RAID!!!

Berkeley EECS
ELECTRICAL ENGINEERING & COMPUTER SCIENCES

Error Detection and Error Correction

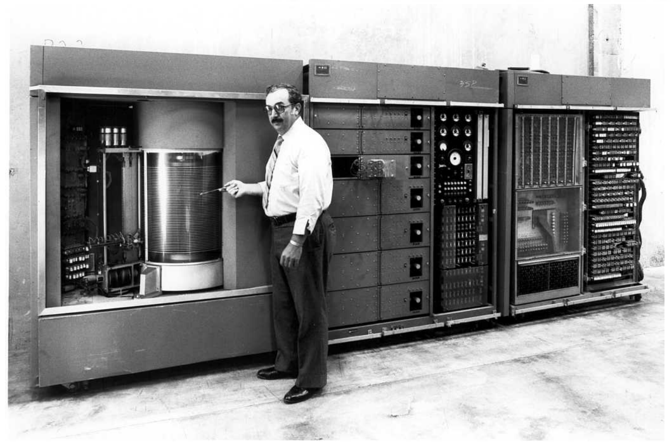
- As a reminder:
  - It takes a lot of bits but one can do error correcting codes
    - EG, Hamming codes
  - But a lot fewer bits to just detect errors
- Therefore you get a lot of noisy systems with CRC or cryptographic hash
  - Result is either CORRECT or ERROR
  - You can't get INCORRECT
  - This applies to network, disk, etc etc etc.

Why Worry About Disk At All?

- Spinning disk is still a critical technology
  - Although worse latency than SSD…
- Disk has equal or greater bandwidth and an order of magnitude better storage density (bits/cm^3) and cost density (bits/$)
- So when you need to store a petabyte or three…
  - You need to use disk, not SSDs
- Oh, and SSDs can fail too

Evolution of the Disk Drive

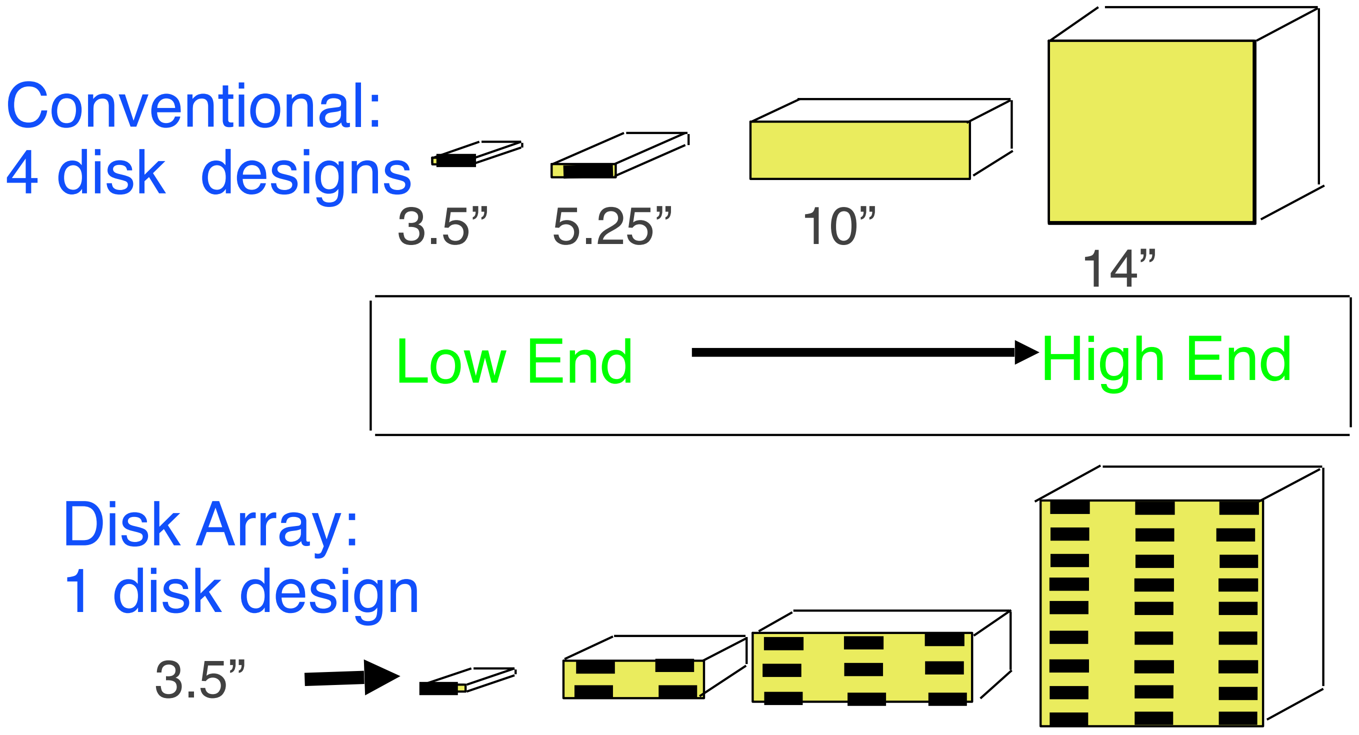
IBM 3390K, 1986


IBM RAMAC 305, 1956

Apple SCSI, 1986

Arrays of Small Disks

Can smaller disks be used to close gap in performance between disks and CPUs?



Replace Small Number of Large Disks with Large Number of Small Disks! (1988 Disks)

	IBM 3390K	IBM 3.5" 0061	x70	
Capacity	20 GBytes	320 MBytes	23 GBytes	
Volume	97 cu. ft.	0.1 cu. ft.	11 cu. ft.	9X
Power	3 KW	11 W	1 KW	3X
Data Rate	15 MB/s	1.5 MB/s	120 MB/s	8X
I/O Rate	600 I/Os/s	55 I/Os/s	3900 IOs/s	6X
MTTF	250 KHrs	50 KHrs	??? Hrs	
Cost	$250K	$2K	$150K	

Disk Arrays have potential for large data and I/O rates, high MB per cu. ft., high MB per KW, but what about reliability?

Clicker Question: MTTF

- You have a disk with MTTF of 10,000 hours...
  - Failures are independent, random, and uniformly distributed
- What is the MTTF for a single disk in a set of 100 disks?
  - a: 10,000 hours
  - b: 1,000 hours
  - c: 100 hours
  - d: 10 hours
  - e: I hate clicker questions

But MTTF goes through the roof…

- If 1 disk as MTTF of 50k hours…
  - 70 disks will have a MTTF of ~700 hours!!!
  - This is assuming failures are independent…
- But fortunately we know when failures occur!
  - Disks use a lot of CRC coding, so we don't have corrupted data, just no data
- We can have both "Soft" and "Hard" failures
  - Soft failure just the read is incorrect/failed, the disk is still good
  - Hard failures kill the disk, necessitating replacement
    - Most RAID setups are "Hot swap":
      Unplug the disk and put in a replacement while things are still going
    - Most modern RAID arrays also have "hot spares":
      An already installed disk that is used automatically if another disk fails.

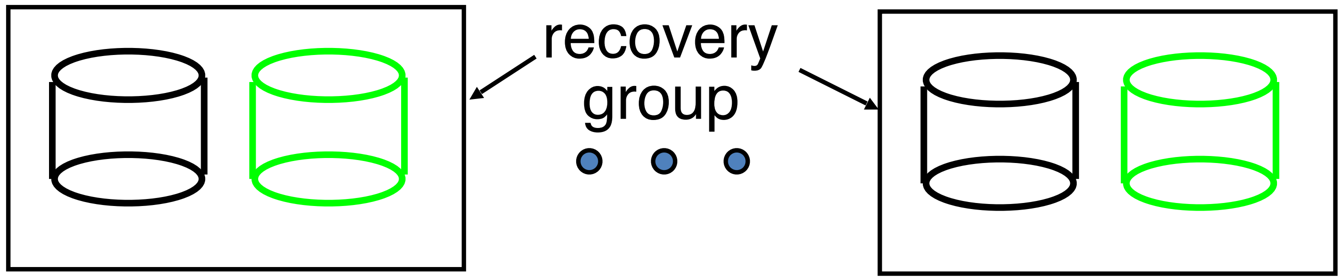
RAID: Redundant Arrays of (Inexpensive) Disks

- Files are "striped" across multiple disks
- Redundancy yields high data availability
  - Availability: service still provided to user, even if some components failed
- Disks will still fail
- Contents reconstructed from data redundantly stored in the array
  - Capacity penalty to store redundant info
  - Bandwidth penalty to update redundant info on writes

Raid 0: Striping

- "RAID 0" is not actually RAID
  - Its simply spreading the data across multiple disks
- So, e.g, for 4 disks, address 0 is on disk 0, address 1 is on disk 1, address 2 is on disk 2, address 4 on disk 0...
- Improves bandwidth linearly
  - With 4 disks you have 4x the disk bandwidth
- Doesn't really help latency
  - Still have the individual disks seek and rotation time
- And well, failures happen...

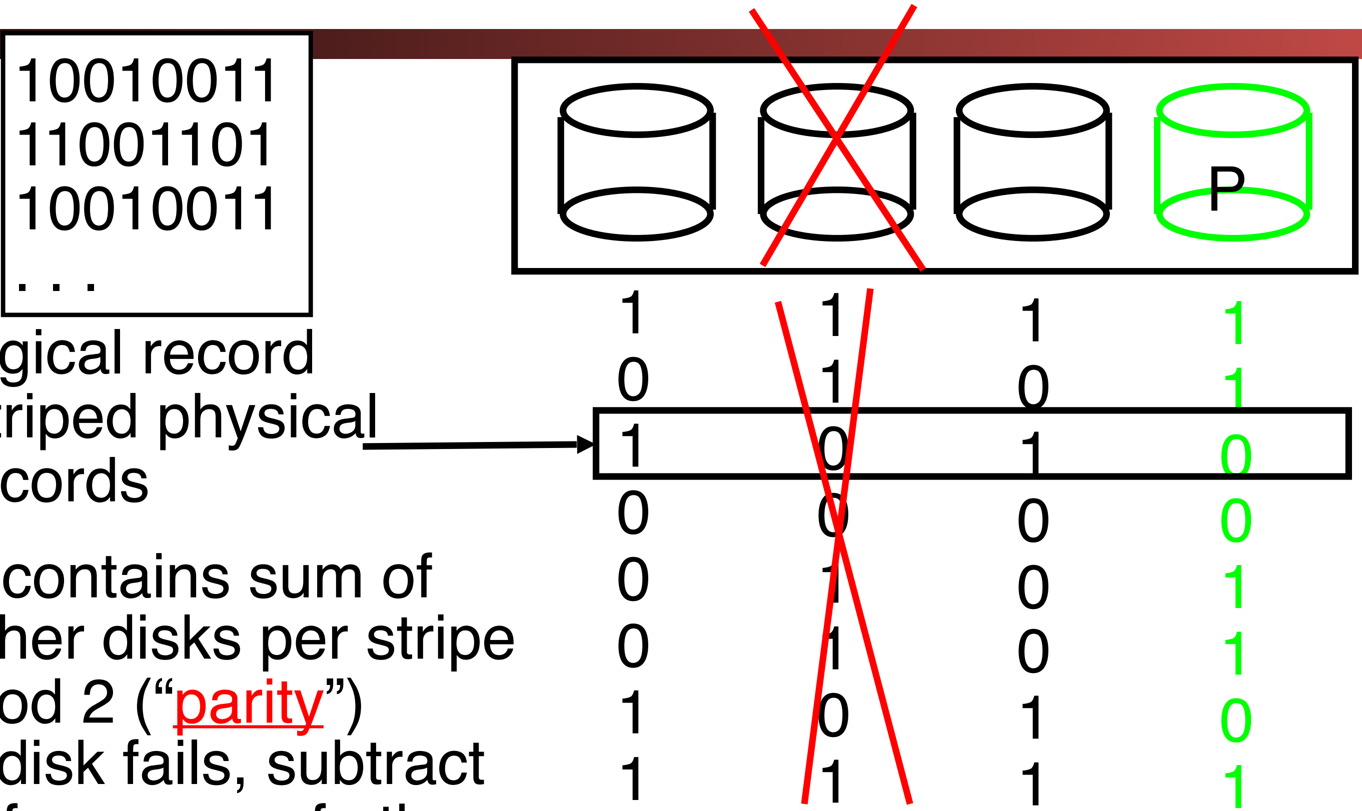
Redundant Arrays of Inexpensive Disks RAID 1: Disk Mirroring/Shadowing



- Each disk is fully duplicated onto its "mirror"
  Very high availability can be achieved
- Writes limited by single-disk speed
- Reads may be optimized

Most expensive solution: 100% capacity overhead

Redundant Array of Inexpensive Disks RAID 3: Parity Disk



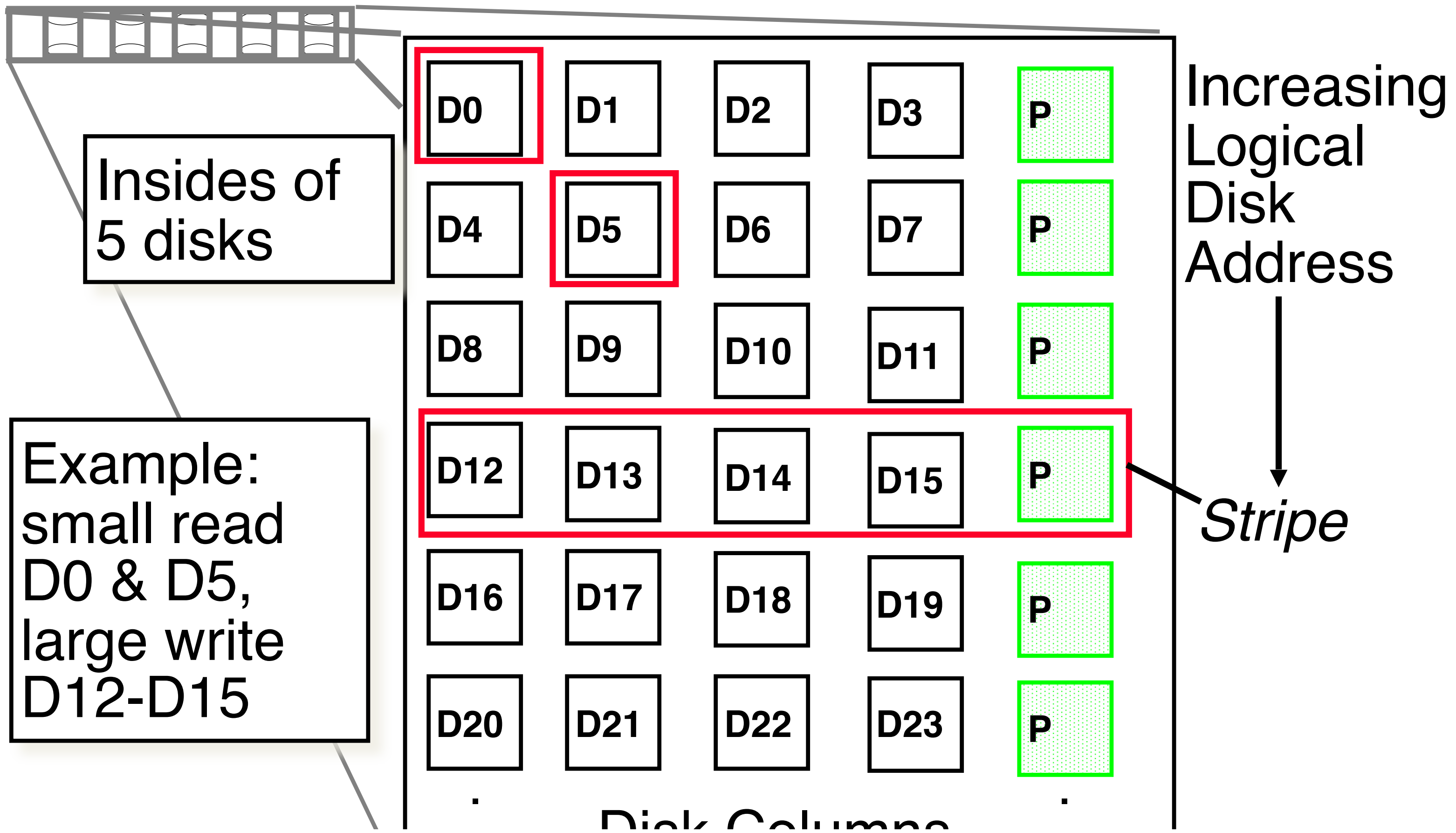
logical record

Striped physical records

P contains sum of other disks per stripe mod 2 ("parity")

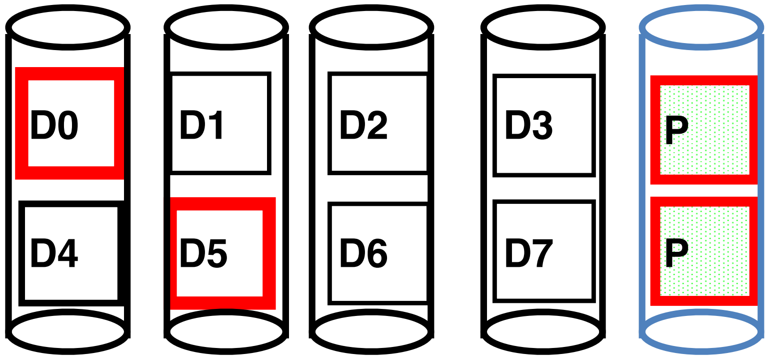
If disk fails, subtract P from sum of other disks to find missing information

Redundant Arrays of Inexpensive Disks RAID 4: High I/O Rate Parity



Inspiration for RAID 5

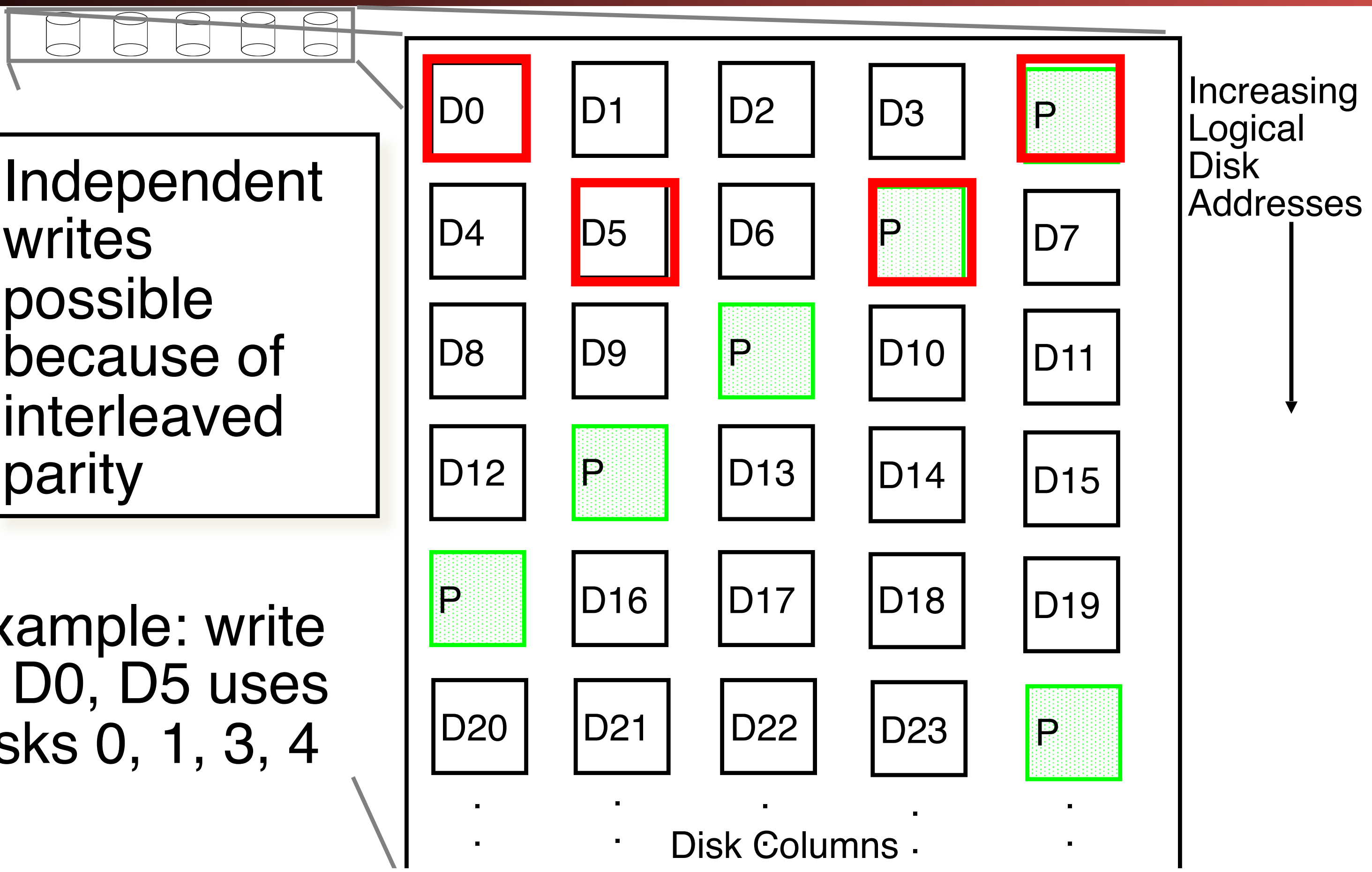
- RAID 4 works well for small reads
- Small writes (write to one disk):
  - Option 1: read other data disks, create new sum and write to Parity Disk
  - Option 2: since P has old sum, compare old data to new data, add the difference to P
- Small writes are limited by Parity Disk: Write to D0, D5 both also write to P disk



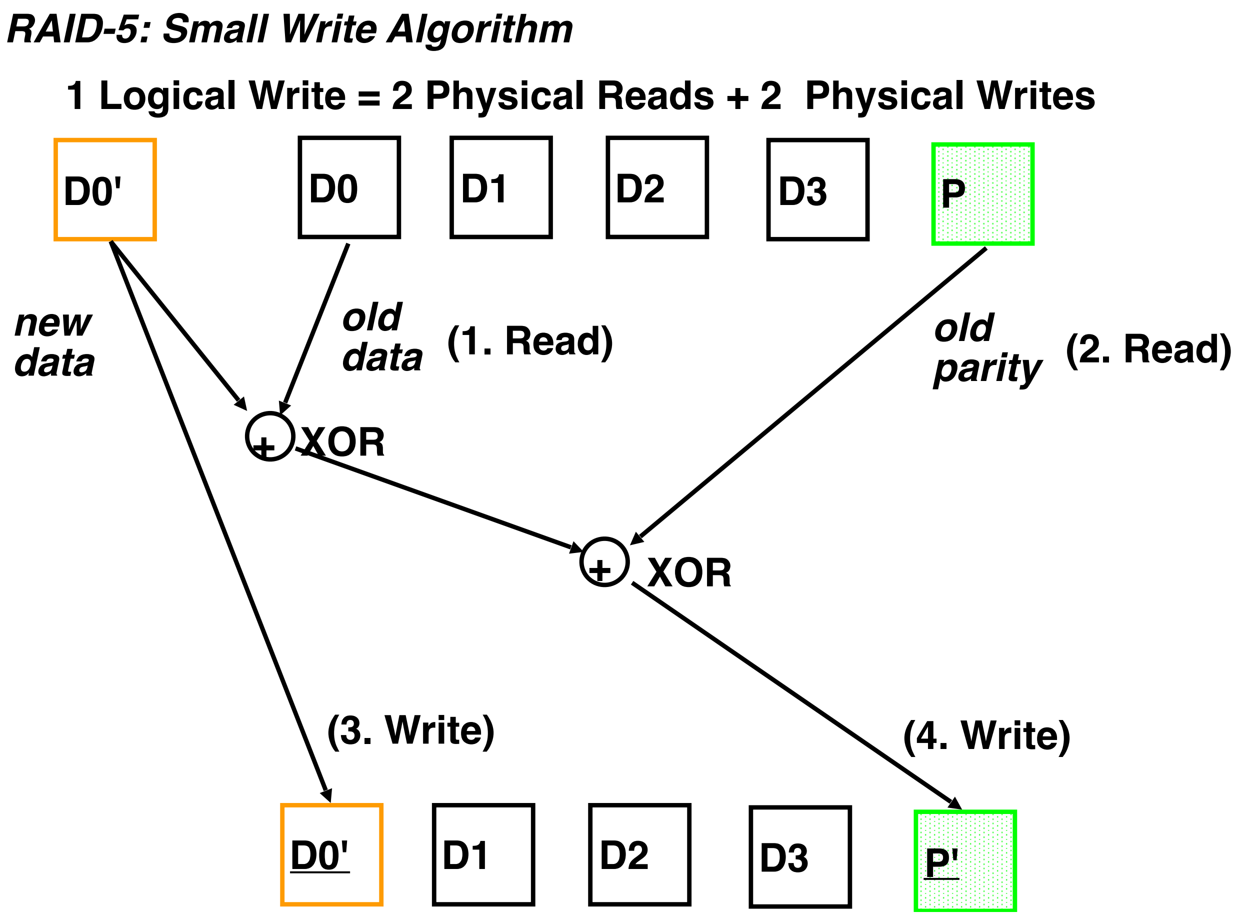
RAID 5: High I/O Rate Interleaved Parity

Independent writes possible because of interleaved parity

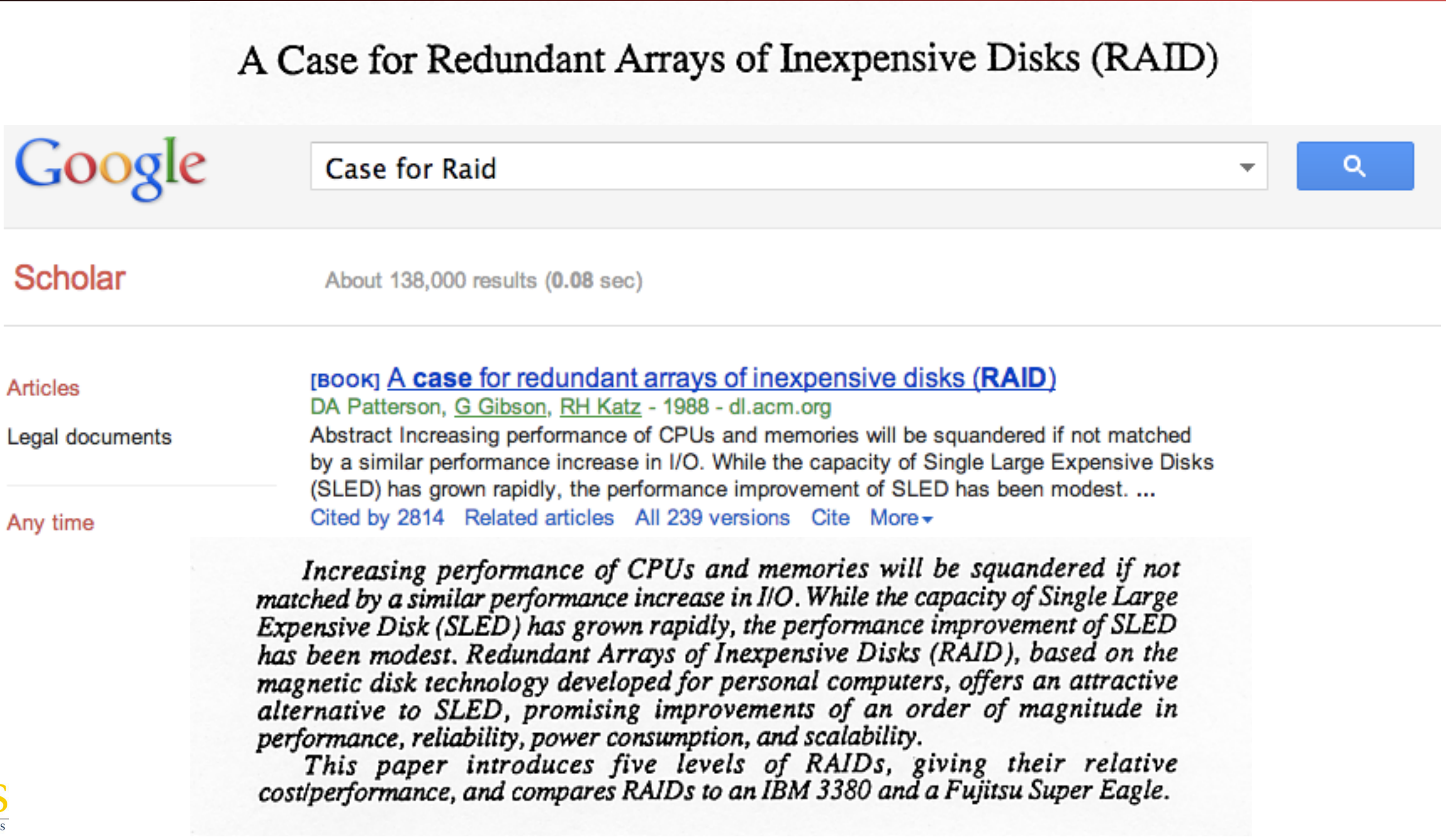
Example: write to D0, D5 uses disks 0, 1, 3, 4



Problems of Disk Arrays: Small Writes



Tech Report Read 'Round the World (December 1987)



A Case for Redundant Arrays of Inexpensive Disks (RAID)

About 138,000 results (**0.08** sec)

[BOOK] A **case** for redundant arrays of inexpensive disks (**RAID**)
DA Patterson, G Gibson, RH Katz - 1988 - dl.acm.org
Abstract Increasing performance of CPUs and memories will be squandered if not matched by a similar performance increase in I/O. While the capacity of Single Large Expensive Disks (SLED) has grown rapidly, the performance improvement of SLED has been modest. ...
Cited by 2814 Related articles All 239 versions Cite More

Increasing performance of CPUs and memories will be squandered if not matched by a similar performance increase in I/O. While the capacity of Single Large Expensive Disk (SLED) has grown rapidly, the performance improvement of SLED has been modest. Redundant Arrays of Inexpensive Disks (RAID), based on the magnetic disk technology developed for personal computers, offers an attractive alternative to SLED, promising improvements of an order of magnitude in performance, reliability, power consumption, and scalability.

This paper introduces five levels of RAIDs, giving their relative cost/performance, and compares RAIDs to an IBM 3380 and a Fujitsu Super Eagle.

RAID-I

- RAID-I (1989)
    - Consisted of a Sun 4/280 workstation with 128 MB of DRAM, four dual-string SCSI controllers, 28 5.25-inch SCSI disks and specialized disk striping software

RAID II

- 1990-1993
- Early Network Attached Storage (NAS) System running a Log Structured File System (LFS)
- Impact:
  - $25 Billion/year in 2002
  - Over $150 Billion in RAID device sold since 1990-2002
  - 200+ RAID companies (at the peak)
  - Software RAID a standard component of modern OSs

Oh, and disk failures are often correlated...

- You don't just have one disk die...
  - You can have more die in a short period of time
  - Thank both the "bathtub curve" and common environmental conditions
- If you care about your data, RAID isn't sufficient!!!
  - You need to also consider a separate backup solution

So Lets Put it In Action…

- Say I'm working for some spy agency…
- The targeting problem: I need the capability to ***target*** anybody
  - For whatever use of "target": advertising, profiling, surveilling, drone-striking…
- The retrospective corollary: I won't know until tomorrow what I wish I knew today
- So ***collect it all, on everybody, always*** proves the optimal solution if you can afford it…

Consider a Spherical Cow…
Is ***Collect It All*** affordable?

- An extreme case: I want to track ***everybody!*** so 10B people
- Interesting records:
  - 8B -> Unique ID #
  - Name & Address -> 100B
  - Location Record -> 64B
  - Decent ID photo -> 50kB
  - Website Pageview -> 200B
  - Purchase Record -> 200B
  - etc etc etc…

And Types of Computations: Mostly-Write Only Datasets

- Query Focused Datasets (QFDs) in NSA-parlance
  - Requires the ability to specify a limited query that can be hash-computed
- Write data into hash buckets
  - Store buckets in memory and only flush to disk when they reach the target block size (probably 64 MB), without bothering to sort in the bucket
- Read data by searching the appropriate buckets
  - Read entire blocks and search through for desired record
- Optimizes for spinning disk and mostly write-only:
  - Writes are batched up and done in bulk
  - Writes are unsorted to save time
  - Search in read is not a problem: Disk latency dominates anyway

Streaming Computation…

- This is more the classic "map/reduce" flow we all know and love
  - For every element in the dataset do X
- Mostly makes sense on data ingress
  - Split data into light ("metadata") and heavy flows on the data itself
    - With different retention times for different types of data
- The problem: it will ***always*** be (relatively) slow unless you can restrict operation to a smaller dataset

Building Block #1: Storage Pods

- EG Storinator S45 Turbo storage pod…
  - Dual core computer w 32 GB RAM
  - Multiple 10 Gbps Ethernet interfaces
  - 45 8 TB hot-swap disks
    - Say ~300TB usable storage after RAIDing
    - Say ~100TB usable after HDFS triple-redundancy
- Result is a "storage budget" of 10kB per person
  - For ~$40k in cash and 4u of space

Building Block #2: 1u Servers

- Commodity 1u, 10 disk compute-node servers
  - 2 processor, 9 2TB drives, 128 GB of RAM
- Used for both computation and in-memory indexing
  - Lot skimpier data budget however:
  - 10B/person in RAM (no redundancy)
  - 400B/person on disk (HDFS redundancy)
  - ~$15k/node budget

Phase 1:
Two Racks, ~$1M

- 80u space -> 10 storage pods and 40 nodes
- Computational budget:
  - 400B/person RAM
  - 16kB/person computation disk
  - 100kB/person storage disk
- OK, so this is starting to get a little interesting.

What to do with 100kB a person?

- Can have a really ***nice*** mostly static profile
  - Photos for image recognition
  - Current and previous address history
  - List of known accounts and identifiers
  - List of known associates and relationships
  - Airline travel history
  - Border crossing history
- Updating a minor pain
  - “Append-info” buckets and then occasionally prune/reduce

Phase 2: One Container (20 racks) $10M

- Basic strategy is “replicate”, so now its time to switch to even bigger blocks
  - Standard solution these days: just add power & AC
- Gives us a lot more storage per person
  - 4 kB RAM/person
  - 160 kB/person compute storage
  - 1 MB/person data storage
- At this point, we can now start talking continuous tracking…

So at 1 MB per person…

- A day’s location history is perhaps 10 kB
  - So that’s 100 days, but…
    - Why not store “typical days and deviations”
    - Or decay the older data…
- A good insight into web browsing history or email metadata
  - DNS lookups, pageviews, etc

Phase 3: 10 Containers (200 racks) $100M

- Now we are talking!
  - 40 kB RAM/person
    - Major in-memory indexes!
  - 1.6 MB computation disk/person
  - 10 MB storage/person
- If you haven't guessed by now, the scaling is pretty darn linear…
  - And the utility may grow superlinear, or at least with interesting steps… 10 MB/person worldwide is an impressive dossier