Flash Memory Summit

Making Elephants Dance

Agility at hyperscale

About Me

CEO / exec team / BOD of several startups

- DSP, 10GBase-T PHY, backplane PHY, MEMS, TCP/FCP offload
- IPO, 100x sale, several smoking craters…

Tours of duty in some public companies

- Vitesse (first Gbit FibreChannel phy)
- IDT (GPU)

Venture Capital

- Investor (semis, CAE, low power wireless)
- CEO coach

Consultant to Flash/SSD companies

- SanDisk, Intel, Toshiba
- Cultural translator: SSD makers ⇔ hyperscale architects

Agenda

The Hard Thing about Hyperscale
- How to get fired from Facebook

Climbing Mt. Frugal
- Wish list & Prerequisites

This Changes Everything
- Four Forklift Upgrades

Unsolved Problems
- & Fearless Prognostications

The Hard Thing about Hyperscale

"The most amazing achievement of the computer software industry is its continuing cancellation of the steady and staggering gains made by the computer hardware industry."

-- Henry Peteroski

Life at Hyperscale: Layman's View

*GOOG, Azure, AWS, FB

Writing code

Playing ping-pong

Eating catered gourmet dinners

Revolutionizing stuff, changing the world…

Flash Memory Summit

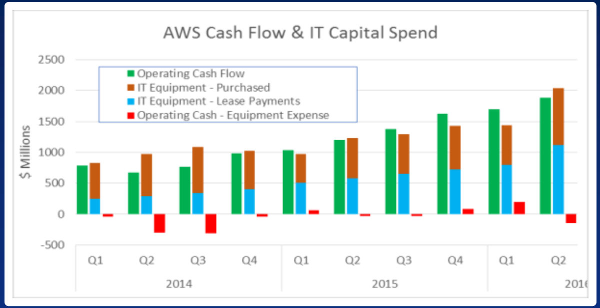
The Hard Thing about Hyperscale

Demand growing faster than Moore's Law

Flash Memory Summit

Fun Financial Tidbits

AWS:



Network traffic CAGR: 100%

Spends ~100% of operating cash flow on IT gear. (!)

Facebook:

Q1 2016 Annualized, **Per-User**		
Revenue		$15
Profit	Gross	$13
	Pretax	$6
IT Spend	Total	$2
	Storage	$1.2

Each new user costs $2, adds $15 of revenue. (!)

Efficiency is not about saving money.
It's about keeping up with demand.

Aside: The Bezos Algorithm

1. Find an infinitely large market, where *scale* wins.
2. Spend *every nickel you make or can borrow* to get bigger.

Forever.

"Your margin is my opportunity."
-- Jeff Bezos

What Price Efficiency?

How many Engineers would you invest to reduce hardware spend by 10%?

Crude Assumptions:

- Compute Node TCO = $5,000, one time
- Engineer TCO = $250,000/year

Category	# Nodes	Capacity Demand Growth	$/year, new Hardware	Breakeven headcount to reduce *growth* 10%
Large Enterprise	10k	10%	$5M	2 full time engineers
Small CSP	100k	15%	$75M	30 full time engineers
Tier-1 Hyperscale	1 million	25%	$1.25B	**500 full time engineers**

How to get Fired from Facebook

Thought experiment:

- You're SVP of Infrastructure. What's the one thing you *never* want to say to Zuck?

✘ Flash memory prices are up 200%

✘ Proposed site for Antarctic datacenter fell into the ocean

✓ Sorry boss, we're full up. Can't take any more new users.

IT at the Big 4, Summary

Relentless demand growth

- Last year's minor bottleneck becomes this year's existential crisis. (Every year)

Moore's law + lavish spending sometimes not enough to keep up.

At Hyperscale,

Efficiency Improvement is a Survival Skill

The Holy Grail of Hyperscale IT

100% resource utilization

This means 100% of:

- CPU cycles
- Bytes of cache
- DRAM capacity & bandwidth
- Storage capacity, IOPS, bandwidth
- Network packets/bandwidth

Simultaneously

Why This Matters to US: Trickle-Down ~~Economics~~ IT

Big innovations now originate at hyperscale

- Requires fleet > 1mm nodes to justify development

Eventually donated to or emulated by OSS

- 3rd party support happens
- “No devs required” deployment

When “safe”, adopted by the hoi-polloi

Corollary: To predict trends in enterprise IT, read about what Google was doing 10 years ago

Climbing Mt. Frugal

"I think frugality drives innovation, just like other constraints do. One of the only ways to get out of a tight box is to invent your way out."

-- Jeff Bezos

Frugality Wish List (1 of 2)

100% **Resource Utilization**

No *Reserved* Resources

Reserved Resources:

- Extra capacity provisioned to handle demand surges

Goal: Respond to demand spikes (on a timescale of seconds) with *zero reserved capacity*.

Frugality Wish List (2 of 2)

100% Resource Utilization

No *Reserved* Resources

No *Stranded* Resources

Reserved Resources:

- Extra capacity provisioned to handle demand surges

Goal: Respond to demand spikes (on a timescale of seconds) with *zero reserved capacity*.

Stranded Resources:

- Unused DRAM in a CPU-bound node
- Unused CPU in an I/O bound node
- Unused storage …

Goal: Use every CPU cycle, byte of DRAM/storage, bps of memory bandwidth… *Simultaneously*

Implications (1)

100% Resource Utilization

No *Reserved* Resources

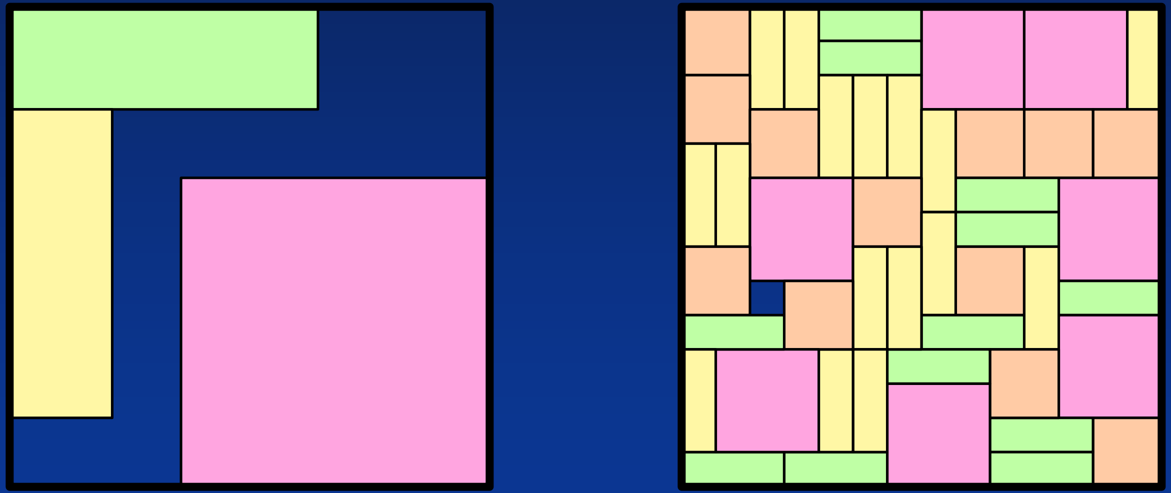
No *Stranded* Resources

Optimal Load Balancing

Scale individual workloads up/down keeping total resources constant

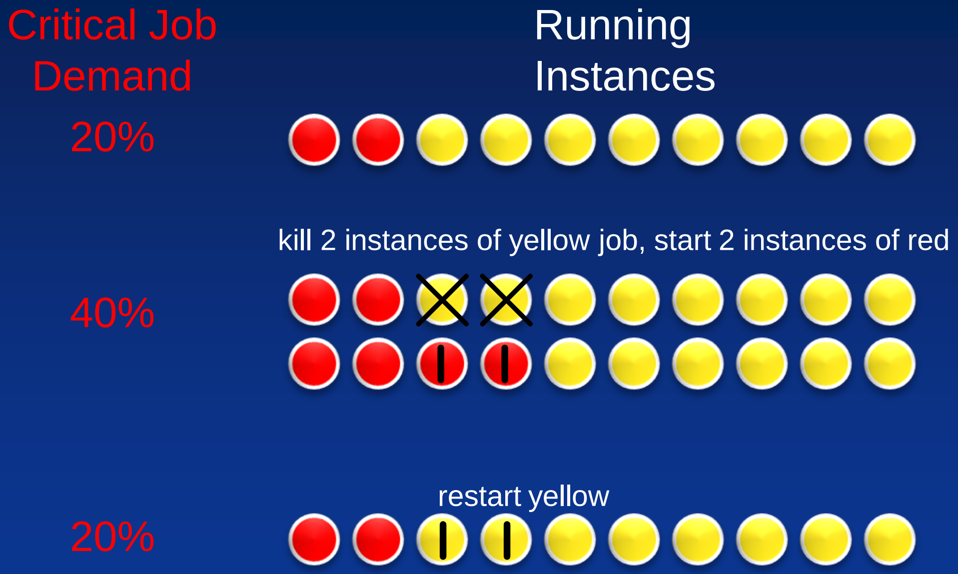
Multitenancy + Fine-Grained Job Decomposition

Efficient Bin-Packing is Easier with Small Objects



Scale by Replication, Restartable Instances

Load Balancing is easier if instances of low-priority jobs can be killed & restarted later.



Flash Memory Summit

Live Migration

Live Migration:

- Relocating a running job to another compute node.

Benefits:

- More 9’s of availability (Biggest source of downtime is reboots due to *planned* maintenance)
  - Network, power grid, infrastructure maintenance and upgrades
  - Host OS and BIOS upgrades
  - Security-patches

~Prerequisite:

- Networked storage (copying large private volumes not impossible, but very costly)

Flash Memory Summit

100% Resource Utilization

No *Reserved* Resources

No *Stranded* Resources

Optimal Load Balancing

Scale individual workloads up/down keeping total resources constant

Scaling by replication

- Restartable instances
- Fine-grained job decomposition

Live Migration

- No local state

Implications (2)

100% Resource Utilization

No *Reserved* Resources

No *Stranded* Resources

<u>Optimal Load Balancing</u>

Scale individual workloads up/down keeping total resources constant

<u>Scaling by replication</u>

- Restartable instances
- Fine-grained job decomposition

<u>Live Migration</u>

- No local state

<u>Optimal Workload Blending</u>

Simultaneously consume 100% of CPU cycles, cache, DRAM, I/O

Flash Memory Summit

Large-scale cluster management at Google with Borg

Borg: Google's (gen n-2) work scheduling system.

- Conceptual ancestor of Kubernetes.

Experiment:

- Workloads from actual traces
- Mixture of long-running/high priority & lower priority batch jobs
- Sensitivity analysis: each trace re-mapped to cluster multiple times, while varying constraints

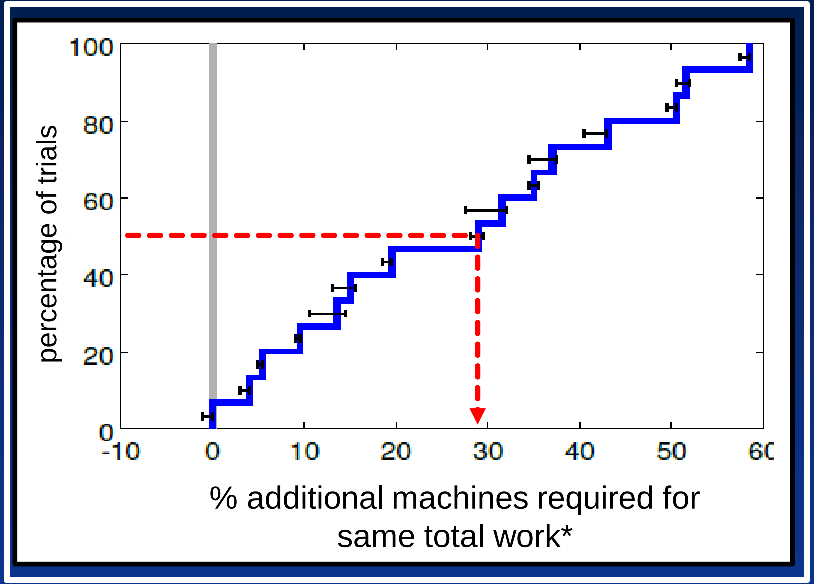
Results expressed as % more machines for the same work

- Vermay, Pedrosaz, Korupolu, Oppenheimer, Tune, Wilkes.
EuroSys'15, April 21–24, 2015

Flash Memory Summit

Segregated Workloads

Segregating critical workloads on dedicated clusters required 30% more machines for the same work

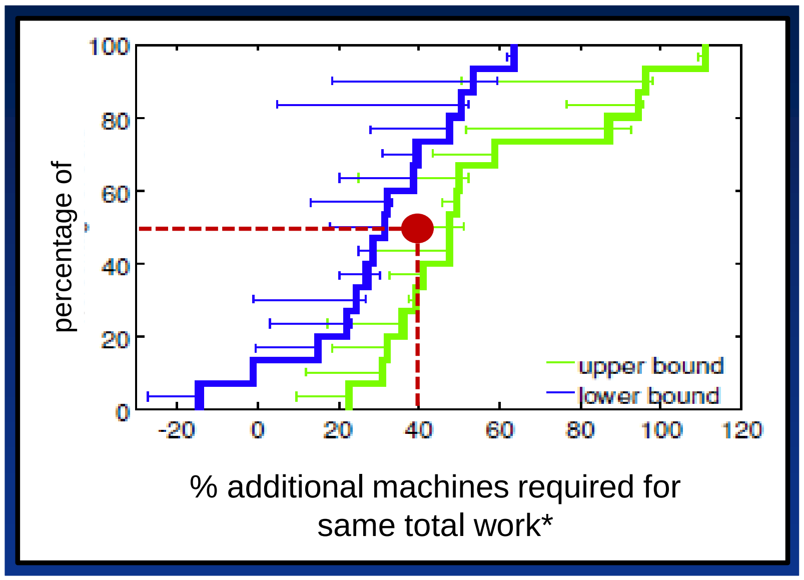


Large-scale cluster management at Google with Borg
EuroSys'15, April 21–24, 2015

Flash Memory Summit

Fine-Grained Resource Allocation

"Bucketing" resource allocation to powers of 2 required 40% more machines for the same work

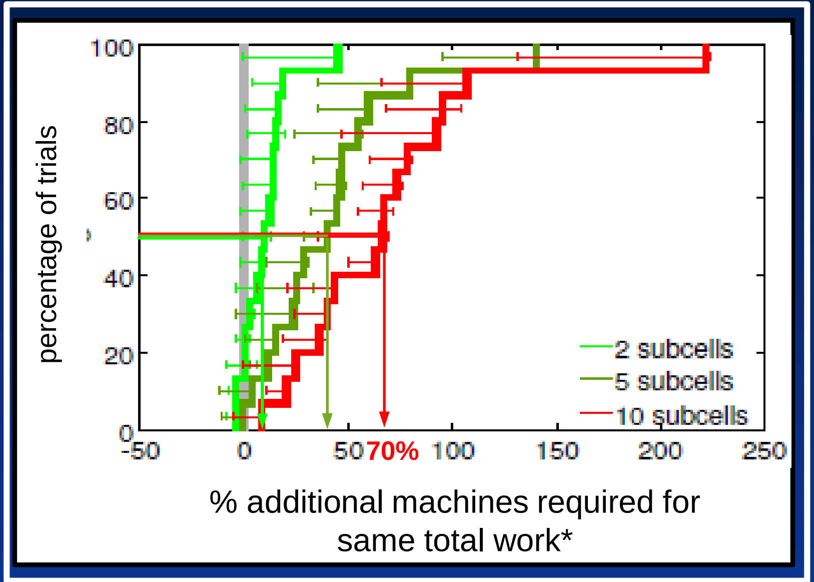


Large-scale cluster management at Google with Borg
EuroSys'15, April 21–24, 2015

Peer Locality

Reducing “cell” (network-neighborhood) size from 10,000 ⇨1,000 nodes

required 70% more machines for the same work



Large-scale cluster management at Google with Borg
EuroSys’15, April 21–24, 2015

Flash Memory Summit

Desiderata

No dedicated hardware

- Avoids stranded resources

No placement constraints. Any job ⇔ any node.

- More freedom to blend work optimally. Implies:
  - No node affinity
  - No peer-peer network locality constraints

Fine-grain resource allocation

- Coarse quantization is wasteful

Flash Memory Summit

100% Resource Utilization

No *Reserved* Resources

No *Stranded* Resources

Optimal Load Balancing

Scale individual workloads up/down keeping total resources constant

Optimal Workload Blending

Simultaneously consume 100% of CPU cycles, cache, DRAM, I/O

Scaling by replication

- Restartable instances
- Fine-grained job decomposition

Live Migration

- No local state

Multitenancy

Any job ⇔ any node

- No node affinity ⇨ No local state
- No locality ⇨ "flat" network

Implications (3)

100% Resource Utilization

No *Reserved* Resources

No *Stranded* Resources

Optimal Load Balancing
Scale individual workloads up/down keeping total resources constant

Optimal Workload Blending
Simultaneously consume 100% of CPU cycles, cache, DRAM, I/O

Optimal Storage Provisioning
Per-instance capacity, IOPS, bandwidth, resilience cost

Scaling by replication
- Restartable instances
- Fine-grained job decomposition

Live Migration
- No local state

Multitenancy

Any job ⬄ any node
- No node affinity ⇨ No local state
- No locality ⇨ “flat” network

Flash Memory Summit

Key Storage Inefficiency Drivers

Direct-attached drives (aka hyperconverged)

- If larger than node requires, strands storage capacity
- If smaller, strands CPU, memory

One-size-fits-all resilience (e.g. RAID at array level)

- Many workloads are ephemeral
  - Intermediate results in analytics calculations
  - Cache
- Some need even more protection (multi-zone)

Coarsely quantized allocation

- Remember Borg…

Flash Memory Summit

Flexible Storage Semantics

Renegotiating the Application:Storage "Contract"

(some examples):

- When a write is acknowledged, the data is ~~safe~~ cached in DRAM
- Overwrites ~~are~~ may not be idempotent
- Write order is ~~preserved~~ not guaranteed
- Failed writes ~~will be automatically retried~~ are the app's problem
- ACID is ~~guaranteed~~ is negotiable
- ~~C~~AP

New semantics with nontraditional (relaxed) guarantees enable hardware simplicity & scale

Flash Memory Summit

Desiderata

Disaggregate drives
- Avoids stranded resources

Per-job resilience
- Replicate etc. only when really needed

Allocate bytes, not GB

(and never trust Job-owners' claims about what they need)

Embrace eventual consistency

100% Resource Utilization

No *Reserved* Resources

No *Stranded* Resources

Optimal Load Balancing

Scale individual workloads up/down keeping total resources constant

Optimal Workload Blending

Simultaneously consume 100% of CPU cycles, cache, DRAM, I/O

Optimal Storage Provisioning

Per-instance capacity, IOPS, bandwidth, resilience cost

Scaling by replication

- Restartable instances
- Fine-grained job decomposition

Live Migration

- No local state

Multitenancy

Any job ⇔ any node

- No node affinity ⇨ No local state
- No locality ⇨ "flat" network

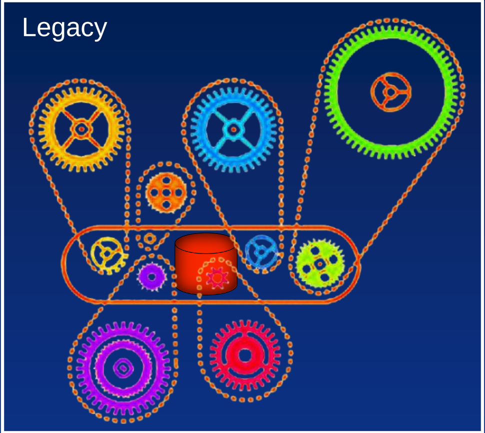
Networked Storage + Flexible semantics

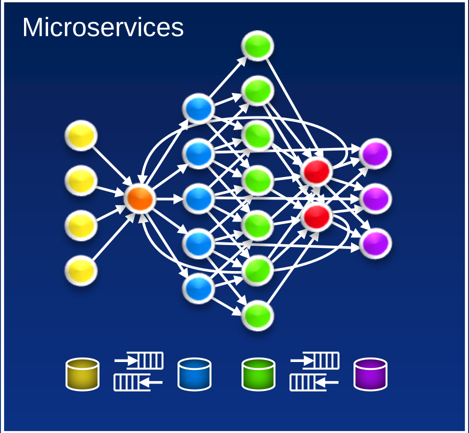
- No stranded capacity/IOPS
- Variable resilience, consistency

This Changes Everything

Four ~~Forklift~~ Bulldozer Upgrades

A New Application Architecture: Microservices





- ✓ Restartable instances
- ✓ Fine-grained job decomposition
- ✓ No local state
- ✓ Scales by replication

Single function: each instance processes one action per invocation

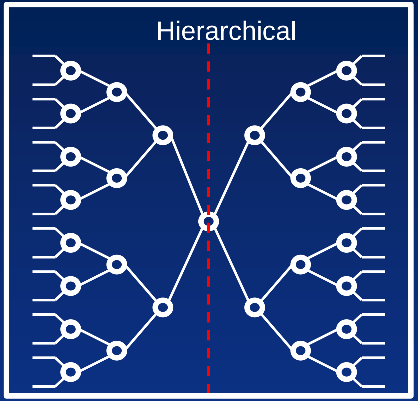
- Application logic is external (a library, not a framework)

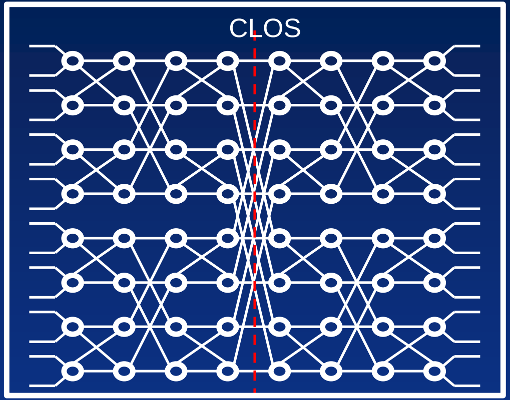
Instances retain no internal state between invocations

Services are self contained

- Don't access external DB's
- Local replica, updated via message queues

A New Network Architecture: Fat-Tree (CLOS)





✓ "Flat" Network – same bandwidth, # hops between any two endpoints

Links are bidirectional, so actual implementations are "folded" about the centerline

Uplinks oversubscribed

- Routes to "nearby" nodes less congested
- Locality matters

One path from any input to any output

(neglecting redundancy not shown)

No architectural oversubscription

- Locality irrelevant

Many paths between any pair of endpoints

- Good at handling bursty/unbalanced traffic

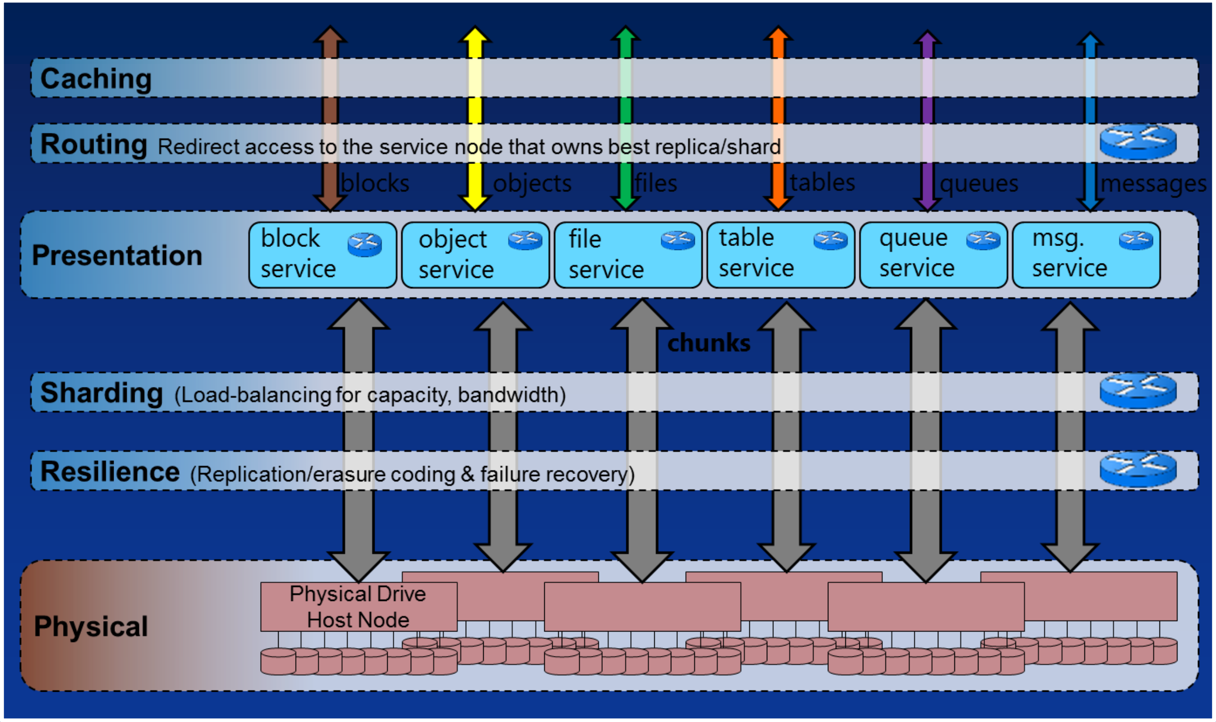
Large hardware cost

A New Storage Architecture: Software Defined Storage

Many abstractions from one stored format

Heavily layered

- ✓ Scales by replication
- ✓ No stranded capacity/IOPS
- ✓ Flexible resilience, consistency



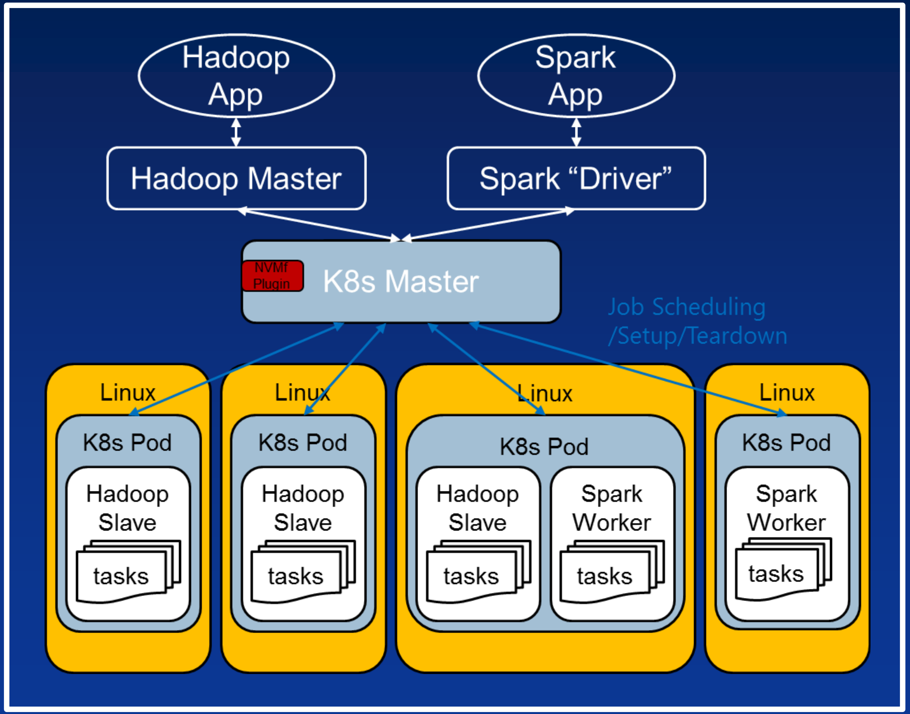
A New Way to Organize & Schedule Work: Datacenter Orchestration

Self service

Onramp to PaaS

- Google open-sourced Kubernetes (*but not workload-blending*)
- *Very* rapid evolution, active community

(Compute more mature than storage)



Flash Memory Summit

Foundations of Hyperscale Efficiency

100% Resource Utilization

No *Reserved* Resources

No *Stranded* Resources

Optimal Load Balancing
Scale individual workloads up/down keeping total resources constant

Optimal Workload Blending
Simultaneously consume 100% of CPU cycles, cache, DRAM, I/O

Optimal Storage Provisioning
Per-instance capacity, IOPS, bandwidth, resilience cost

Scaling by replication
- Restartable instances
- Fine-grained job decomposition

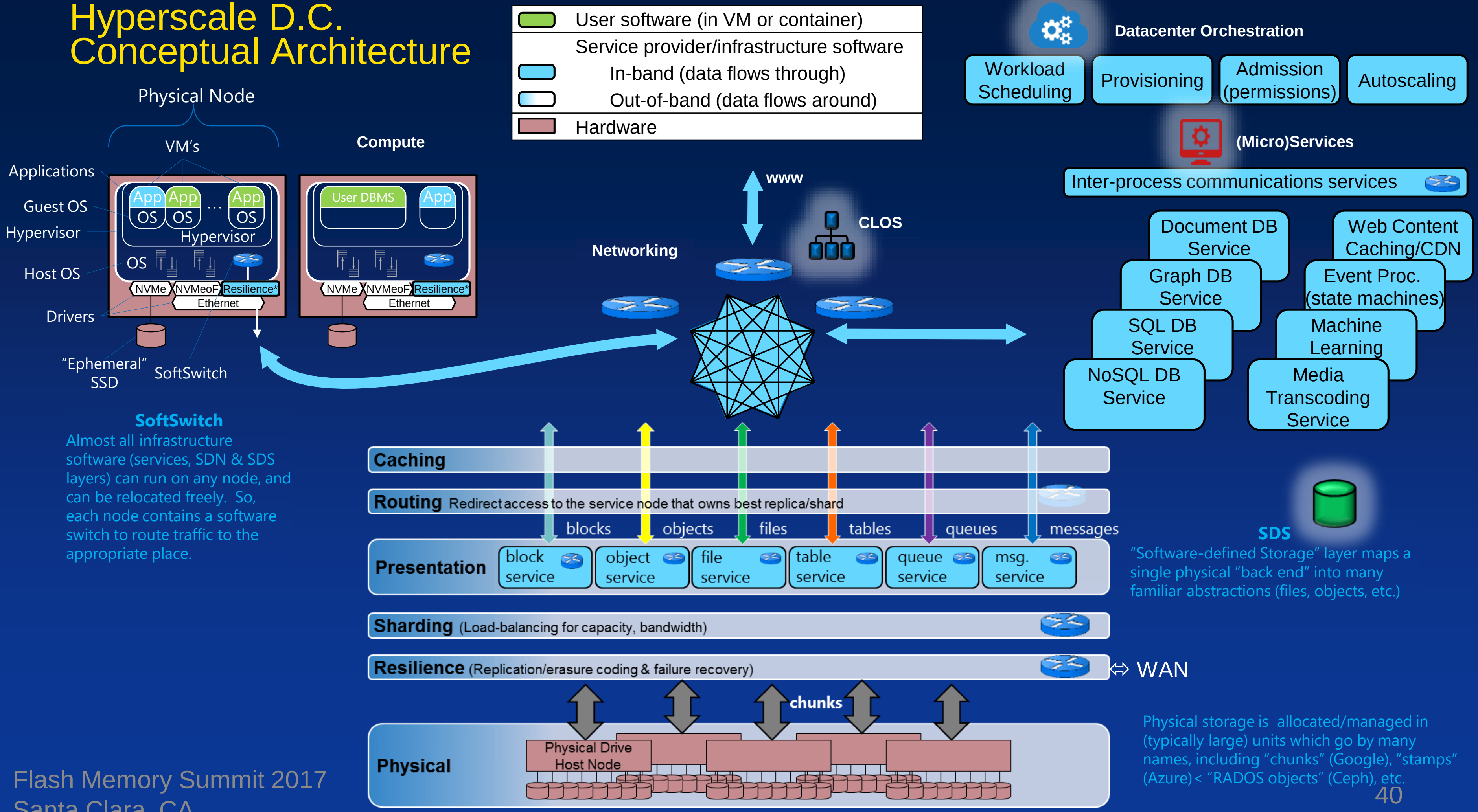
Live Migration
- No local state

Multitenancy
Any job ⇔ any node
- No node affinity ⇨ No local state
- No locality ⇨ "flat" network

Networked Storage + Flexible semantics
- No stranded capacity/IOPS
- Variable resilience, consistency

Microservices Architecture | **Orchestration** | **CLOS/ Fat Tree** | **Software Defined Storage**

Hyperscale D.C. Conceptual Architecture



SoftSwitch
Almost all infrastructure software (services, SDN & SDS layers) can run on any node, and can be relocated freely. So, each node contains a software switch to route traffic to the appropriate place.

SDS
"Software-defined Storage" layer maps a single physical "back end" into many familiar abstractions (files, objects, etc.)

Physical storage is allocated/managed in (typically large) units which go by many names, including "chunks" (Google), "stamps" (Azure) < "RADOS objects" (Ceph), etc.

Unsolved Problems & Fearless Prognostications

In anything at all, perfection is finally attained not when there is no longer anything to add, but when there is no longer anything to take away.

-- Antoine de Saint-Exupery

Typical "Back Ends"

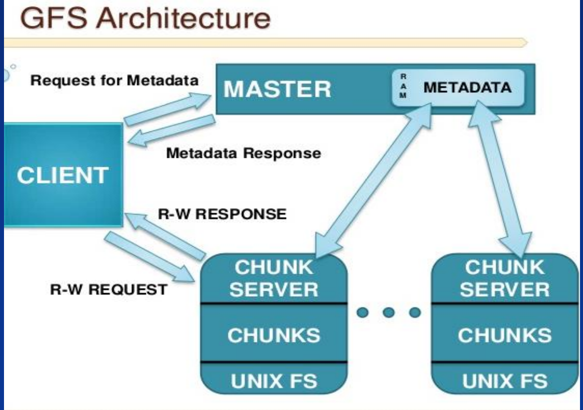
Flash Memory Summit

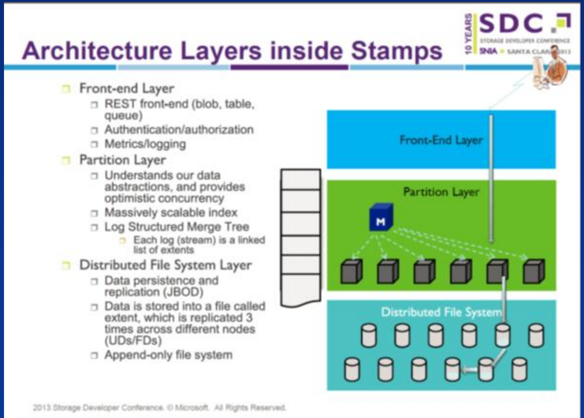
Google GFS/Colossus ("chunks"):

- Modify supported, but append-mostly
- Eventual consistency
- Colossus reduced backend object size from 64MB to 1MB

MS Azure ("stamps")

- Append-only, then immutable
- Variable size, typically 1GB





Append Only Back-Ends

Sequential:Random IOPS ratio

- NVMe SSD: < 2:1
- HDD ~250:1

Large, log-structured back-end chunks are a remnant of hard-disc centric storage.

Neither appropriate nor helpful for SSD's.

Cloud Storage Services, Today

Many abstractions, one stored format

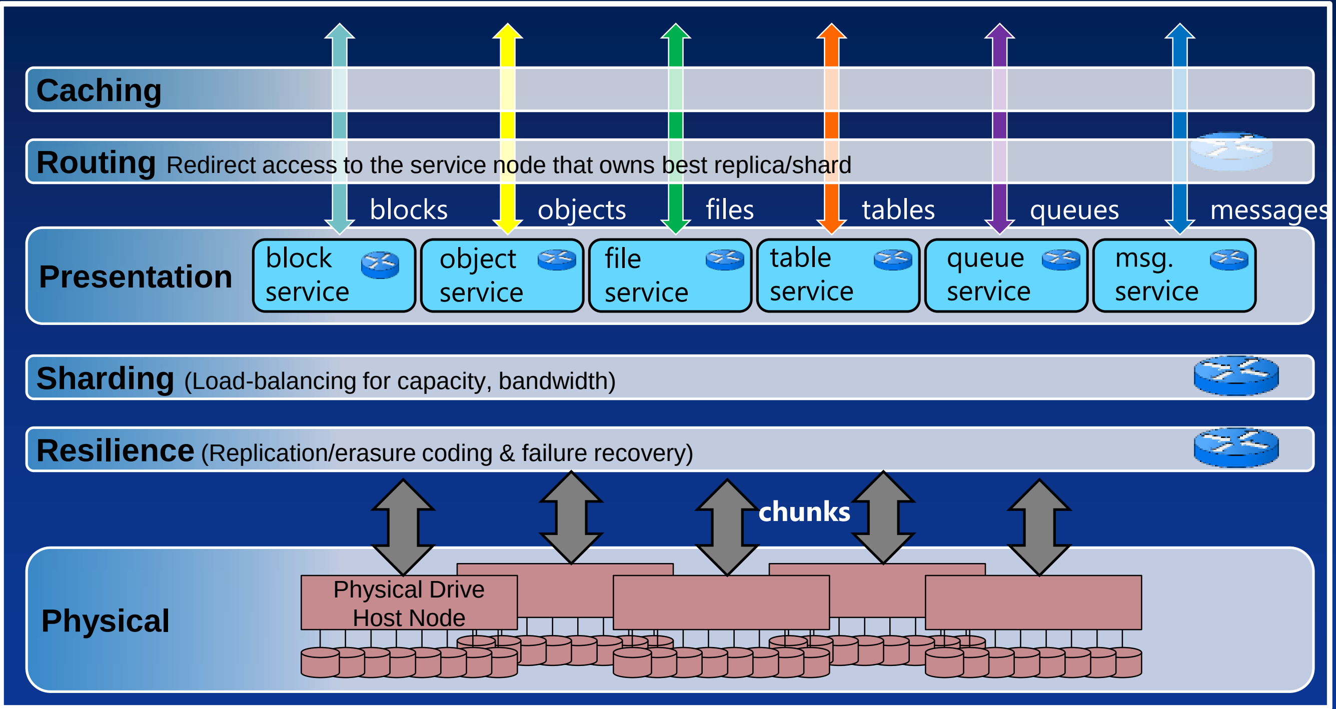
⇨ *simple, flexible, scalable*

Lots of layers

⇨ *poor latency*

Very large back-end chunk size

⇨ *Ineffcient use of storage bandwidth*



What About Applications that need High-Performance Block Access?

IaaS customers often want to run SQL database applications

- Row (stored item) size for many applications is very small; 100 bytes or less

How can we deliver random reads of 100 byte items if the “back end” can only read/write 64MB chunks?!

Partial Answer: Local SSD Cache

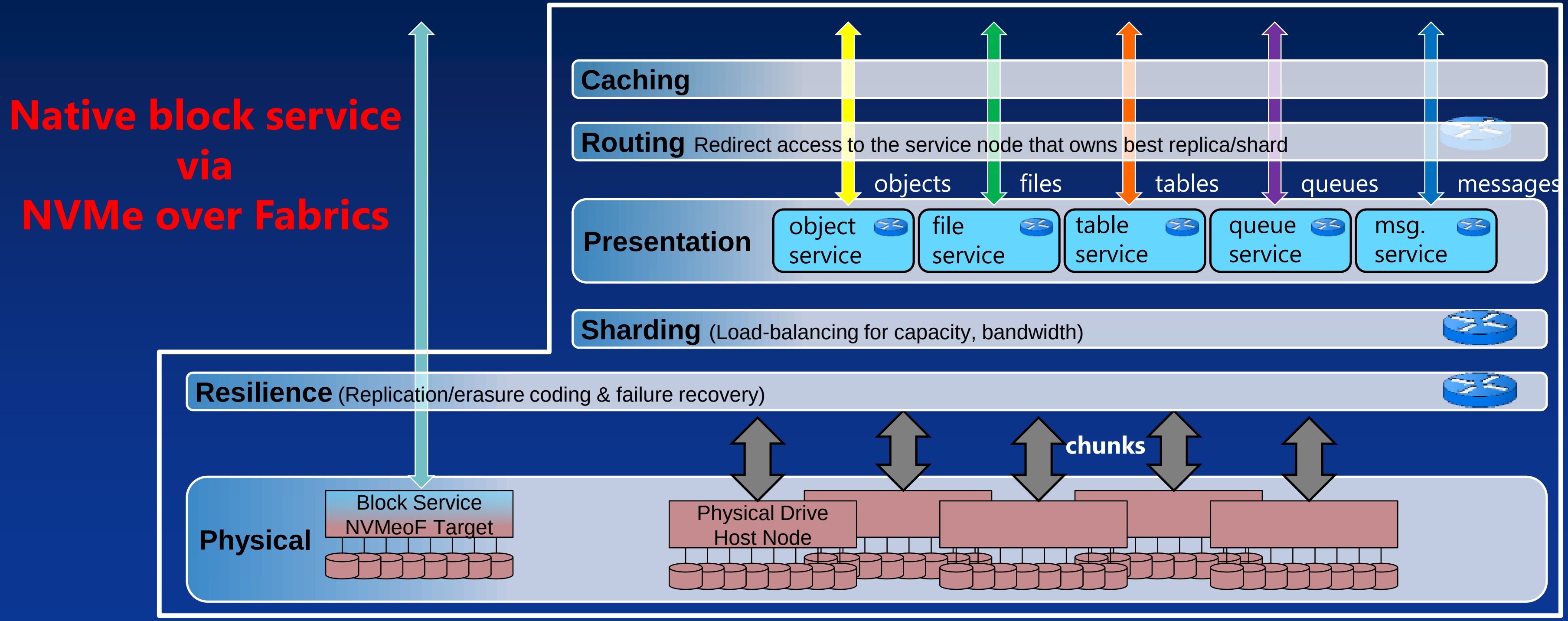
Cache block traffic in local NVMe SSD.

- A big win: One machine with PCIe SSD cache matched performance of several machines with networked HDD storage.

Problems: Local State

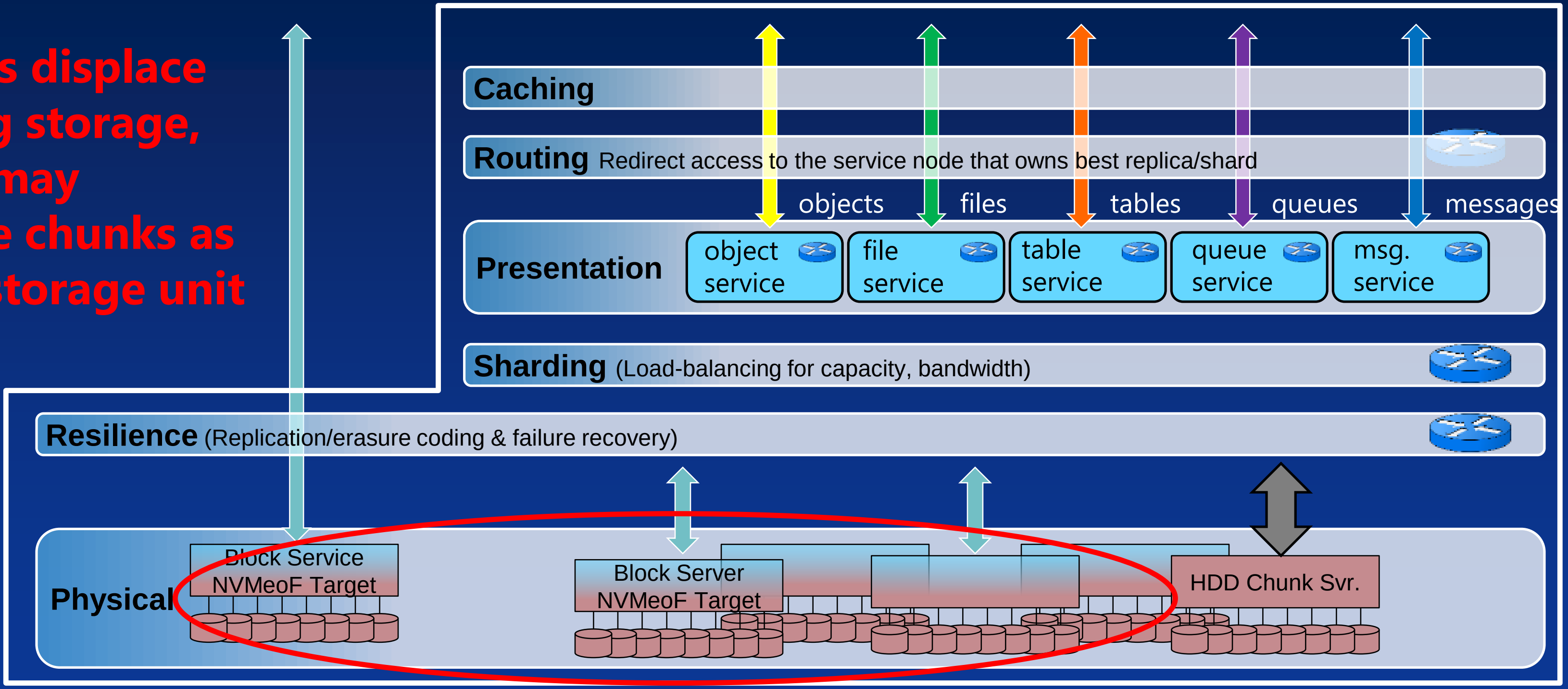
  - Never the right size (stranded capacity)
  - Restricts job placement (violates “no node affinity”)
  - Restricts live migration

Next Generation



Deja Vu?

As SSD's displace rotating storage, blocks may displace chunks as native storage unit



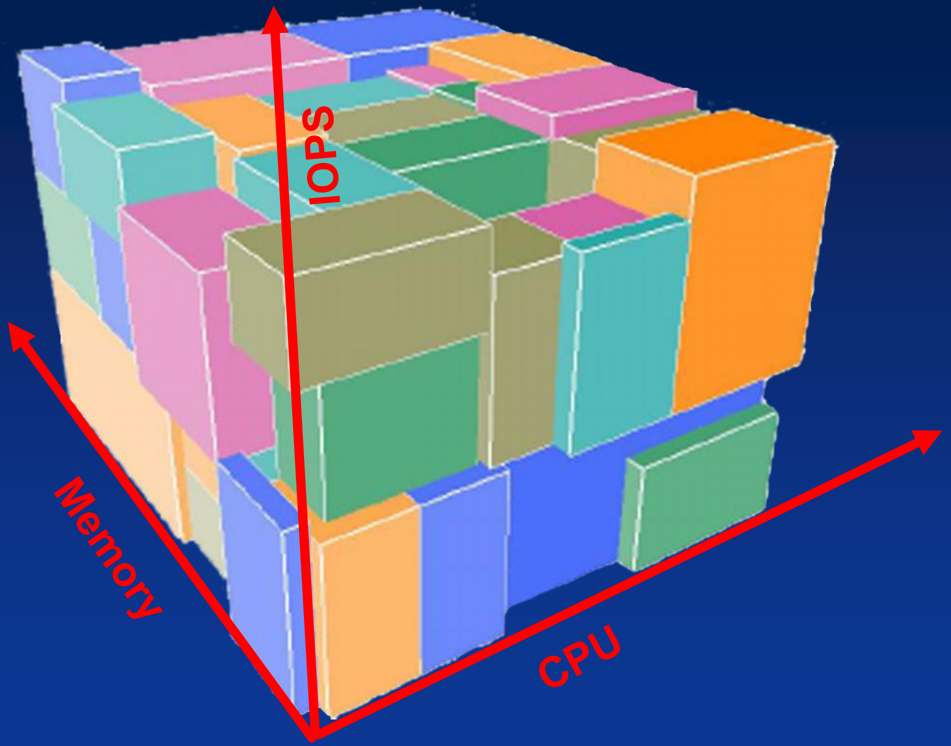
A Small Exercise Left to the Interested Listener

Kubernetes is now OSS

A win-win

- We get orchestration tech.
- Google trains potential customers

But they didn't give away their "n-dimensional bin-packing" technology



Flash Memory Summit

Thank You!

"Lately it occurs to me
What a long, strange trip it's been."

"Truckin'" – The Grateful Dead