Improving the Efficiency of Public Clouds through Enhanced Service Flexibility

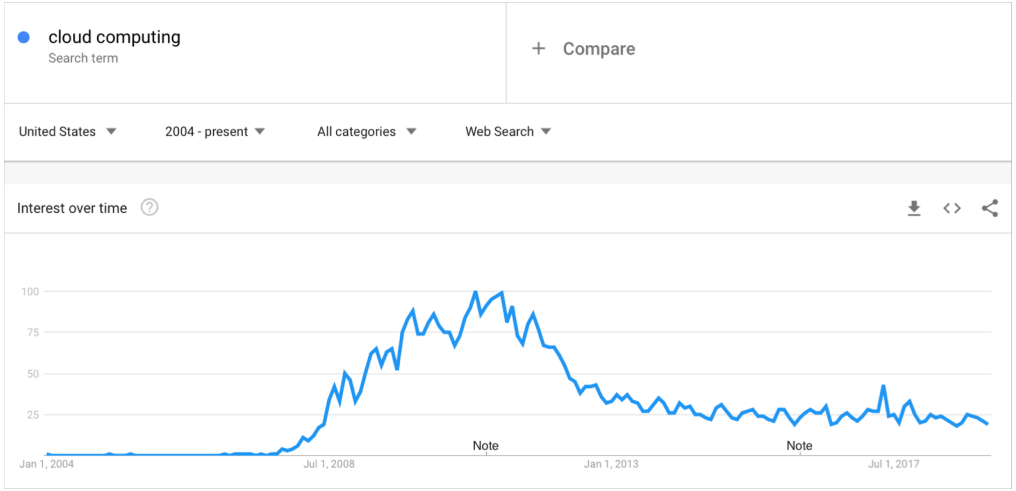
Mohammad Shahrad

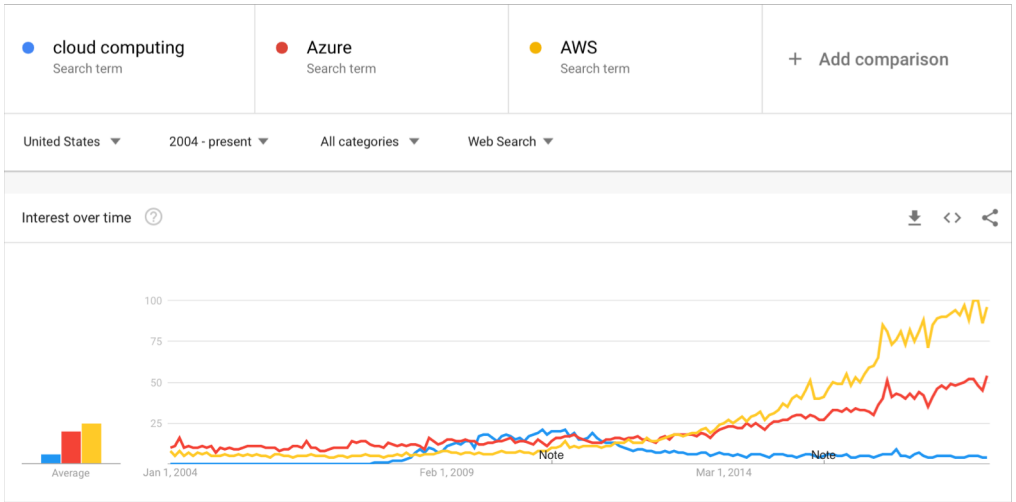
PRINCETON UNIVERSITY

PRINCETON
School of Engineering and Applied Science

Thursday, January 24, 2019

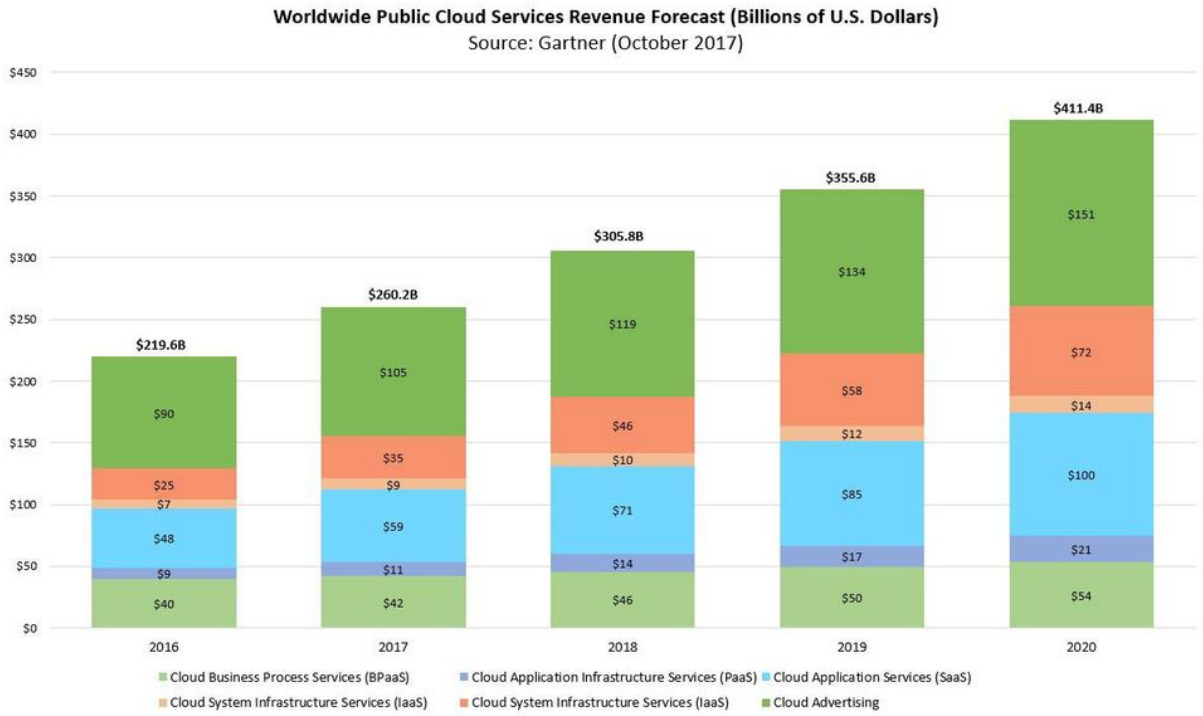
“You got to change your research area! Cloud was for the past decade.”





Retrieved on 1/21/19 from https://trends.google.com/trends.

Public Cloud Revenue Projected To Reach $411B By 2020

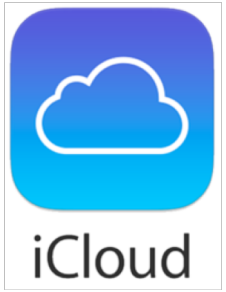


$411B is not trivial!

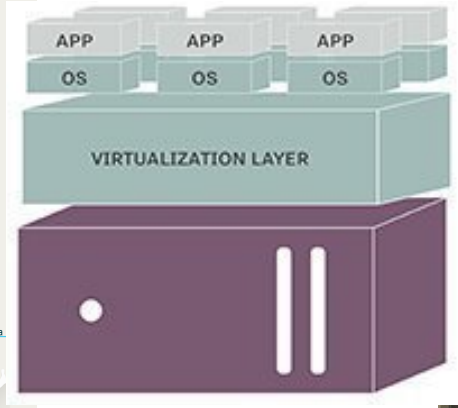
Rank	Country	Nominal GDP
26	Iran	$ 439.5 B
27	Austria	$ 416.6 B
28	Norway	$ 398.8 B
29	UAE	$ 382.6 B

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

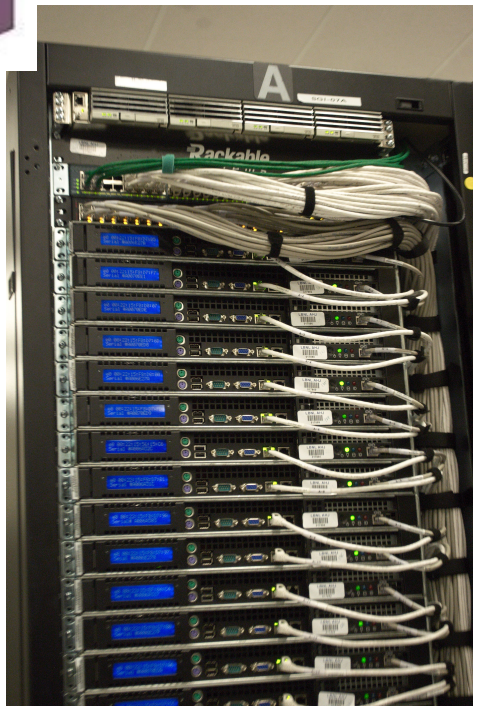
Prevalence of Cloud Services



Global Network of Data Centers



Virtualized Servers



Rack Full of Servers

Data Center

Aisles of Racks

Image sources: [1] https://upload.wikimedia.org/wikipedia/commons/3/3c/KVH_Tokyo_Data_Center_2.png
[2] https://upload.wikimedia.org/wikipedia/commons/2/2e/123Net_Data_Center_%28DC2%29.jpg
[3] https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcQjdXJUXGN__odNPl2ar7SBPBZwaHLguSbHweHSRTPQUslrdQYmDQ
[4] http://www.datacenterknowledge.com/sites/datacenterknowledge.com/files/wp-content/uploads/2016/09/aws-azure-dc-map.png

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

Cloud providers care about maximizing profit.

Maintain relevance

Maximize return on investment (ROI)

Efficiency

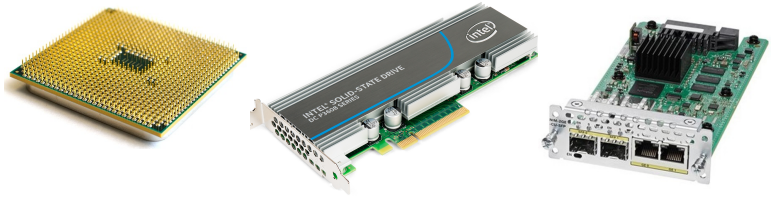
Efficiency Across the Stack

Applications

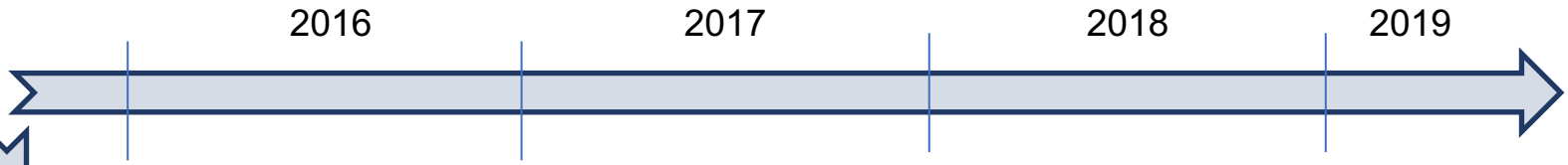
Less control in public clouds

Platform
(Scheduling, Management, etc.)

Infrastructure



My Cloud Efficiency Journey



Hardware Architecture

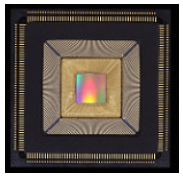
Systems

System Architecture

Mathematical Models

OpenPiton
[ASPLOS '16]

Piton
[IEEE Micro]



Piton Power
[HPCA '18]

LLC Allocation to VMs
[In Submission]

Architectural Implications of Serverless
[In Submission]

Incentivizing Self-Capping
[SoCC '17]

Availability Knob
[SoCC '16]

Using Old Smartphones in the Cloud
[HotCloud '17]

Modeling Burstable VMs
[INFOCOM '19]

A manageable degree of

Service flexibility can improve efficiency.

Case 1: VM availability offerings

Case 2: Flexible capacity pricing to shape resource usage

Case 1: VM Uptime

We know:

- Users' uptime valuation vary

- Heterogeneous infrastructure

- Delivering high uptime costs more

Case 1: VM Uptime

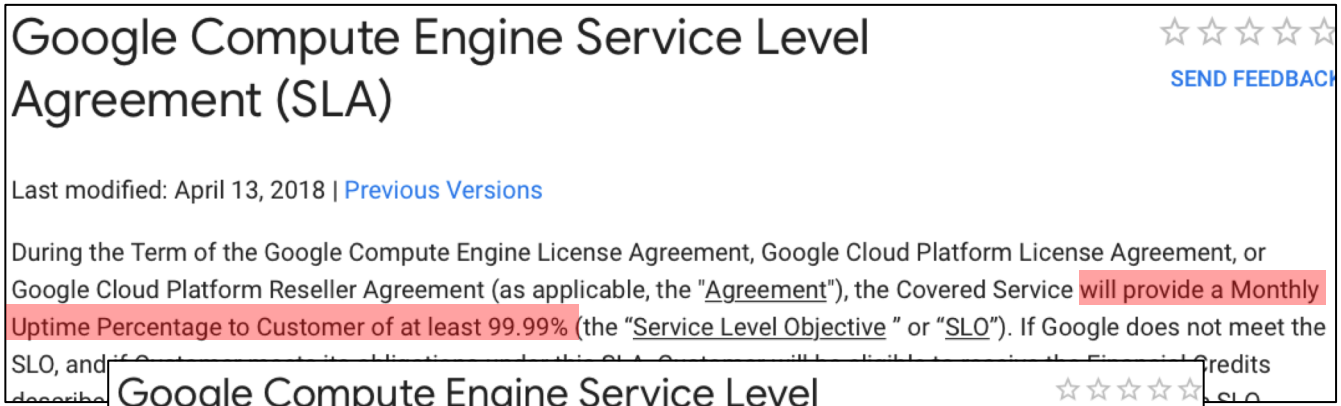
Solution?

1. Deliver fix uptime to everyone, and increase that fix value every year.

2. Have an optional API for users to express their uptime demand, charge them accordingly.

State of Uptime Service Level Objectives (SLO)

Downtime: 4min 20sec per month



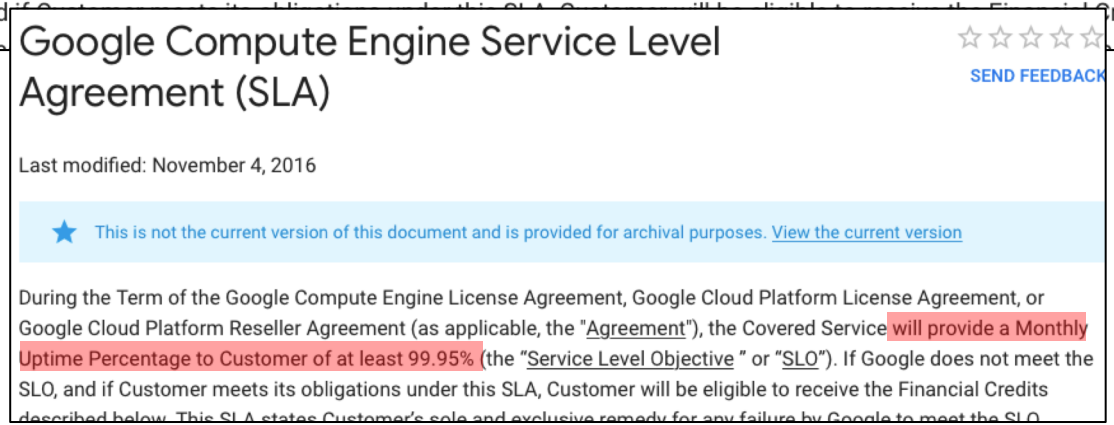
Google Compute Engine Service Level Agreement (SLA)

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

And they all do it.

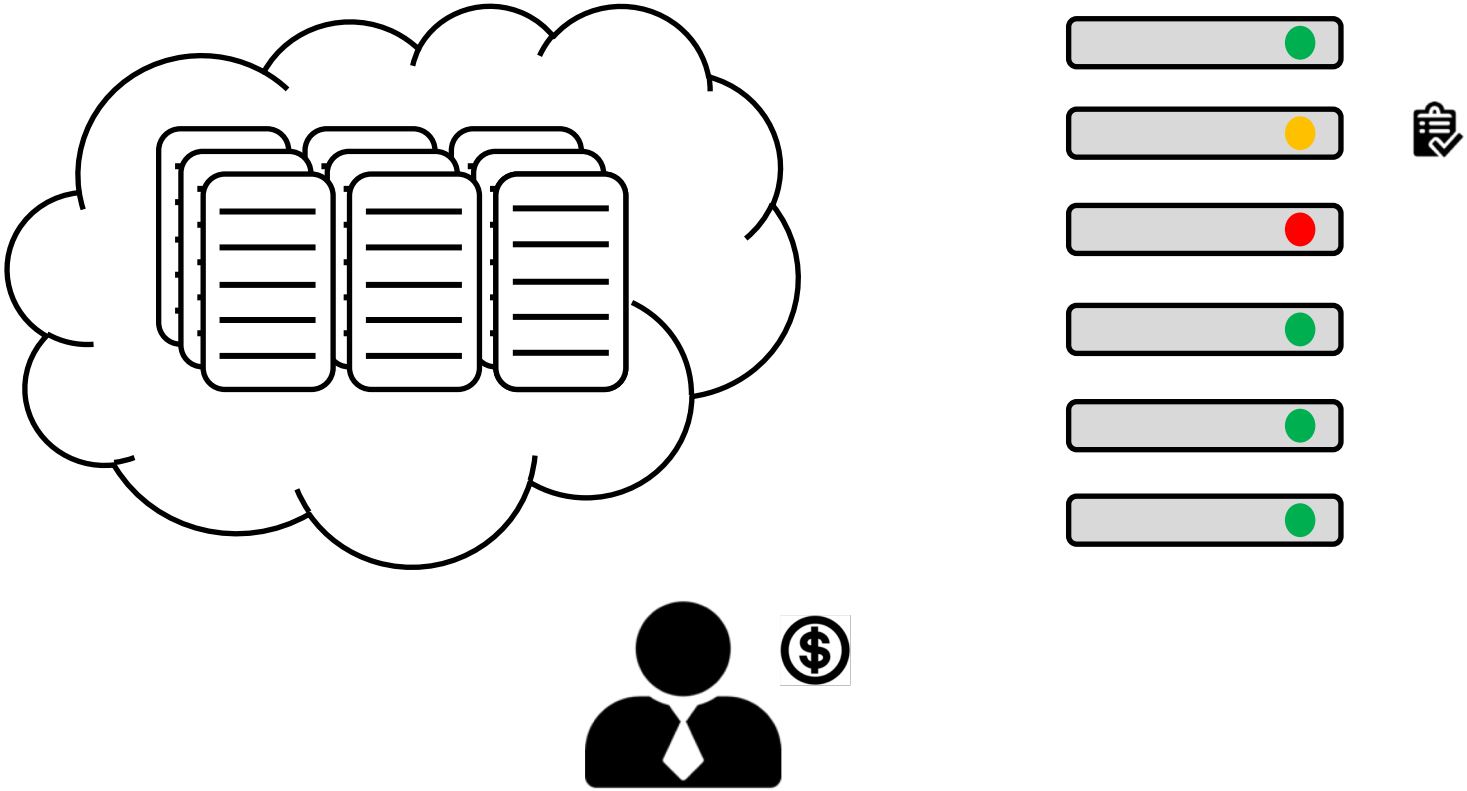
Monthly Uptime Percentage	Service Credit Percentage
Less than 99.99% but equal to or greater than 99.0%	10%
Less than 99.0%	30%

Google Cloud Platform

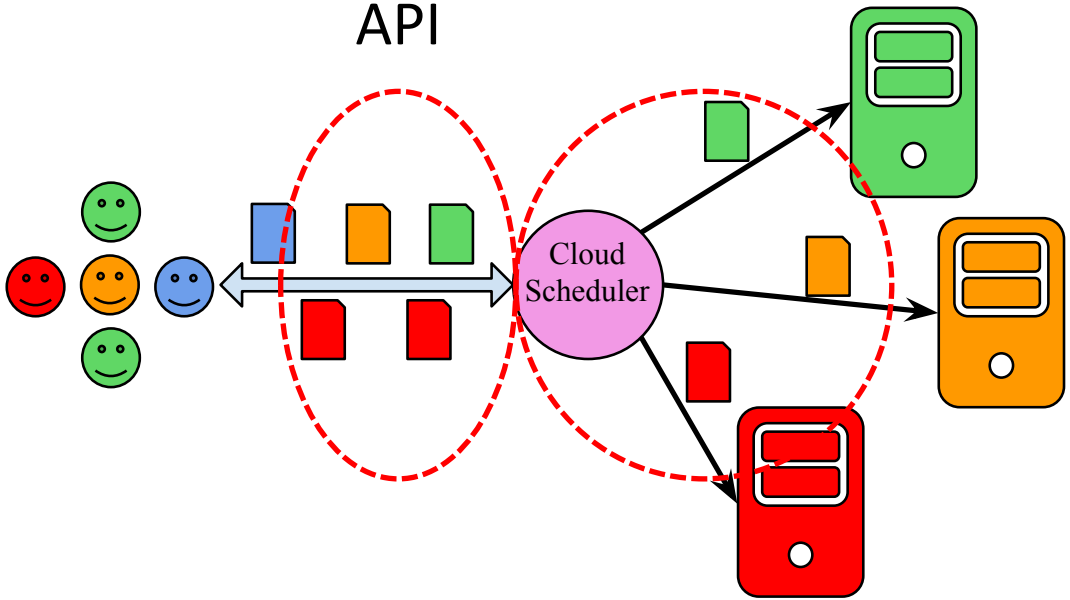
Monthly Uptime Percentage	Percentage of monthly bill for the respective Covered Service in the Region affected which did not meet SLO that will be credited to future monthly bills of Customer
99.00% - < 99.99%	10%
95.00% - < 99.00%	25%
< 95.00%	50%

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

The Availability Knob (AK)

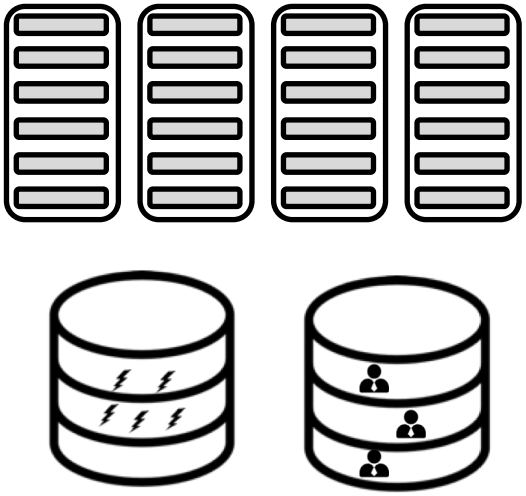


Changes Required to Support AK



Management { Failure stats, Scheduling

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

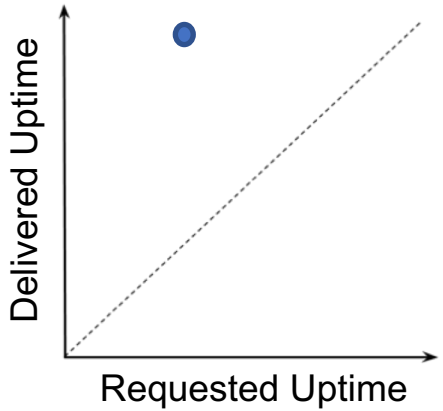
Benign VM Migration

Periodic migration of over-served VMs to cheaper resources.

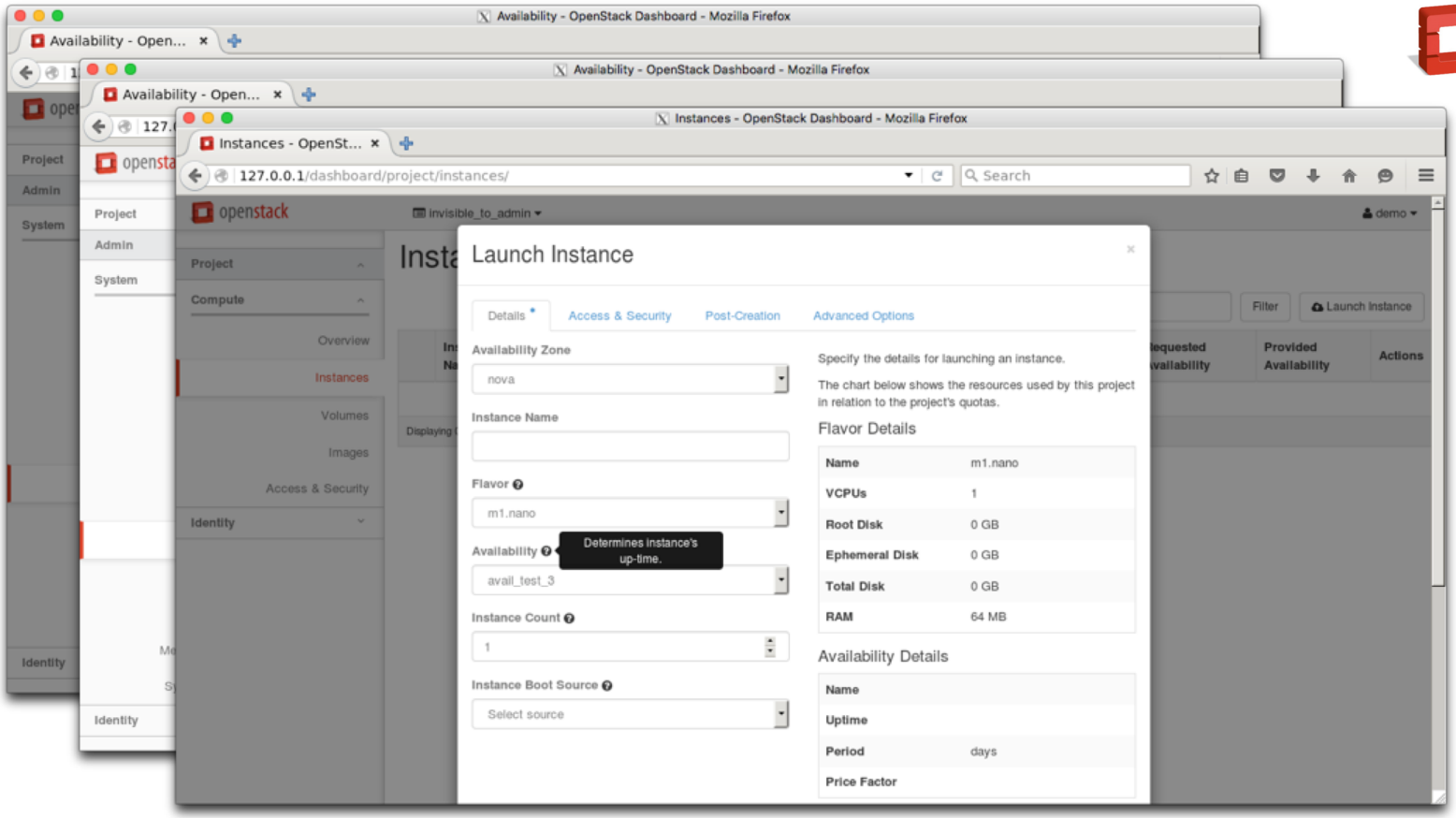
Deliberate Downtime

Deliberately fail VMs near the end of their period to:

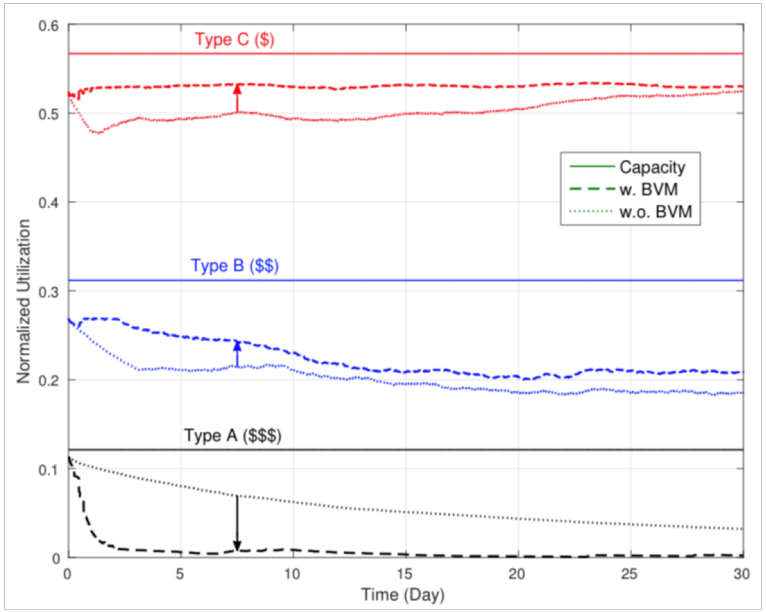
- Build market incentives
- Bid redeemed resources

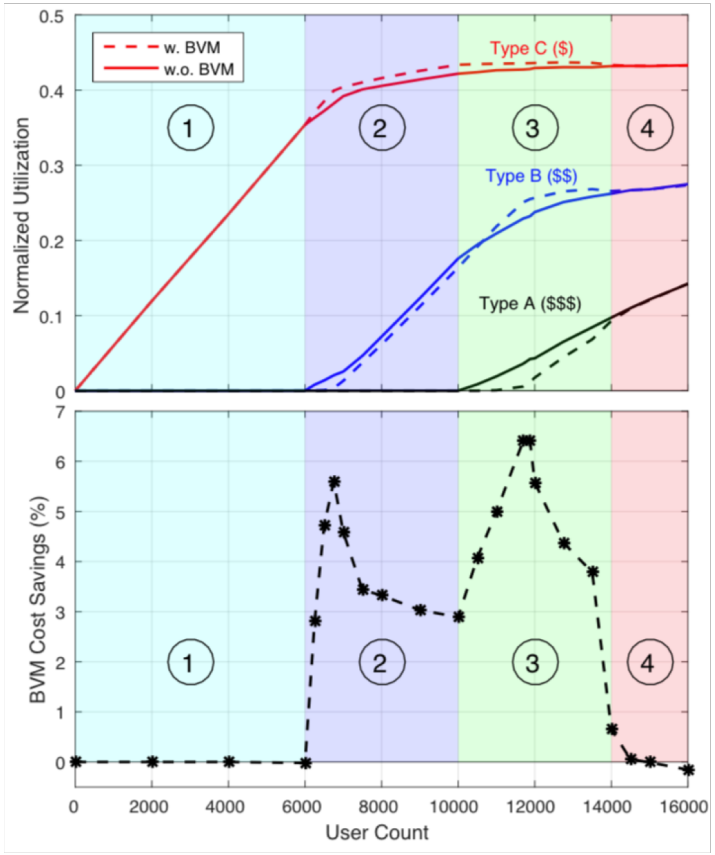


A Peek into the OpenStack Prototype



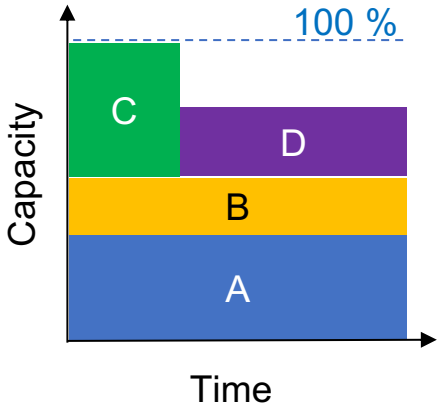
Offloading VMs to Cheaper Machines

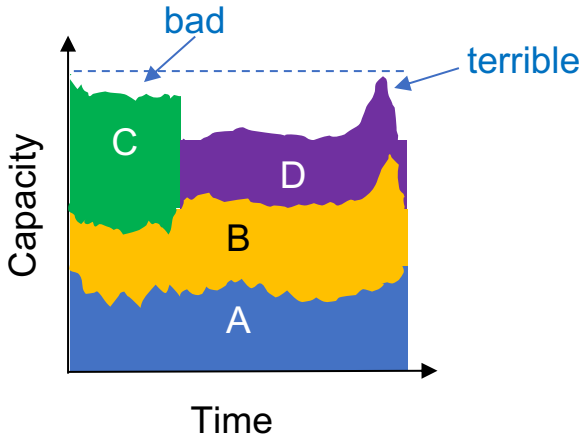




- >10% cost reduction
- >20% profit margin increase using variable prices

Case 2: Incentivizing Self-Capping





Demand Fluctuations

QoS guarantees

→ **Spare Capacity**

Can we motivate tenants to fluctuate less?

Self-adaptation mechanisms to control demand fluctuations.

Economic incentives to change behavior.

Graceful Degradation (GD) Methodology

Adaptive Bitrate Streaming

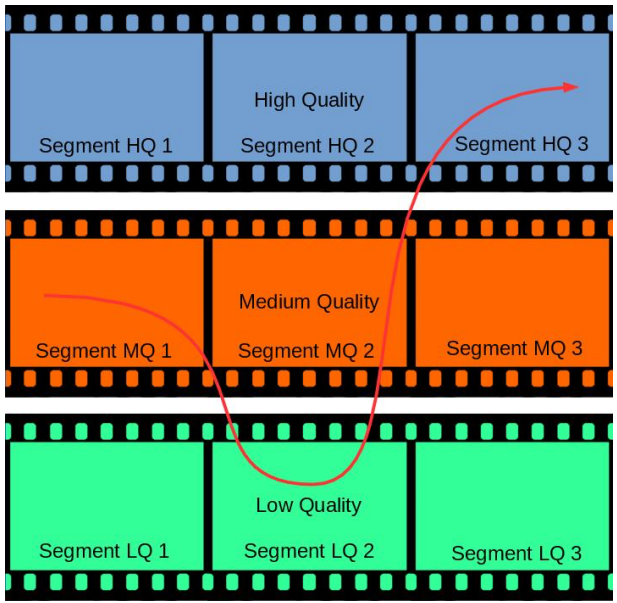
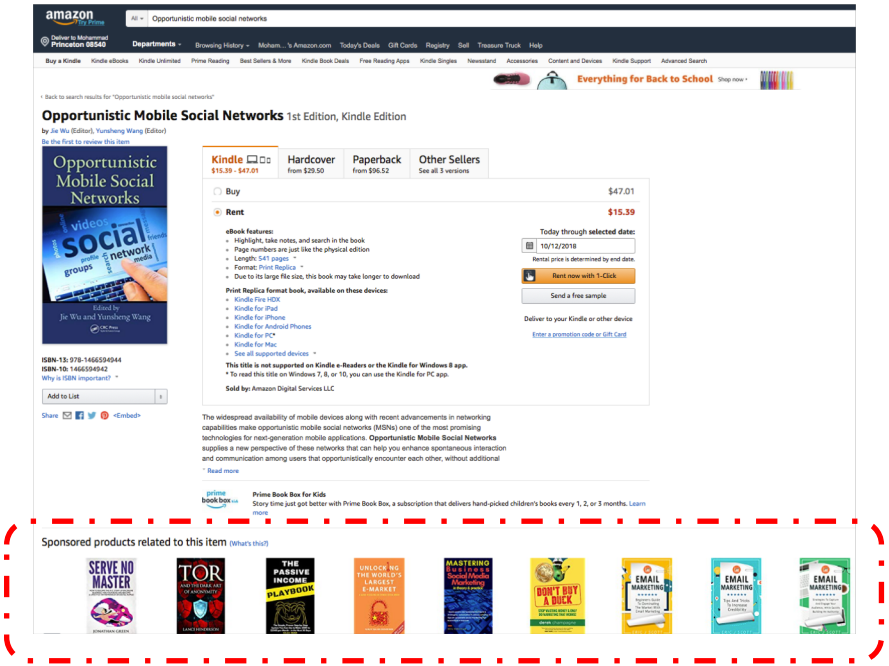


Image source: https://www.ubik-ingenierie.com/blog/adaptive-bitrate-streaming-watching-videos/

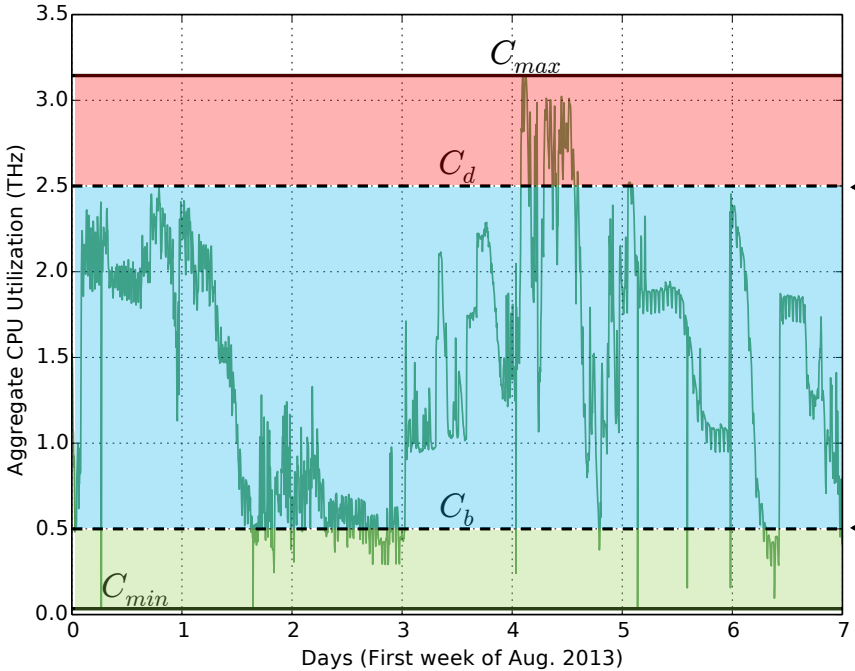
Brownout Self-Adaptation

Deactivating non-essential content to maintain response time.



Shaping the Capacity Demand

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



Total Deliverable Capacity

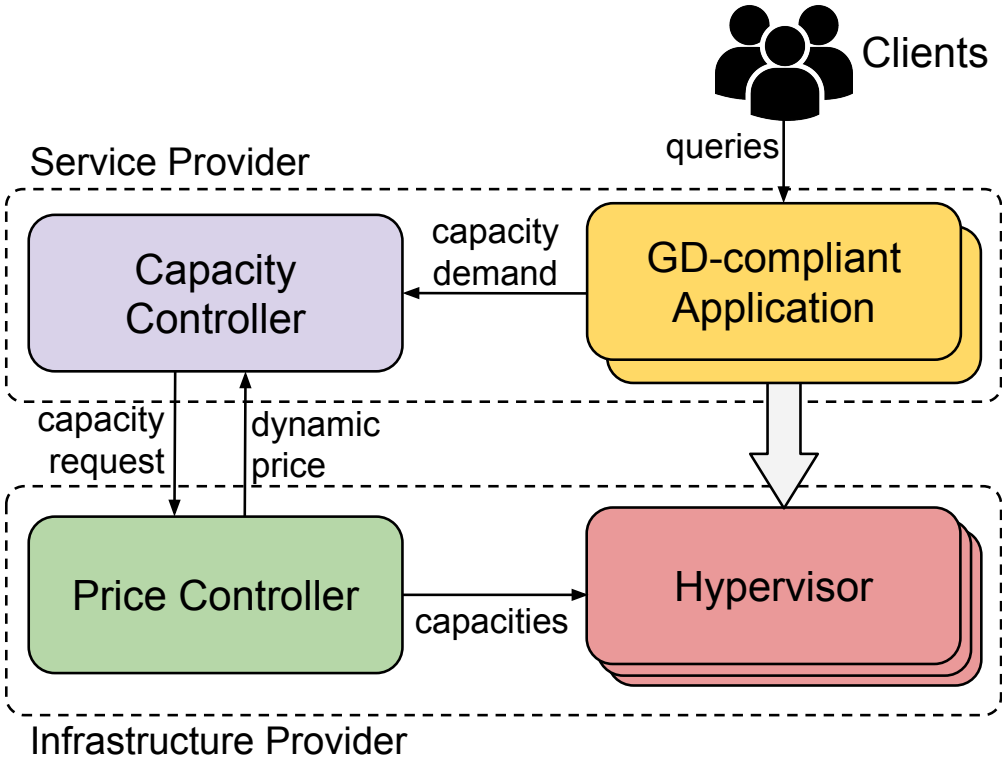
Not delivered (too expensive)

Dynamic Higher Price

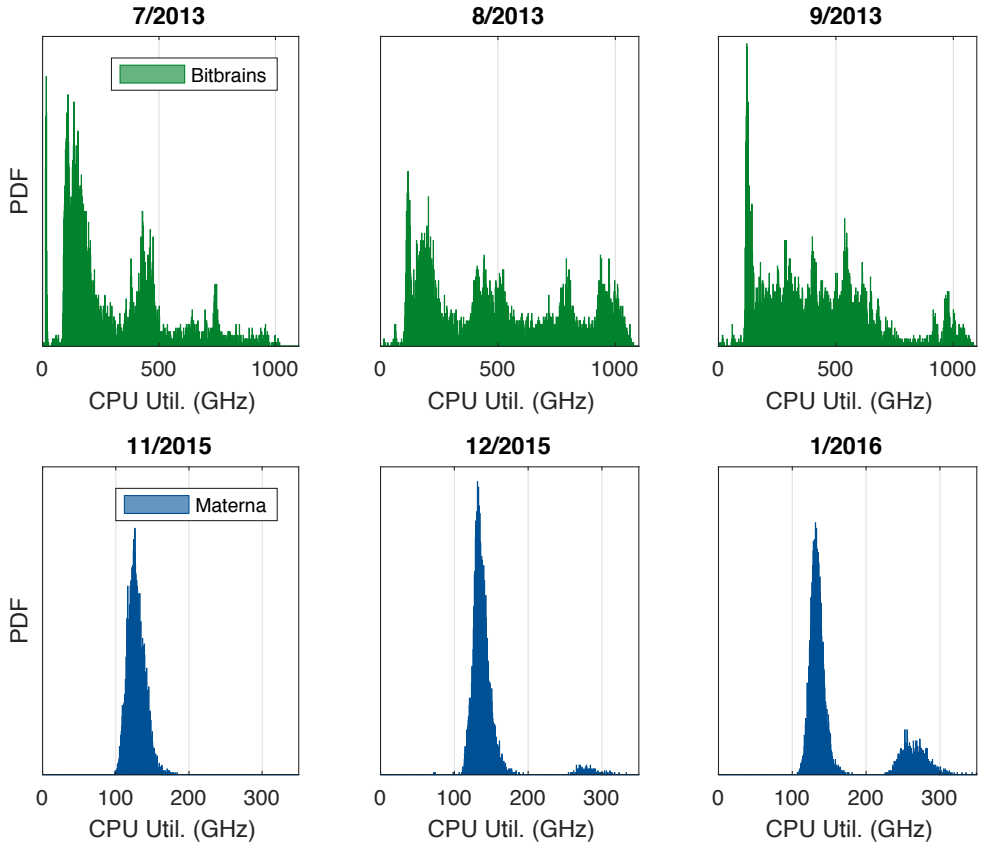
Reserved Base Capacity

Fixed Low Price

System Overview



Demand Distributions Allow Planning for Future



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

$$R(c, \lambda\theta) = \lambda^k R(c, \theta)$$

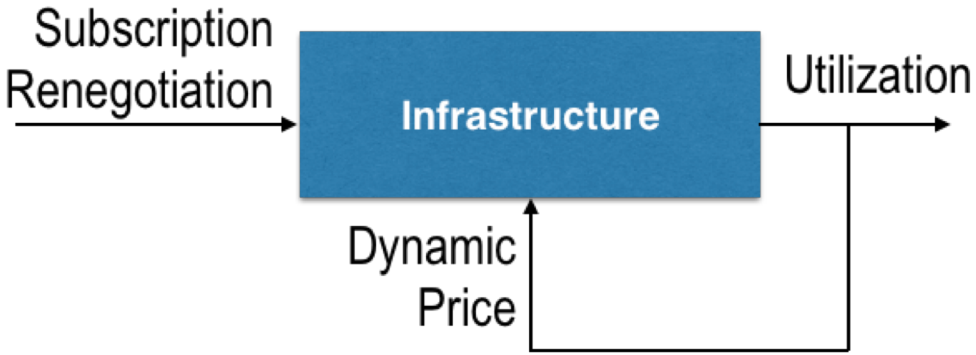
Dynamic Pricing to Control Utilization

Regardless of revenue function:

$$\textbf{\textit{Reserved Capacity}} \propto \frac{\textbf{\textit{Dynamic Price}}}{\textbf{\textit{Base Price}}}$$

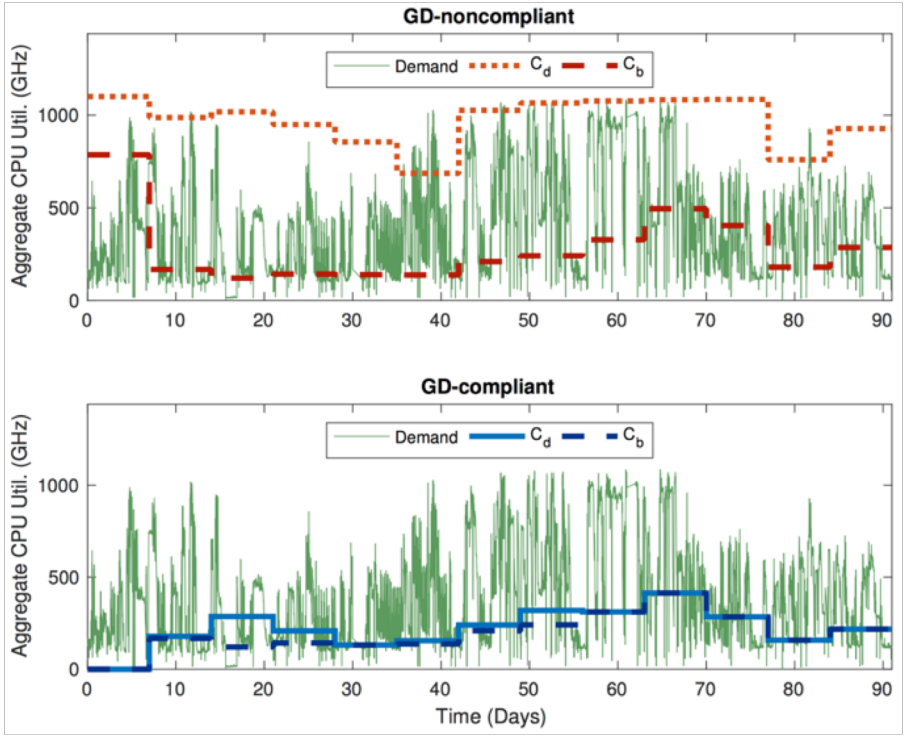
$$\textbf{\textit{Total Capacity}} \propto \frac{\mathbf{1}}{\textbf{\textit{Dynamic Price}}}$$

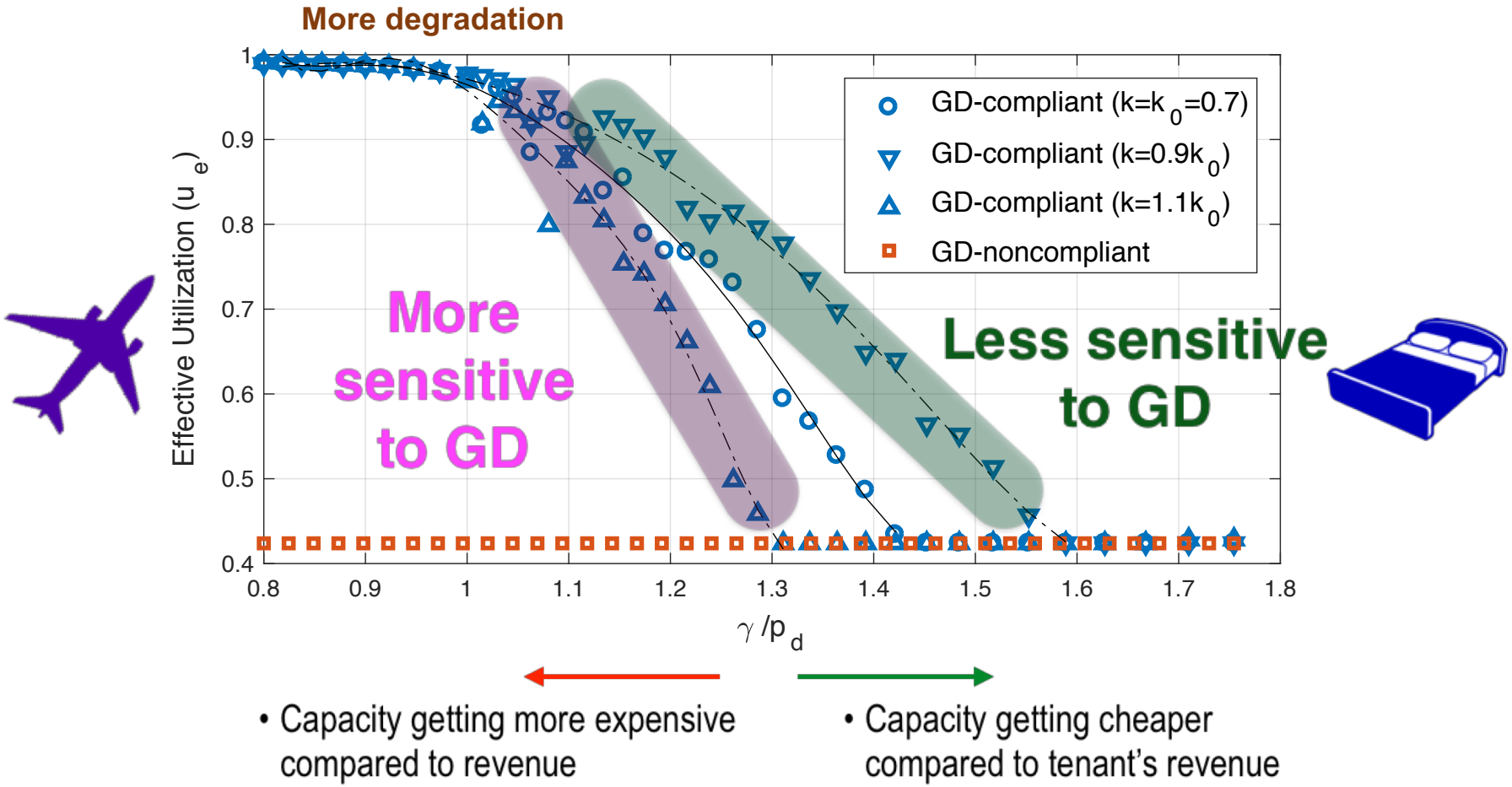
Monolithic incentives



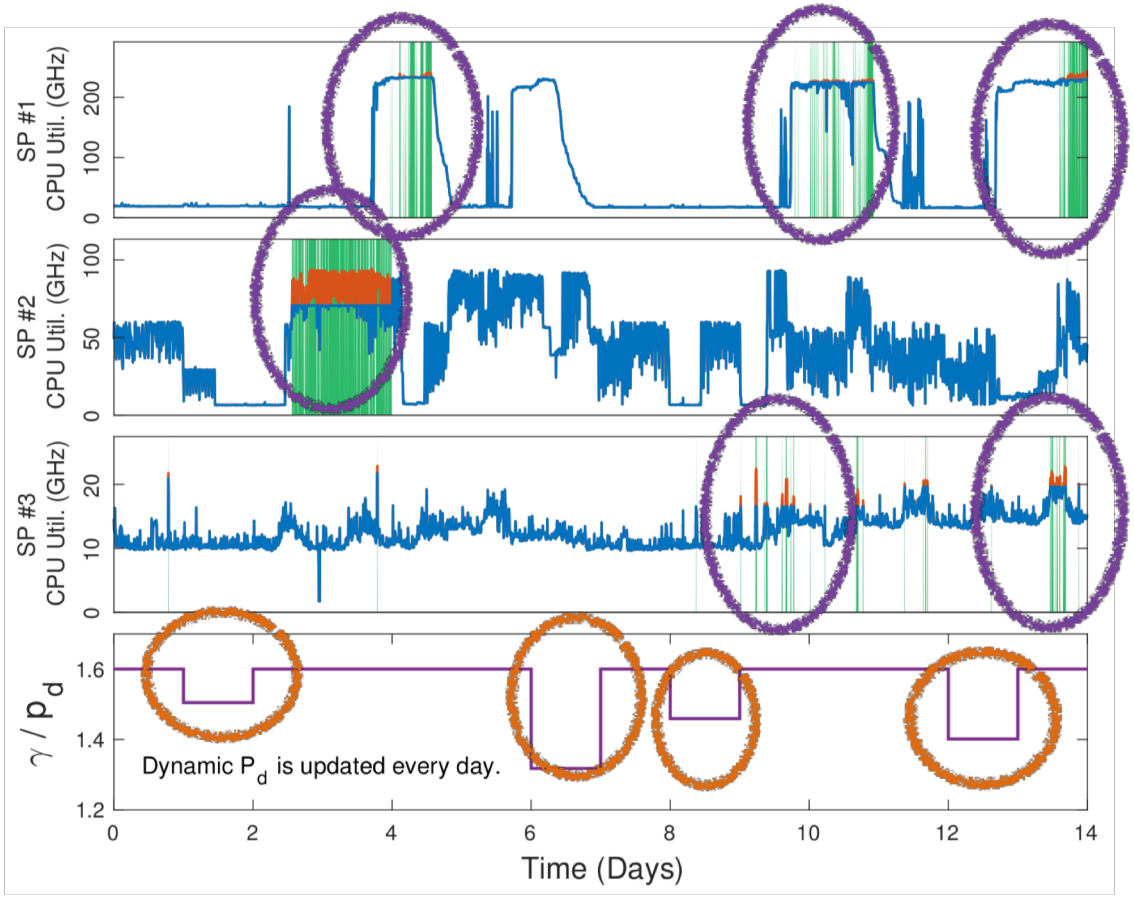
Evaluation

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




More degradation
Effective Utilization (u_e)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.8
0.9
1
1.1
1.2
1.3
1.4
1.5
1.6
1.7
1.8
γ / p_d
GD-compliant (k=k_0=0.7)
GD-compliant (k=0.9k_0)
GD-compliant (k=1.1k_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



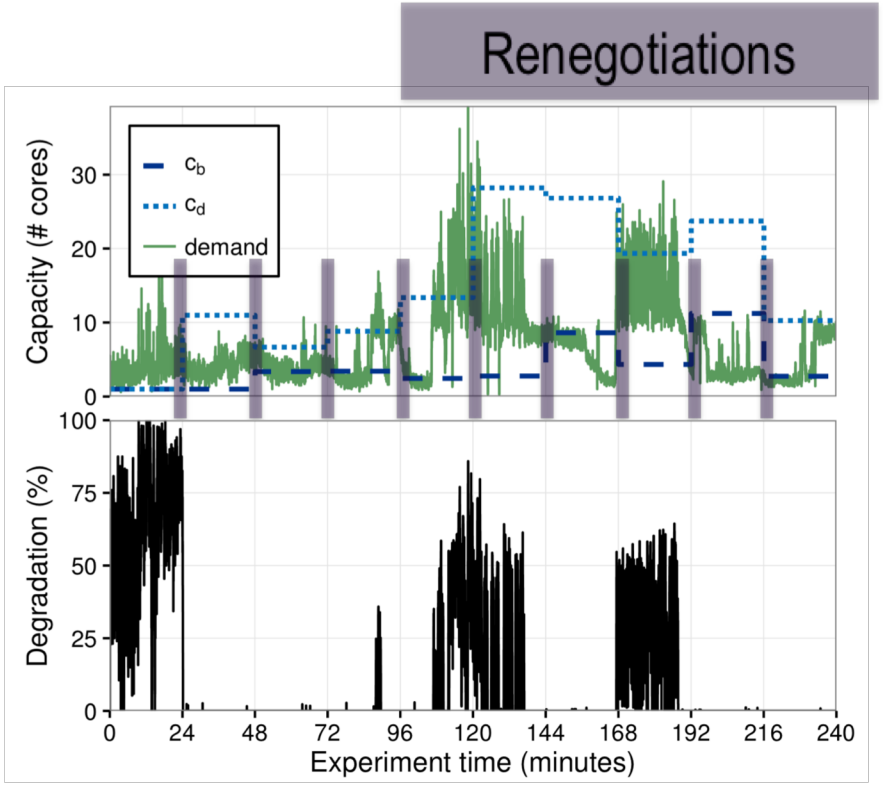
More degradation

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

Currently Pursing FaaS, a.k.a. Serverless

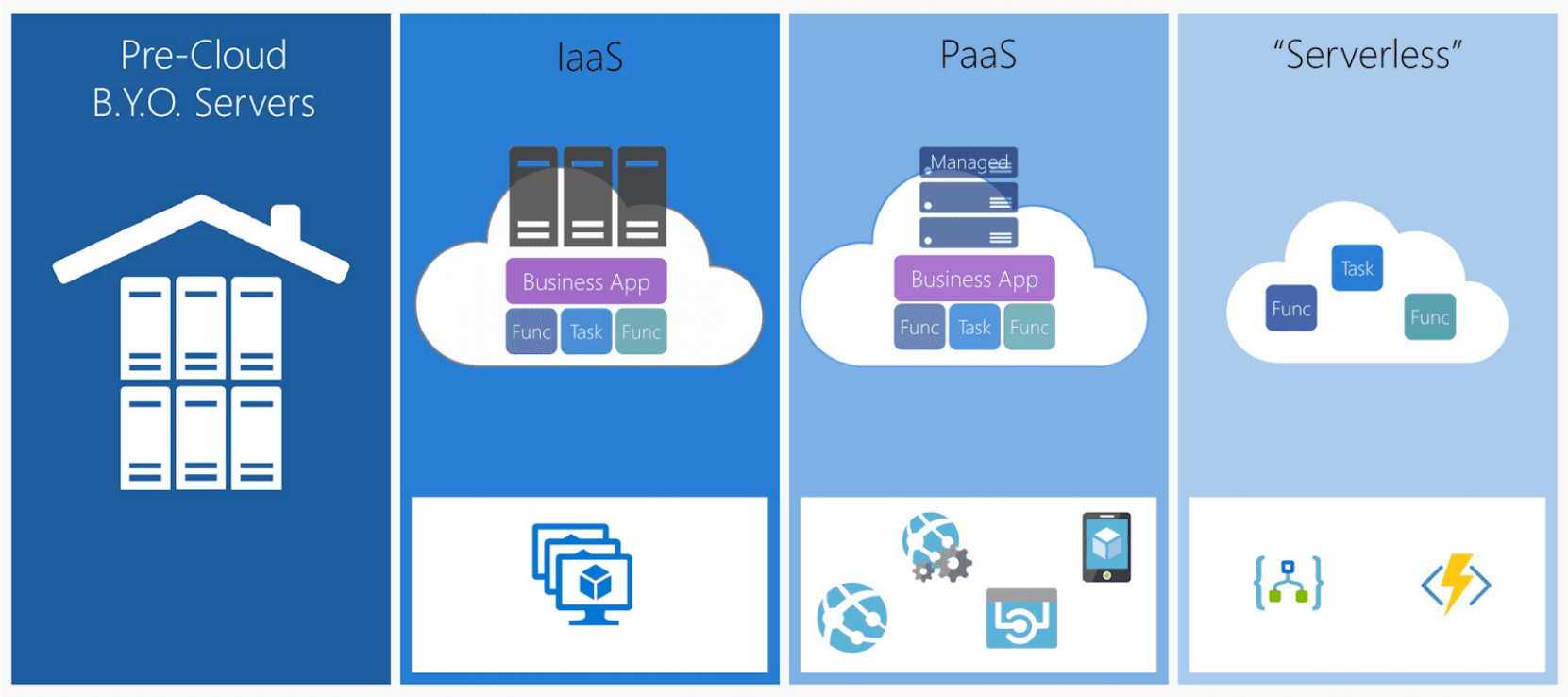


Image source: https://stackify.com/function-as-a-service-serverless-architecture/

The FaaS Wonderland

Knowns

- Building FaaS systems that work
  - Reliable
  - Secure (hopefully)
  - Scalable

Unknowns

- Delivering QoS
- Dedicated servers?
- Best way to co-locate with other cloud services

FaaS is inherently different:

Deeply Virtualized

Short Executions

High Memory Footprint

Low Temporal Locality

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

Big guys can't scratch their back!

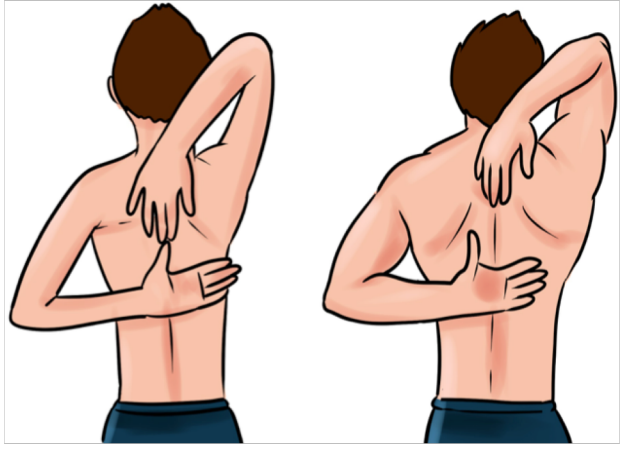


Image source: https://www.kaa-yaa.com/how-to-avoid-shoulder-imbalances-and-improve-shoulder-flexibility-to-avoid-injuries/big-guys-back-1/