Lecture S7: Reliability

****
Review -- 1 min
****
Transactions
- ACID: atomicity, consistency, isolation, durability
- logging
- LFS,

****
Outline - 1 min
****
Reliability
- disk reliablity
- RAIDs

****
Preview - 1 min
****

****
Lecture - 20 min
****

1. Disk Reliability

Simple model

Disk advertises: MTTF and nonrecoverable read error.

E.g.,

1,500,000 hours MTTF
1 unrecoverable read-sector error per 10^16 bits read

What does this mean?

MTTF -- mean time to (total device) failure
-- head failure (e.g., electrostatic discharge, physical impact with contaminating particles, heat damage from rubbing against platter)
-- damage to track positioning information
-- bad electronics, capaciters, etc.

1.5M hours ~= 171 years (!)

MTTF v. Lifetime
-- MTTF estimate is during "useful life" of device (e.g., 5 years)
-- Bathtub model

--> 1.5M hours really means "during first 5 years of operation, expect to lose .5%-1% of disks per year"

Note: advertised MTTF failure rates v. reality
Empirical field studies have found
-- higher failure rates (2%-3%/year)
-- increasing failure rates during deployment (not quite bathtub)
-- variable failure rates across different production runs (even for same disk from same manufacturer; certainly across different models and different manufacturers)

Also note: A deployment may have many disks. E.g., cs department has several hundred; google has millions
--> failures are routine (yearly, monthly, daily, hourly occurrance depending on scale.)

Nonrecoverable read error
-- bad media, corrosion, scratch from contamination, manufacturing defect (not always immediately apparent. e.g., material flakes off --> bad sector + particle that may damange other sectors)
-- "high fly write" - head gets too far away during a write
-- disk head impact (thermal damage)

For high-throughput sequential reads/writes
1 error / 10^16 bits * 120 MB/s
~= 1 error / 10^16 bits * 10^9 bits/s
~= 1 error/10^7 seconds
~= 1 error/115 days (!)

Challenges (basic model):
-- disk failure (lose one disk)
-- sector corruption (lose a few sectors on a disk)
    -- detected in HW or not

Remember: manufacturer gives MTTF estimate but, your milage may vary
-- correlated failures
-- bathtub curve
-- environmental challenges
-- Google reports seeing about 2% of disks fail per year

Key techniques
(1) Transactions (see above)
(2) Redundancy: checksums -- End-to-end checksum a la ZFS
-- Previously: rely on disk HW checksums
-- Many recent new file systems include additional checksums (ZFS, google FS, HDFS, ...)
(3) RAID: Redundancy: redundant copies of data

2. Checksums

Each sector encoded with checksum --> detect/corrects some errors (included in "nonrecoverable read error" estimate)

Old file systems relied on hardware checksums
-- occasional false negative
-- doesn't detect data written to wrong sector (e.g., software/firmware error)
-- doesn't detect bus controller bug
etc.

Recent file systems include stronger checksums on data
-- more bits
-- end-to-end principle
e.g., ZFS checksum tree
e.g., google FS, HDFS

ADVICE: Current state of the art is to include end-to-end checksums; advise this for any system you design (and at least consider it for any system you buy)

3. RAIDS and reliability

So, you've detected an error (either through hardware checksum or file system checksum). Now what?

Need to repair the damage.

~~Data stored to disk is supposed to be permanent. Physical reality -- disks fail~~
~~-- Today disks advertise ~1.4million hour MTTF~~
~~-- 1.4M hours/8760 hours/year = annualized failure rate of .6%~~
~~-- expect to lose ~.5-1% of your disks each year~~
~~-- some reports from large deployed systems see higher annualized failure rates (~2%)~~

~~Note MTTF 1.4M hours = 160 year MTTF does not mean disks will last 100+ years. This is failure rate during useful life (bathtub curve)~~

~~If you have 1 disk, this should make you nervous. You shouldn't ignore it.~~
~~If you have 10 or 100 disks, you can't ignore it.~~

Organization may have hundreds or thousands of disks

Suppose you need to store more data than fits on a single disk? How should you arrange data across disks?

Naive option: treat disk as huge pool of disk blocks so that:

disk 1 has blocks 1, k+1, 2k+1, ….
Disk 2 has blocks 2, k+2, …
…

Benefits

- load gets balanced automatically across disks
- can transfer large files at aggregate BW of all disks

Problem: what if one disk fails?
Big problem: for k disks k times as likely to have a failed disk at any given time

Availability v. reliability

Availiability – never lose access to data; system should continue working even if some components are not working (liveness)
Reliability – never lose data (safety)

(Battery runs out on my laptop makes storage unavailable but hopefully not unreliable)

RAID

RAID -- redundant array of inexpensive disks
-- use redundancy to improve reliablity

In RAID, dedicate one disk to hold parity for other disks in stripe

disk 1 has blocks 1, K+1, 2K+1, …
disk 2 has blocks 2, K+2, 2K+2, …
…
parity disk has blocks parity(1…k), parity(K+1…2K)…

details
-- block is at least several KB; can be larger

-- rotate parity -- [better still, rotate parity across disks; disk 0 has parity blocks for stripes 0, N, 2N, ...]; improves small-write performance
-- need transactions to update blocks + parity atomically

If lose any disk, can recover data from other disks plus parity
ex:

disk 1 has 1 0 0 1
disk 2 has 0 1 0 1
disk 3 has 1 0 0 0
parity has 0 1 0 0

What if we lose disk 2? Its contents are parity of remainder!
Thus can lose any disk, and data is still available

Details:

- disk failures are "fail-stop" – disks tell you when they fail
- update – read-modify-write data and parity *atomically*
  - solution – write-ahead logging or log-structure

Preferred Customer
Comment:

Simple (naïve) analysis

why does this work?

Suppose MTTF = 100K hours (11.5 years)
Department has 100 disks → 100K/100 until first failure = 1000 hours = lose data every 41.66 days!

Suppose MTTR = 10 hours and we arrange disks as 99 disks + 1 parity
QUESTION: 1% better? 2x better? 10x better?

Assuming independent failures(*) – need to get unlucky and have a second failure before the first disk is fixed

e.g., 41 days until the first failure happens, then race to fix disk before next one fails. Since I fix the disk in 10 hours and the next disk is expected to fail in 1000 hours, I win this race 99 out of 100 times

→ MTTDL = 100 * 1000 = 100K hours

ANSWER 100x better. 1% reduction in effective space gets 100x improvement in reliability!

Of course, I can improve this further by
(1) using more redundancy (e.g., 1 parity per 10 rather than 1 per 100)

Typical deployments – 1-2 parity per 1-10 disks; (e.g., 3 replicas of data in Google file system)

standard (naive) formula
one parity disk per group:
MTTDL = MTTF^2_disk / (N * (G-1) * MTTR_disk)

e.g., 100 disks in groups of 9 data + 1 parity; 10 hr mttr

100K^2 / (100 * 9 * 10) = 1.1M hours (>100 years)

Intution: MTTF/N = time for first failure
MTTF/G-1 = time to second failure after first occurs
(MTTF/G-1)/MTTR_disk -- probability second failure occurs before first disk repaired

for 2 parity disks per group:

MTTDL_2 parity ~= MTTF_1 parity * MTTF/(G-2)MTTR =
MTTF^3_disk / (N * (G-1) * (G-2) * MTTR^2_disk)

e.g., 100 disks in groups of 8+2
100K^3/(100*9*8*10^2) = 10^15/10^5 = 10B hours (>1M years)

--> You can get really gaudy numbers with naive formula and a small amount of redundancy

(2) improving repair time

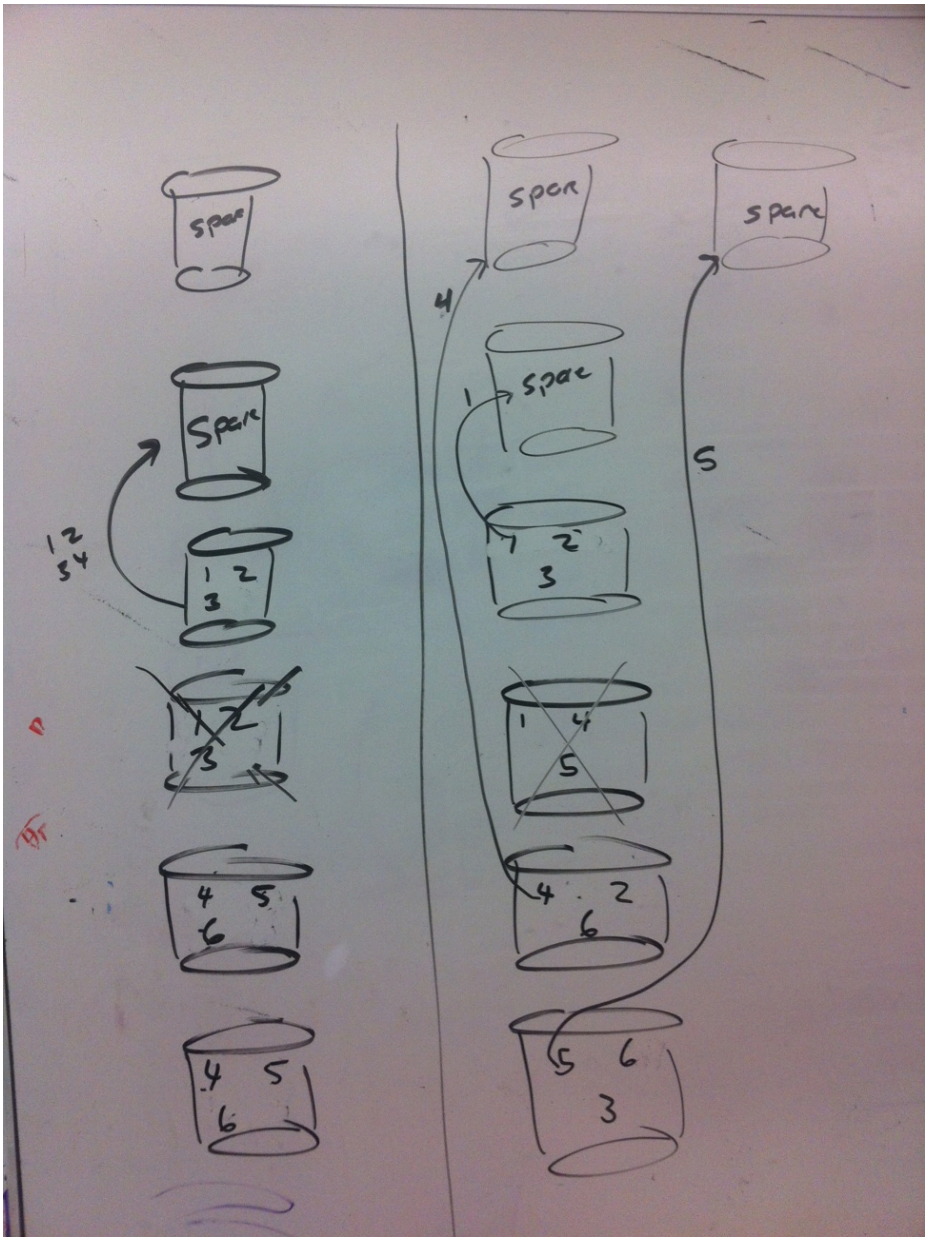
Hot swap
"Hot swap" – immediate switch to new disk in seconds/minutes (possibly w/o operator intervention)

Unfortunately, physical limits of disk put lower bound on repair time. E.g.,

Limited by time to read/write data – if dedicate 100% of disk BW to repair, 1TB/50MB/s = 20K seconds – 6 hours; more if reads are slowed down because of non-repair traffic; technology trends – this number is rising
→
(a) Have enough redundancy to survive multi-hour MTTR

(b) **Declustering** – spread repair load – instead of organizing 100 disks into 10 groups of 10, send each data item to 10 random disks out of 100 ; if a disk fails, send the repair traffic to a random disk (excluding the ones already used for that data) → each remaining disk supplies 1% of the repair reads and receives 1% of the repair writes → repair in minutes not hours

PICTURE:



(Note: For simplicity picture shows separate spare disks; enhancement is to just leave spare storage capacity on existing disks. I'll leave that as an exercise...)

3.1 Problem: Naive estimate is wildly optimistic

The above equation is wildly optimistic. 100 disks in groups of 9+1 → 100 years? No way.

NOTE: independent failure assumption is way too optimistic

(1) "bathtub" lifetime – quoted MTTF only valid during intended service lifetime (e.g., first 3-5 years of services possibly not including burn-in)

(2) Field data v. measured data (2-4x failure rate; increasing failure rate over time)

(3) environmental **correlation** – power surges, vibration, manufacturing defect, faulty controller or server, …

(4) Unrecoverable read errors

Early raid formula ignored these. On modern drives, they can be more common than total disk failures (see above).

we had
whole disk failure + whole disk failure = data loss

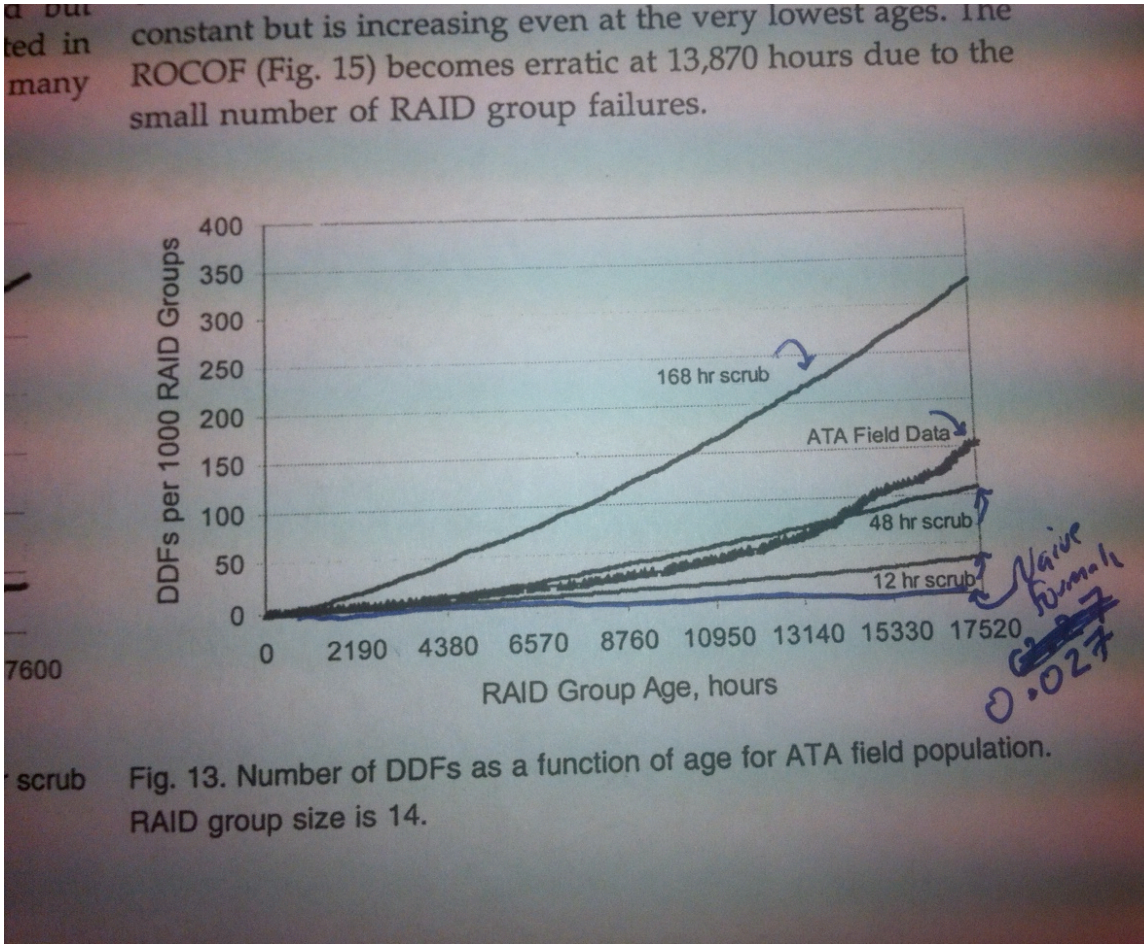
now we also have
whole disk failure + nonrecoverable error = data loss

(Note: We ignore nonrecoverable error + nonrecoverable error because odds of losing the same sector on multiple disks is tiny)

Problem: Over time bad sectors accumulate at some rate; eventually, it is inevitable that when you try to recover from a failed disk, you will find that one of the sectors you need to read is gone.

Solution: Scrubbing
Periodically read all data blocks; detect errors; correct them

Scrubbing rate is limited by disk size v. bandwidth and by what fraction of bandwidth you are willing to give up to maintenance (scrubbing)



[[Elerath and Pecht "A Highly Accurate Method for Assessing Reliability of Redundant Arrays of Inexpensive Disks (RAID)" IEEE Transactions on Computers V 58 n 3 march 2009]]
17520 hours = 2 years;
naive v. field data is ~100x after year 1

3.2 Common configurations today:

Mirrored: 2 identical disks (write to both, read from either)
3-5 data + 1 parity
3-way replication
5-10 data + 2 parity

Advice: double redundancy increasingly sensible for important data (You may be able to get by with 1 parity or mirroring + additional backup copy)

3.3 Environmental factors

Disk failure is not the only thing you need to worry about

Other "advanced" sources of failure

- Operator error
- Malicious operator
- Malware: Virus, ransomware
- Fire, flood, hurricane, …
- Bankruptcy of outsourced storage provider
- FBI raid on collocation center (!)
  http://blog.wired.com/27bstroke6/2009/04/data-centers-ra.html
-
-

One solution: SafeStore – geographic, operator, organization, software diversity; restrict interface;

http://www.cs.utexas.edu/users/dahlin/papers/SafeStore-USENIX07.pdf

****

Summary - 1 min

****

Key idea: redundancy