# Improving the Efficiency of Cloud Systems through Enhanced Service Flexibility

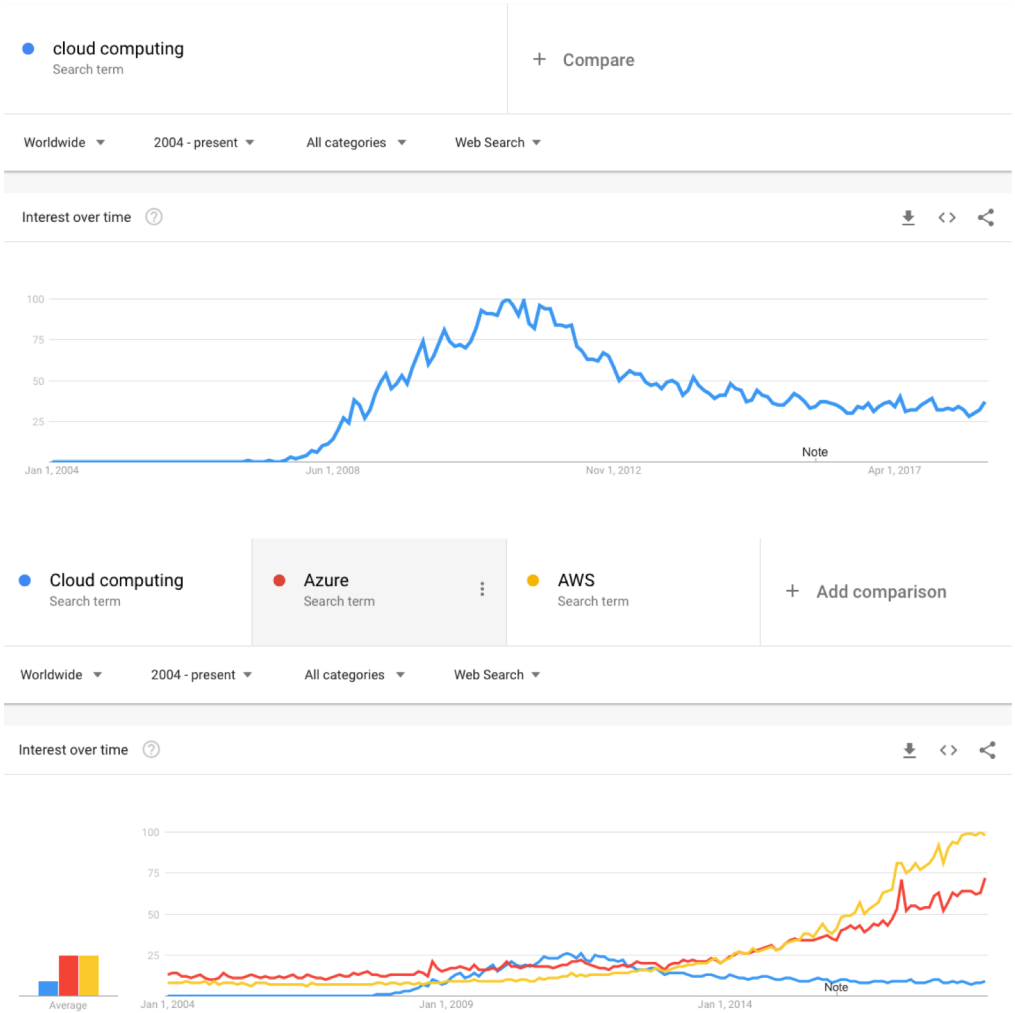
Mohammad Shahrad

PRINCETON UNIVERSITY

PRINCETON
School of Engineering and Applied Science

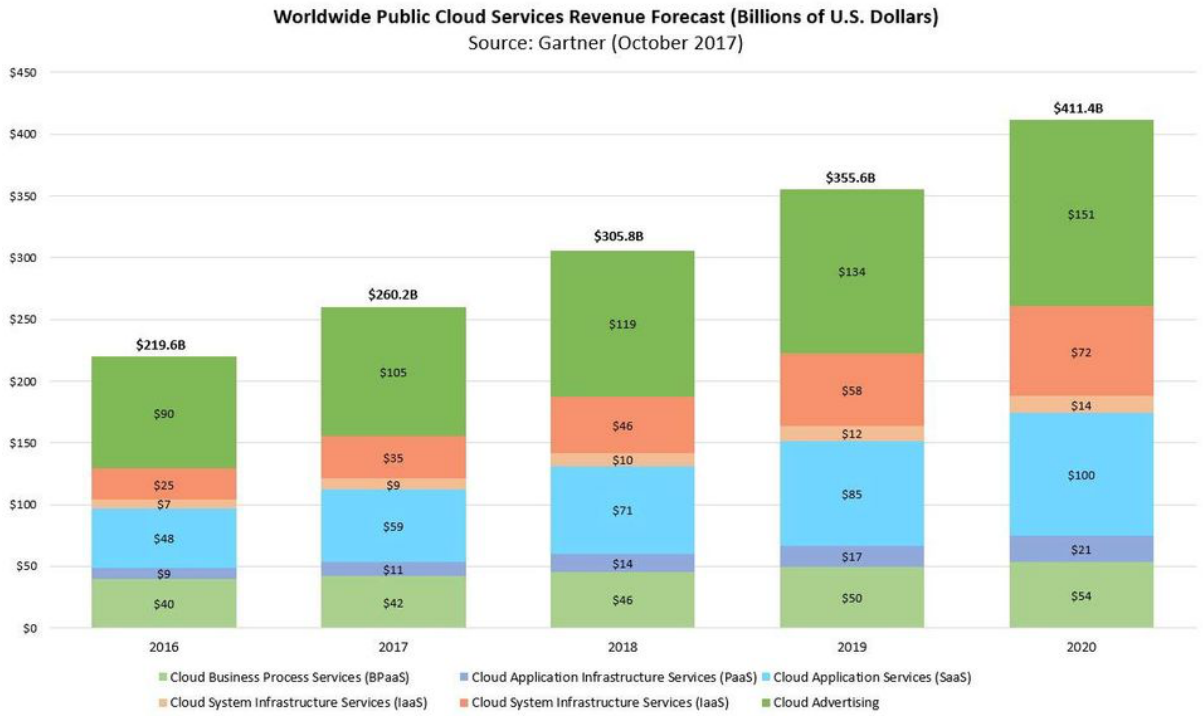
Thursday, September 13, 2018

*“You got to change your research area! Cloud was for the past decade.”*




Retrieved on 9/9/18 from https://trends.google.com/trends.

# Public Cloud Revenue Projected To Reach $411B By 2020



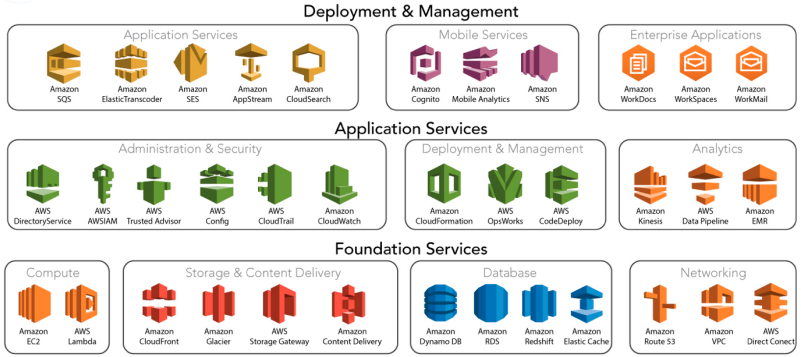

**$411B is not trivial!**

| Rank | Country | Nominal GDP |
|---|---|---|
| 26 | Iran | $ 439.5 B |
| 27 | Austria | $ 416.6 B |
| 28 | Norway | $ 398.8 B |
| 29 | UAE | $ 382.6 B |

Source: Louis Columbus, Cloud Computing Market Projected To Reach $411B By 2020, **Forbes**, Oct. 2017, https://www.forbes.com/sites/louiscolumbus/2017/10/18/cloud-computing-market-projected-to-reach-411b-by-2020

# Growing Services

AWS Services

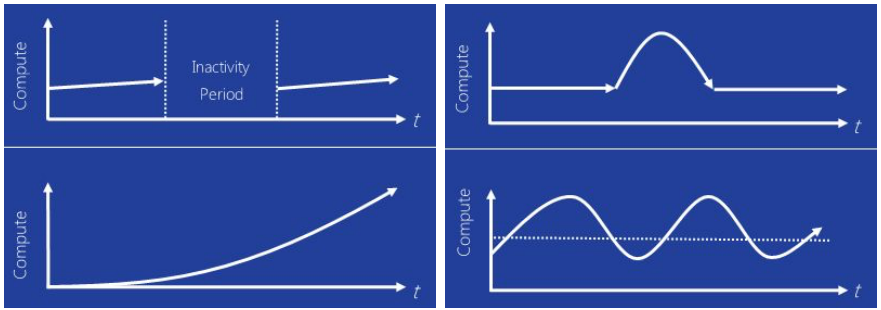



# Service Valuation

Santander

github:pages

# Usage Patterns




# Heterogeneous Infrastructure

Image sources: [1] https://www.clickittech.com/aws/why-aws-cloud/, [2] Thilina Gunarathne, Introduction to Windows Azure Cloud, https://slideplayer.com/slide/6619318/, [3] PHD Comics

# Flexibility is a must in the cloud.

# Various VM Types to Suit Everyone

**Amazon Elastic Compute Cloud (EC2) offers 106 instance types. (on 9/18)**

| Instance Family | Current Generation Instance Types |
|---|---|
| General purpose | t2.nano \| t2.micro \| t2.small \| t2.medium \| t2.large \| t2.xlarge \| t2.2xlarge \| t3.nano \| t3.micro \| t3.small \| t3.medium \| t3.large \| t3.xlarge \| t3.2xlarge \| m4.large \| m4.xlarge \| m4.2xlarge \| m4.4xlarge \| m4.10xlarge \| m4.16xlarge \| m5.large \| m5.xlarge \| m5.2xlarge \| m5.4xlarge \| m5.12xlarge \| m5.24xlarge \| m5d.large \| m5d.xlarge \| m5d.2xlarge \| m5d.4xlarge \| m5d.12xlarge \| m5d.24xlarge |
| Compute optimized | c4.large \| c4.xlarge \| c4.2xlarge \| c4.4xlarge \| c4.8xlarge \| c5.large \| c5.xlarge \| c5.2xlarge \| c5.4xlarge \| c5.9xlarge \| c5.18xlarge \| c5d.xlarge \| c5d.2xlarge \| c5d.4xlarge \| c5d.9xlarge \| c5d.18xlarge |
| Memory optimized | r4.large \| r4.xlarge \| r4.2xlarge \| r4.4xlarge \| r4.8xlarge \| r4.16xlarge \| r5.large \| r5.xlarge \| r5.2xlarge \| r5.4xlarge \| r5.12xlarge \| r5.24xlarge \| r5d.large \| r5d.xlarge \| r5d.2xlarge \| r5d.4xlarge \| r5d.12xlarge \| r5d.24xlarge \| x1.16xlarge \| x1.32xlarge \| x1e.xlarge \| x1e.2xlarge \| x1e.4xlarge \| x1e.8xlarge \| x1e.16xlarge \| x1e.32xlarge \| z1d.large \| z1d.xlarge \| z1d.2xlarge \| z1d.3xlarge \| z1d.6xlarge \| z1d.12xlarge |
| Storage optimized | d2.xlarge \| d2.2xlarge \| d2.4xlarge \| d2.8xlarge \| h1.2xlarge \| h1.4xlarge \| h1.8xlarge \| h1.16xlarge \| i3.large \| i3.xlarge \| i3.2xlarge \| i3.4xlarge \| i3.8xlarge \| i3.16xlarge \| i3.metal |
| Accelerated computing | f1.2xlarge \| f1.16xlarge \| g3.4xlarge \| g3.8xlarge \| g3.16xlarge \| p2.xlarge \| p2.8xlarge \| p2.16xlarge \| p3.2xlarge \| p3.8xlarge \| p3.16xlarge |

$$\frac{\sim 2\ GB\ Mem}{1\ Virtual\ CPU}$$

$$\frac{[8, 30]\ GB\ Mem}{1\ Virtual\ CPU}$$




Source: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-types.html

# Current cloud offerings are far from fully flexible!

**All-or-nothing Service-level Agreements (SLAs):**

- Uptime guarantees [ SoCC ’16 ]
- Isolation mechanisms

**Lack of provider/client information sharing:**

- Demand uncertainty [ SoCC ’17 ]
- Application demand

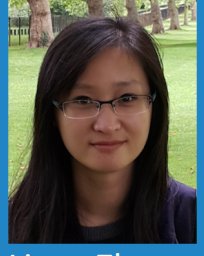
# 1.

# Incentivizing Self-Capping

## to Increase Cloud Utilization



Cristian Klein


Liang Zheng


Mung Chiang


Erik Elmroth


David Wentzlaff

PRINCETON UNIVERSITY

UMEÅ UNIVERSITY

PURDUE UNIVERSITY

# A Quick Survey

How much is the installment cost of a typical cloud datacenter?

(A) O($1M) (B) O($10M) (C) O($100M) ✓

What is the typical CPU utilization of cloud servers?

(A) ~10% ✓

Latency-sensitive interactive services

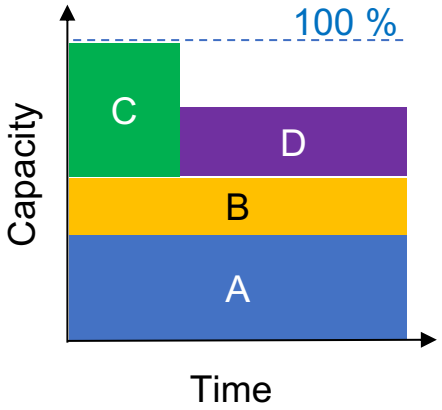
(B) ~30% ✓

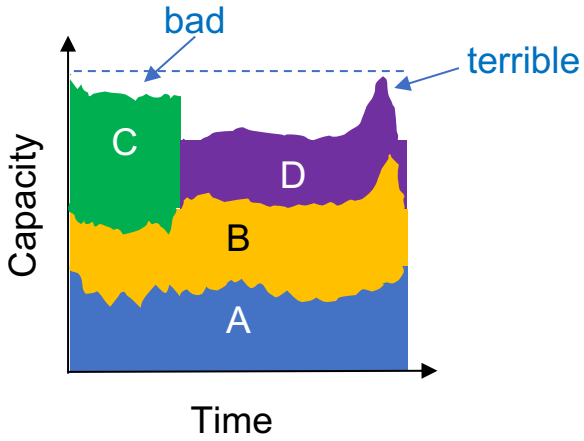
Mix of workloads including online services

(C) ~75% ✓

Large continues batch workloads

# Managing Uncertainty is Fundamentally Challenging



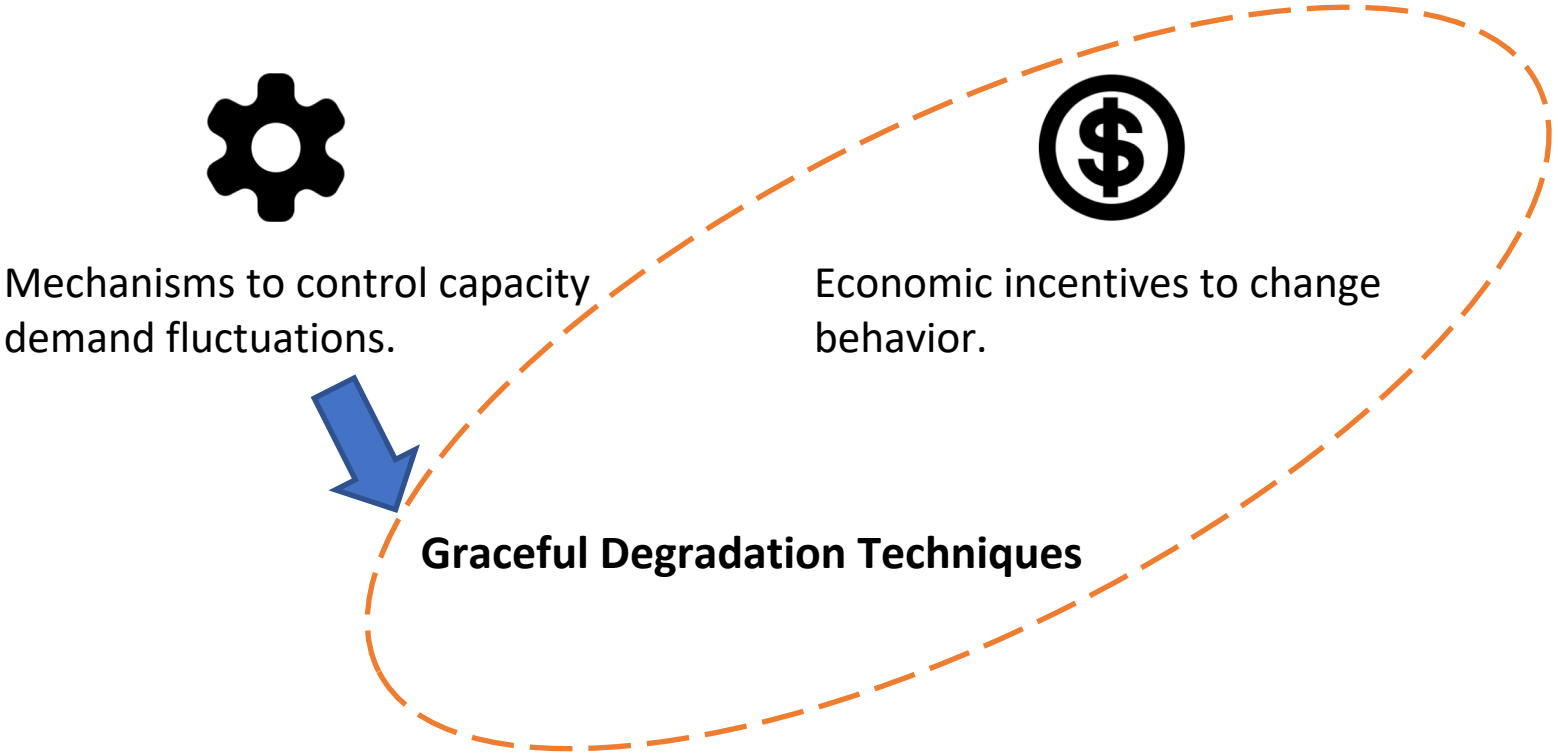




Demand Fluctuations

QoS guarantees

→ **Spare Capacity**

# Can we motivate tenants to fluctuate less?



# Graceful Degradation (GD) Methodology

## Adaptive Bitrate Streaming

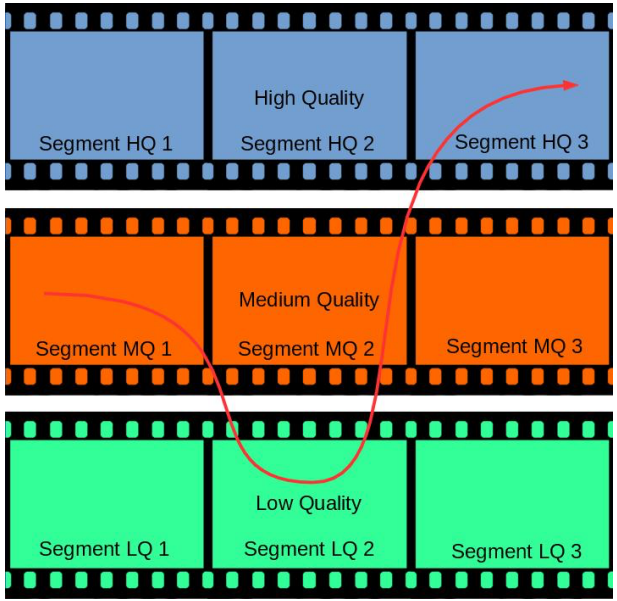
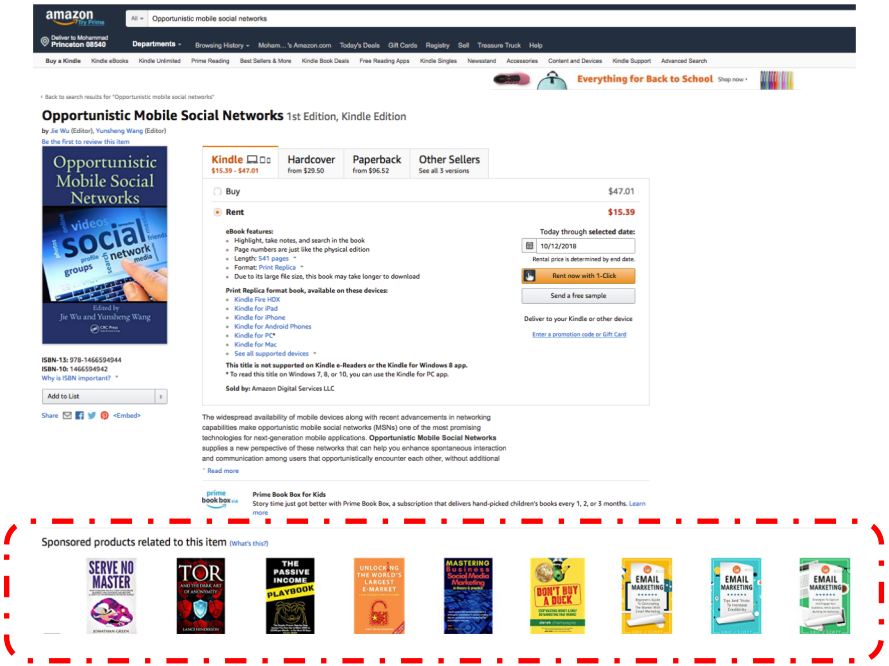



Image source: https://www.ubik-ingenierie.com/blog/adaptive-bitrate-streaming-watching-videos/

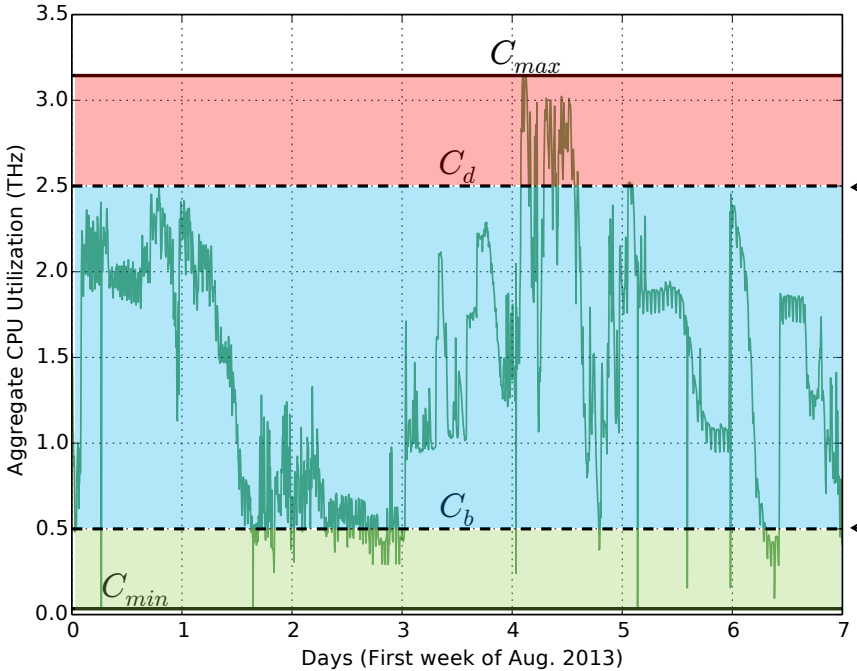
## Brownout Self-Adaptation

Deactivating non-essential content to maintain response time.



# Shaping the Capacity Demand

Capacity demand variation of a real service provider (Bitbrains) using 1,250 VMs.




Capacity pair ( $C_b$, $C_d$ )
Price pair ($p_b$, $p_d$ )

Total Deliverable Capacity

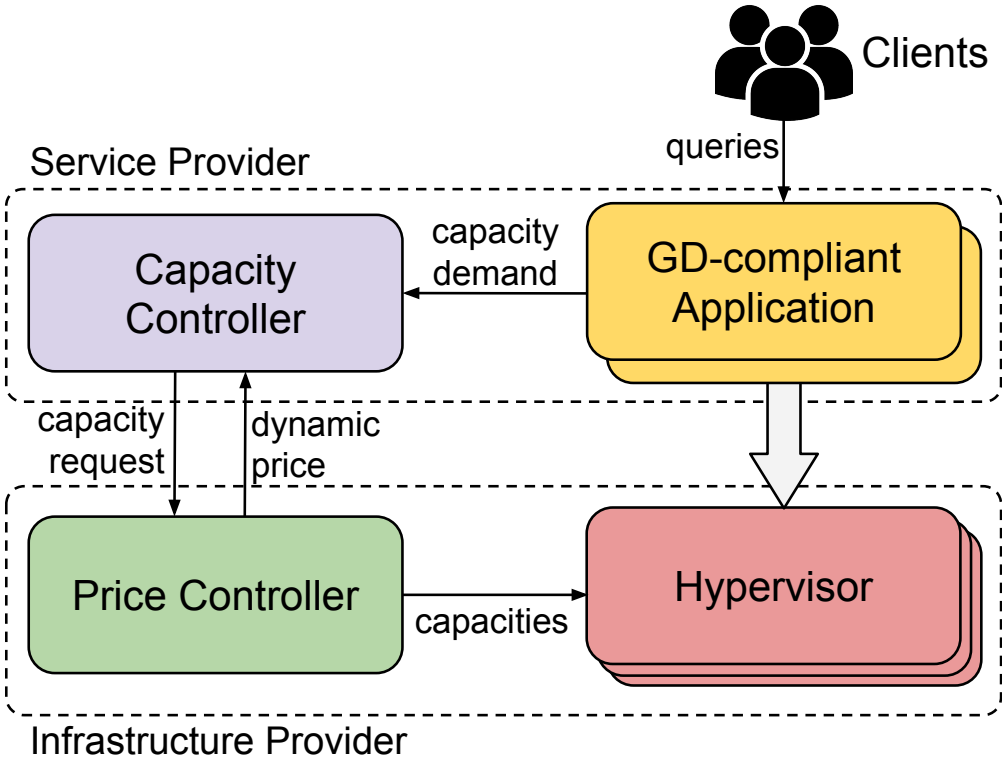
Not delivered (too expensive)

Charged based on usage ($p_d$)

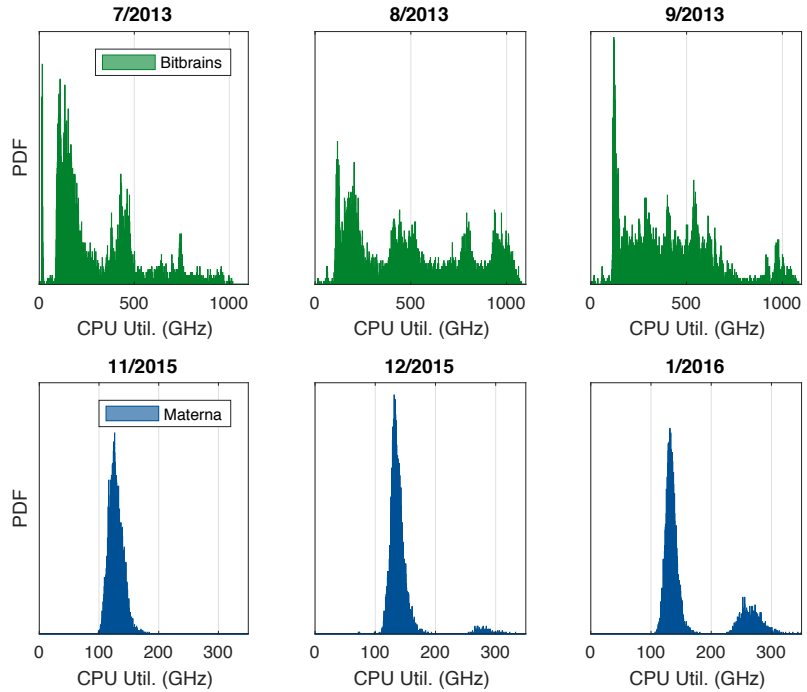
Reserved Base Capacity

Always Charged ($p_b$)

# System Overview



# Demand Distributions Generally Consistent



# Tenant's Profit Maximization

Given a price pair, tenants can select the best capacity pair:

Optimal Reserved Capacity

$$c_b^\star = \int_{c_b^\star}^{c_{max}} f(c)\mathrm{d}c = \frac{p_b}{p_d},$$

Optimal Capacity Limit

$$c_d^\star = \frac{\int_{c_d^\star}^{c_{max}} kR(c, c_d^\star/c) f(c)\mathrm{d}c}{\int_{c_d^\star}^{c_{max}} p_d f(c)\mathrm{d}c}$$

Price

Demand Distribution

Revenue (Valuation)

$k$: Degree of homogeneity for revenue function $\quad R(c, \lambda\theta) = \lambda^k R(c, \theta)$

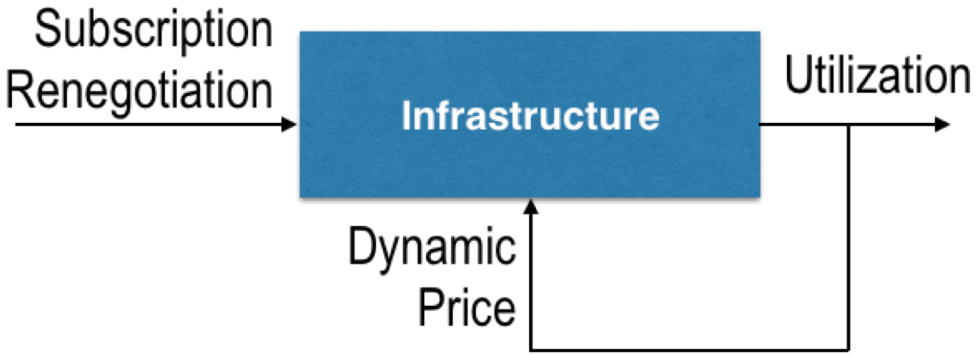
# Dynamic Pricing to Control Utilization

Regardless of revenue function:

$$\textbf{\textit{Reserved Capacity}} \propto \frac{p_d}{p_b}$$

$$\textbf{\textit{Total Capacity}} \propto \frac{1}{p_d}$$

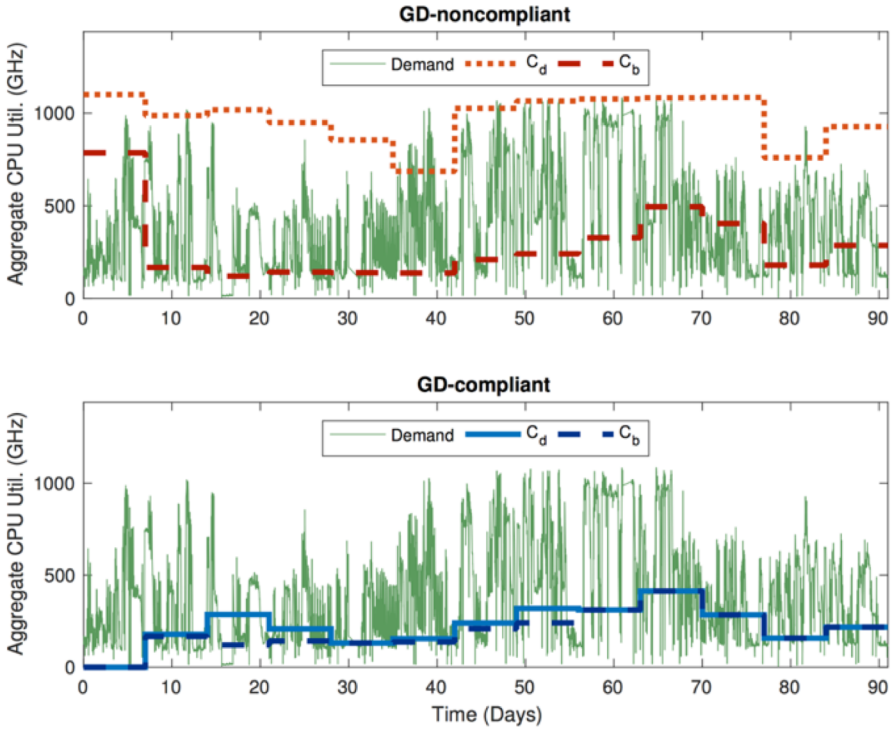
**Monolithic incentives**

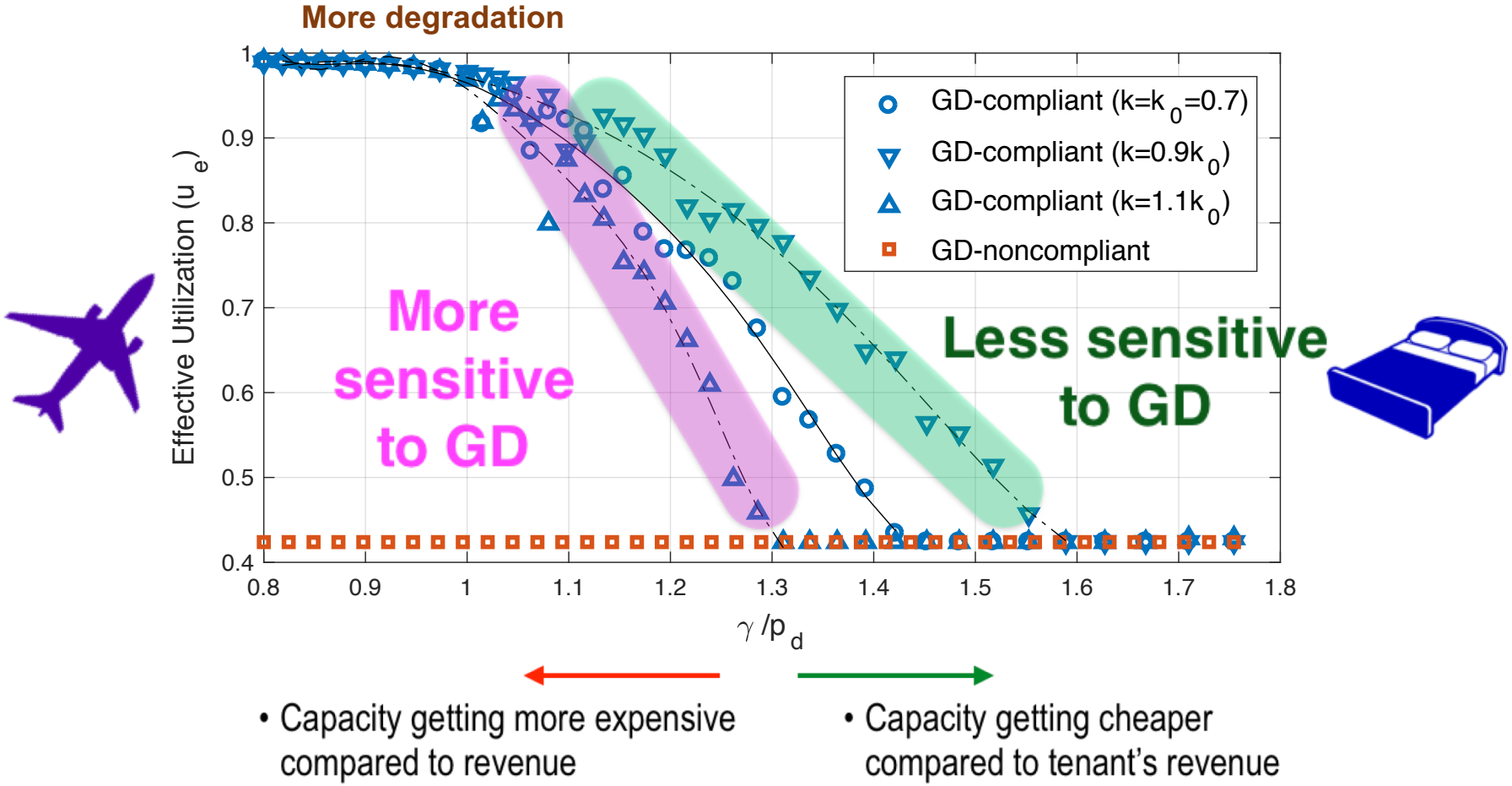


# Evaluation

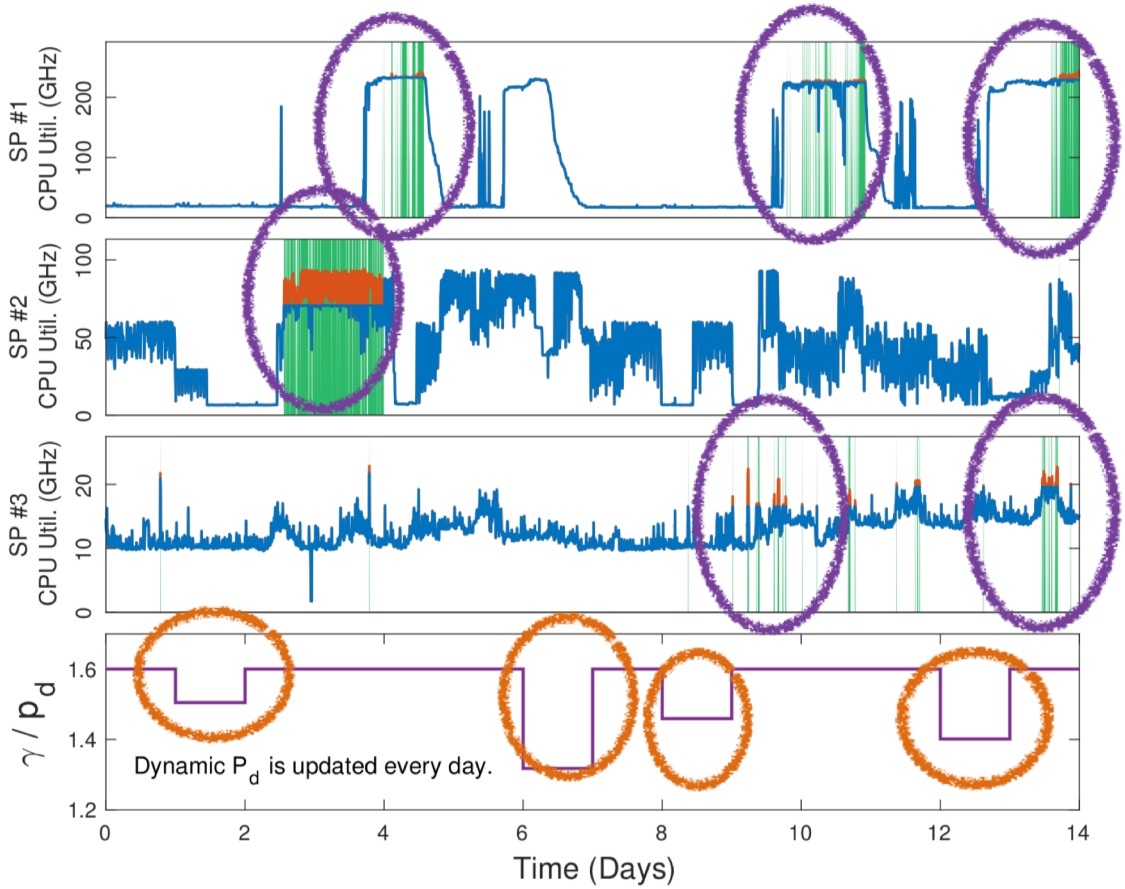
- Simulations using real-world traces
  - Fast design space exploration
- Implemented and tested a prototype
  - Validation of simulations
  - Scalability tests

| | Cutting Peaks | Filling Valleys |
|---|---|---|
| GD-compliant | ✓ | ✓ |
| GD-noncompliant | ✗ | ✓ |




More degradation
Effective Utilization ($u_e$)
$\gamma / p_d$
GD-compliant (k=$k_0$=0.7)
GD-compliant (k=0.9$k_0$)
GD-compliant (k=1.1$k_0$)
GD-noncompliant
More sensitive to GD
Less sensitive to GD
• Capacity getting more expensive compared to revenue
• Capacity getting cheaper compared to tenant's revenue

# A Multi-tenant Scenario



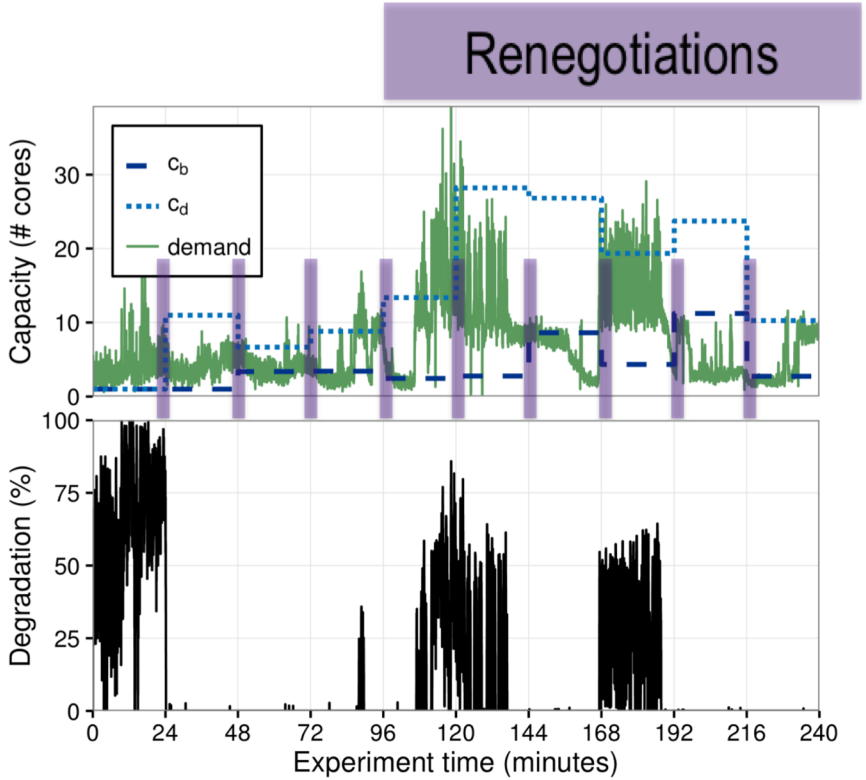

**More degradation**

**Increased on-demand price**

# Prototype Evaluation

- Used **Xen** hypervisor (best CPU scaling)
- GD-enabled RUBiS (eBay-like benchmark)
- Scaled down traces in two dimensions:
  - Time-wise (60X faster)
  - Magnitude-wise

    (50,000X less, to fit in 32 cores)




**https://github.com/cristiklein/gdinc-experiment**

# Takeaways

**Demand uncertainty** decreases utilization.

A flexible service model can **incentivize** tenants to fluctuate less.

**Graceful degradation** resilience methodology can be used.

A well-defined pricing model allows:

- Profit maximization for tenants;
- Control utilization for infrastructure providers.

# 2.

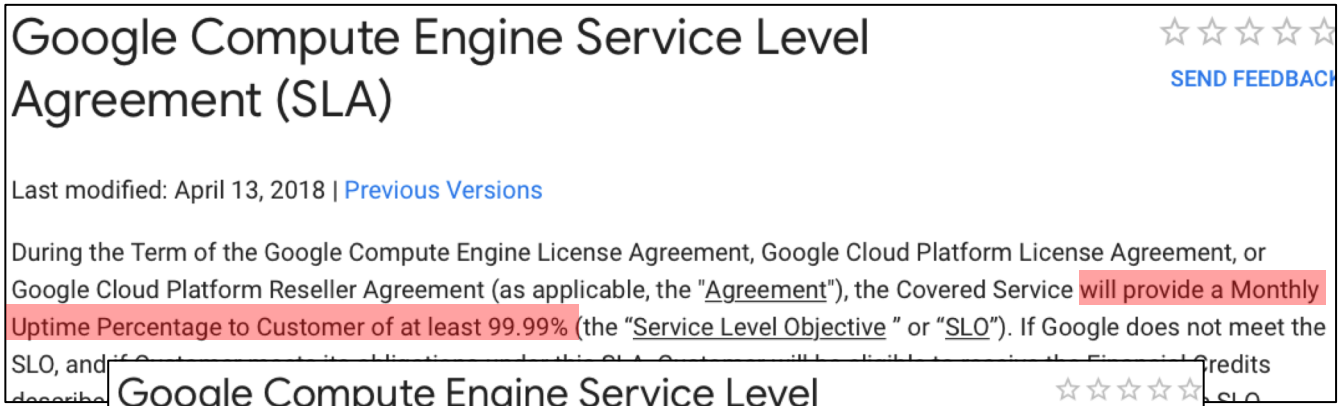
# Availability Knob

## Flexible User-defined Availability in the Cloud

David Wentzlaff

# State of Uptime Service Level Objectives (SLO)

Downtime: 4min 20sec per month



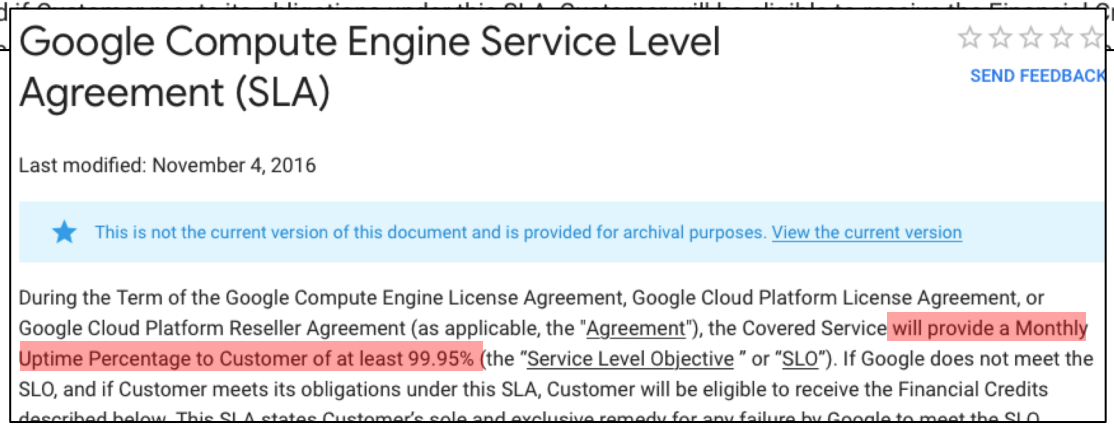
Google Compute Engine Service Level Agreement (SLA)

SEND FEEDBACK

Last modified: April 13, 2018 | Previous Versions

During the Term of the Google Compute Engine License Agreement, Google Cloud Platform License Agreement, or Google Cloud Platform Reseller Agreement (as applicable, the "Agreement"), the Covered Service will provide a Monthly Uptime Percentage to Customer of at least 99.99% (the "Service Level Objective " or "SLO"). If Google does not meet the SLO, and

Downtime: 21min 40sec per month



Google Compute Engine Service Level Agreement (SLA)

SEND FEEDBACK

Last modified: November 4, 2016

This is not the current version of this document and is provided for archival purposes. View the current version

During the Term of the Google Compute Engine License Agreement, Google Cloud Platform License Agreement, or Google Cloud Platform Reseller Agreement (as applicable, the "Agreement"), the Covered Service will provide a Monthly Uptime Percentage to Customer of at least 99.95% (the "Service Level Objective " or "SLO"). If Google does not meet the SLO, and if Customer meets its obligations under this SLA, Customer will be eligible to receive the Financial Credits

# In case the uptime SLO not met, …


| Monthly Uptime Percentage | Service Credit Percentage |
|---|---|
| Less than 99.99% but equal to or greater than 99.0% | 10% |
| Less than 99.0% | 30% |


| Monthly Uptime Percentage | Percentage of monthly bill for the respective Covered Service in the Region affected which did not meet SLO that will be credited to future monthly bills of Customer |
|---|---|
| 99.00% - < 99.99% | 10% |
| 95.00% - < 99.00% | 25% |
| < 95.00% | 50% |


| MONTHLY UPTIME PERCENTAGE | SERVICE CREDIT |
|---|---|
| < 99.99% | 10% |
| < 99% | 25% |
| < 95% | 100% |

Is there also a dark side to cloud industry's achievement culture?

THE DARK SIDE OF AMERICA'S ACHIEVEMENT CULTURE

# What's wrong with fixed high availability?

Cloud customers:

- Various downtime demands
- Different service valuations



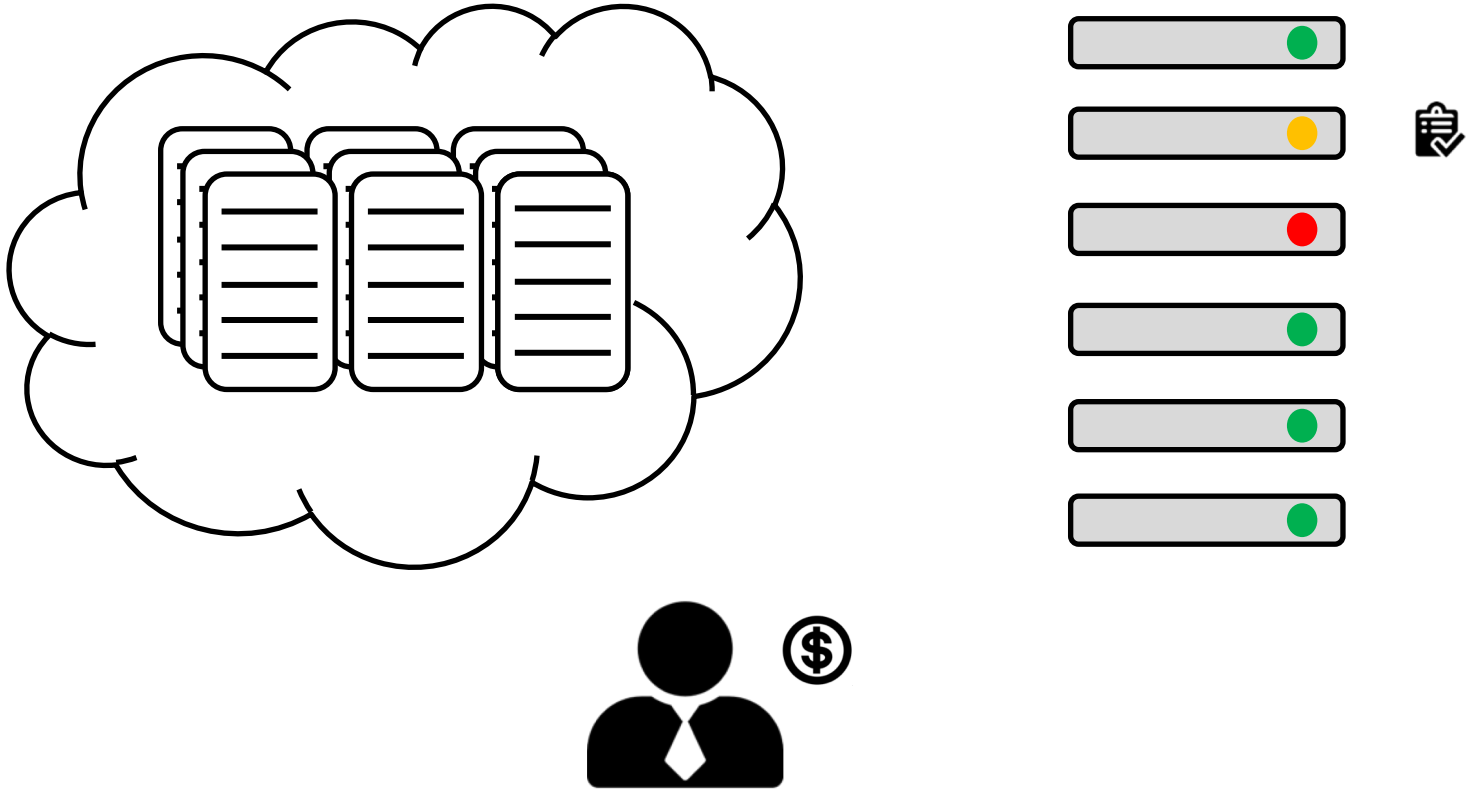
Adobe Creative Cloud

Heterogeneous cloud infrastructure

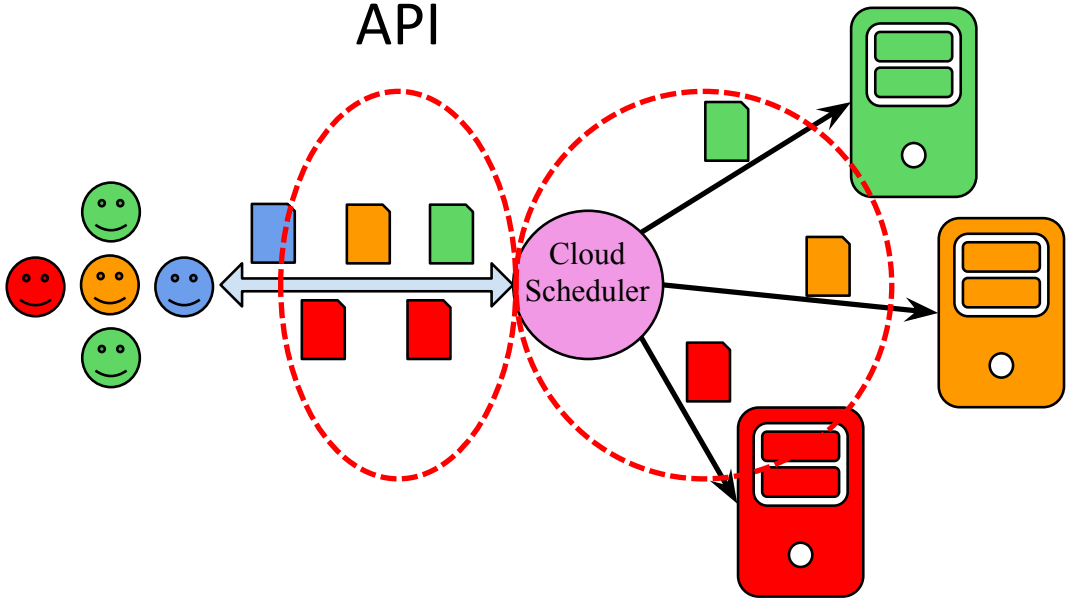
- Network redundancy
- Server type (CPU, storage, etc.)

Redundancy is not free!

# The Availability Knob (AK)



# Changes Required to Support AK



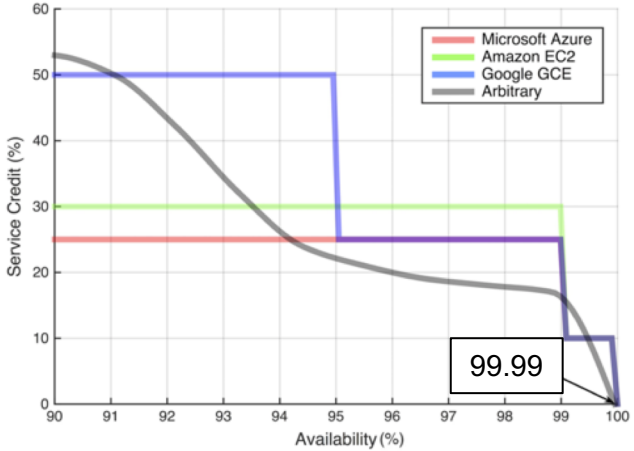

Management { Failure stats, Scheduling

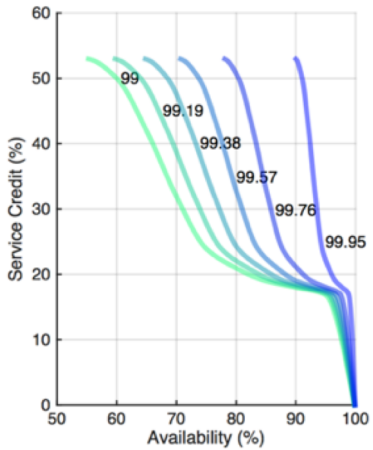
# Flexible Availability SLAs

1. Desired Availability & Calculation Period
(e.g., 99.8% / 7 days)

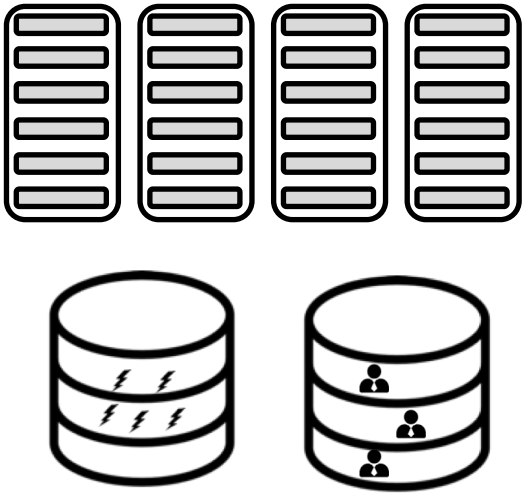
2. Availability Price Scale
e.g., (99.99%, 1), (99.9%, 0.95)

3. Variable Service Credit



# The AK Scheduler




Scheduling policy:

For candidate servers to host a VM, find the cheapest resource so that expected time to next failure meets requested downtime.

Extra run-time policies:

- Benign VM Migration
- Deliberate Downtime

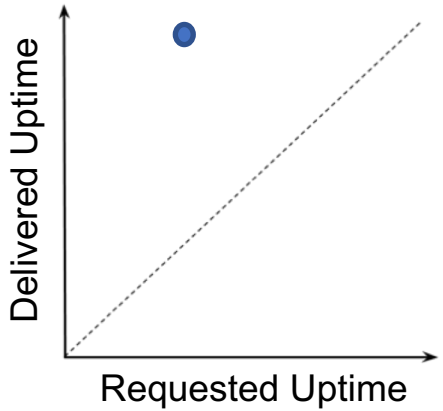
**Benign VM Migration**

Periodic migration of over-served VMs to cheaper resources.

**Deliberate Downtime**

Deliberately fail VMs near the end of their period to:

- Build market incentives
- Bid redeemed resources



# Incentive Compatibility

Providers can:

- Neglect meeting SLOs

Customers can:

- Run buggy applications
- Cause deliberate downtime
- Ask for unreal uptime

AK uses **game theory** to build a pricing scheme:

1. Providers maximize profit by not violating SLOs
2. Clients pay less by asking for their true demands

# Evaluation of AK

Infrequency of Failures

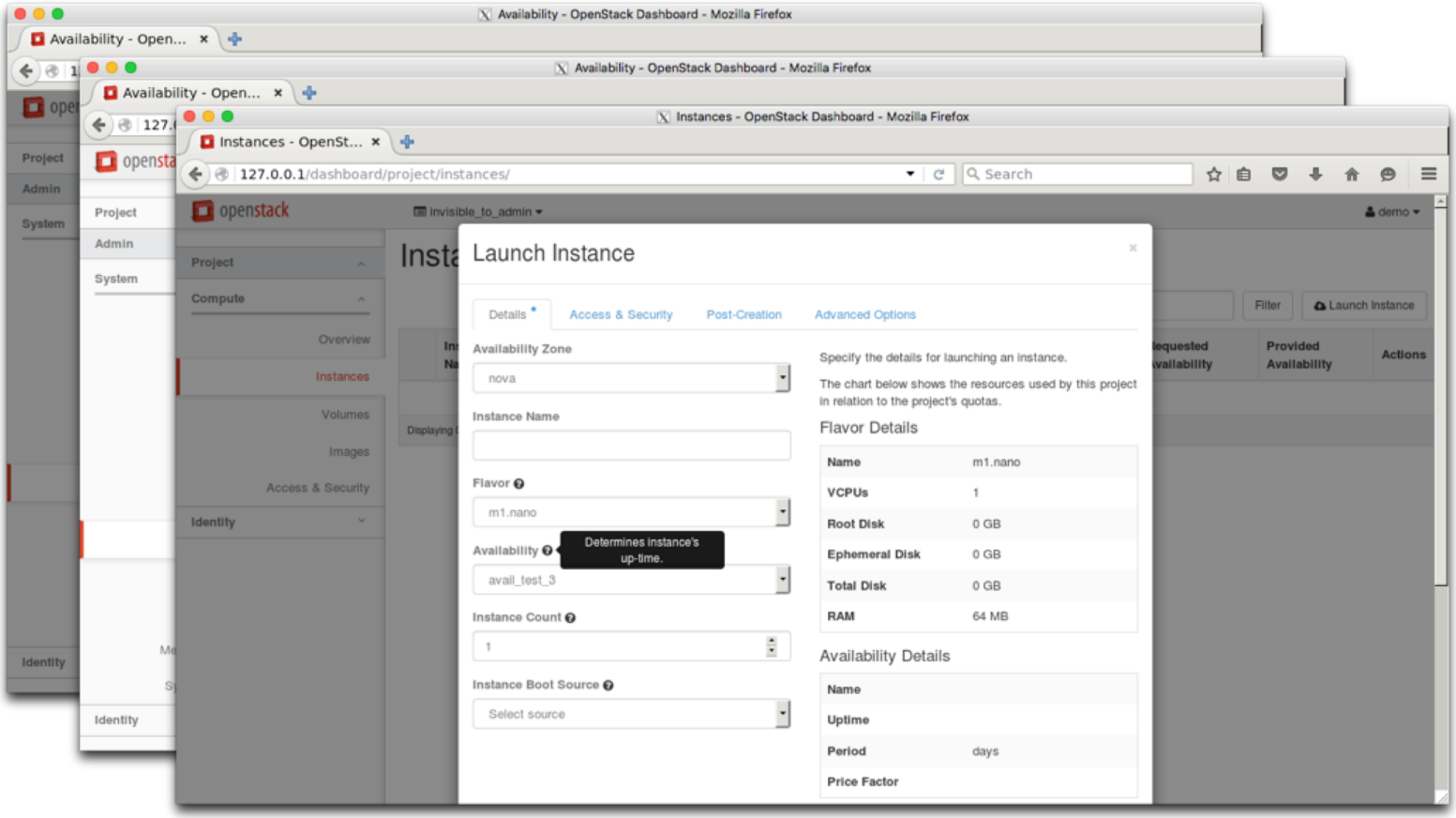
Accelerated testing

Large scale simulations

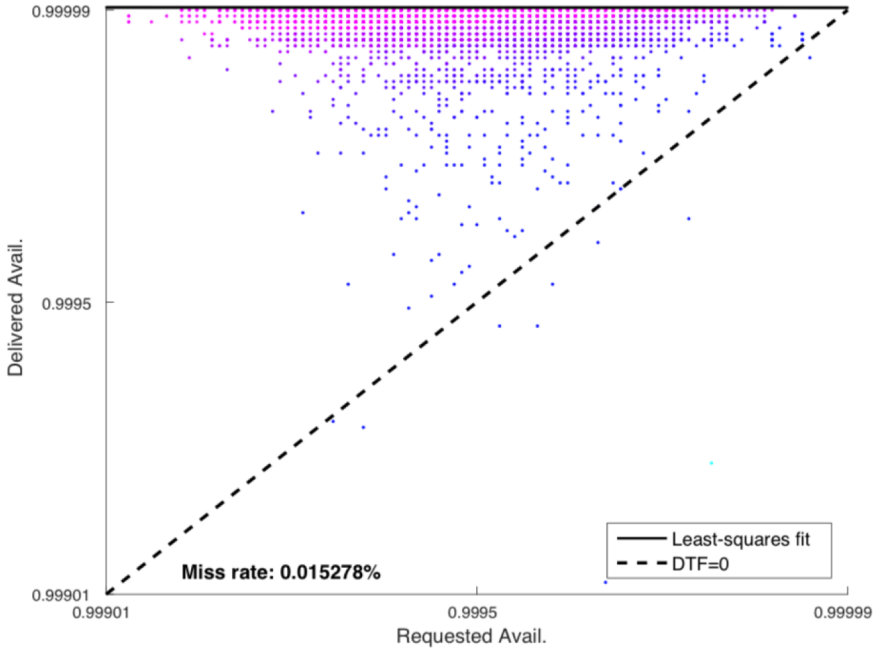

1. Stochastic simulations in MATLAB

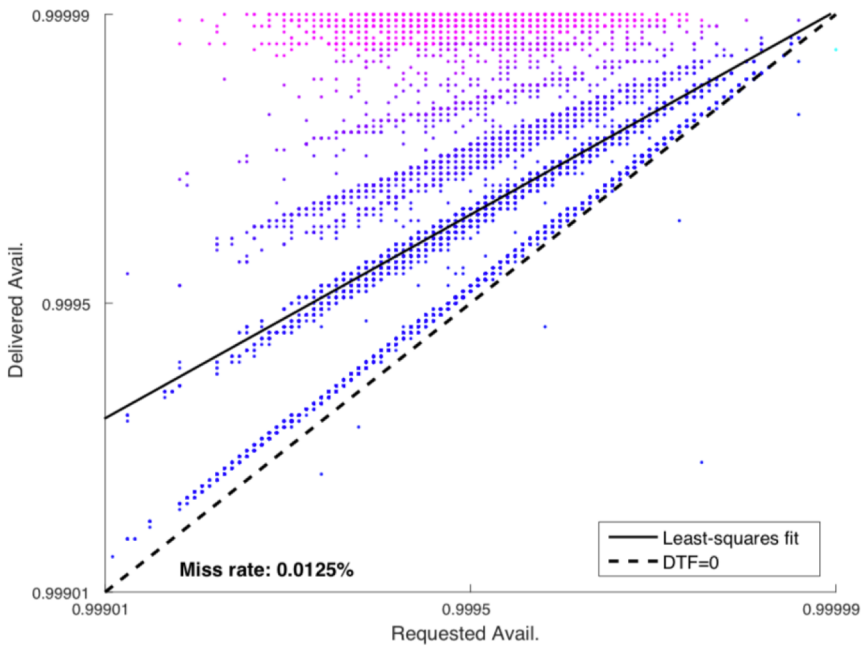
2. Prototype implementation with OpenStack

# A Peek into the OpenStack Prototype



# Availability-aware Scheduler

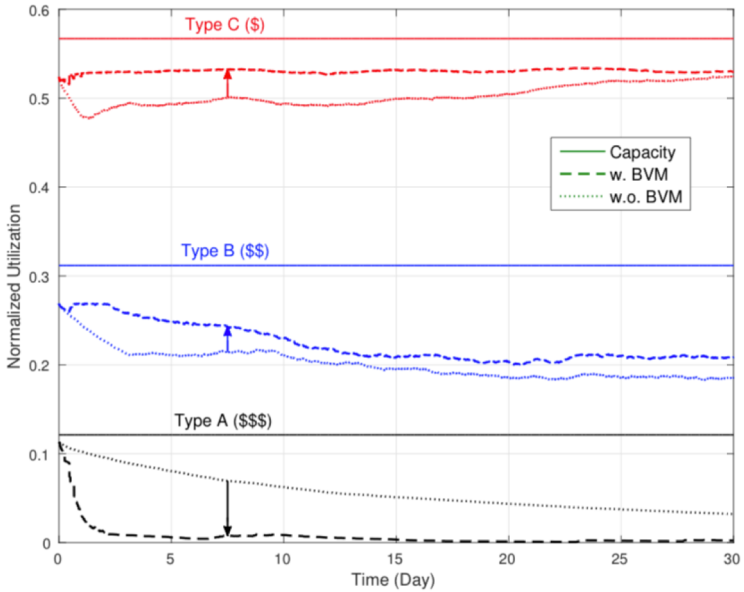


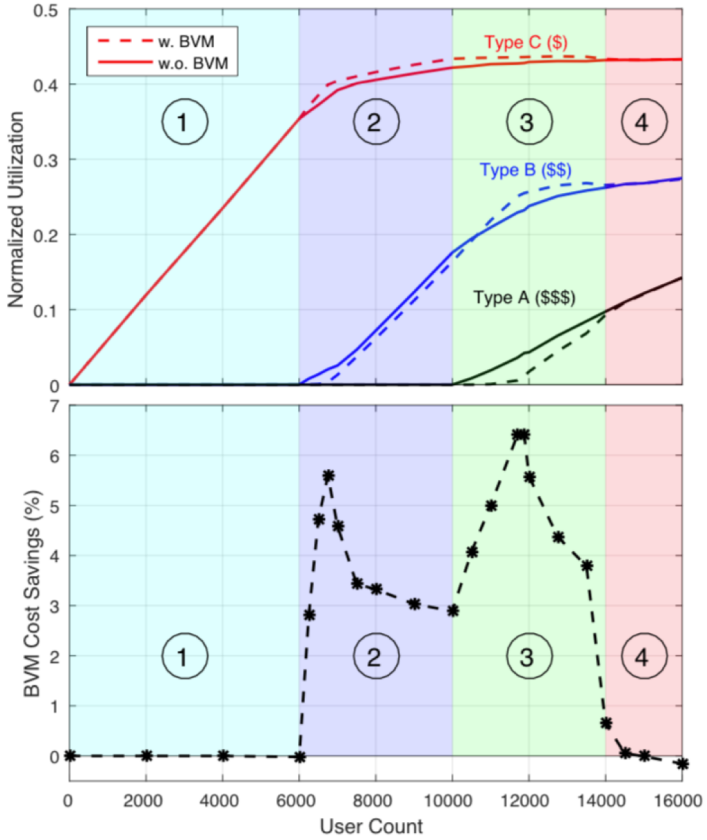




**With Deliberate Downtime**

1000 machines, 12000 users, Normal demand distribution in [99.9, 99.999],
6 months long, BVM every 1hr for top 10% of over-served clients

# Benign Migrations Offload VMs to Cheaper Machines



# Takeaways

Supply chain flexibility → market efficiency

Knowing user demand can enable better management.

Game theory to ensure mutual economic incentives.

Leveraging reliability/cost trade-offs

# There are numerous inefficiency corner cases in the cloud, but …

*Big guys can't scratch their back!*

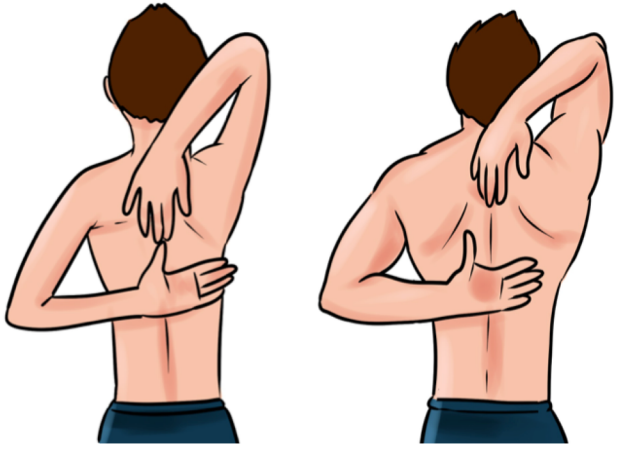


Image source: https://www.kaa-yaa.com/how-to-avoid-shoulder-imbalances-and-improve-shoulder-flexibility-to-avoid-injuries/big-guys-back-1/