS	Azure	Google Cloud	IBM Cloud
Service Name	EC2 Spot Instances	Low Priority VMs	Preemptible VMs	Transient Servers
Pricing	Variable	Fixed	Fixed	Fixed
Shutdown Lead Time	2 Mins	30 Secs	30 Secs	None
Maximum Time Limit	None (depends on extra capacity)	None (depends on extra capacity)	24 Hour Limit and certain instances within 6 hours	None (depends on extra capacity)
Capacity Management	Spot Fleet	No	Instance Groups	No
Cost Visibility	Spot Instance Advisor	Fixed Pricing	Fixed Pricing	Fixed Pricing

Epilogue

Amazon EC2 Spot introduces new pricing model and the ability to launch Spot instances via RunInstances API

Posted On: Nov 28, 2017

Amazon EC2 simplified the Amazon EC2 Spot instance pricing by moving to a model that delivers low, predictable prices that adjust gradually based on long-term trends in supply and demand. You will continue to save up to 90% off the On-Demand instance price and you will continue to pay the Spot price that's in effect at the beginning of each instance-hour for your running instance.

Amazon EC2 Spot now allows you to launch Spot instances via the RunInstances function, run-instances command or AWS management console by simply indicating you want to use Spot. Unlike the old model that required an understanding of Spot markets, bidding and calls to a standalone asynchronous API, the new model is synchronous and as easy to use as On-Demand. To launch a Spot instance from the command line, simply specify **Spot** for **InstanceMarketOption** parameter in the call to run-instances command and you will receive an instance ID immediately if the capacity is fulfilled.

You now have the option to request Spot instances without a bid price. Applications that use Spot and currently submit a bid price will continue to work as is, with no changes.

All the new features are now available in all the Spot supported regions and you can start using it today via the SDK, CLI or AWS console.

To learn more about Spot New pricing model and launching instances via RunInstances, visit the Amazon EC2 Spot page and read the blog post. To learn about how Spot instances work, visit here.

Reservations

Long-term reservations

a1.medium

STANDARD 1-YEAR TERM					
Payment Option	Upfront	Monthly*	Effective Hourly**	Savings over On-Demand	On-Demand Hourly
No Upfront	$0.00	$11.75	$0.016	37%	$0.0255
Partial Upfront	$67.00	$5.62	$0.015	40%	
All Upfront	$131.00	$0.00	$0.015	41%	

CONVERTIBLE 1-YEAR TERM					
Payment Option	Upfront	Monthly*	Effective Hourly**	Savings over On-Demand	On-Demand Hourly
No Upfront	$0.00	$13.50	$0.018	27%	$0.0255
Partial Upfront	$77.00	$6.42	$0.018	31%	
All Upfront	$151.00	$0.00	$0.017	32%	

STANDARD 3-YEAR TERM					
Payment Option	Upfront	Monthly*	Effective Hourly**	Savings over On-Demand	On-Demand Hourly
No Upfront	$0.00	$8.03	$0.011	57%	$0.0255
Partial Upfront	$134.00	$3.72	$0.01	60%	
All Upfront	$252.00	$0.00	$0.01	62%	

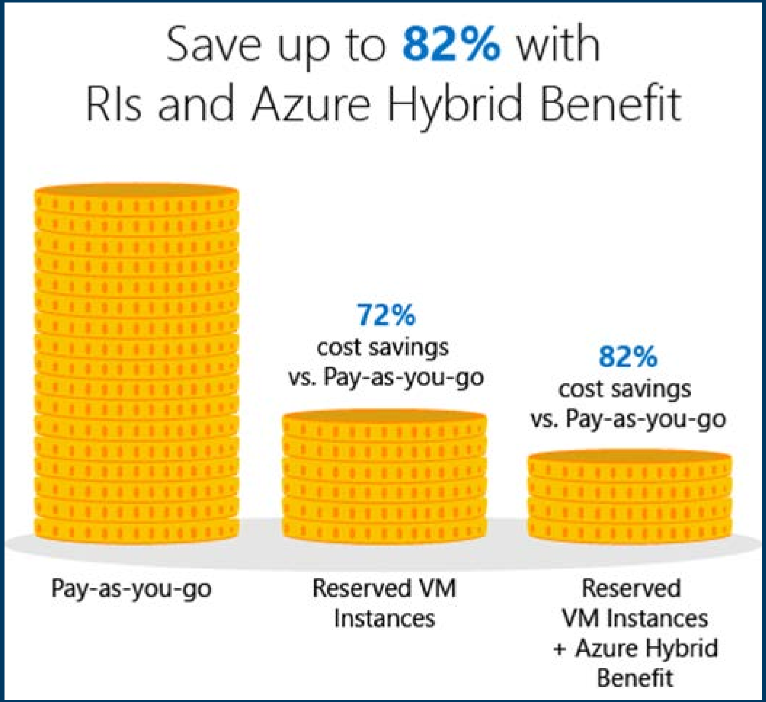
Long-term reservations

Characteristic	Standard	Convertible
Terms (avg. discount off On-Demand)	1yr (40%), 3yr (60%)	1yr (31%), 3yr (54%)
Change Availability Zone, instance size (for Linux OS), networking type	Yes (Using ModifyReservedInstances API and console)	Yes (Using ExchangeReservedInstances API and console)
Change instance families, operating system, tenancy, and payment option		Yes
Benefit from Price Reductions		Yes
Sellable on the Reserved Instance Marketplace	Yes (After linking account with a US bank account)	Coming soon

Standard and Convertible RI Payment Attributes

- **Offering class:** There are two classes of RIs: Convertible and Standard. Convertible RIs can be exchanged for different Convertible RIs of equal or greater value.
- **Term:** AWS offers Standard RIs for 1-year or 3-year terms. Reserved Instance Marketplace sellers also offer RIs often with shorter terms. AWS offers Convertible RIs for 1-year or 3-year terms.
- **Payment option:** You can choose between three payment options: All Upfront, Partial Upfront, and No Upfront. If you choose the Partial or No Upfront payment option, the remaining balance will be due in monthly increments over the term.

Long-term reservations



Part-time Reservations

Spot Instances | Defined Duration for Linux | **Defined Duration for Windows**

Region: US East (Ohio)

	1 hour	6 hours
General Purpose - Current Generation		
m4.large	$0.142 per Hour	$0.157 per Hour
m4.xlarge	$0.284 per Hour	$0.314 per Hour
m4.2xlarge	$0.568 per Hour	$0.628 per Hour
m4.4xlarge	$1.136 per Hour	$1.256 per Hour
m4.10xlarge	$2.84 per Hour	$3.14 per Hour
m4.16xlarge	$4.544 per Hour	$5.024 per Hour

	Windows (Peak Hours)	Windows (Off-Peak Hours)
General Purpose - Current Generation		
m4.large	$0.24 per Hour	$0.234 per Hour
m4.xlarge	$0.479 per Hour	$0.467 per Hour
m4.2xlarge	$0.959 per Hour	$0.935 per Hour
m4.4xlarge	$1.918 per Hour	$1.87 per Hour
m4.10xlarge	$4.794 per Hour	$4.675 per Hour

Length-based Pricing - Model

- One server
- One job arrives per time period
- Jobs want to use the server for 1+ time periods
- Shared value per unit time distribution

"Simple Pricing Schemes for the Cloud" Kash, Key, Suksompong 2017

Length-based Pricing - Options

- Complex: one price per job length
- Simple: on price per unit time
- Simpler: that price is chosen from among those used by the complex policy

"Simple Pricing Schemes for the Cloud" Kash, Key, Suksompong 2017

Length-based Pricing - Results

- Simpler pricing gets at least 50% of the benefits
- This is tight
- Simple pricing does too under somewhat less restrictive assumptions but only with optimal pricing

"Simple Pricing Schemes for the Cloud" Kash, Key, Suksompong 2017

Length-based Pricing - Intuitions

- Longer jobs have higher opportunity cost
- With 2 lengths: low price gets at least the revenue from the short jobs and high price from the long ones
  - One of these must be half the revenue
- With >2 lengths: more careful about revenue from other lengths

"Simple Pricing Schemes for the Cloud" Kash, Key, Suksompong 2017

Online Scheduling

Each job has:

- An arrival time a_j
- A duration l_j
- A deadline d_j
- A value v_j with density $\rho_j = \frac{v_j}{l_j}$

$$\mathrm{cr}(\mathcal{A}) \leq \begin{cases} 3 + O\left(\frac{1}{(s-1)^2}\right) & 1 < s < 2 \\ 2 + O\left(\frac{1}{\sqrt[3]{s}}\right) & s \geq 2 \end{cases}$$

Key assumption:

- Slack parameter s: $d_j - a_j \geq s \cdot l_j$

"Efficient Online Scheduling for Deadline-Sensitive Jobs." Lucier et al. 2013

Online Scheduling

Algorithm 1: Single Server Algorithm $\mathcal{A}$

Event: On arrival of job j at time $t = a_j$:

1. Call the threshold preemption rule.

Event: On job completion at time t:

1. Resume execution of the preempted job with highest value-density.
2. Call the threshold preemption rule.

Threshold Preemption Rule (t)**:**

1. $j \leftarrow$ job currently being processed.
2. $j^* \leftarrow \arg\max \{\rho_{j^*} \mid j^* \in A^{-\mu}(t)\}$.
3. if $(\rho_{j^*} > \gamma \cdot \rho_j)$
   3.1. Preempt j and run j^*.

"Efficient Online Scheduling for Deadline-Sensitive Jobs." Lucier et al. 2013

Making this Truthful

ALGORITHM 1: Truthful Non-Committed Algorithm $\mathcal{A}_T$ for a Single Server

$\forall t,\quad J^P(t) = \{ j \in \mathcal{J} \mid j$ partially processed by $\mathcal{A}_T$ at time $t \wedge t \in [a_j, d_j]\}$.

$J^E(t) = \{ j \in \mathcal{J} \mid j$ unallocated by $\mathcal{A}_T$ at time $t \wedge t \in [a_j, d_j - \mu D_j]\}$.

Event: On arrival of job j at time $t = a_j$:

1. call ClassPreemptionRule(t).

Event: On completion of job j at time t:

1. resume execution of job $j' = \arg\max \{\rho_{j'} \mid j' \in J^P(t)\}$.
2. call ClassPreemptionRule(t).
3. delay the output response of j until time d_j.

ClassPreemptionRule **(t):**

1. $j \leftarrow$ job currently being processed.
2. $j^* \leftarrow \arg\max \{\rho_{j^*} \mid j^* \in J^E(t)\}$.
3. if $(j^* \succ j)$:
   3.1. preempt j and run j^*.

"Truthful Online Scheduling with Committtments." Azar et al. 2015

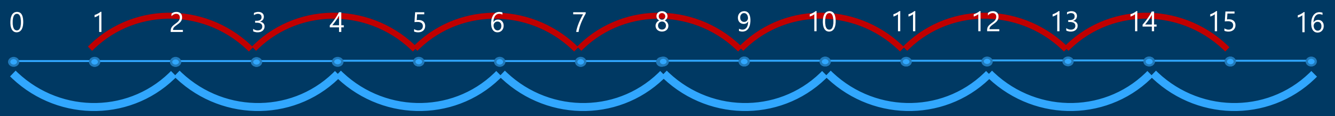
Stochastic online scheduling

- At each time $t \in \{0, \ldots, T\}$, a job j is realised from the known distribution D_t.
- We have to accept or reject the job right away.

Theorem: There is a mechanism for stochastic online scheduling on a single machine with uniform lengths that gives a 4 approximation in expectation

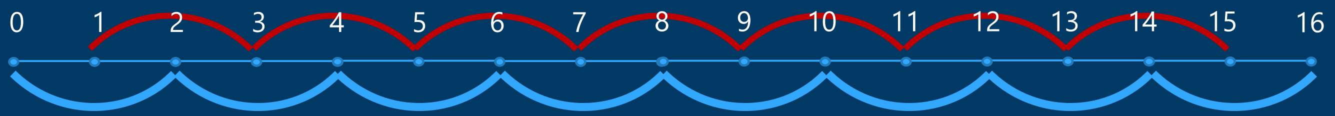
Trick #1: Discretization

- Assume every job has the same length l.
- We partition the time into time slots of size $2l$.
- We consider two partitions: even partition (blue) and odd partition(red).



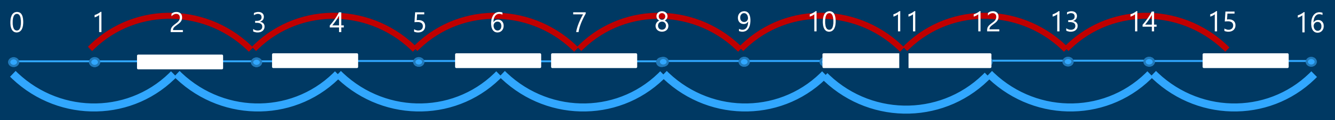
Trick #1: Discretization

- Claim: Given a subset of scheduled jobs S,
  there is a matching from each job in S to exactly one partition.



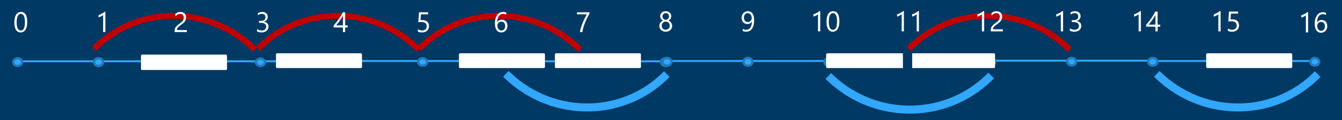
Trick #1: Discretization

- Claim: Given a subset of scheduled jobs S, there is a matching from each job in S to exactly one partition.



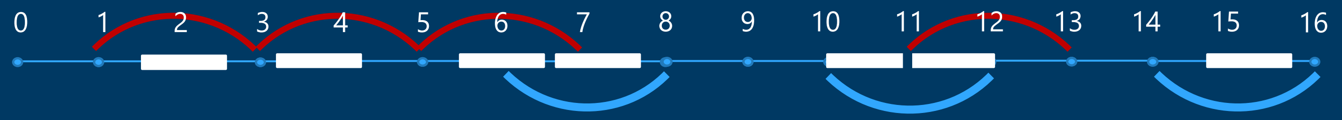
Trick #1: Discretization

- Claim: Given a subset of scheduled jobs S,
 there is a matching from each job in S to exactly one partition.



Trick #1: Discretization

- Choose one of the partitions randomly.
- The value we get is exactly half in expectation.



Trick #2: Expected LPs

$$\begin{aligned} &\text{maximize } c \cdot x \\ &\text{subject to} \\ &A \cdot x \leq b \\ &x \geq 0 \end{aligned}$$

$$\begin{aligned} &\text{maximize } c \cdot x \\ &\text{subject to} \\ &A \cdot x \leq E[b] \\ &x \geq 0 \end{aligned}$$

Thm [DJSW11,AHL12]: The value of the right LP is an upper bound on the expected value of the left LP

Trick #3: Prophet Inequalities

$U[0,2]$ $U[0,1]$ $U[0,2]$

Trick #3: Prophet Inequalities

$U[0,2]$

$U[0,1]$

$U[0,2]$

Trick #4: Bellman Equation

- AKA Dynamic Programming
- Calculate a price for each time slot at each time

Mechanism

Algorithm 1 PRICING

Offline Process:

1: $\mathcal{S} \in \{\mathcal{S}_1, \mathcal{S}_2\}$ uniformly at random.
2: $x^* \leftarrow$ an optimal solution of $ELP(\mathbb{IP}1)$.
3: Recursively Compute $H_{s,t} = \sum_{j \in \mathcal{J}} x^*_{sjt} \max\{H_{s,t+1}, u_{sj}\}$ for every time slot $s \in \mathcal{S}$ and time $0 \leq t \leq T$.

Online Scheme; assuming the current time is t **and job** $j = \langle a_j, l_j, d_j, v_j \rangle$ **is arrived:**

1: $\theta(s) \leftarrow H_{s,t+1}$ if the slot has not been allocated and ∞ otherwise,
2: **if** $v_j \geq \min_s \theta(s)$ **then**
3: Schedule j at minimum price time slot
4: **else**
5: Reject j.

Length Heterogeneity

- Assume $l_j \in \{1, \dots, L\}$.
- Consider $\mathbf{log}(L)$ layers / servers.
- k^{th} layer is responsible for jobs with length $2^{k-1} \leq l < 2^k$.
- In each layer the ratio of the longest job to the shortest job is at most 2.

Value Heterogeneity

- Assume $v_j \in [1, V]$.
- Do the same trick.
- Consider $\log(V)$ layers / servers.
- k^{th} layer is responsible for jobs with value $2^{k-1} \leq v < 2^k$.
- In each layer the ratio of the highest valued job to the lowest valued job is at most 2.

Algorithm

Price of the machine

Duration of the job	1$	2$	4$	8$	16$	.	.	$2^{[\log(V)]}$$
[1, 2)								
[2, 4)								
[4, 8)								
[8, 16)								
.								
.								
.								
$[2^l, 2^{l+1})$								

Algorithm

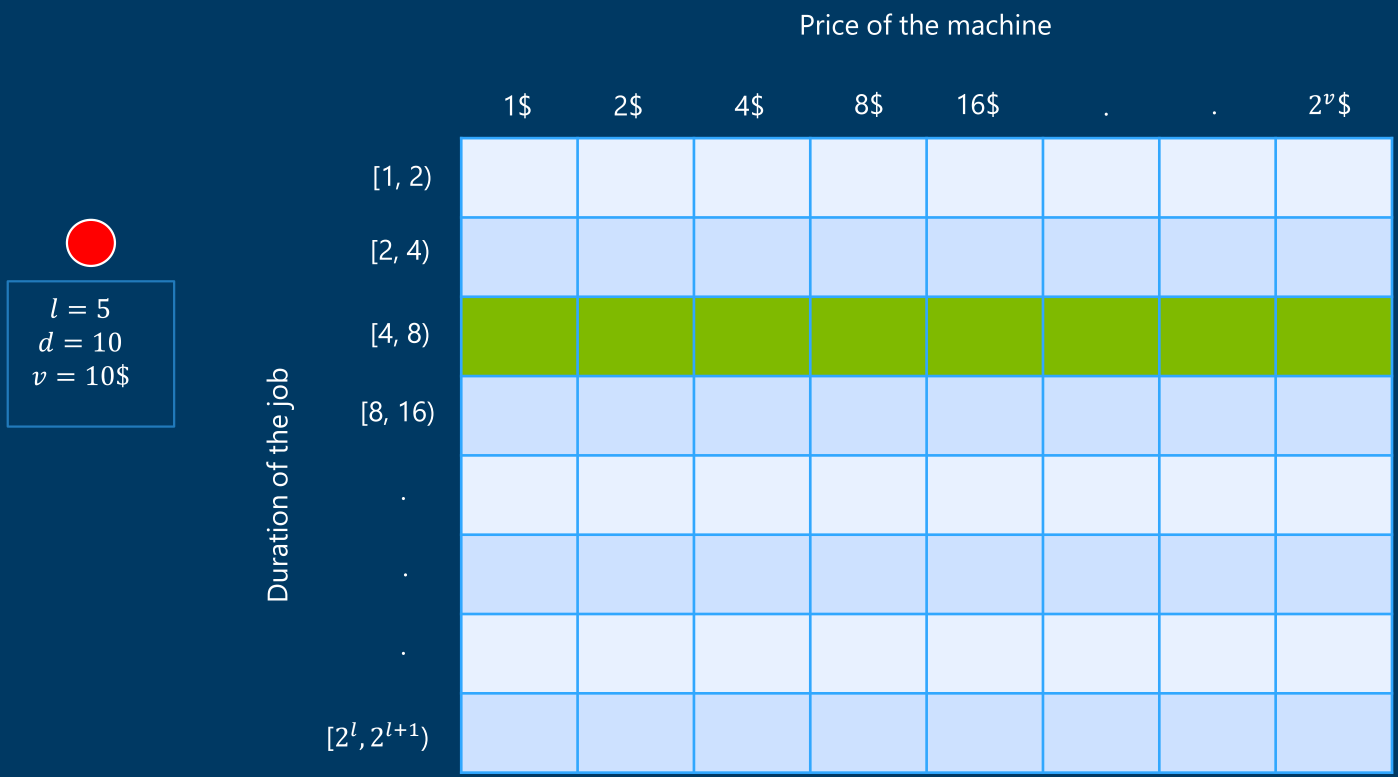
$l = 5$
$d = 10$
$v = 10\$$

Price of the machine

Duration of the job

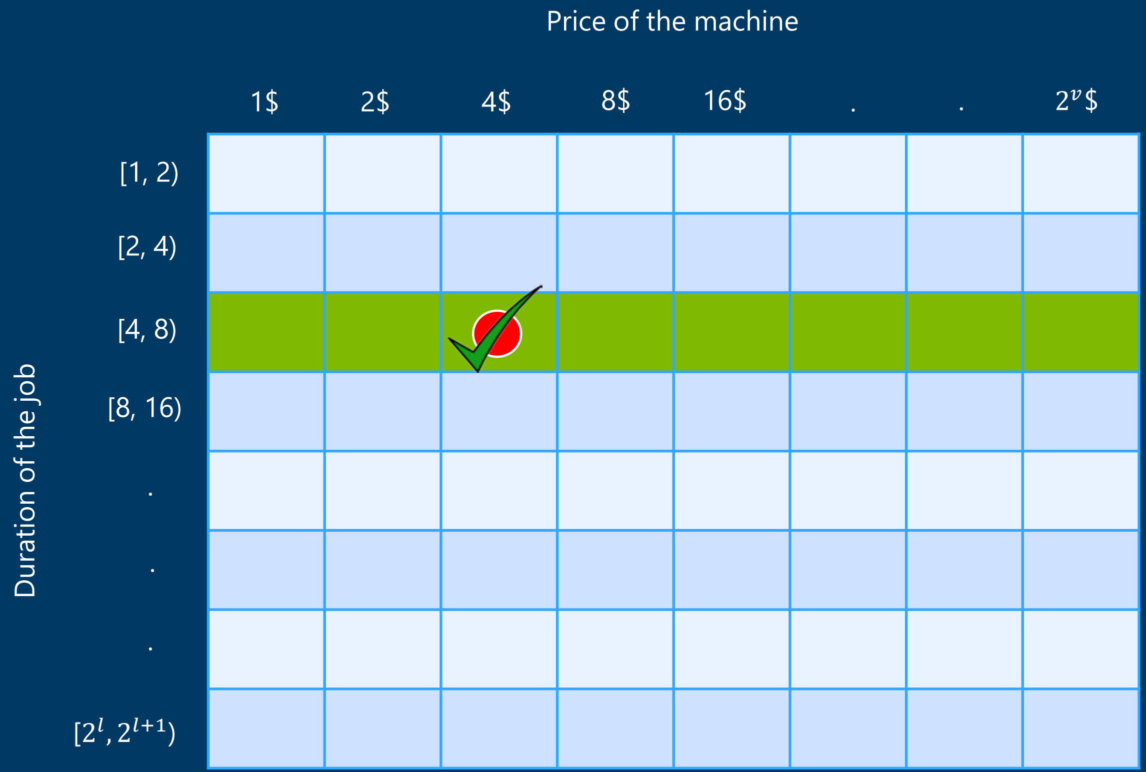
	1$	2$	4$	8$	16$	.	.	$2^v$$
[1, 2)								
[2, 4)								
[4, 8)								
[8, 16)								
.								
.								
.								
$[2^l, 2^{l+1})$								

Algorithm


$l = 5$
$d = 10$
$v = 10\$$
Price of the machine
1$
2$
4$
8$
16$
.
.
$2^v\$$
Duration of the job
[1, 2)
[2, 4)
[4, 8)
[8, 16)
.
.
.
$[2^l, 2^{l+1})$

Algorithm

$l = 5$
$d = 10$
$v = 10\$$



Other Related Work

- “A Truthful Mechanism for Value-Based Scheduling in Cloud Computing” Jain et al. 2017
- “Truth and Regret in Online Scheduling” Chawla et al. 2017
- “Stability of Service under Time-of-Use Pricing” Chawla et al. 2017
- “Selling reserved instances in cloud computing” Wang et al. 2015

ERA

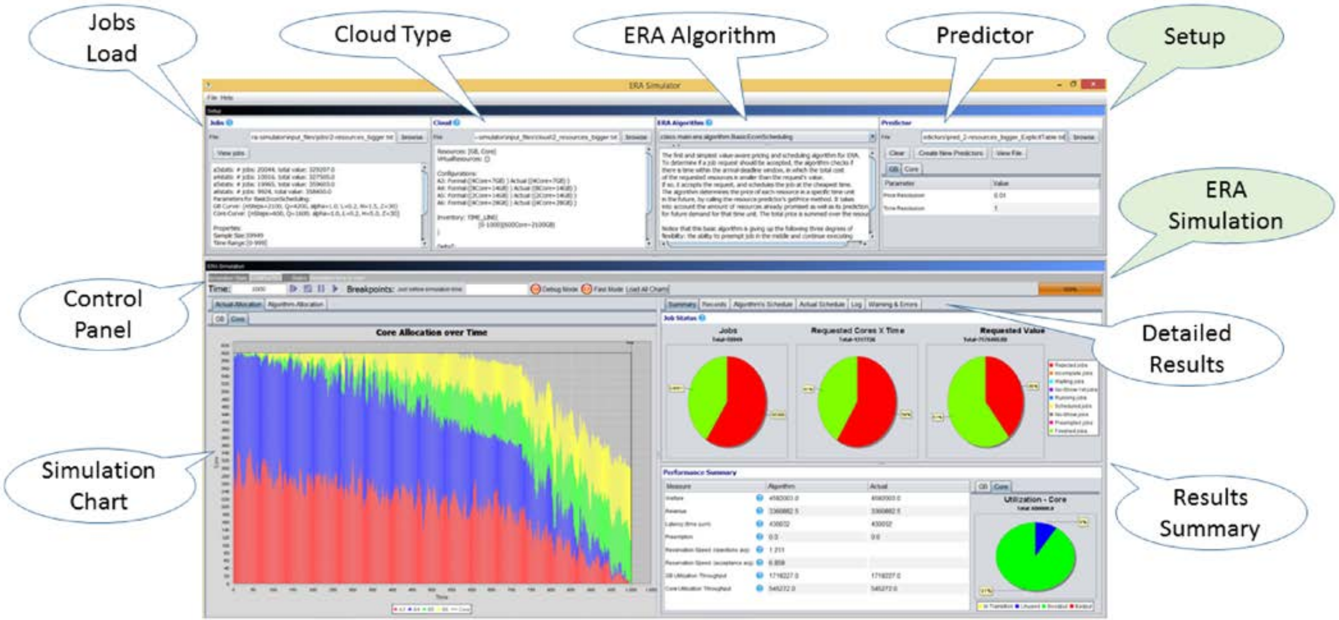
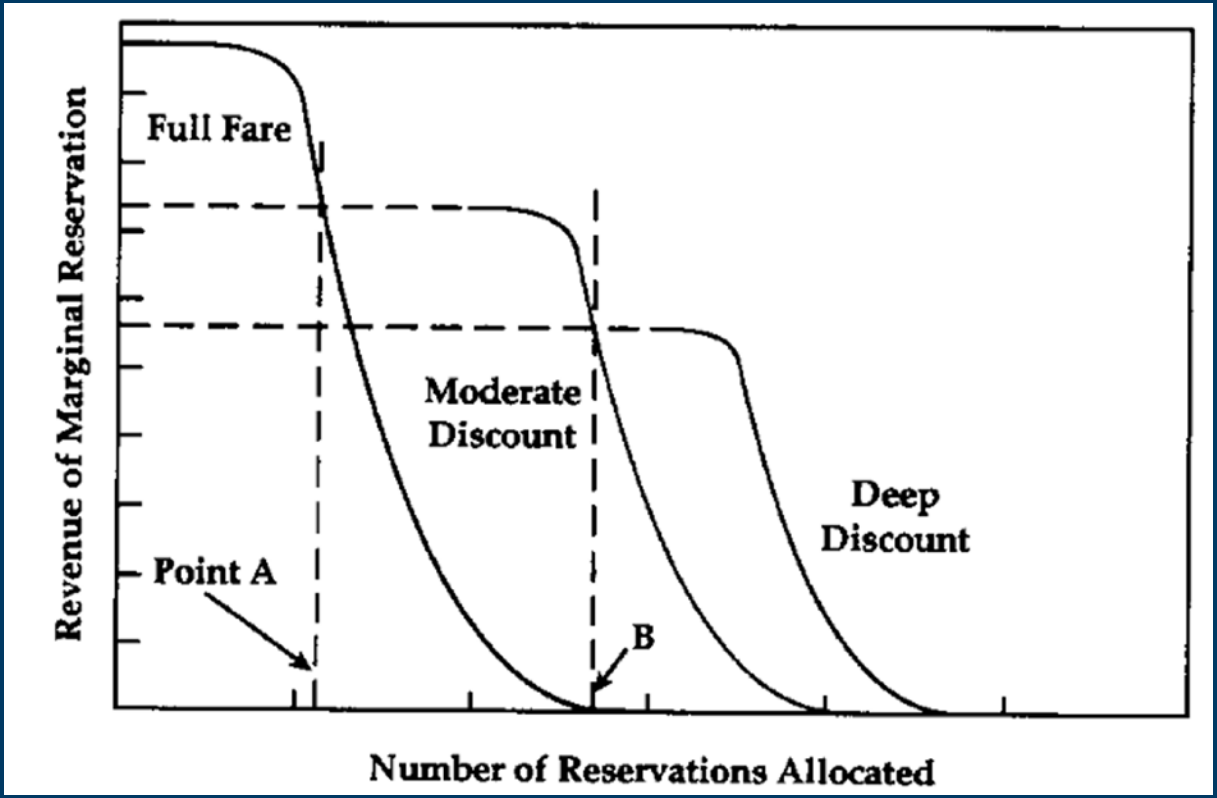


Figure 2: ERA Simulator Screenshot

"ERA: A Framework for Economic Resource Allocation for the Cloud" Babaioff et al. 2017

Airline Pricing?



"Yield Management at American Airlines" Smith, Leimkuhler, Darrow 1992

Coffee!

Cloud Economics

Ian Kash

Part II:
Beyond Abstract Compute

Cluster Scheduling

Public Cloud DC

Lots of Options

VM Series

Type	Sizes	Description
General purpose	B, Dsv3, Dv3, DSv2, Dv2, Av2, DC	Balanced CPU-to-memory ratio. Ideal for testing and development, small to medium databases, and low to medium traffic web servers.
Compute optimized	Fsv2, Fs, F	High CPU-to-memory ratio. Good for medium traffic web servers, network appliances, batch processes, and application servers.
Memory optimized	Esv3, Ev3, M, GS, G, DSv2, Dv2	High memory-to-CPU ratio. Great for relational database servers, medium to large caches, and in-memory analytics.
Storage optimized	Lsv2, Ls	High disk throughput and IO ideal for Big Data, SQL, NoSQL databases, data warehousing and large transactional databases.
GPU	NV, NVv2, NC, NCv2, NCv3, ND, NDv2 (Preview)	Specialized virtual machines targeted for heavy graphic rendering and video editing, as well as model training and inferencing (ND) with deep learning. Available with single or multiple GPUs.
High performance compute	H	Our fastest and most powerful CPU virtual machines with optional high-throughput network interfaces (RDMA).

Generations

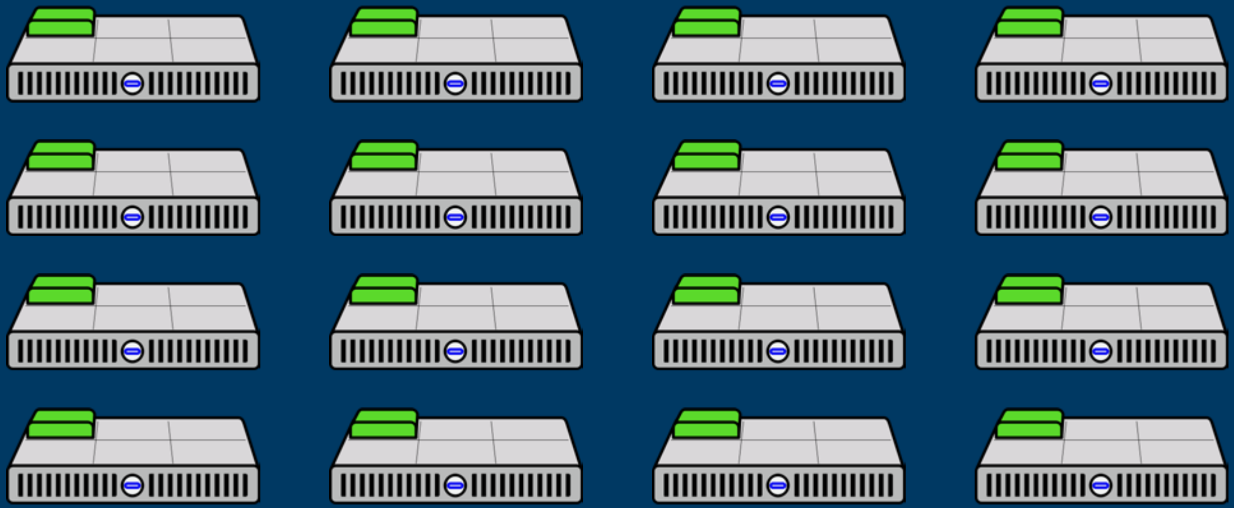
Previous Generation Instances

AWS offers Previous Generation Instances for users who have optimized their applications around these instances and have yet to upgrade. Previous Generation Instances are still fully supported and retain the same features and functionality. Previous Generation Instances are available through the AWS Management Console, AWS CLI, and EC2 API tools.

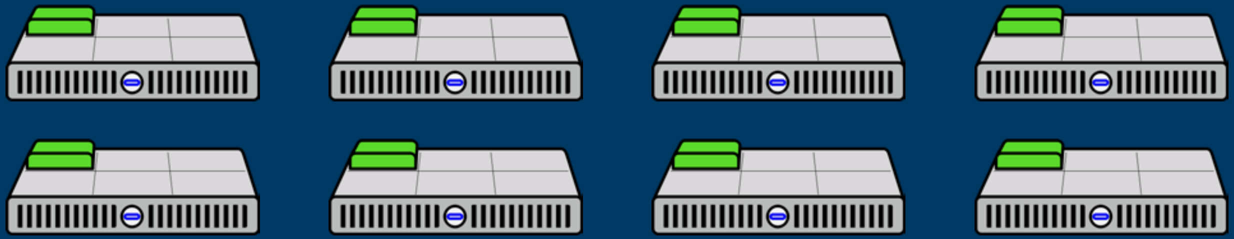
Cluster Scheduling Contraints

- Heterogeneous Clusters
- What to do about old generations on new CPUs?
  - Underclock?
  - Share cores?
  - Unreliable Performance?
- Failure Domains
- Fragmentation
  - Cores
  - Memory
  - Specialized Hardware

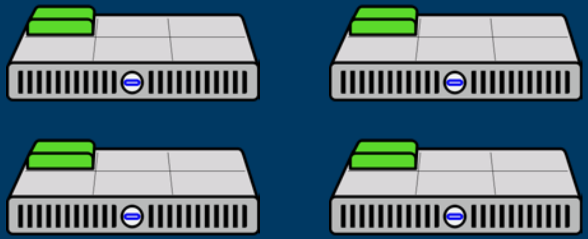
Fragmentation



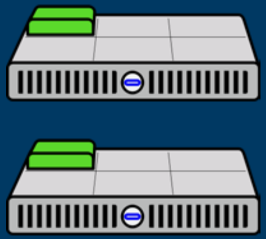
Fragmentation



Fragmentation



Fragmentation



Fragmentation

Work on Cluster Scheduling

- “Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms” Cortez et al. 2017
- “More Than Bin Packing: Dynamic Resource Allocation Strategies in Cloud Data Centers.” Wolke et al. 2015

When to Introduce Next Generation?

- Technology improves at a linear rate with time
- Users live for $\mathbf{2}$, have value $\boldsymbol{\theta t}$ for time $\boldsymbol{t}$ technology
- $\boldsymbol{\theta \sim F}$ with monotone hazard rate
- New generations cost $\boldsymbol{C}$ to introduce, $\boldsymbol{c}$ to adopt

"Optimal Pricing and Introduction Timing of New Virtual Machines" Kash, Key, Zoumpoulis 2018

Myerson Pricing

- Revenue of only offering technology t:
$$\left(1 - F(p)\right)pt$$
- Charge optimal price p^*:
$$p^* = (1\ - F(p^*))/p^*$$
- Do this for every technology

"Optimal Pricing and Introduction Timing of New Virtual Machines" Kash, Key, Zoumpoulis 2018

Myerson Pricing => Periodic Introductions

- New customers choose the latest technology
- Existing customers may switch, depending on the *time since last introduction*
- If we instead assume periodic introductions, this also shows Myerson is asymptotically optimal

"Optimal Pricing and Introduction Timing of New Virtual Machines" Kash, Key, Zoumpoulis 2018

With arbitrary introductions

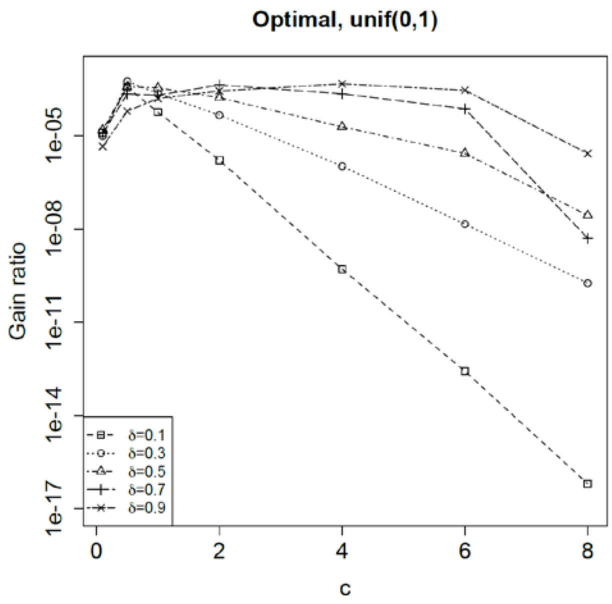
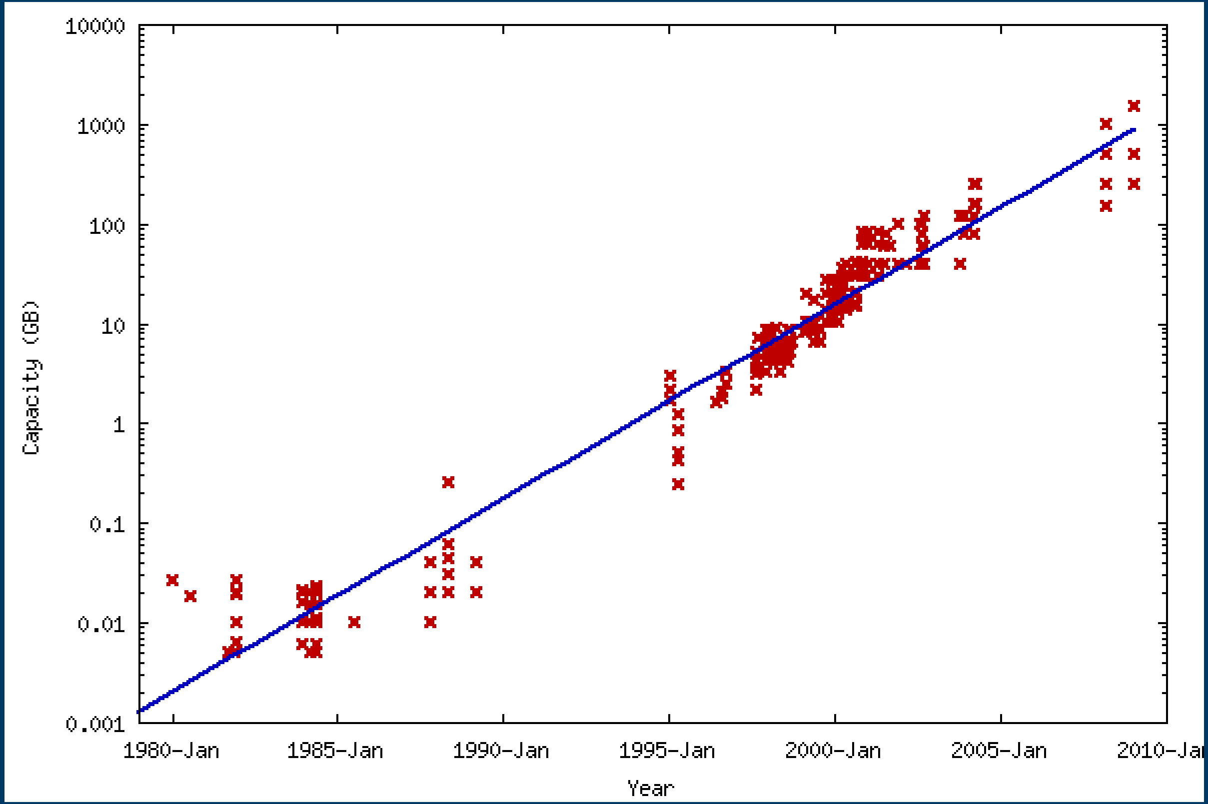


Fig. 3. The average of the gain ratio of optimal total revenue over Myerson total revenue, over 100 simulations, against the switching cost c, for discount rate $\delta = 0.1, 0.3, 0.5, 0.7, 0.9$, for the uniform distribution on $[0, 1]$.

"Optimal Pricing and Introduction Timing of New Virtual Machines" Kash, Key, Zoumpoulis 2018

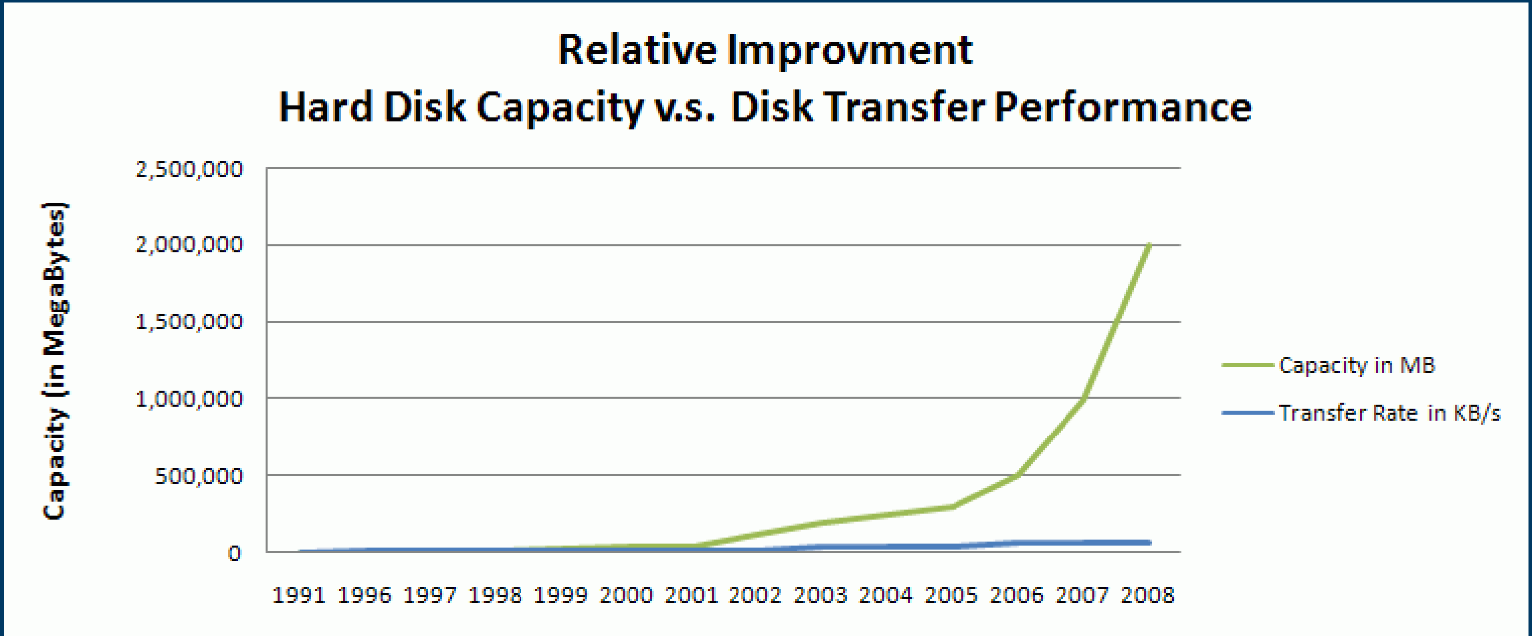
Storage

Kryder's Law



Souce: Wikimedia Commons (Public Domain)

But not throughput…



Souce: R1Soft

"Max" Contracts

Amazon Web Services

PRODUCTS & SERVICES

- Amazon Glacier
- Product Details
- Pricing
- Getting Started
- Resources
- FAQs

RELATED LINKS

- Documentation
- Management Console
- Release Notes
- Discussion Forum

Get Started for Free

Create Free Account

- $0.01 per GB

Except as otherwise noted, our prices are exclusive of applicable taxes and duties, including VAT and applicable sales tax. For customers with a Japanese billing address, use of the Asia Pacific (Tokyo) Region is subject to Japanese Consumption Tax. Learn more.

Request Pricing

Region: US East (N. Virginia)

	Pricing
UPLOAD and RETRIEVAL Requests	$0.050 per 1,000 requests
LISTVAULTS, GETJOBOUTPUT, DELETE† and all other Requests	Free
Data Retrievals	Free †

† Glacier is designed with the expectation that retrievals are infrequent and unusual, and data will be stored for extended periods of time. You can retrieve up to 5% of your average monthly storage (pro-rated daily) for free each month. If you choose to retrieve more than this amount of data in a month, you are charged a retrieval fee starting at $0.01 per gigabyte. Learn more. In addition, there is a pro-rated charge of $0.03 per gigabyte for items deleted prior to 90 days. Learn more.

Except as otherwise noted, our prices are exclusive of applicable taxes and duties, including VAT and applicable sales tax. For customers with a Japanese billing address, use of the Asia Pacific (Tokyo) Region is subject to Japanese Consumption Tax. Learn more.

“Max” Contracts

First we calculate your *peak retrieval rate*. Your peak hourly retrieval rate each month is equal to the greatest amount of data you retrieve in any hour over the course of the month. If you initiate several retrieval jobs in the same hour, these are added together to determine your hourly retrieval rate. We always assume that a retrieval job completes in 4 hours for the purpose of calculating your peak retrieval rate. In this case your peak rate is 140 GB/4 hours, which equals 35 GB per hour.

Then we calculate your *peak billable retrieval rate* by subtracting the amount of data you get for free from your peak rate. To calculate your free data we look at your daily allowance and divide it by the number of hours in the day that you retrieved data. So in this case your free data is 128 GB /4 hours or 32 GB free per hour. This makes your billable retrieval rate 35 GB/hour – 32 GB per hour which equals 3 GB per hour.

To calculate how much you pay for the month we multiply your peak billable retrieval rate (3 GB per hour) by the retrieval fee ($0.01/GB) by the number of hours in a month (720). So in this instance you pay 3 GB/Hour * $0.01 * 720 hours, which equals $21.60 to retrieve 140 GB in 3-5 hours.

Pelican Rack

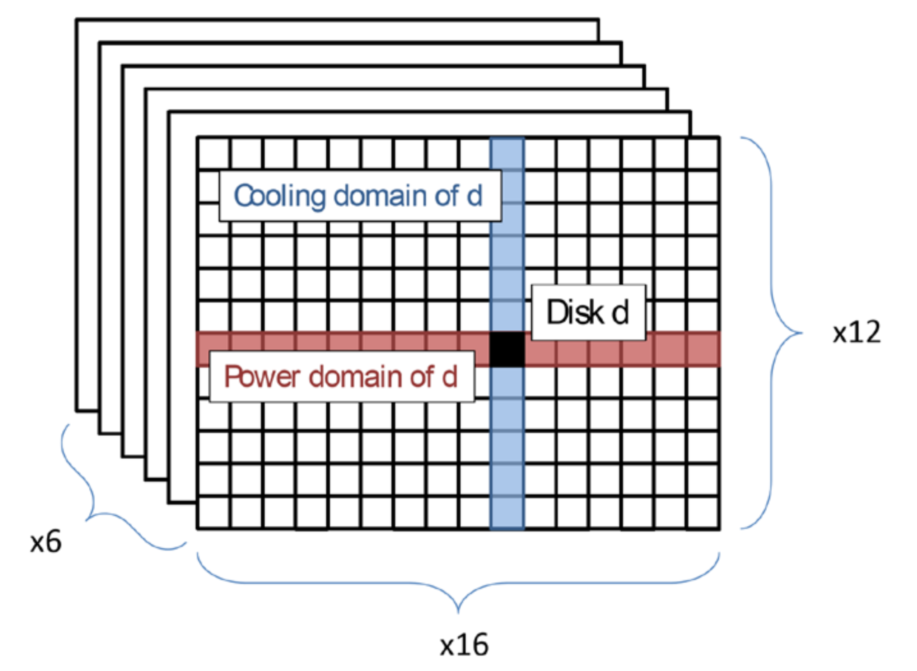


Figure 1: Schematic representation of the Pelican rack

"Pelican: A Building Block for Exascale Cold Data Storage" Balakrishnan et al. 2014

Erasure Coding

- Group data blocks into sets of k=15
- Add r=3 redundancy blocks
- Any 15/18 suffice to recover the data

Pelican Rack

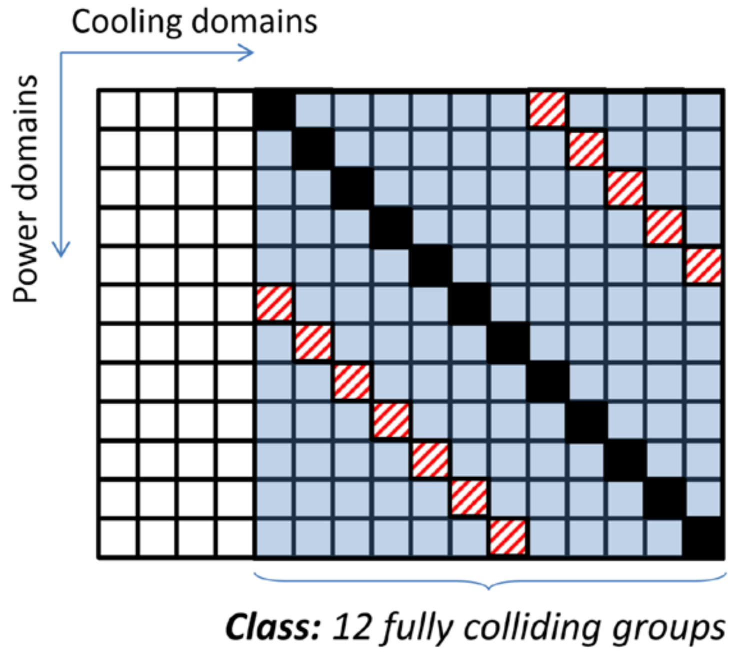


Figure 2: Two fully colliding groups.

"Pelican: A Building Block for Exascale Cold Data Storage" Balakrishnan et al. 2014

Pelican Rack

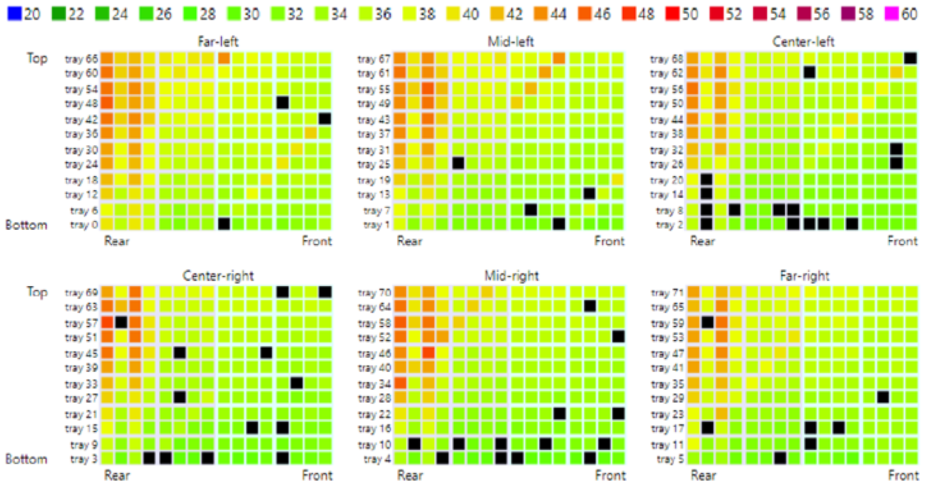


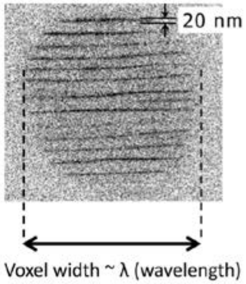
Figure 4: Temperature snapshot from a single rack. The squares are HDDs, colored by their temperature.

"Feeding the Pelican" Black et al. 2016

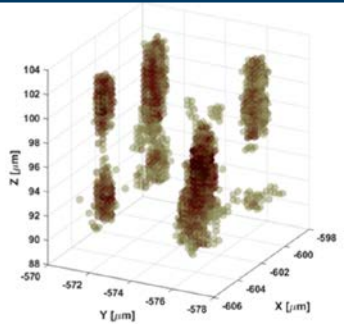
Glass: A New Media for a New Era?

Patrick Anderson,[1] Richard Black,[1] Aušra Čerkauskaitė,[2] Andromachi Chatzieleftheriou,[1] James Clegg,[1] Chris Dainty,[1] Raluca Diaconu,[1] Austin Donnelly,[1] Rokas Drevinskas,[1] Alexander L. Gaunt,[1] Andreas Georgiou,[1] Ariel Gomez Diaz,[1] Peter G. Kazansky,[2] David Lara,[1] Sergey Legtchenko,[1] Sebastian Nowozin,[1] Aaron Ogus,[1] Douglas Phillips,[1] Antony Rowstron,[1] Masaaki Sakakura,[2] Ioan Stefanovici,[1] Benn Thomsen,[1] Lei Wang,[2] Hugh Williams,[1] and Mengyang Yang[1]

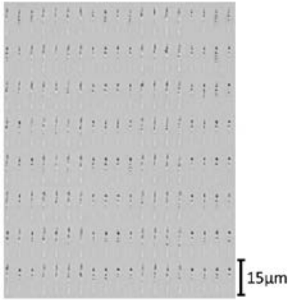
[1]Microsoft Research
[2]Optoelectronics Research Centre, University of Southampton



(a) Single voxel (top view)

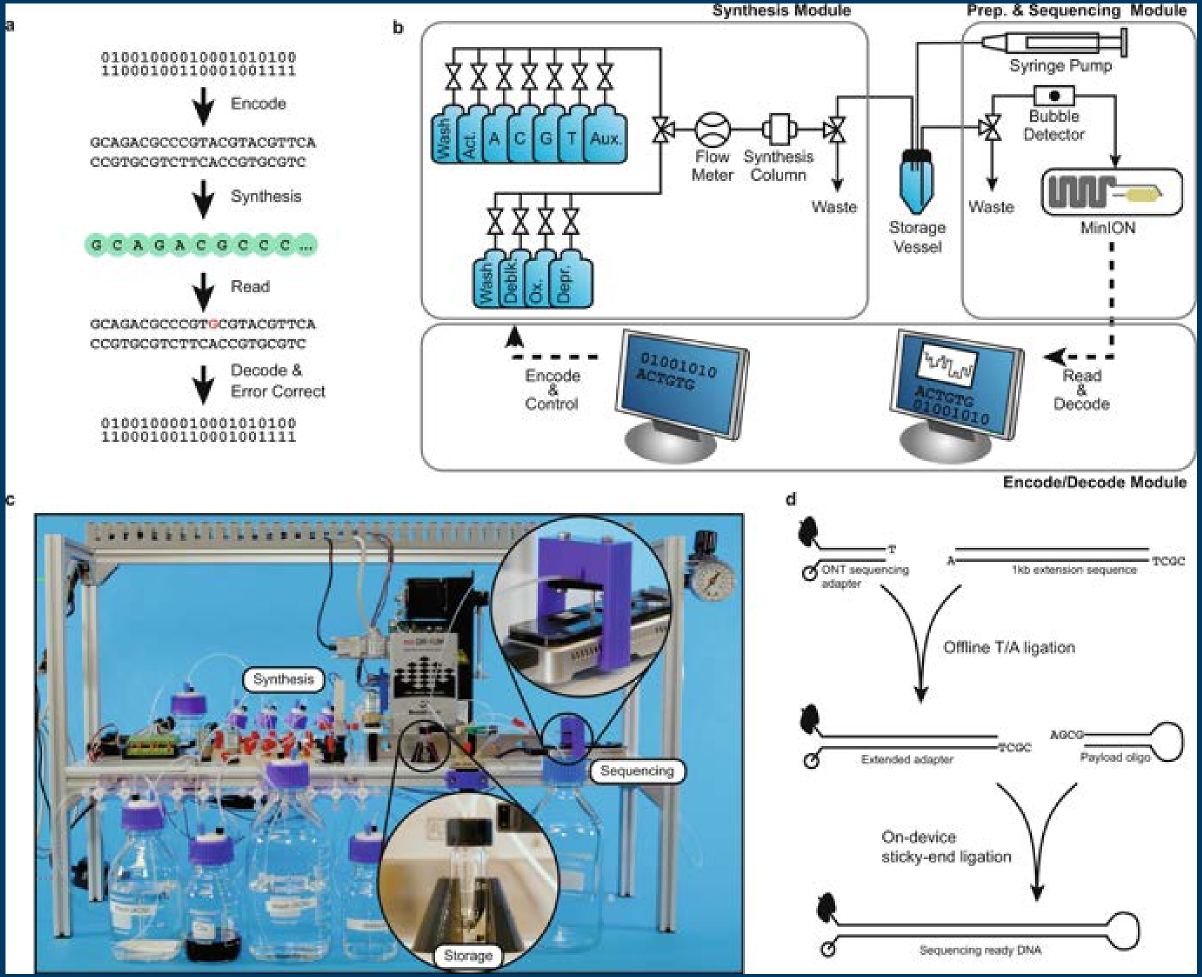


(b) 3D reconstruction of four measured voxels



(c) Side view (8 layers)

Figure 1: Voxels in fused silica



"Demonstration of End-to-End Automation of DNA Data Storage" Takahashi et al. 2019

Storage Economics

- High throughput and low latency are expensive
- Initial pricing policies try and capture this
- Lots of need to improve on both the technology and pricing sides

Network

A Cautionary Tale…

https://github.com/stickfigure/blog/wiki/The-Unofficial-Google-App-Engine-Price-Change-FAQ

Performance Isolation is Hard

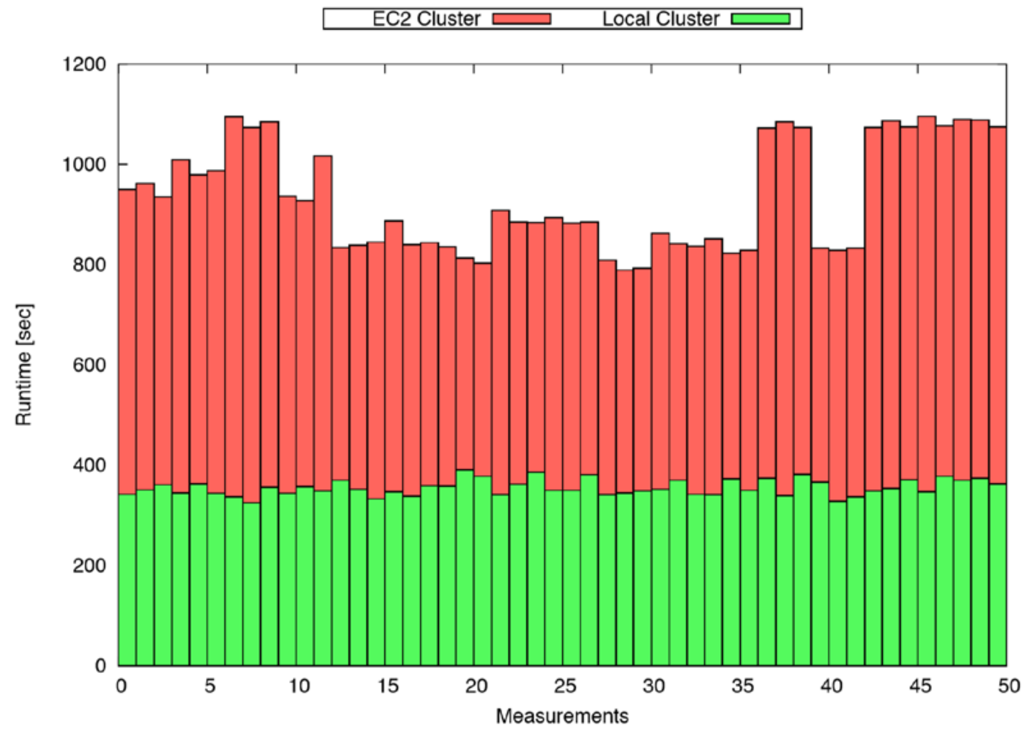


Figure 1: Runtime for a MapReduce job.

"Runtime Measurements in the Cloud: Observing, Analyzing, and Reducing Variance"
Schad et al. 2010

Performance Isolation is Hard

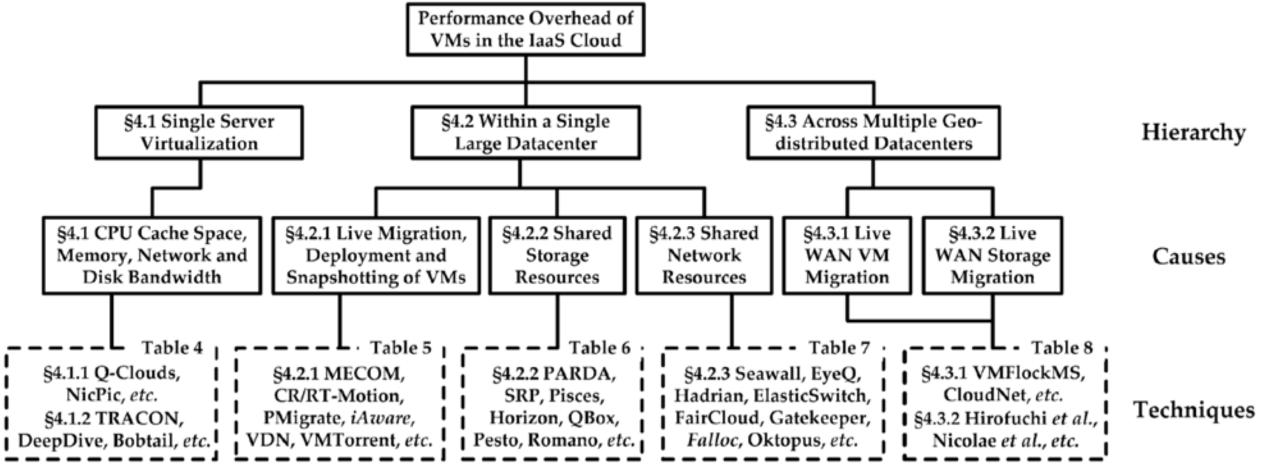


Fig. 1. *Classification of causes and mitigation techniques of VM performance overhead from the viewpoint of IaaS cloud hierarchy.*

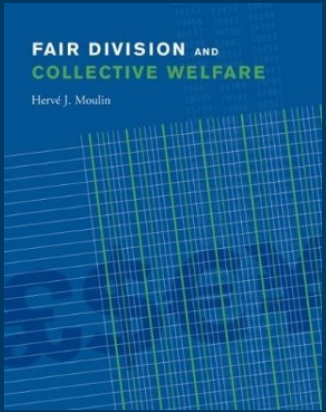
12 PROCEEDINGS OF THE IEEE | *Vol. 102, No. 1, January 2014*

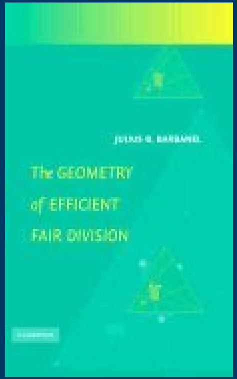
"Managing Performance Overhead of Virtual Machines in Cloud Computing"
Xu et al. 2014

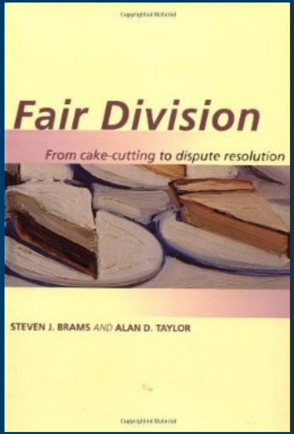
Other Performance Isolation Work

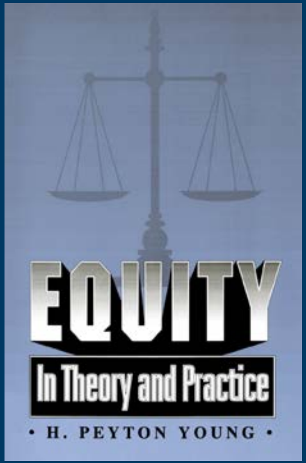
- "Better Never than Late: Meeting Deadlines in Datacenter Networks" Wilson et al. 2011
- "The Price Is Right: Towards Location-Independent Costs in Datacenters" Ballani et al. 2011
- "Performance Isolation and Fairness for Multi-Tenant Cloud Storage" Shue, Friedman, and Shaik 2012
- "Chatty Tenants and the Cloud Network Sharing Problem" Ballani, Jang, Karagiannis 2013

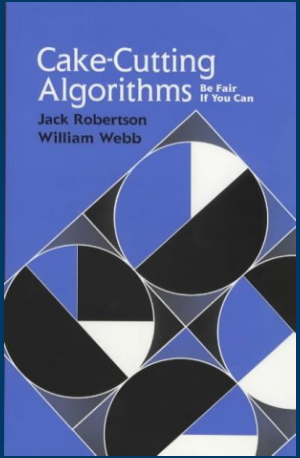
What is fair?


FAIR DIVISION AND
COLLECTIVE WELFARE
Hervé J. Moulin


The GEOMETRY
of EFFICIENT
FAIR DIVISION


Fair Division
From cake-cutting to dispute resolution
STEVEN J. BRAMS AND ALAN D. TAYLOR


EQUITY
In Theory and Practice
• H. PEYTON YOUNG •


Cake-Cutting
Algorithms
Be Fair If You Can
Jack Robertson
William Webb

Dominant Resource Fairness: Fair Allocation of Multiple Resource Types

Ali Ghodsi, Matei Zaharia, Benjamin Hindman, Andy Konwinski, Scott Shenker, Ion Stoica
University of California, Berkeley
{alig,matei,benh,andyk,shenker,istoica}@cs.berkeley.edu

Beyond Dominant Resource Fairness: Extensions, Limitations, and Indivisibilities

DAVID C. PARKES, Harvard University
ARIEL D. PROCACCIA and NISARG SHAH, Carnegie Mellon University

Homogeneous Divisible Goods

Axiomatic Approach

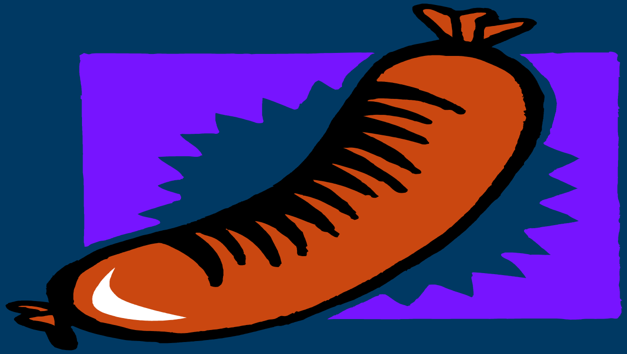
1) Sharing Incentives (SI) – Everyone gets 1/n

2) Envy Freeness (EF) – Everyone prefers his own

3) Strategyproofness (SP) – Truth-telling is optimal

4) Pareto Optimality (PO) – Nothing wasted

Leontief Utilities



Leontief Utilities

Leontief Utilities

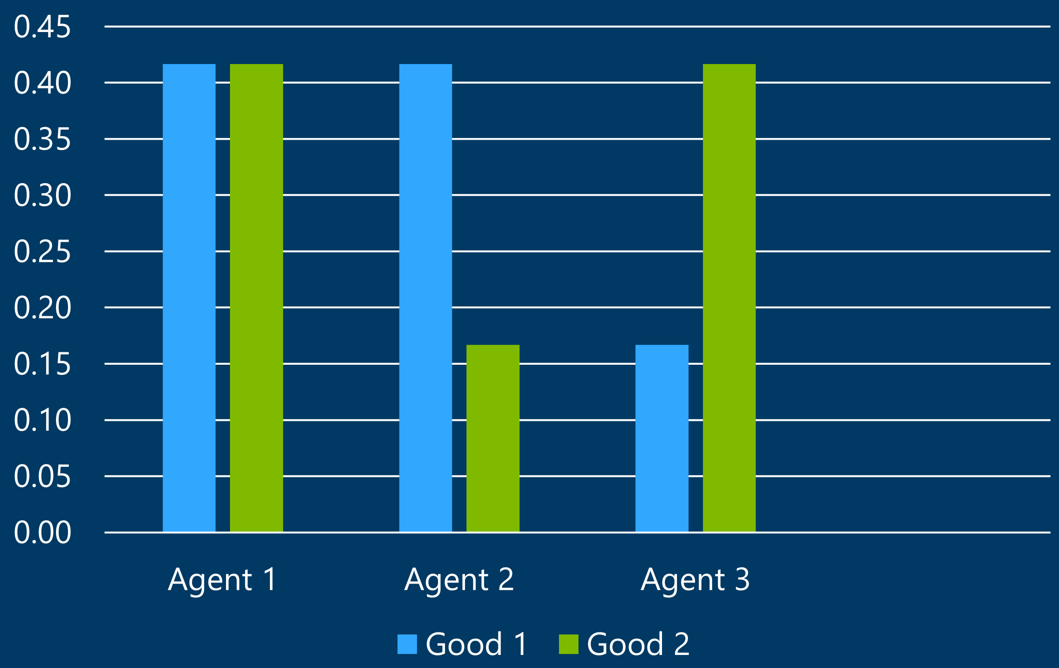
$$U(x) = \min_r \frac{x_r}{d_r}$$

Dominant Resource Fairness

"Everyone gets the same share of his dominant resource"

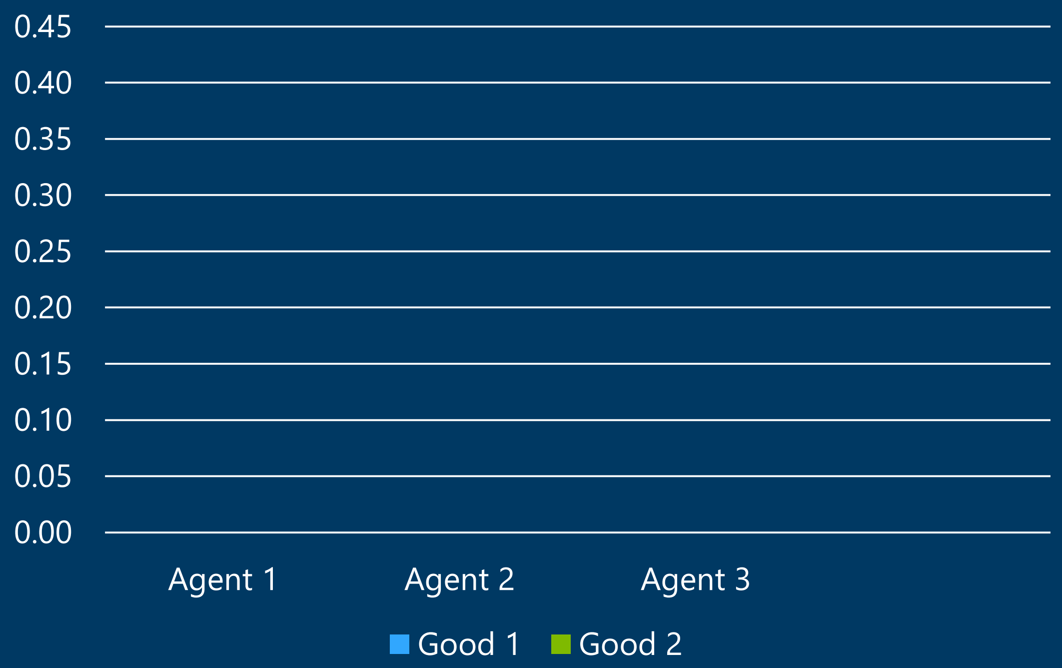
Dominant Resource Fairness

Dominant Resource Fairness

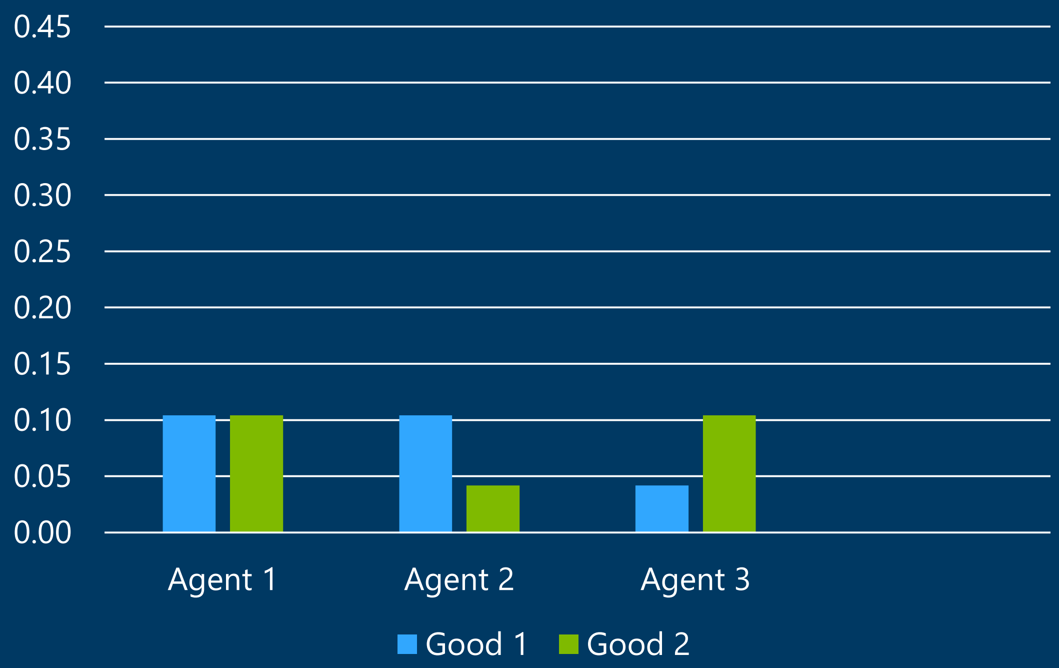


Dominant Resource Fairness

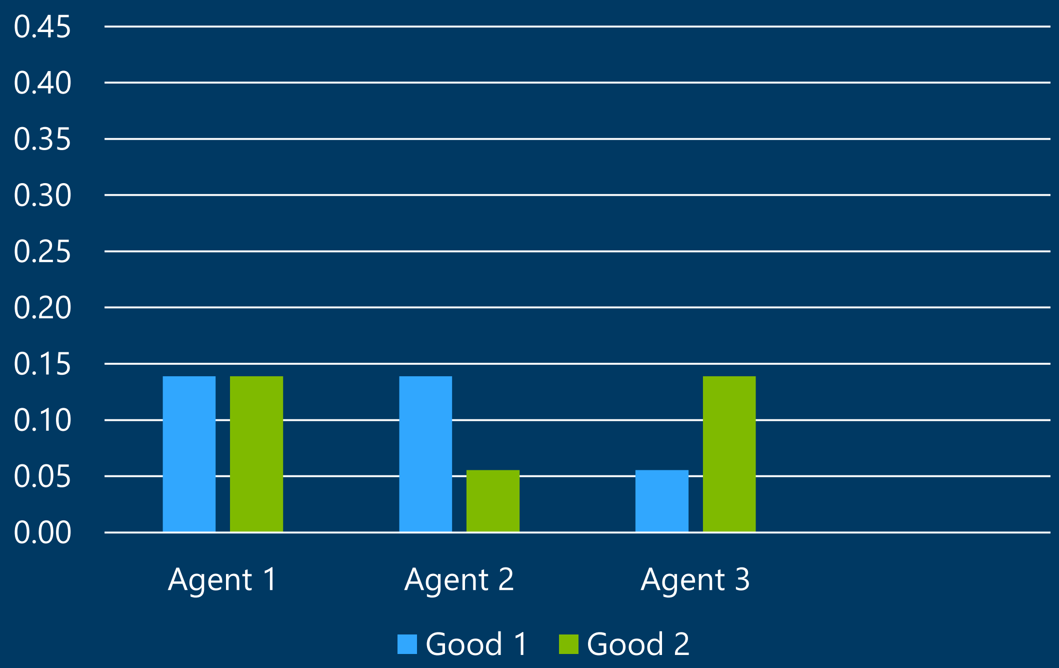
Dominant Resource Fairness



Dominant Resource Fairness

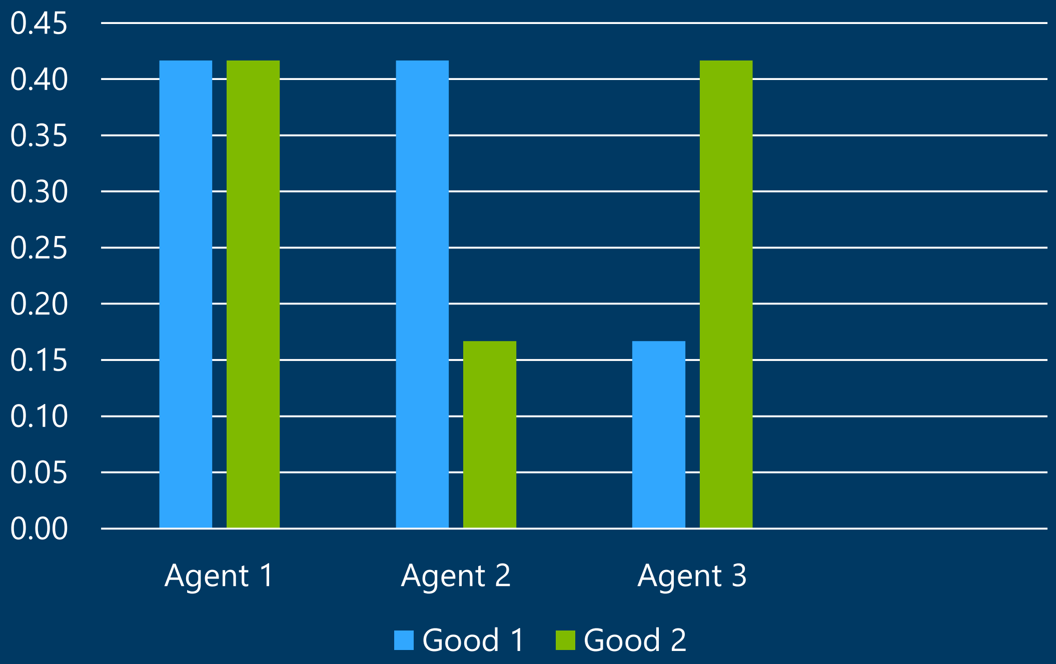


Dominant Resource Fairness



Dominant Resource Fairness

Dominant Resource Fairness



Dominant Resource Fairness

$$\max x$$

Subject to

$$\sum_i x\, d_{ir} \leq 1\ \forall r$$

Dominant Resource Fairness

Theorem: DRF satisfies SI + EF + SP + PO

Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center

Benjamin Hindman, Andy Konwinski, Matei Zaharia,
Ali Ghodsi, Anthony D. Joseph, Randy Katz, Scott Shenker, Ion Stoica
University of California, Berkeley

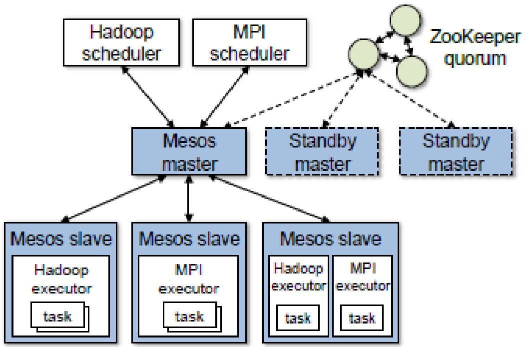


Figure 2: Mesos architecture diagram, showing two running frameworks (Hadoop and MPI).

Journal of Artificial Intelligence Research 51 (2014) 579-603 Submitted 4/14; published 11/14

No Agent Left Behind: Dynamic Fair Division of Multiple Resources

Ian Kash IANKASH@MICROSOFT.COM
Microsoft Research Cambridge, UK

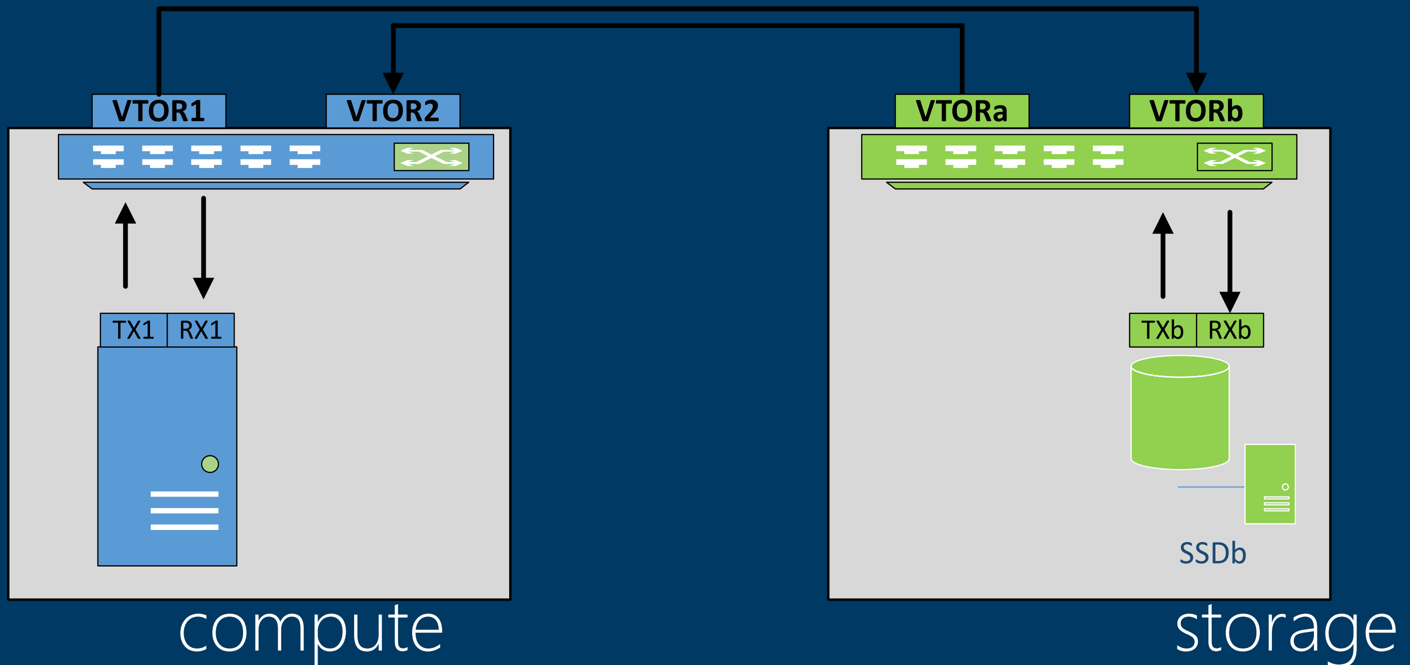
Ariel D. Procaccia ARIELPRO@CS.CMU.EDU
Carnegie Mellon University, USA

Nisarg Shah NKSHAH@CS.CMU.EDU
Carnegie Mellon University, USA

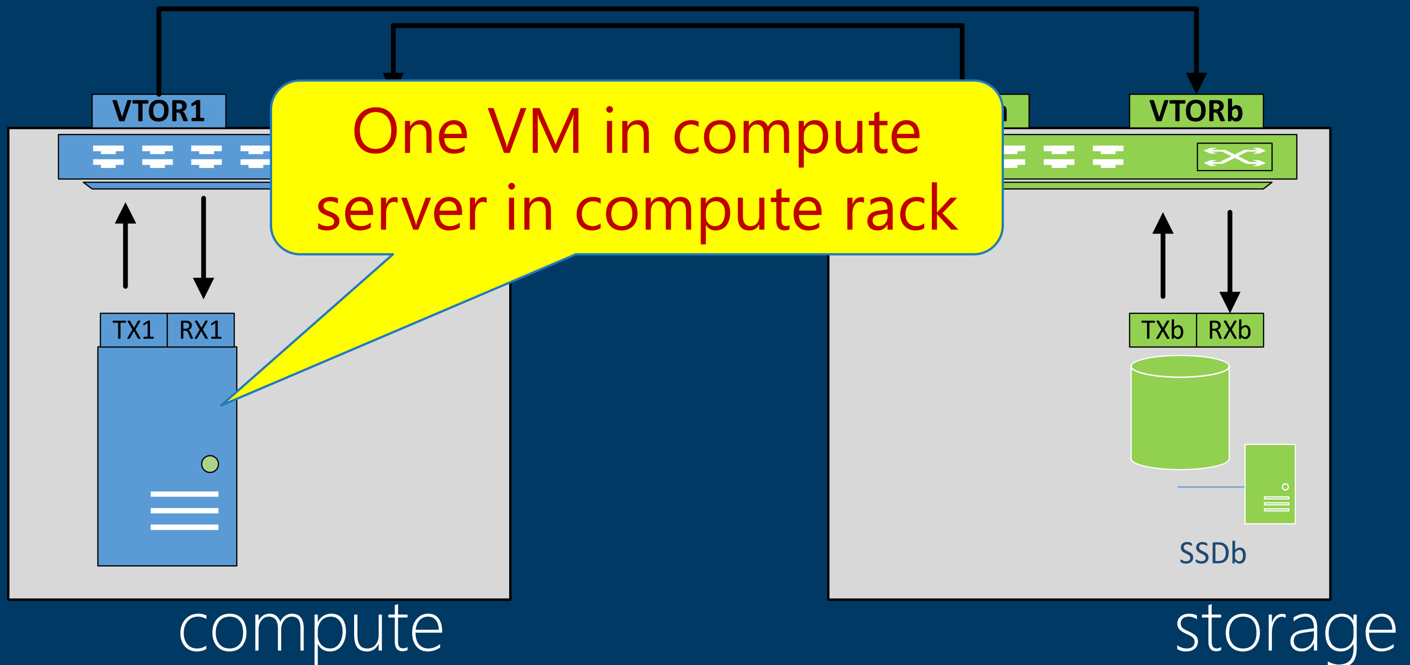
End-to-end Performance Isolation through Virtual Datacenters

Sebastian Angel[*], Hitesh Ballani, Thomas Karagiannis, Greg O'Shea, Eno Thereska
Microsoft Research [*]*The University of Texas at Austin*

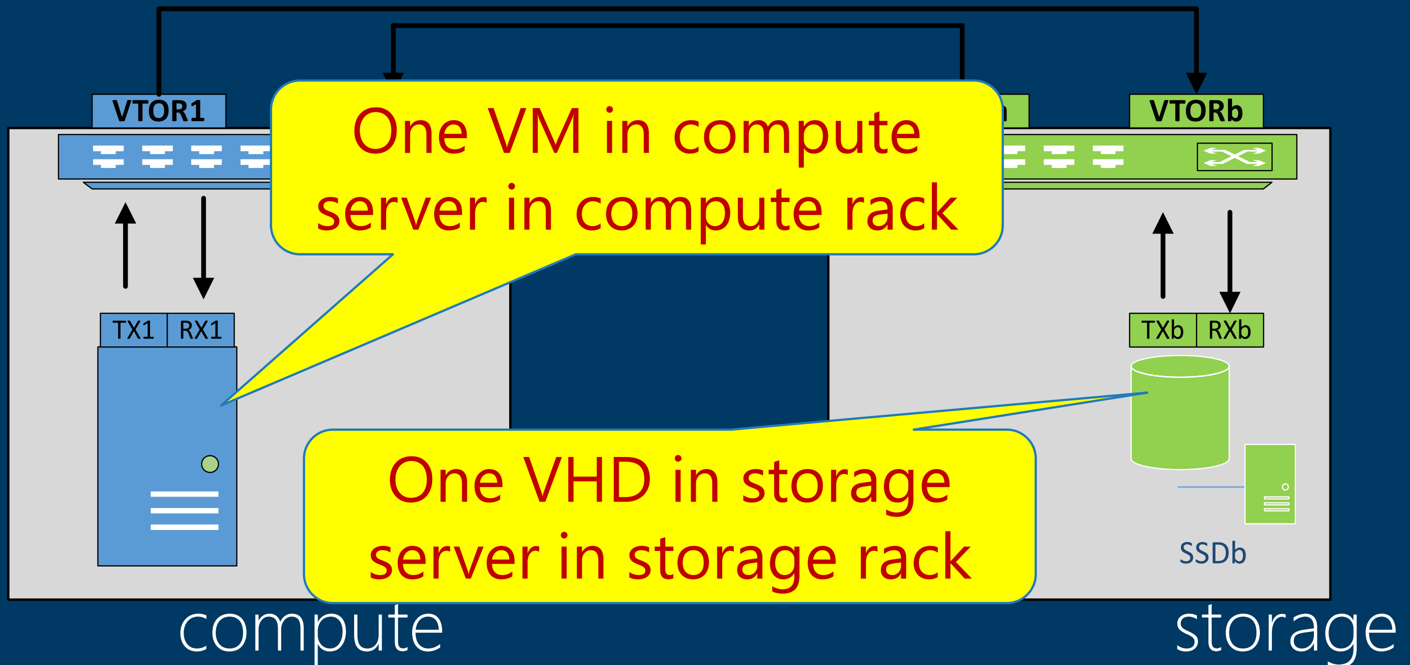
Small customer : one VM accessing storage



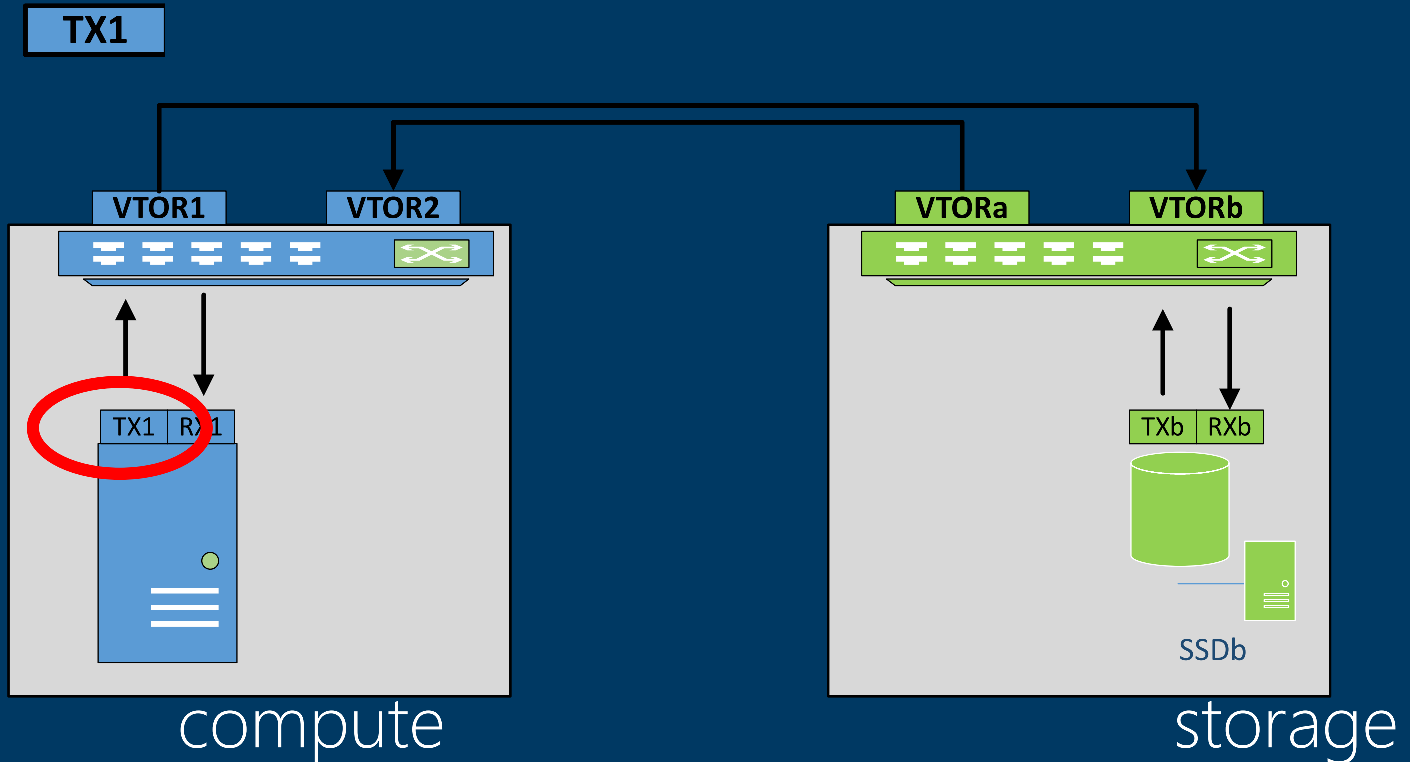
Small customer : one VM accessing storage



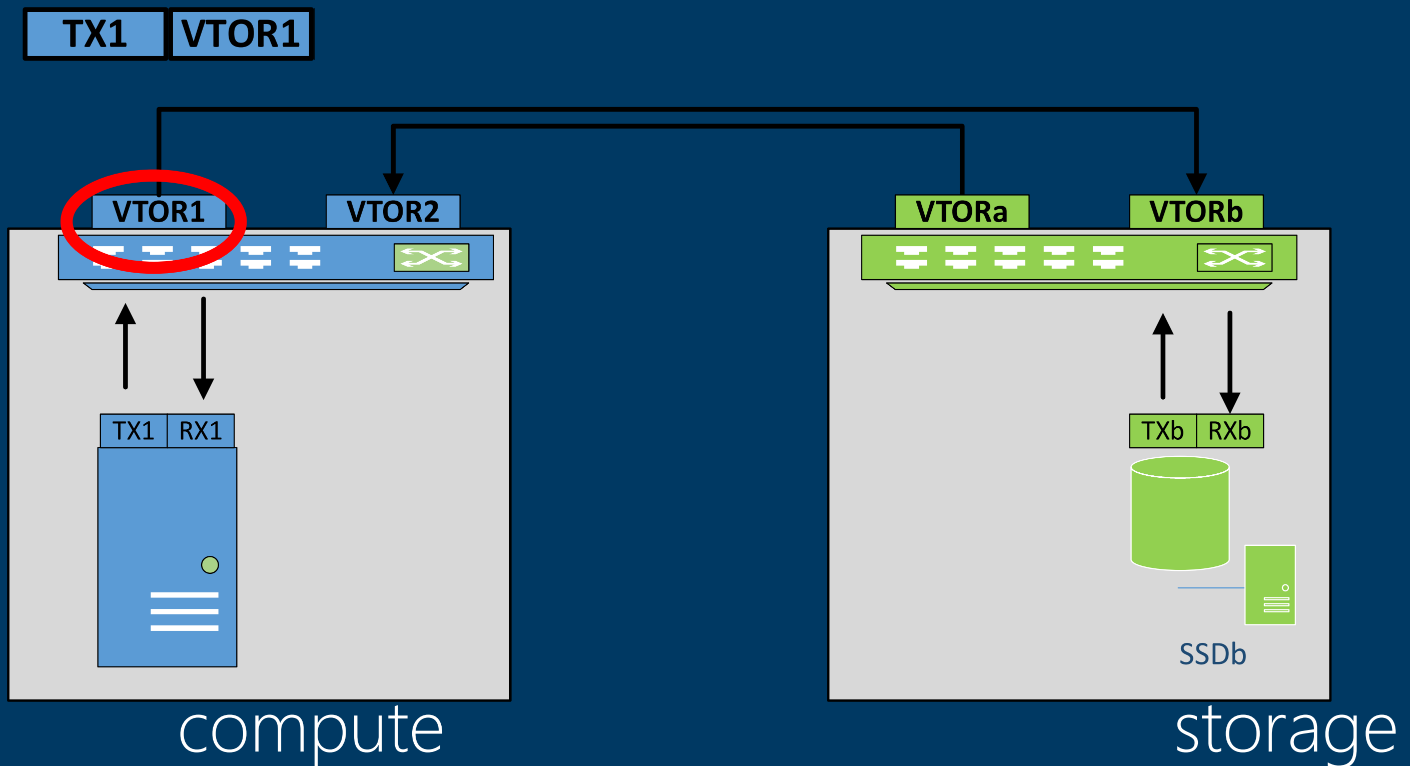
Small customer : one VM accessing storage



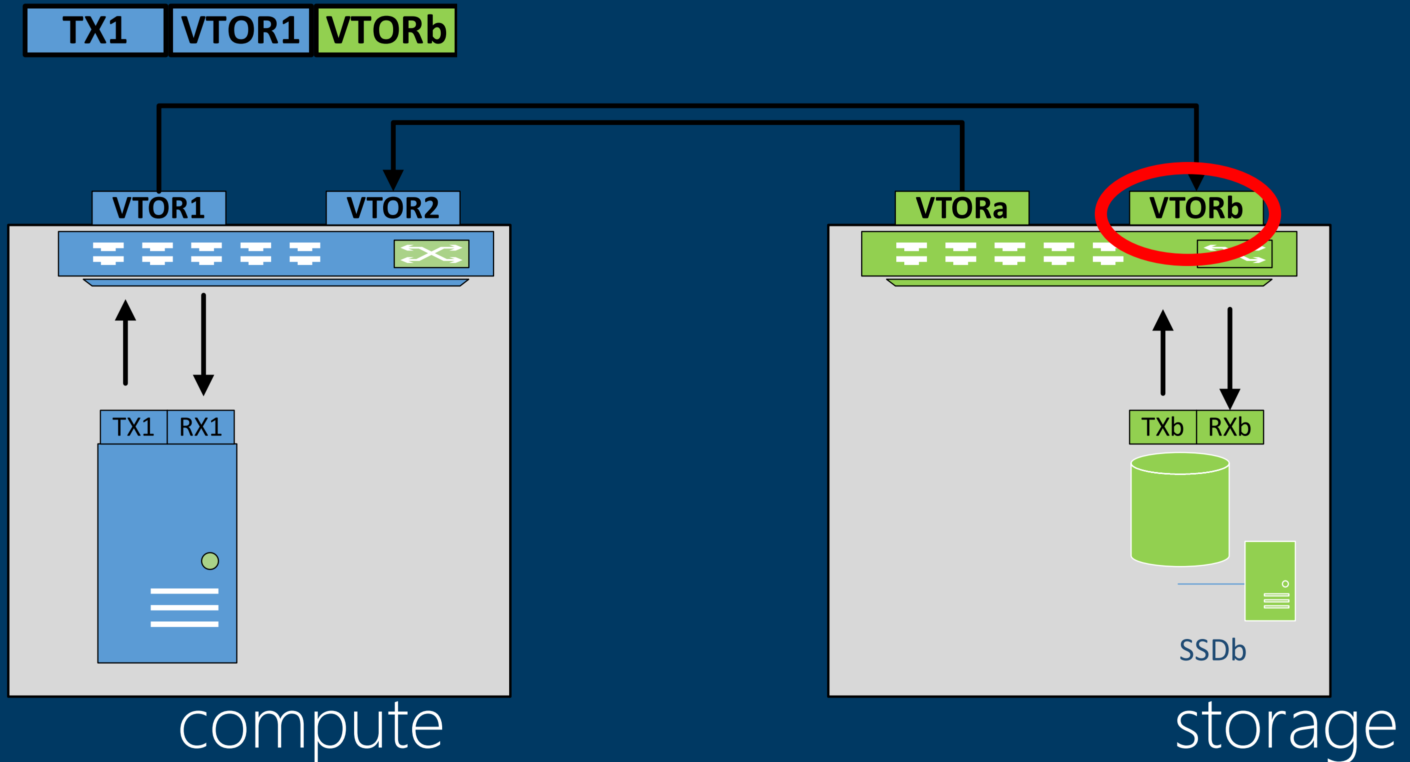
Small customer : one VM accessing storage



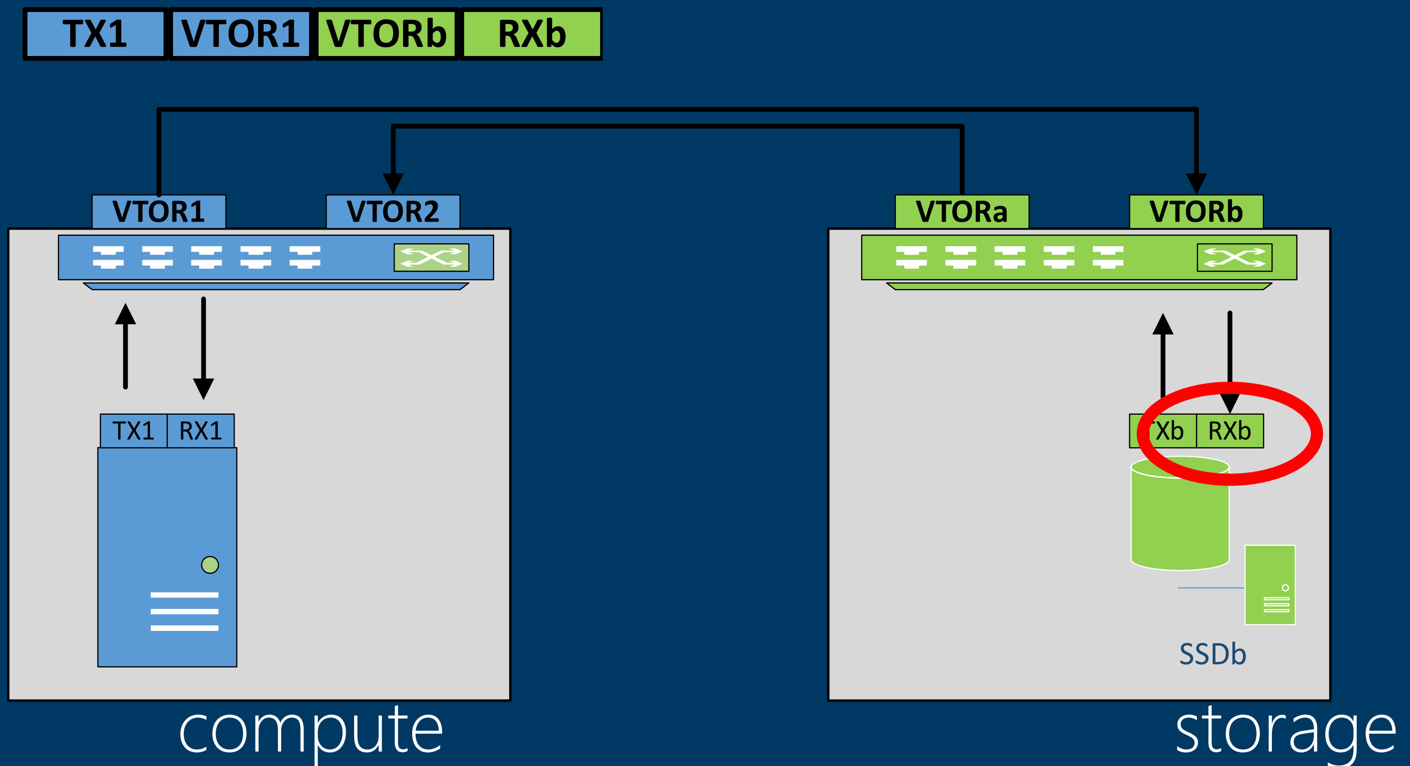
Small customer : one VM accessing storage



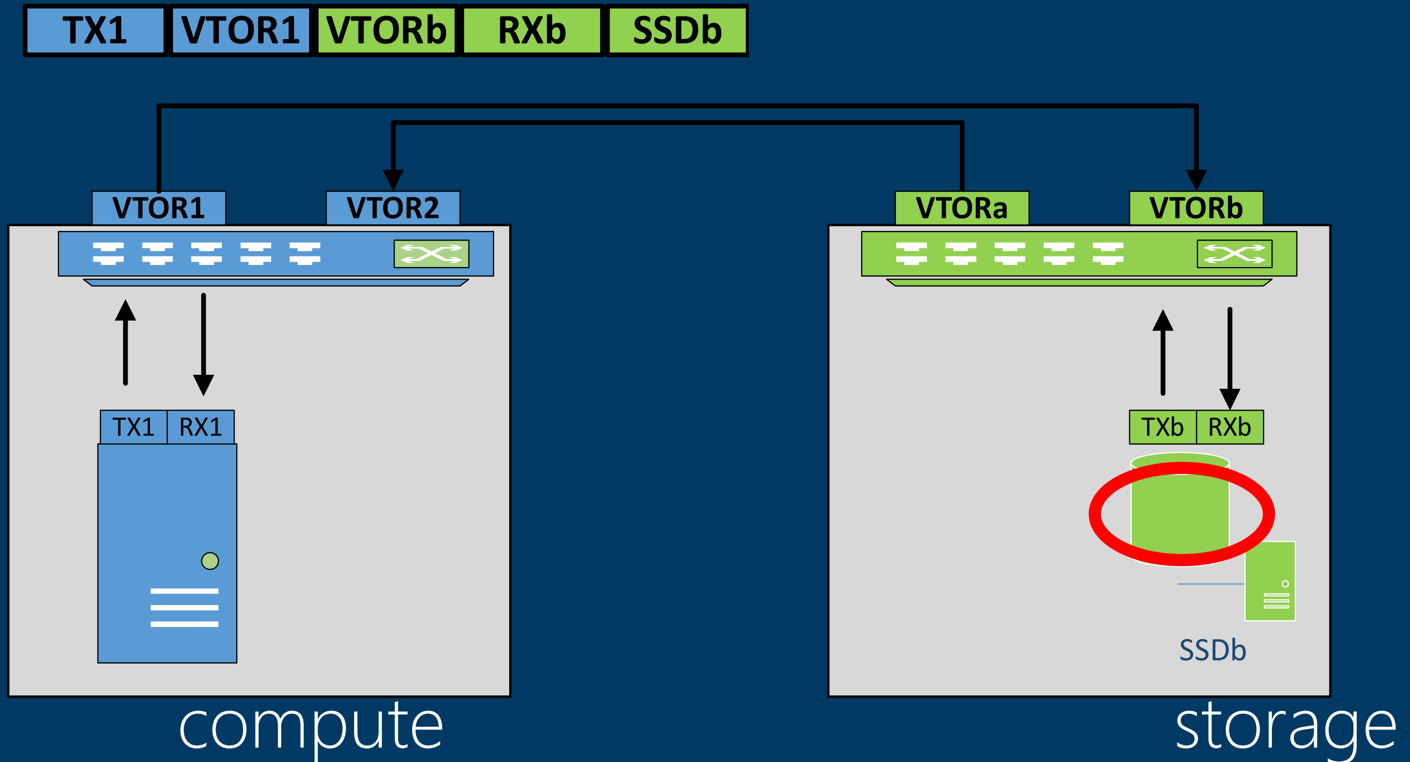
Small customer : one VM accessing storage



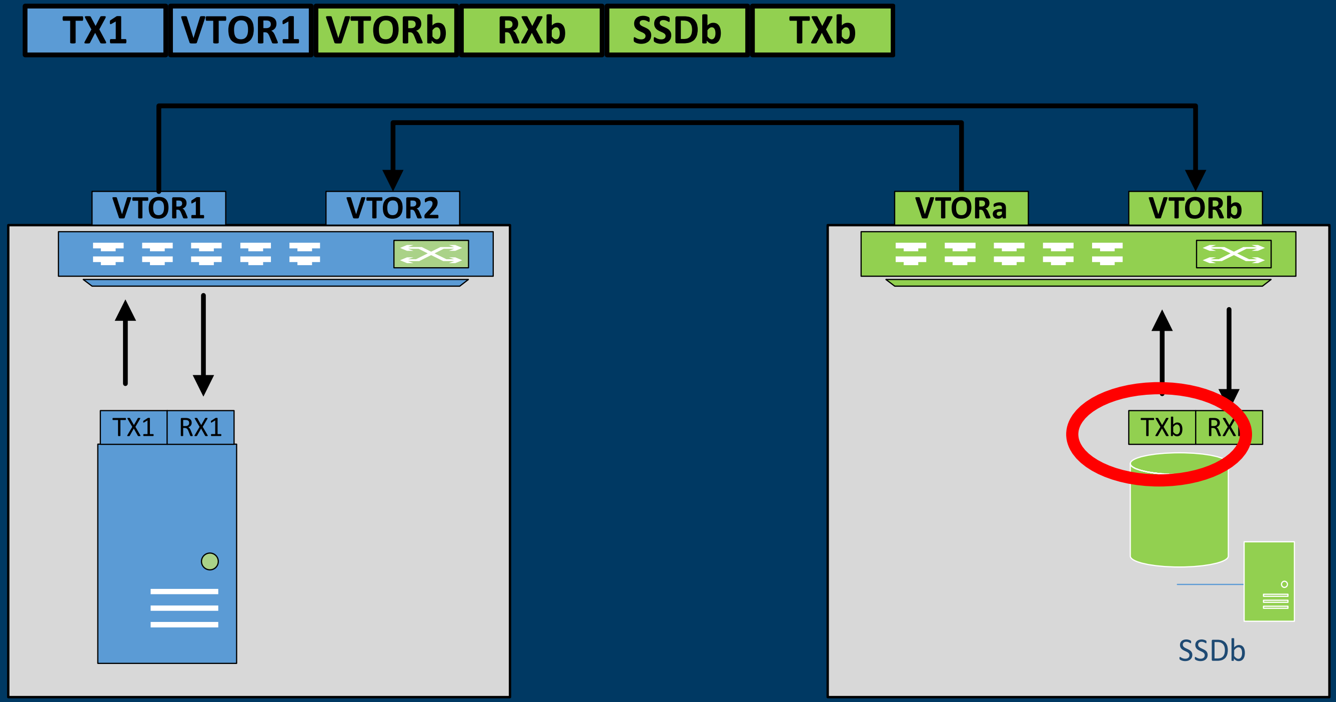
Small customer : one VM accessing storage



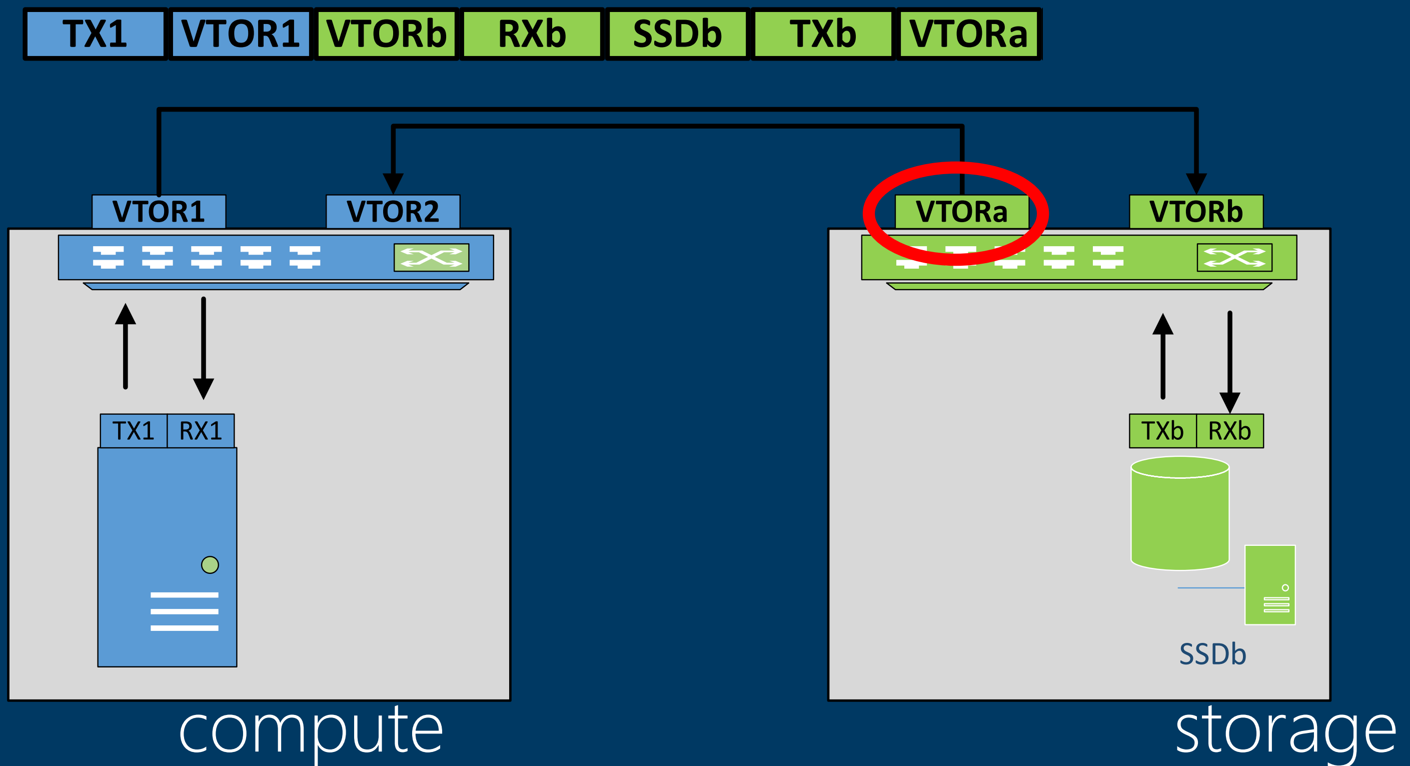
Small customer : one VM accessing storage



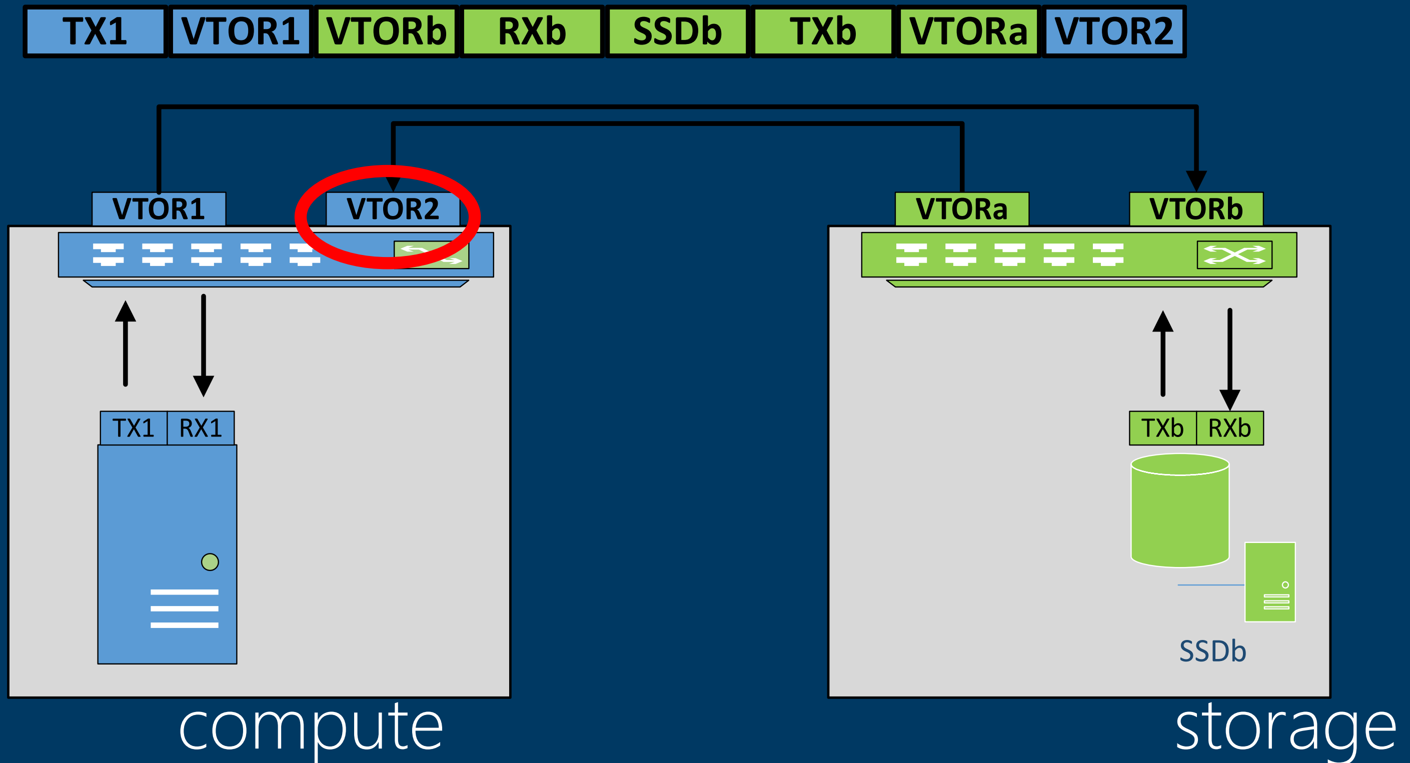
Small customer : one VM accessing storage



Small customer : one VM accessing storage

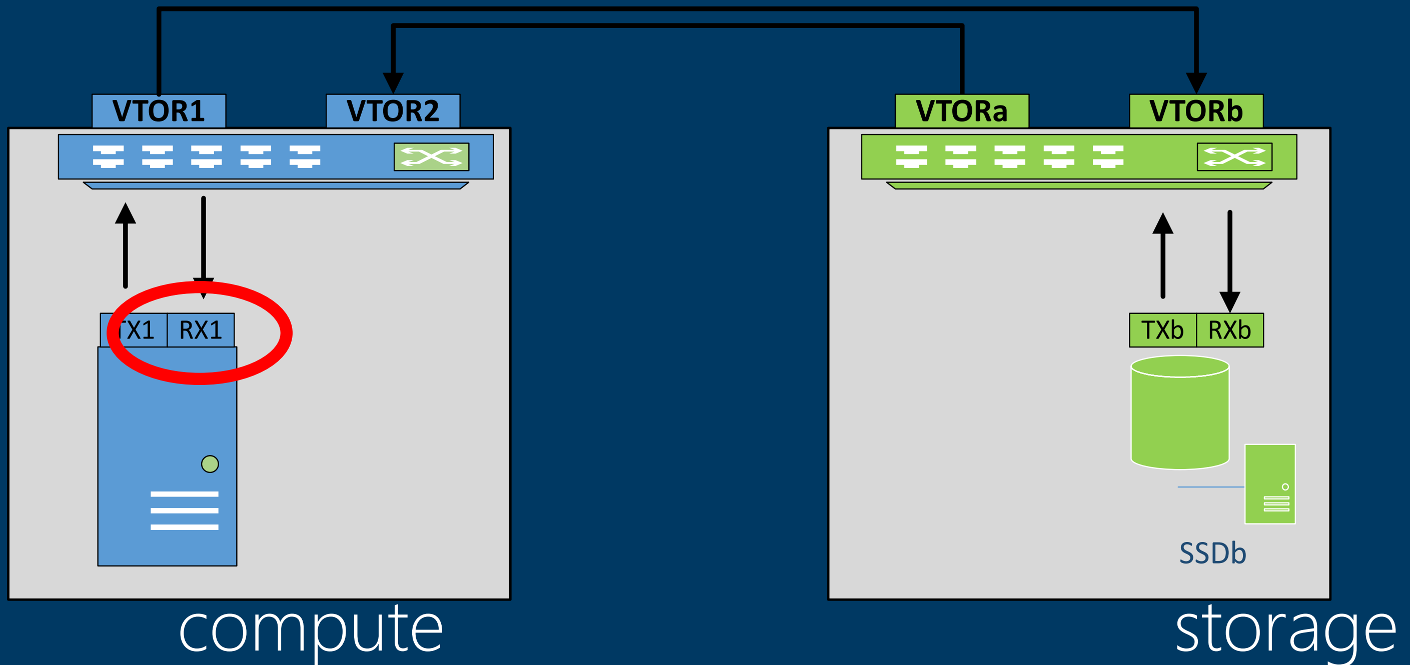


Small customer : one VM accessing storage

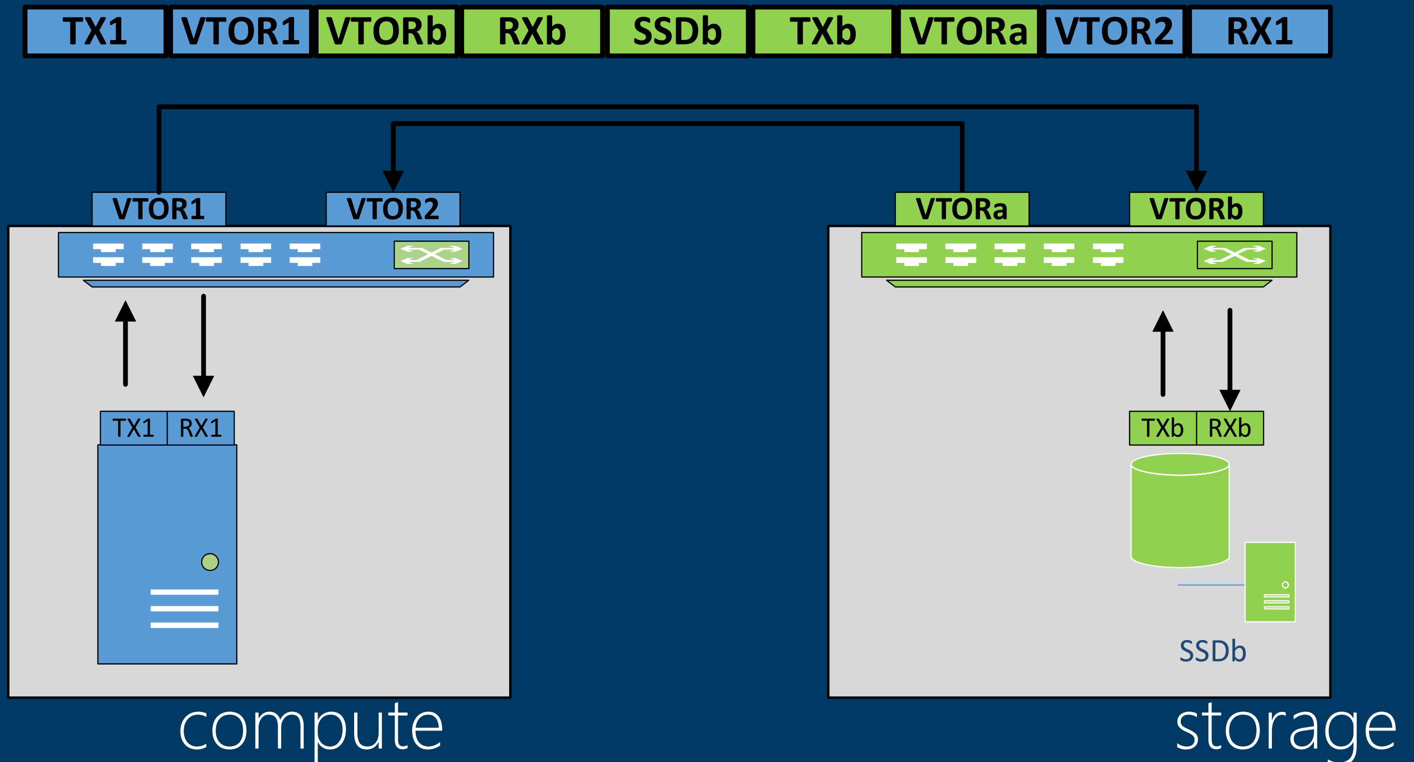


Result: a multi-resource “demand vector”

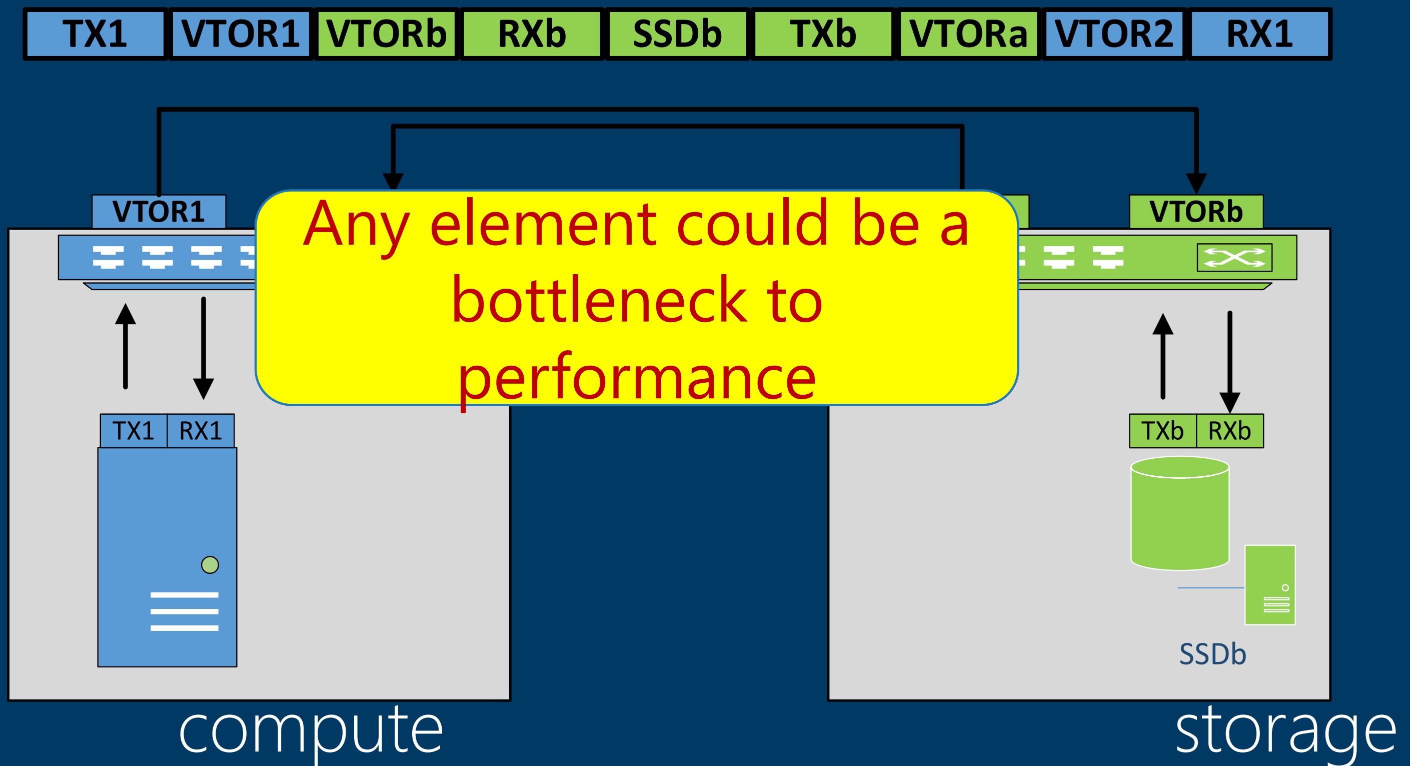
TX1	VTOR1	VTORb	RXb	SSDb	TXb	VTORa	VTOR2	RX1



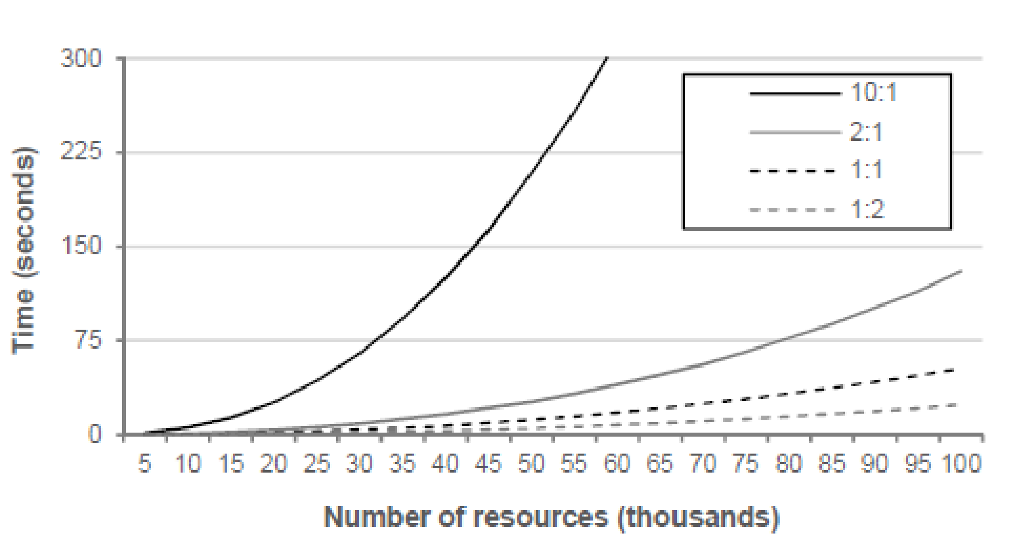
Encodes resource id and proportions



Encodes resource id and proportions



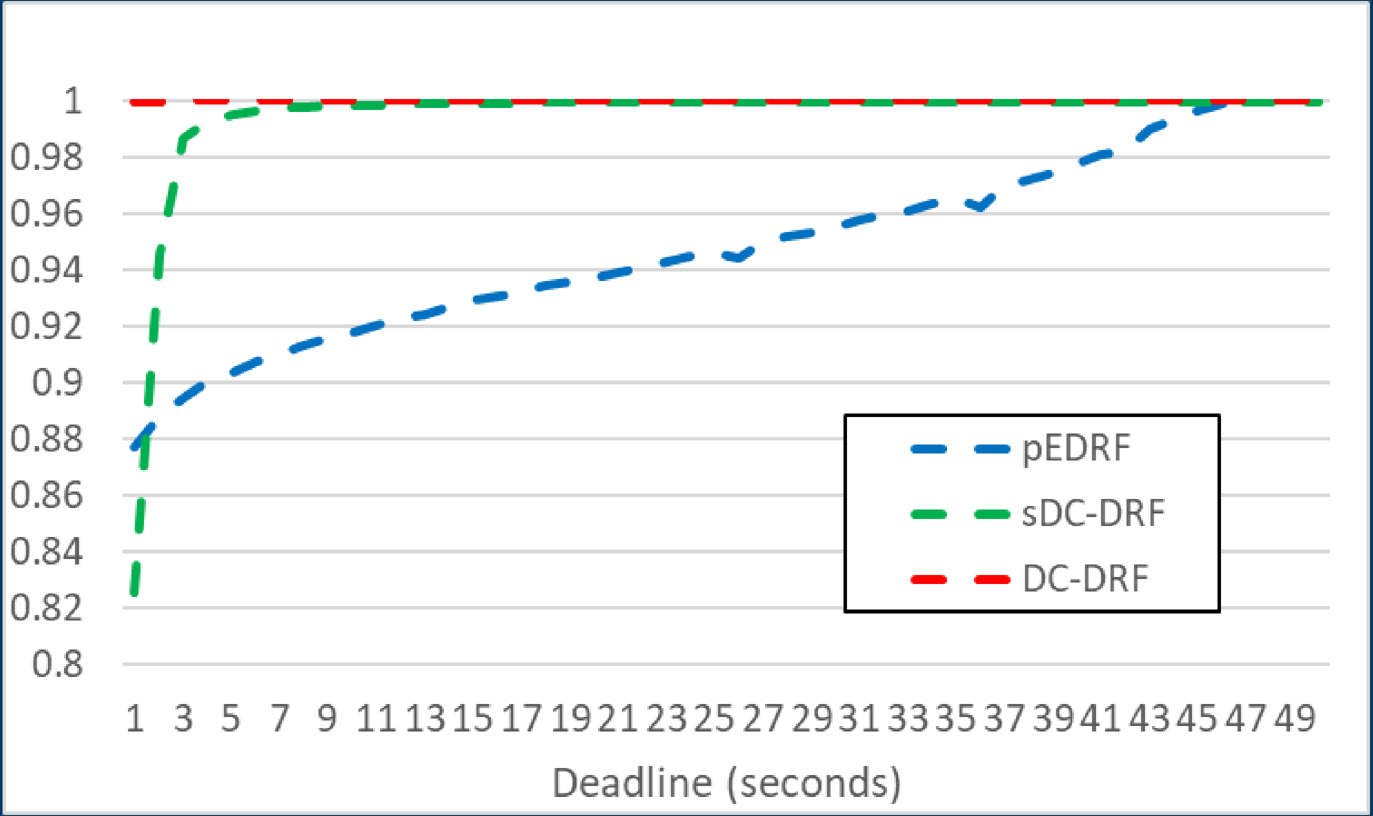
- ~10^5-10^6 agents in a datacenter
- Similar number of resources
- ~1-10 second control interval
- DRF is Quadratic



DC-DRF: two tactics to improve scalability

1. Algorithmic: extending EDRF
   - Operate to a time deadline chosen by operator ("control interval")
   - Variable degree of approximation: trading resource utilization for time
   - Treat any resource that is ϵ-close to exhausted as exhausted
2. HPC: maximize rate of computation
   - Parallel where possible
   - Optimize for thread and NUMA locality
   - SIMD vector instructions

Utilization relative to baseline



Altruistic Scheduling in Multi-Resource Clusters

Robert Grandl[1], Mosharaf Chowdhury[2], Aditya Akella[1], Ganesh Ananthanarayanan[3]
[1] *University of Wisconsin-Madison* [2]*University of Michigan* [3]*Microsoft*

- Jobs are DAGs
- Room for efficiency at no cost to fairness
- More generally fairness-efficiency tradeoffs

Example

- 2 jobs
- 1 processor
- Each requires 1 unit of processor time

- DRF:
- Give each ½ the processor

- Efficient:
- One then the other

Data and Machine Learning

Paying for ML Models

aws

Contact Sales | Support | English | My Account | Create an AWS Account

Products | Solutions | Pricing | Documentation | Learn | Partner Network | AWS Marketplace | Explore More

Amazon Rekognition | Overview | AI/ML Services | Features | Pricing | Getting Started | Resources | FAQs | Customers

US-EAST (N. VIRGINIA)

Image Analysis Tiers	Price per 1,000 Images Processed
First 1 million images processed* per month	$1.00
Next 9 million images processed* per month	$0.80
Next 90 million images processed* per month	$0.60
Over 100 million images processed* per month	$0.40

*Each API that accepts 1 or more input images, counts as 1 image processed. Learn more »

Rekognition Face Metadata Storage Pricing

Rekognition's IndexFaces API analyzes an image (face crop or whole image) and stores the vector representation of faces in a collection. Storage charges are applied monthly and pro-rated for partial months.

Face Metadata Storage	Price per 1,000 face metadata stored per month
Face metadata stored	$0.01

Paying for Data?

Potential Issues with ML Market Design

- “Model Stealing”
- Combining Models
  - Granger causality?
  - Credit assignment
  - Connections to explanability

The Future

Non-linear pricing?

Linear Pricing

Microsoft Azure

INSTANCE	CORES	RAM	PRICE
A1	1	1.75 GB	$0.09/hr (~$67/mo)
A2	2	3.5 GB	$0.18/hr (~$134/mo)
A3	4	7 GB	$0.36/hr (~$268/mo)
A4	8	14 GB	$0.72/hr (~$536/mo)

amazon webservices™

	vCPU	Memory (GiB)	Linux/UNIX Usage
m3.medium	1	3.75	$0.067 per Hour
m3.large	2	7.5	$0.133 per Hour
m3.xlarge	4	15	$0.266 per Hour
m3.2xlarge	8	30	$0.532 per Hour

Reasons for non-linear costs

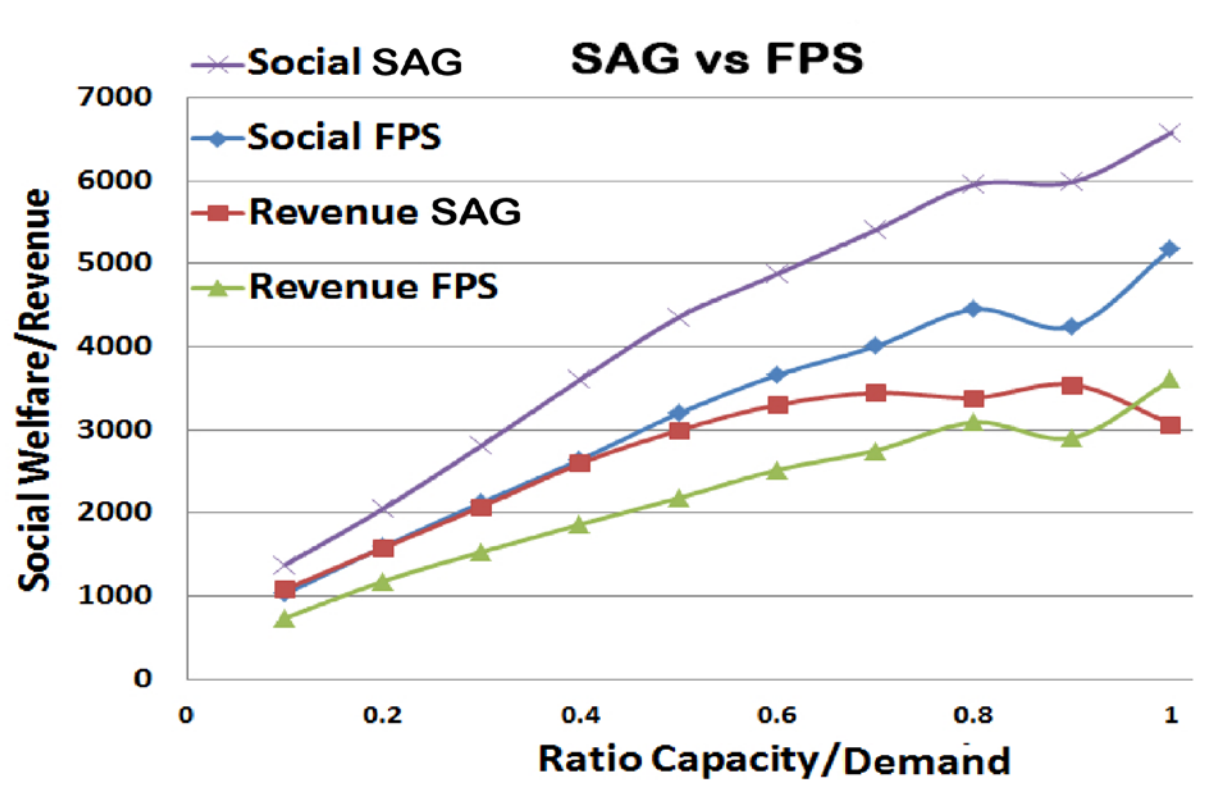
- VM type
- Service Size
- Availibility needs
- Duration
- Scale-outs

Shapley Value / Cost

$$\phi_i(v) = \sum_{\mathbb{S} \subseteq \mathbb{N} \setminus \{i\}} \frac{|\mathbb{S}|!\,(|\mathbb{N}| - |\mathbb{S}| - 1)!}{|\mathbb{N}|!} (v(\mathbb{S} \cup \{i\}) - v(\mathbb{S}))$$

- Consider all possible arrival orders
- For each order, compute marginal cost
- Pay average marginal cost

Fair Cost Sharing



"The Shared Assignment Game" Blocq, Bachrach, and Key 2014

"Max" Contracts

On Economic Heavy Hitters: Shapley value analysis of 95th-percentile pricing

Rade Stanojevic
Telefonica Research

Nikolaos Laoutaris
Telefonica Research

Pablo Rodriguez
Telefonica Research

A. THE 95TH-PERCENTILE PRICING

The 95th-percentile pricing is the most prevalent method that transit ISPs use for charging their customers. A billing cycle, typically one month, is split in constant-size intervals (e.g. 5-min or 1-hour) and number of bytes transferred in each interval is recorded, and the 95th-percentile of the distribution of recorded samples is used for billing. Thus, in a billing cycle of 30 days, 36 hours (5% of time) of the heaviest traffic is filtered out, and then the maximal traffic of the remaining 684 hours is used for billing. Usually, the downstream and upstream 95th-percentile are computed independently, and the lower value is neglected.

The 95th-percentile is also a good measure of how utilized the network is, and is often used as an indicator for dimen-

Pretium – WAN Bandwidth Pricing

Cloud	Traffic billed	Price kind			Service kind		Price (USD/GB)		
		Dyn?	Geo. diff?	Bulk Dis-count?	Tiers?	SLAs	US/EU	Asia	South America
Amazon	between sites	No	No	No	No	No	0-0.02	0-0.09	0-0.16
	to Internet	No	Yes	Yes	No	No	0.05-0.09	0.08-0.14	0.19-0.25
Azure	to Internet	No	Yes	Yes	No	No	0.05-0.09	0.12-0.14	0.16-0.18
Google	to G prod.	No	Yes	No	No	No	0-0.01		
	to Internet	No	Yes	Yes	No	No	0.08-0.12	0.15-0.21	0.08-0.12
Rackspace	all out	No	No	Yes	No	No	0.06-0.12		

Table 2: Pricing for WAN bandwidth by cloud providers (current as of 1/25/2016).

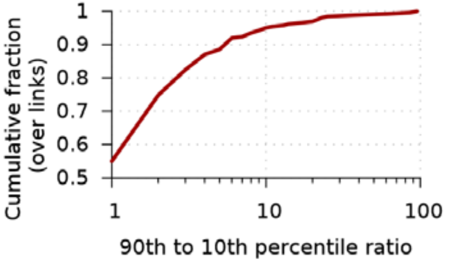


Figure 1: Ratio of 90^{th} percentile to 10^{th} percentile link utilization shown as a cumulative distribution function.

"Dynamic Pricing and Traffic Engineering for Timely Inter-Datacenter Transfers" Jalaparti et al. 2016

Job-based pricing

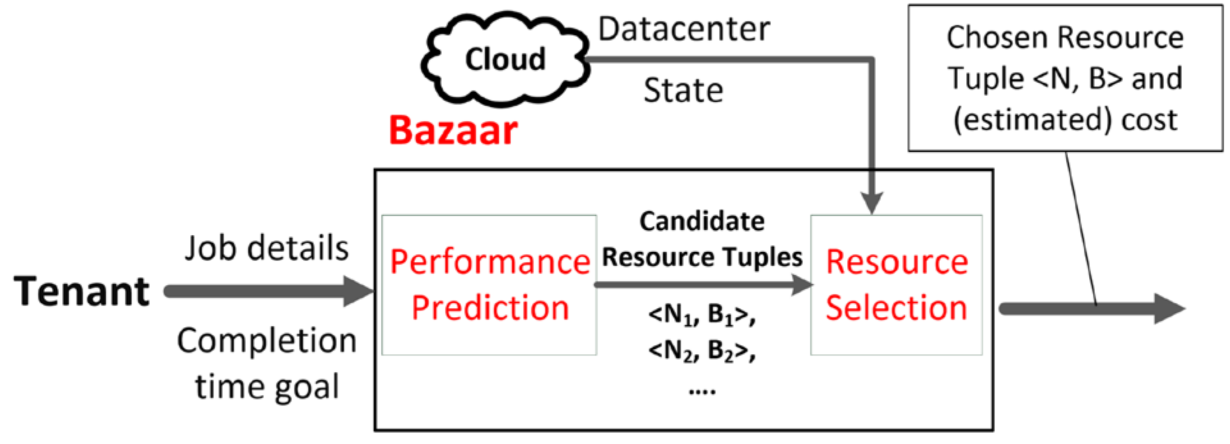


Figure 3: Bazaar offers a job-centric interface.

"Bridging the Tenant-Provider Gap in Cloud Services" Jalaparti et al. 2012

Job-based pricing

"Exploiting Time-Malleability in Cloud-based Batch Processing Systems" Mai, Kalyvianaki, Costa 2013

Job-based pricing

"Exploiting Time-Malleability in Cloud-based Batch Processing Systems" Mai, Kalyvianaki, Costa 2013

Information Elicitation

What not to do

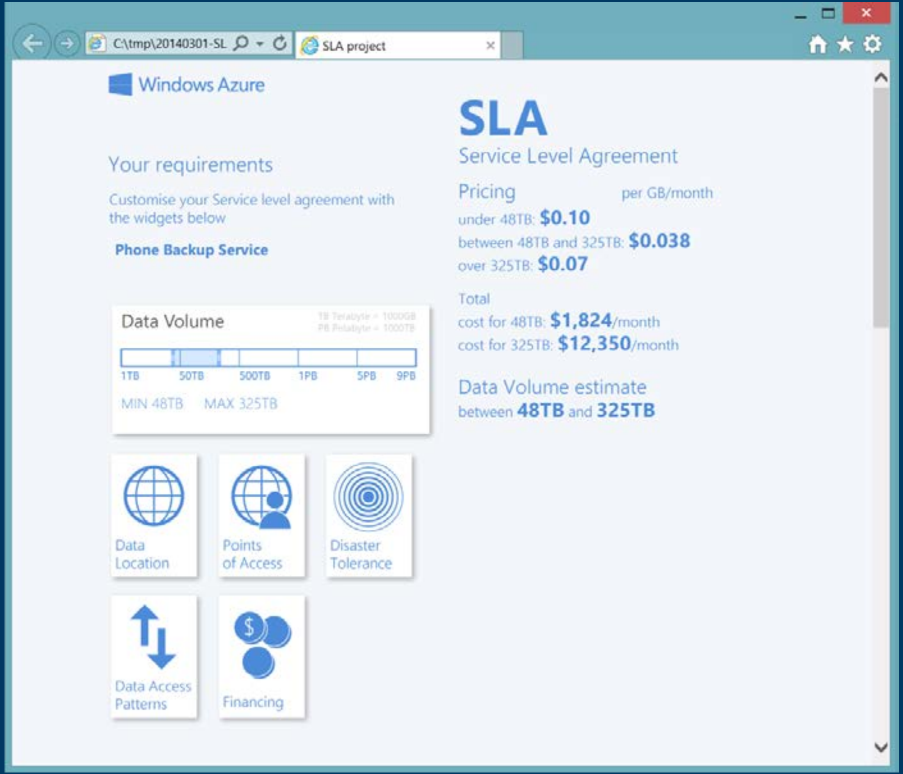
Examples of intent elicitation

Product	Information
Pay as You Go (Azure, AWS, Google)	None
Quota (Azure, AWS, Google)	Peak Demand
Fine-grained budgets (AWS)	Bound on total usage
Reserved Instances (AWS, Azure)	Heavy / light workload
Scheduled Reserved Instances (AWS)	Heavy use in a particular pattern
Sport Market (AWS) / Evictable (Azure, Google)	High / low value jobs
Tiered Storage pricing (Azure) / Glacier (AWS)	Data hot / cold
???	VM short-lived / long-lived
???	Usage steady / bursty
???	Heavy usage at a particular time

Quotas

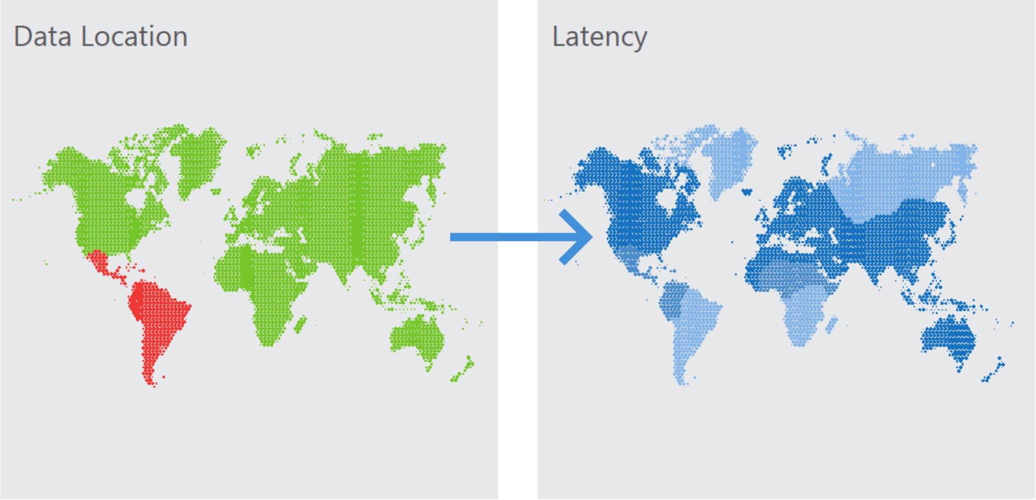
- Provide a "guarantee" to customers
- Provide information about peak usage
- Allow Azure to do capacity control
- Enable customer governance of end users
- But always a headache for someone
  - Small quotas require customer management
  - Big quotas are costly for Azure
  - Manual negotiation process

Storage SLAs



User Input

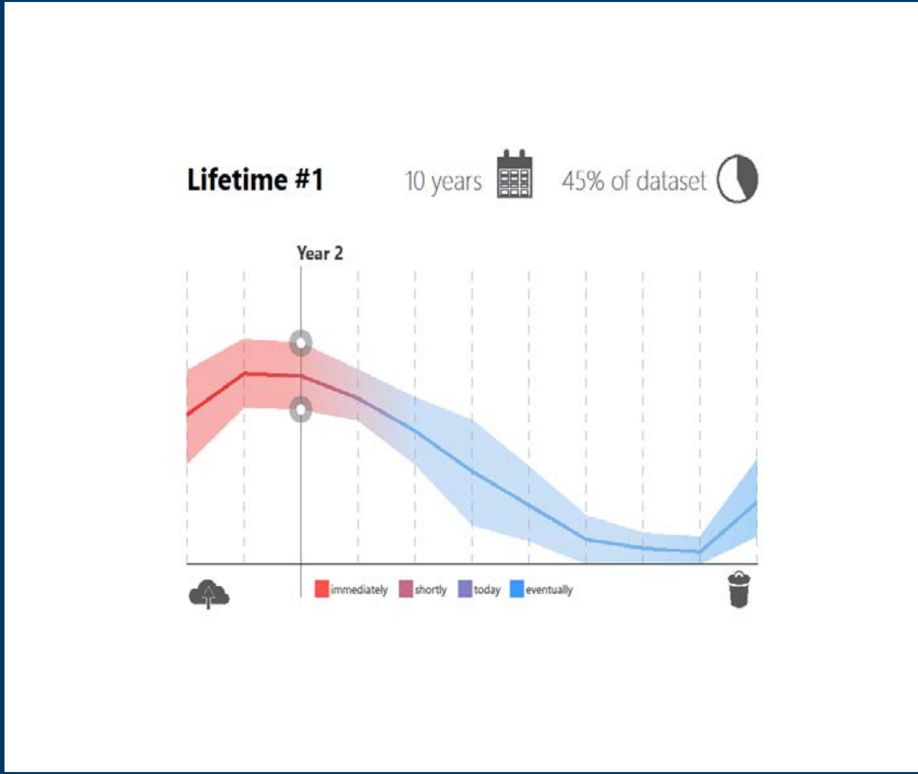
SLA



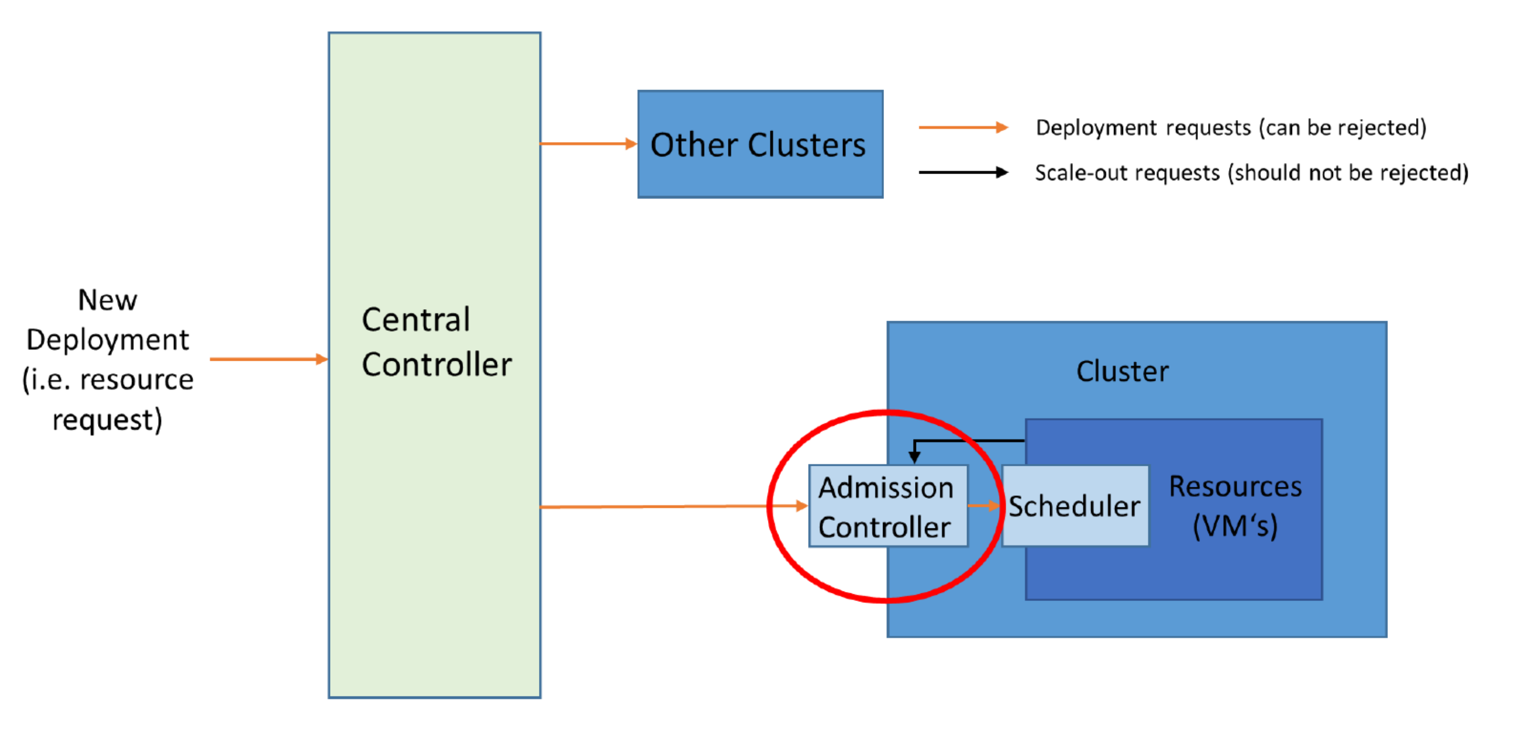
Disaster Tolerance

city 50km	country 1000km	region 3000km	continent 5000km	2 continents 10000km

Explain Future Access Patterns

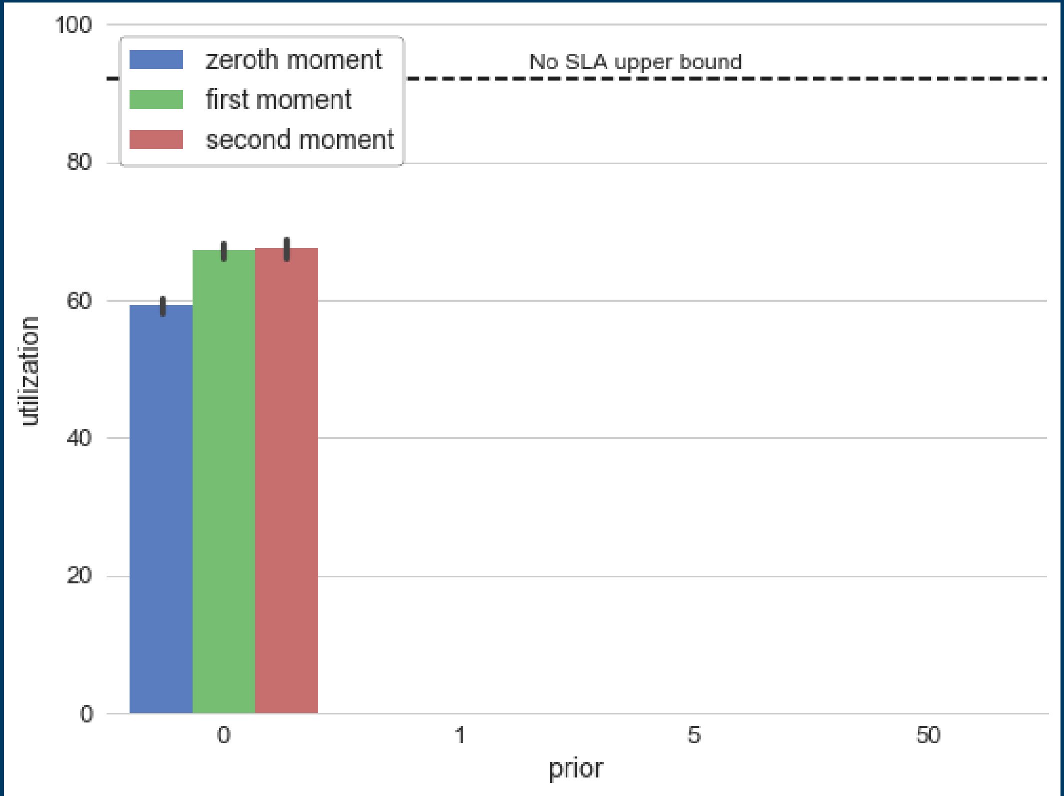


Cluster Admission



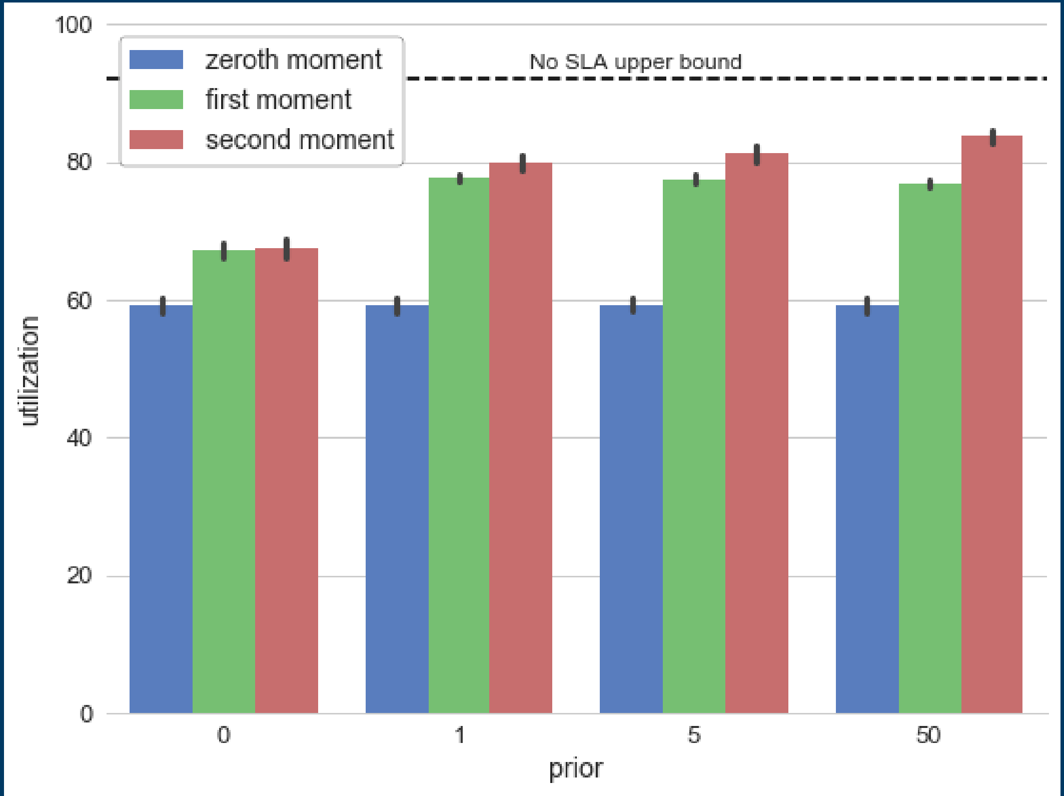
"On the Cluster Admission Problem for Cloud Computing" Dierks, Kash, Seuken 2019

Cluster Admission



"On the Cluster Admission Problem for Cloud Computing" Dierks, Kash, Seuken 2019

Cluster Admission



"On the Cluster Admission Problem for Cloud Computing" Dierks, Kash, Seuken 2019

Cluster Admission - Pricing

- Sell Options that permit scale outs

- Variance-based pricing

$$\pi(X) = \kappa_1 C^X + \kappa_2 Var(X)$$

"On the Cluster Admission Problem for Cloud Computing" Dierks, Kash, Seuken 2019

Thanks!