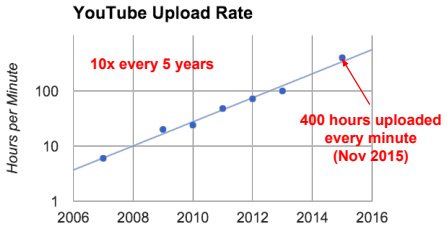
Google

Disks and their Cloudy Future

Eric Brewer
VP Infrastructure, Google



1GB per hour ⇒ 1 PB new storage *every day*
(+ multiple copies, multiple formats)

Most disks will be "Data Center" Disks

Not just YouTube

- Many fast growing storage services
- Google alone also has Drive, Photos, GMail, Cloud, ...

Plus "big data" moving to the Cloud

At the same time, less use of (spinning) disks in PCs and none in mobile

"Disks for Data Centers" White Paper

Released this morning to go with this talk:

http://research.google.com/pubs/pub44830.html (you can Google it by title)

With Lawrence Ying, Lawrence Greenfield, Robert Cypher and Theodore T'so

Covers this material in greater depth

Start of a broad open discussion on "data center" disks

What's different in a Data Center?

1) Collection View
   - Disks are always part of a large collection
   - Optimize for the collection, not the server

2) Focus on Tail Latency
   - Live services: users are waiting for data
   - Metric: 99th-percentile read latency

Current Tail Latency is Poor

Reading data: mean = 10 ms

99% = 100s of ms
99.9% = seconds

Lots of reasons, but disk tells you little (we need profiling)

- sector remapping
- writes delaying reads
- non-data commands
- background tasks in the firmware

Tail Latency: Parallel Requests

Sometimes we send the *same request* to multiple disks
- Wastes some work
- But can still be worthwhile for tail latency

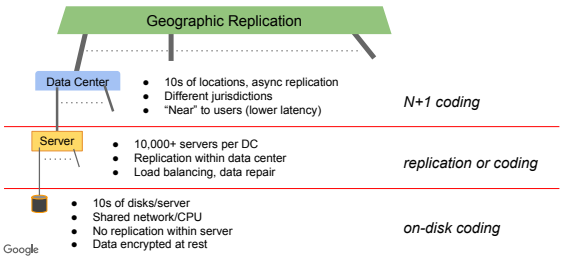
Sometimes we *cancel* pending requests
- … because we got the answer elsewhere
- Limits the wasted work
- Works more often than you might think

Five Metrics for Data-Center Disks

1. Higher I/Os per second (**IOPS**), typically limited by seeks,
2. Higher **capacity**, in GB
3. Lower **tail latency**
4. Meet **security** requirements, and
5. Lower total cost of ownership (**TCO**).

(these are not too novel)

Collection View: Many Layers



Collection View: Durability

Disk promises < 1 in 10^{15} bit error rate (incredible)
- Extensive retries & complex coding
- Background scanning and rewrites

But we already have copies elsewhere!
- Maybe OK to lose a little (more) data
  … if we can trade it for capacity or tail latency

(variation of the "end to end" principle)

Philosophy of New APIs

Goals:
- Control over timing of background work (for tail latency)
- Leverage the disk's extra knowledge of details
  - E.g. what tracks are at risk?
- Teach disk about prioritization, but let it do the scheduling
  - It knows the mechanics
- … but still an abstraction layer
  with multiple implementations

API: Retry Policy

Goal: *per-read* retry policy
1. Traditional: *"really try hard"*
   Use all possible mechanisms to get the data
2. Fast: *"limited retry"*
   Try to read, but fail quickly if you cannot

Improves tail latency (& we have the data elsewhere)
- Can be used with parallel reads
- Can use 1 after 2 if needed

API: Background Work

Lots of background work to do...
Idea: host controls *timing*, but not details

- Disk provides data on *need* for background work
- Host periodically schedules such work
  - … When it will not hurt tail latency
  - Or ideally host can cancel background work if needed
- If host forgets, timer expires & disk can do it
  - But the host should not let this happen

API: fine-grain labeling

We label all I/Os:

- Low latency
- Throughput (think batch workloads)
- Best Effort (background data movement)

We have quotas for the first two (admission control)

- Prerequisite to low tail latency

We track them throughout the systems

API: fine-grain scheduling

If disk doesn't know about labels
... we have to give it one or few I/Os at at time (!)
Otherwise, low latency reads might get delayed

Native Command Queuing (NCQ) is partial solution

- But we need *per-IO* labels and latency targets
- Host should manage quotas via throttling
- Disk should manage I/O reordering based on labels
- Implies real-time scheduler in the disk with reorering

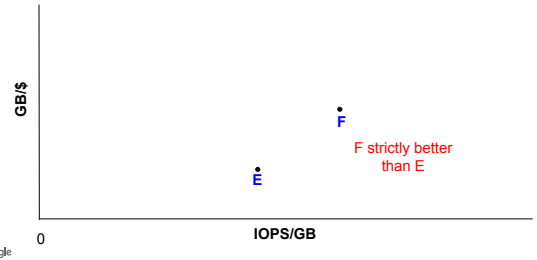
Collection View: Aggregate Mix

We always mix a variety of drives
in part due to incremental deployment of new drives

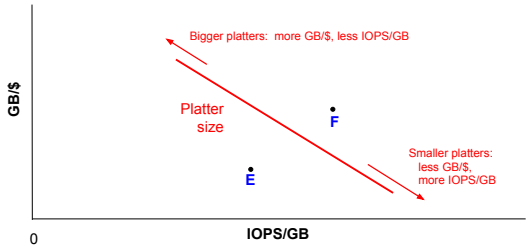
We have overall goals for total IOPS and capacity

We select new disks
to bring the overall fleet mix closer to our goals

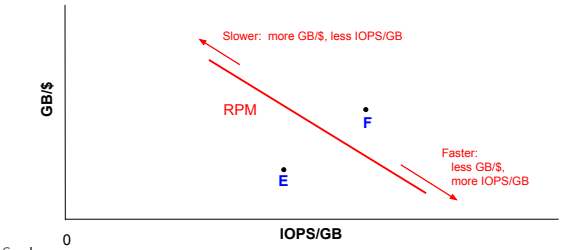
Collection View: Aggregate Mix



Collection View: Aggregate Mix

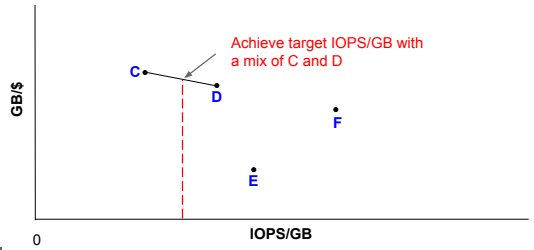


Collection View: Aggregate Mix



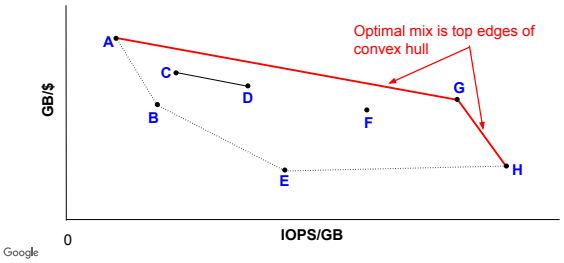
Google

Collection View: Aggregate Mix



Google

Collection View: Aggregate Mix



Google

Collection View: Flexible capacity

We don't need precisely 6TB…

- Just tell us how many sectors are good
- Don't hold any back (disk will get "smaller" over time)

We don't even need constant size over time…

- OK to mark sectors bad
  - please don't remap them to other tracks
- OK to drop capacity when a head/platter dies
  - (might have to reformat to create a new smaller drive)

Google

What about SSDs?

We use them… for caching and high-performance workloads

But not shifting to them en masse anytime soon…

- Cost per GB is too high
- SSDs and HDDs both have good capacity growth *rates*
  - Hard for SSDs to catch up soon
- SSDs have limited program-erase cycles

Most of our storage will be disks for at least 5-10 years

Google

Source of modern disk dimensions: The 3½" PC floppy disk

(might not be optimal)

Wikipedia, Medvedev CC-BY-SA 3.0

Physical Changes

Changing the form factor is a long process
(… and kind of why I am here)
Seems fruitful and getting more so every year…

- What dimensions?
- What power distribution?
- How many heads per platter? Multiple actuators?

Data center volume is high enough to justify a change

Taller Drives?

Assume a fixed total platter area

Taller drives:
⇒ smaller platters, faster RPM & seeks ⇒ higher IOPS
⇒ more platters ⇒ lower GB/$

Areal density improvements ⇒ this a better tradeoff over time

Multi-disk packages?

Not integrated systems like appliances
Instead:

- Multiple disks in combined enclosure
- Combined power distribution
- Shared RAM caching
- Perhaps PCI-E interface

Improves GB/$ at similar IOPS/GB (Maybe 4 disks?)

Summary

Most disks will be in data centers

- Optimize the collection, not the server
- (read) tail latency is really important

Need APIs for vertical integration

- Combine global view with disk's local knowledge
- Disk is more mechanisms and feedback
- Policies in higher-level systems

New form factor(s) probably make sense
Start of a broad, open discussion