

Cluster-Level Storage @ Google

How we use *Colossus* to improve storage efficiency

Denis Serenyi
Senior Staff Software Engineer
dserenyi@google.com

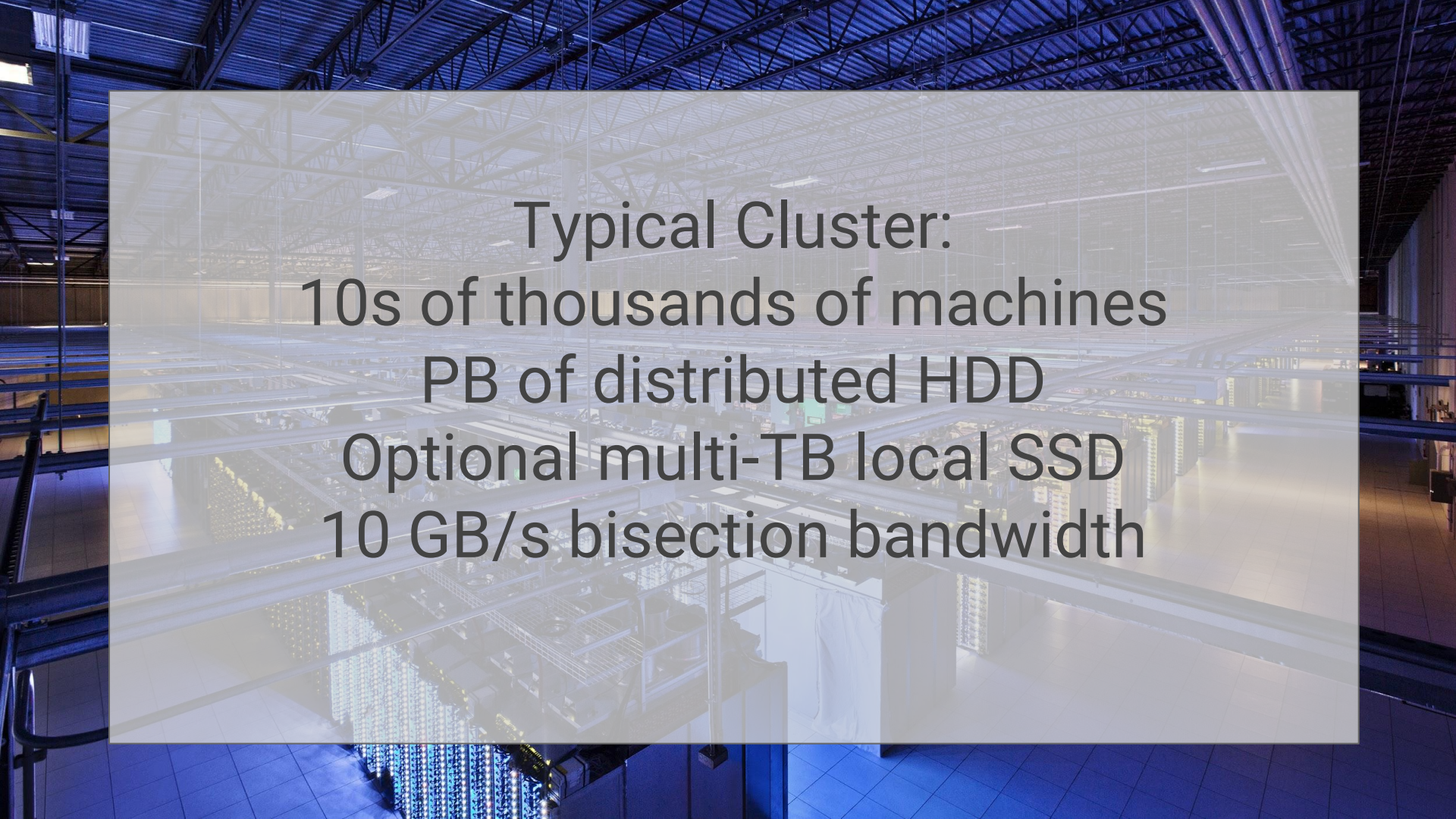
November 13, 2017

Keynote at the 2nd Joint International Workshop on Parallel Data Storage & Data Intensive Scalable Intensive Computing Systems




What do you call a few PB of free space?


What do you call a few PB of free space?
An emergency low disk space condition


Typical Cluster:
10s of thousands of machines
PB of distributed HDD
Optional multi-TB local SSD
10 GB/s bisection bandwidth



Part 1: Transition From GFS to Colossus

GFS architectural problems

GFS master

- One machine not large enough for large FS
- Single bottleneck for metadata operations
- Fault tolerant, not HA



Predictable performance

- No guarantees of latency

Some obvious GFSv2 goals

Bigger!
Faster!
More predictable tail latency

GFS master replaced by **Colossus**
GFS chunkserver replaced by **D**

Solve an easier problem

A “file system” for Bigtable

- Append-only
- Single-writer (multi-reader)
- No snapshot / rename
- Directories unnecessary

Where to put metadata?

Storage options back then

GFS

Sharded MySQL with local disk & replication

- Ads databases

Local key-value store with Paxos replication

- Chubby 

Bigtable (sorted key-value store on GFS)

Storage options back then

GFS ← lacks useful database features
Sharded MySQL ← poor load balancing, complicated

Local key-value store ← doesn’t scale

Bigtable ← hmmmmmm....

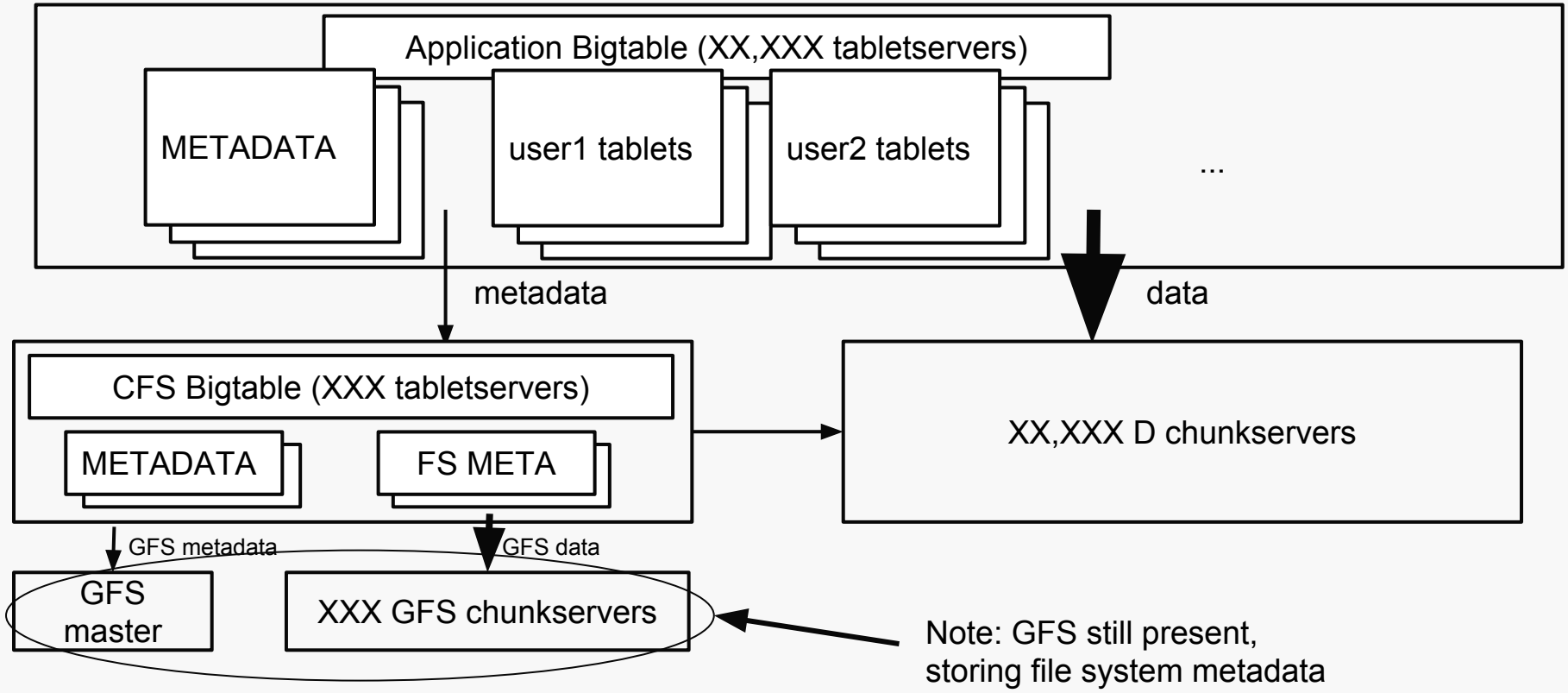
Why Bigtable?

Bigtable solves many of the hard problems:

- Automatically shards data across tablets
- Locates tablets via metadata lookups
- Easy to use semantics
- Efficient point lookups and scans

File system metadata kept in an in-memory locality group

Metadata in Bigtable (!?!?)

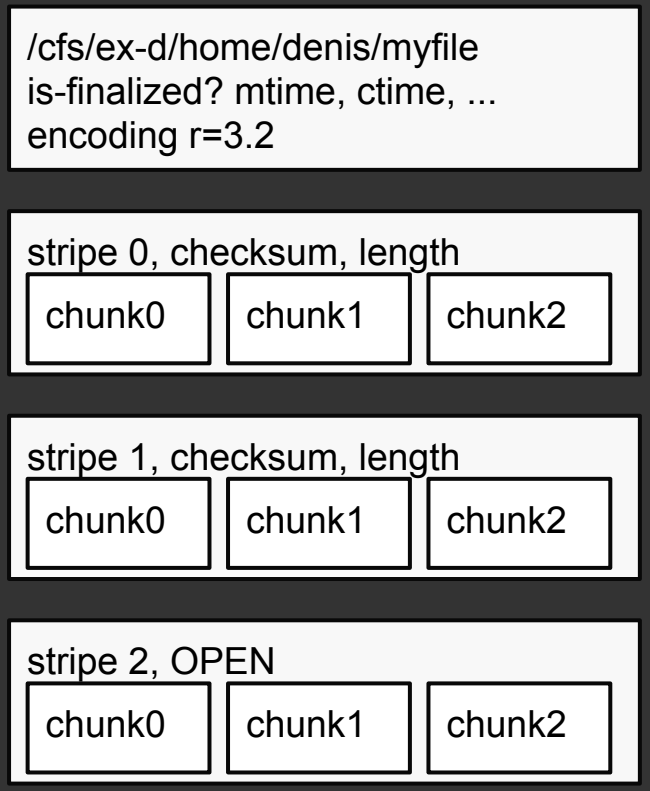


GFS master -> CFS

CFS “curators” run in Bigtable tablet servers

Bigtable row corresponds to a single file

Stripes are replication groups: open, closed, finalized



Colossus for metadata?

Metadata is ~1/10000 the size of data
So if we host a Colossus on Colossus…
100 PB data → 10 TB metadata
10TB metadata → 1GB metametadata
1GB metametadata → 100KB meta...

And now we can put it into Chubby!



Part 2: Colossus and Efficient Storage

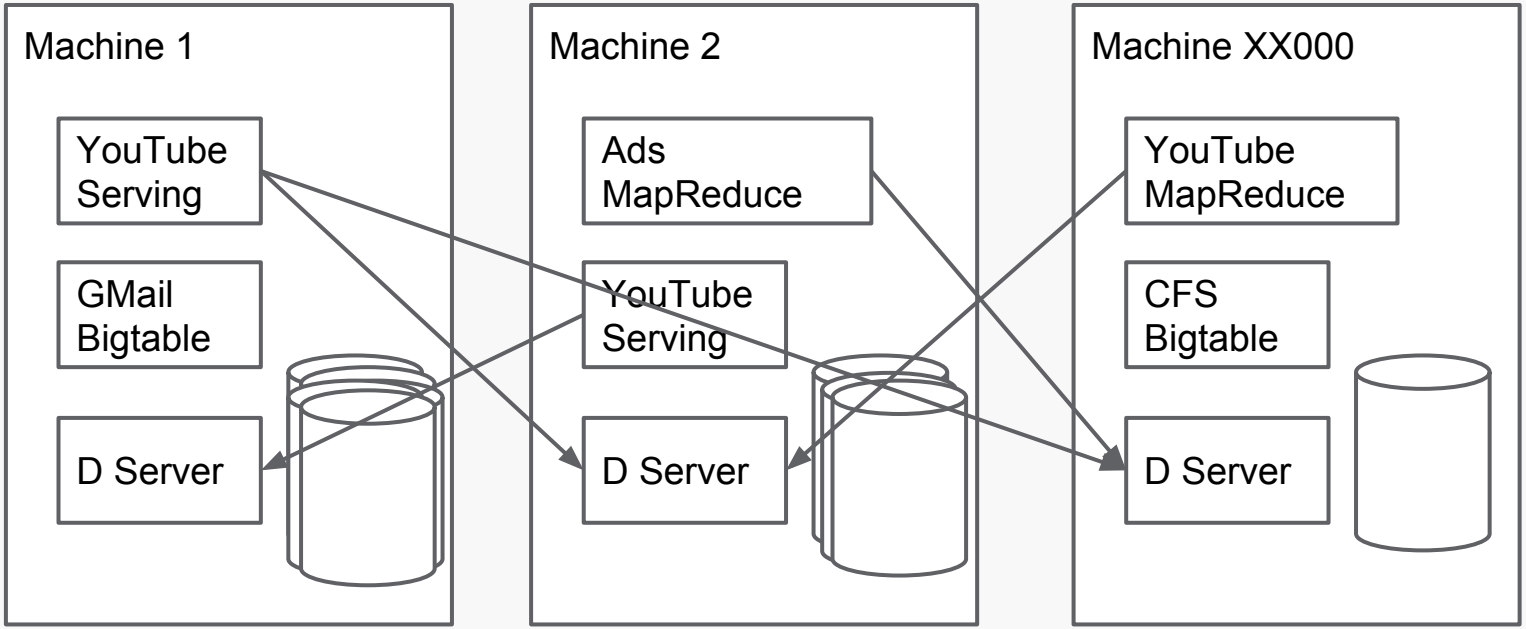
Themes

Colossus enables scale, declustering

Complementary applications → cheaper storage

Placement of data, IO balance is hard

What's a cluster look like?



Let’s talk about money

Total **C**ost of **O**wnership

TCO encompasses much more than the retail price of a disk

A denser disk might sell at a premium $/GB but still cheaper to deploy
(power, connection overhead, repairs)

The ingredients of storage TCO

Most importantly, we care about storage TCO, not disk TCO. Storage TCO is the cost of data **durability** and its **availability**, and the cost of **serving** it

We minimize total storage TCO if we keep the disk **full** and **busy**

What disk should I buy?

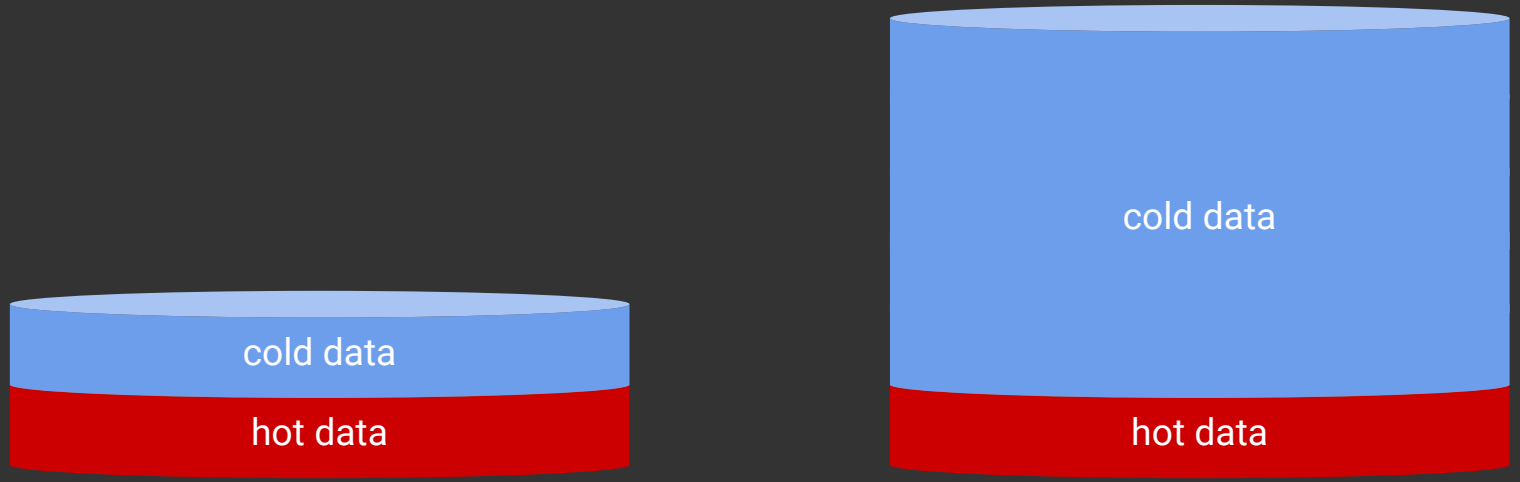
Which disk***s*** should I buy?

We'll have a mix because we're growing

We have an overall goal for IOPS and capacity

We select disks to bring the cluster and fleet closer to our overall mix

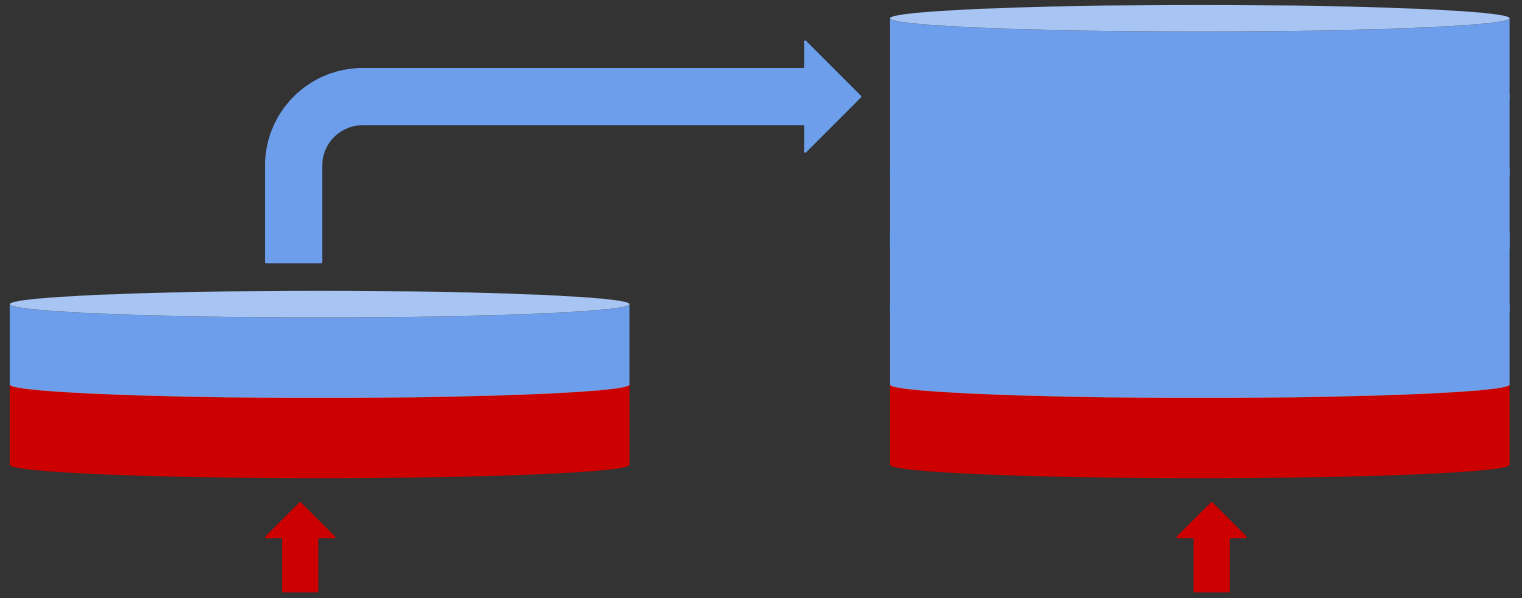
What we want



Equal amounts of hot data (spindle is busy)
Rest of disk filled with cold data (disks are full)

How we get it

Colossus *rebalances* old, cold data



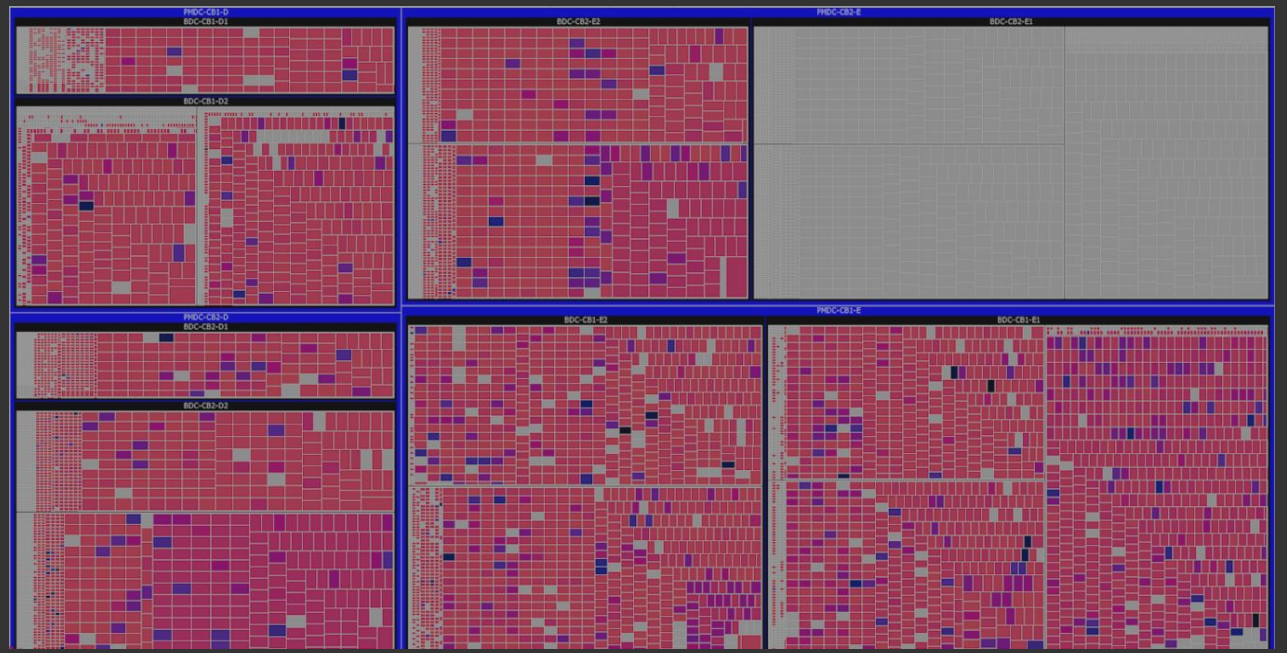
...and distributes newly written data evenly across disks

When stuff works well

Each box is a D server

Sized by disk capacity

Colored by spindle utilization



Rough scheme

Buy flash for caching to bring IOPS/GB into disk range

Buy disks for capacity and fill them up

Hope that the disks are busy

- otherwise we bought too much flash…
- but not too busy…

If we buy disks for IOps, byte improvements don’t help

If cold bytes grow infinitely, we have lots of IO capacity

Filling up disks is hard

Filesystem doesn’t work well when 100% full

Can’t remove capacity for upgrades and repairs without empty space

Individual groups don’t want to run near 100% of quota

Administrators are uncomfortable with statistical overcommit

Supply chain uncertainty

Applications must change

Unlike almost anything else in our datacenters, disk I/O cost is going up

Applications that want more accesses than HDDs offer probably need to think about making their hot data hotter (so flash works well) and cold data colder

An application written X years ago might cause us to buy smaller disks, increasing storage costs

Conclusion

Colossus has been extremely useful for optimizing our storage efficiency

- Metadata scaling enables declustering of resources
- Ability to combine disks of various sizes and workloads of varying types is very powerful

Looking forward, I/O cost trends will require both applications and storage systems to evolve



Thank you!

Denis Serenyi dserenyi@google.com