EECS 262a
Advanced Topics in Computer Systems
Lecture 4

Filesystems (Con't)
February 1st, 2016

John Kubiatowicz
Electrical Engineering and Computer Sciences
University of California, Berkeley

http://www.eecs.berkeley.edu/~kubitron/cs262

Today's Papers

- The HP AutoRAID Hierarchical Storage System (2-up version), John Wilkes, Richard Golding, Carl Staelin, and Tim Sullivan. Appears in *ACM Transactions on Computer Systems*, Vol. 14, No, 1, February 1996, Pages 108-136.
- Finding a needle in Haystack: Facebook's photo storage,Doug Beaver, Sanjeev Kumar, Harry C. Li, Jason Sobel, Peter Vajgel. Appears in *Proceedings of the USENIX conference in Operating Systems Design and Implementation* (OSDI), 2010

- System design paper and system analysis paper
- Thoughts?

Array Reliability

- **Reliability of N disks = Reliability of 1 Disk ÷ N**

  50,000 Hours ÷ 70 disks = 700 hours

  Disk system MTTF: Drops from 6 years to 1 month!

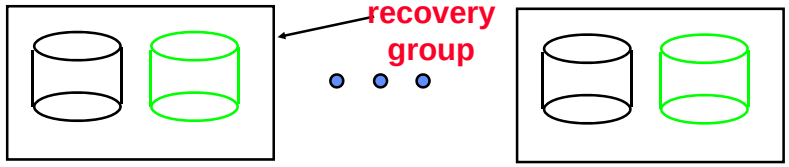
- **Arrays (without redundancy) too unreliable to be useful!**

Hot spares support reconstruction in parallel with access: very high media availability can be achieved

RAID Basics (Two optional papers)

- Levels of RAID (those in **RED** are actually used):
  - RAID 0 (JBOD): striping with no parity (just bandwidth)
  - **RAID 1: Mirroring** (simple, fast, but requires 2x storage)
    - » 1/n space, reads faster (1 to Nx), writes slower (1x) – why?
  - RAID 2: bit-level interleaving with Hamming error-correcting codes (ECC)
  - RAID 3: byte-level striping with dedicated parity disk
    - » Dedicated parity disk is write bottleneck, since every write also writes parity
  - RAID 4: block-level striping with dedicated parity disk
    - » Same bottleneck problems as RAID 3
  - **RAID 5: block-level striping with *rotating* parity disk**
    - » Most popular; spreads out parity load; space 1-1/N, read/write (N-1)x
  - **RAID 6: RAID 5 with two parity blocks (tolerates two drive failures)**
- Use RAID 6 with today's drive sizes! Why?
  - Correlated drive failures (2x expected in 10hr recovery) [Schroeder and Gibson, FAST07]
  - Failures during multi-hour/day rebuild in high-stress environments

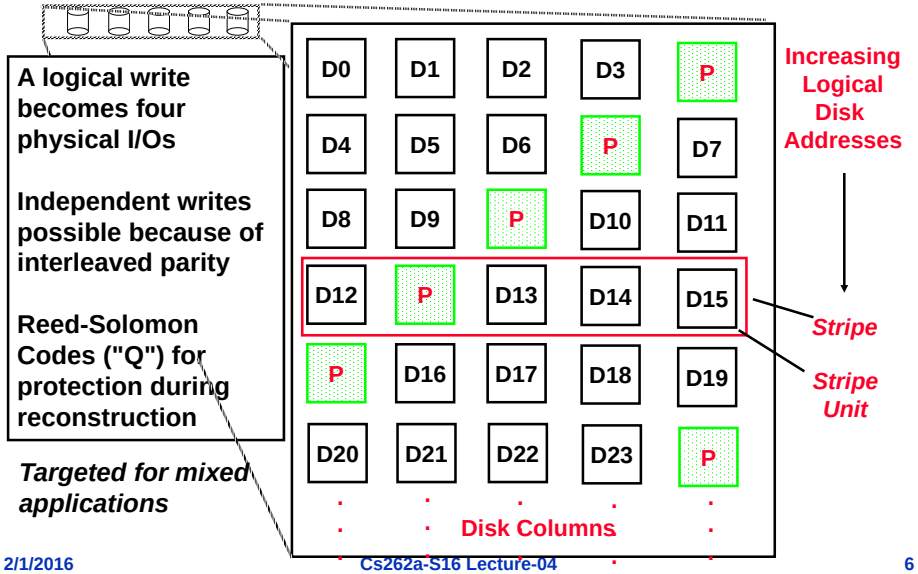
Redundant Arrays of Disks RAID 1: Disk Mirroring/Shadowing



- Each disk is fully duplicated onto its "shadow"
  Very high availability can be achieved
- Bandwidth sacrifice on write:
  Logical write = two physical writes
- Reads may be optimized
- Most expensive solution: 100% capacity overhead

Targeted for high I/O rate , high availability environments

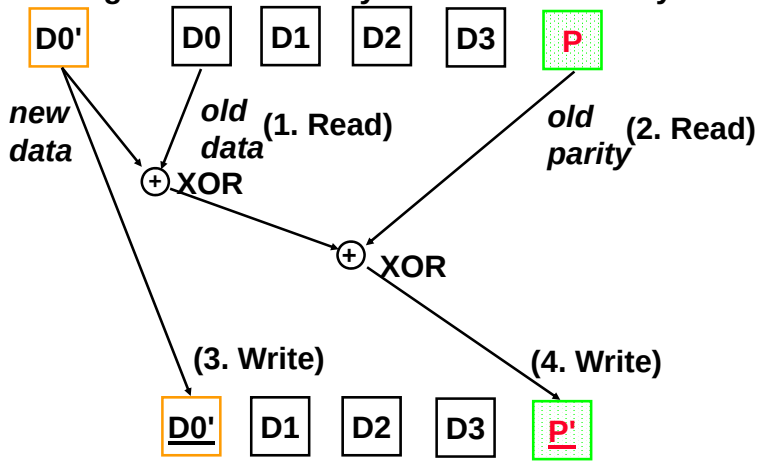
Redundant Arrays of Disks RAID 5+: High I/O Rate Parity



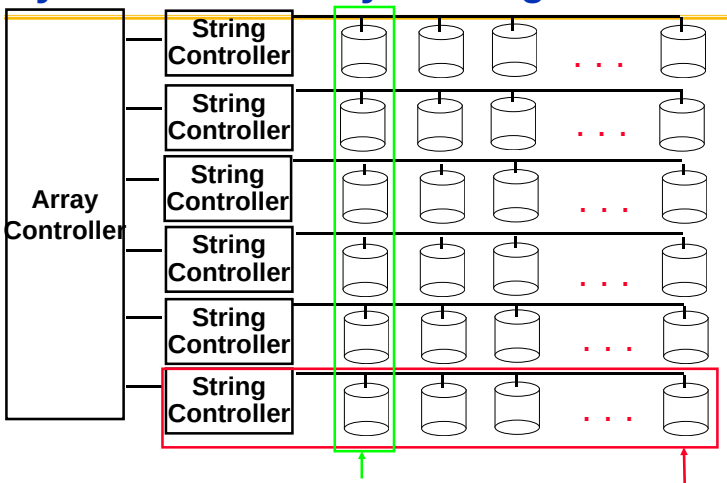
Problems of Disk Arrays: Small Writes

RAID-5: Small Write Algorithm

1 Logical Write = 2 Physical Reads + 2 Physical Writes

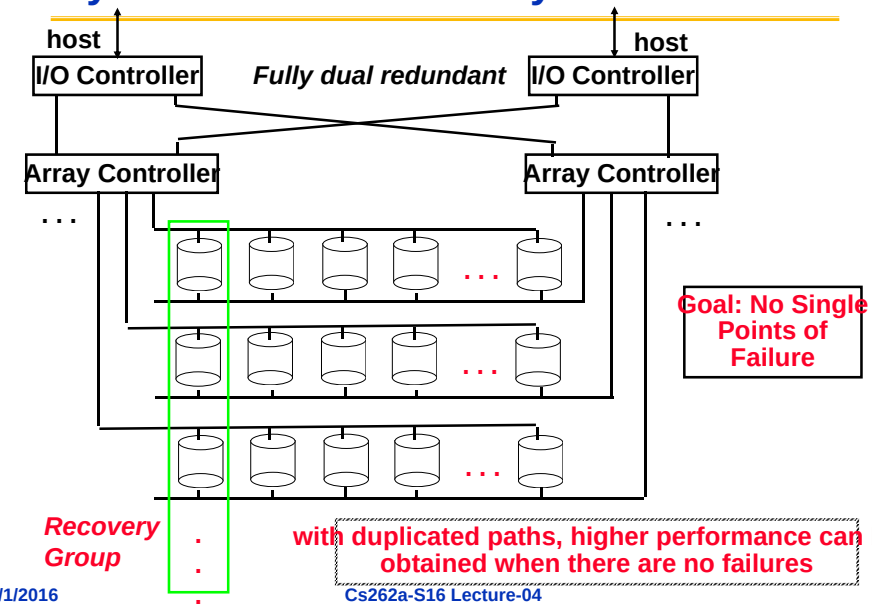


System Availability: Orthogonal RAIDs



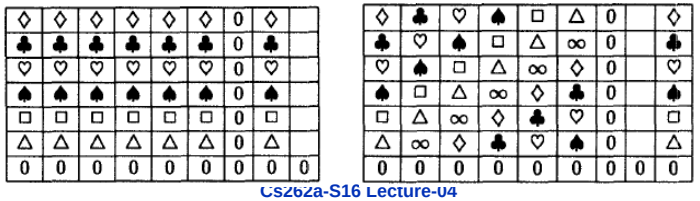
Data Recovery Group: unit of data redundancy
Redundant Support Components: fans, power supplies, controller, cables
End to End Data Integrity: internal parity protected data paths

System-Level Availability



How to get to "RAID 6"?

- **One option: Reed-Solomon codes (Non-systematic):**
  - **Use of Galois Fields (finite element equivalent of real numbers)**
  - **Data as coefficients, code space as values of polynomial:**
  - **$P(x)=a_0+a_1x^1+\ldots a_4x^4$**
  - **Coded: P(1),P(2)….,P(6),P(7)**
  - **Advantage: can add as much redundancy as you like: 5 disks?**
- **Problems with Reed-Solomon codes: decoding gets complex quickly – even to add a second disk**
- **Alternates: lot of them – I've posted one possibility**
  - **Idea: Use prime number of columns, diagonal as well as straight XOR**

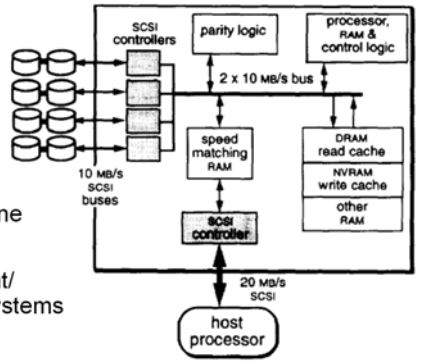


HP AutoRAID – Motivation

- Goals: automate the efficient replication of data in a RAID
  - RAIDs are hard to setup and optimize
  - Mix fast mirroring (2 copies) with slower, more space-efficient parity disks
  - Automate the migration between these two levels
- RAID small-write problem:
  - to overwrite part of a block required 2 reads and 2 writes!
  - read data, read parity, write data, write parity
- Each kind of replication has a narrow range of workloads for which it is best...
  - Mistake ⇒ 1) poor performance, 2) changing layout is expensive and error prone
  - Also difficult to add storage: new disk ⇒ change layout and rearrange data...

HP AutoRAID – Key Ideas

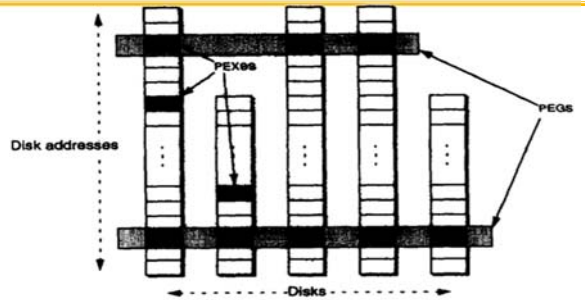
- Key idea: mirror active data (hot), RAID 5 for cold data
  - Assumes only part of data in active use at one time
  - Working set changes slowly (to allow migration)
- How to implement this idea?
  - Sys-admin
    - » make a human move around the files.... BAD. painful and error prone
  - File system
    - » best choice, but hard to implement/ deploy; can't work with existing systems
  - Smart array controller: (magic disk) block-level device interface
    - » Easy to deploy because there is a well-defined abstraction
    - » Enables easy use of NVRAM (why?)



HP AutoRaid – Features

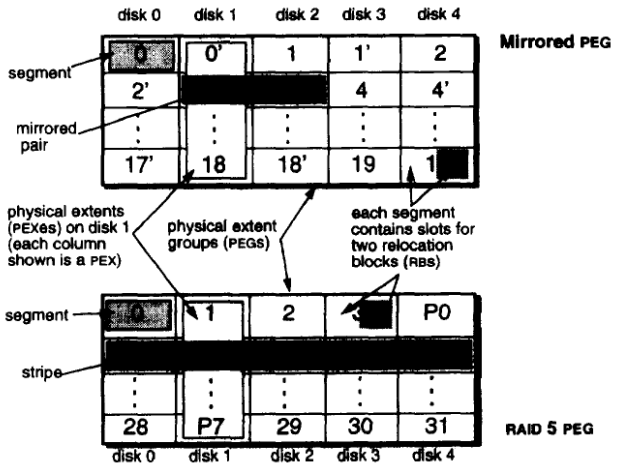
- Block Map
  - level of indirection so that blocks can be moved around among the disks
  - implies you only need one “zero block” (all zeroes), a variation of copy on write
  - in fact could generalize this to have one real block for each unique block
- Mirroring of active blocks
  - RAID 5 for inactive blocks or large sequential writes (why?)
  - Start out fully mirrored, then move to 10% mirrored as disks fill
- Promote/demote in 64K chunks (8-16 blocks)
  - Hot swap disks, etc. (A hot swap is just a controlled failure.)
  - Add storage easily (goes into the mirror pool)
  - useful to allow different size disks (why?)
- No need for an active hot spare (per se);
  - just keep enough working space around
- Log-structured RAID 5 writes
  - Nice big streams, no need to read old parity for partial writes

AutoRAID Details



- **PEX (Physical Extent): 1MB chunk of disk space**
- **PEG (Physical Extent Group): Size depends on # Disks**
  - **A group of PEXes assigned to one storage class**
- **Stripe: Size depends # Disks**
  - **One row of parity and data segments in a RAID 5 storage class**
- **Segment: 128 KB**
  - **Strip unit (RAID 5) or half of a mirroring unit**
- **Relocation Block (RB): 64KB**
  - **Client visible space unit**

Closer Look:



Questions

- When to demote? When there is too much mirrored storage (>10%)
  - Demotion leaves a hole (64KB). What happens to it? Moved to free list and reused
  - Demoted RBs are written to the RAID5 log, one write for data, a second for parity
- Why log RAID5 better than update in place?
  - Update of data requires reading all the old data to recalculate parity.
  - Log ignores old data (which becomes garbage) and writes only new data/parity stripes
- When to promote? When a RAID5 block is written...
  - Just write it to mirrored and the old version becomes garbage.
- How big should an RB be?
  - Bigger ⇒ Less mapping information, fewer seeks
  - Smaller ⇒ fine grained mapping information
- How do you find where an RB is?
  - Convert addresses to (LUN, offset) and then lookup RB in a table from this pair
  - Map size = Number of RBs and must be proportional to size of total storage
- How to handle thrashing (too much active write data)?
  - Automatically revert to directly writing RBs to RAID 5!

Issues

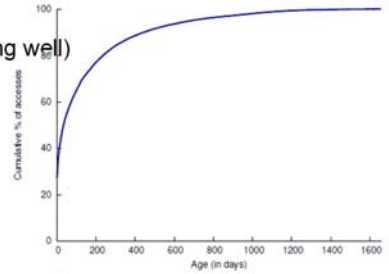
- Disks writes go to two disks (since newly written data is "hot").
  - Must wait for both to complete -- why?
  - Does the host have to wait for both? No, just for NVRAM
- Controller uses cache for reads
- Controller uses NVRAM for fast commit, then moves data to disks
  - What if NVRAM is full? Block until NVRAM flushed to disk, then write to NVRAM
- What happens in the background?
  - 1) compaction, 2) migration, 3) balancing
- Compaction: clean RAID5 and plug holes in the mirrored disks.
  - Do mirrored disks get cleaned? Yes, when a PEG is needed for RAID5; i.e., pick a disks with lots of holes and move its used RBs to other disks. Resulting empty PEG is now usable by RAID5
  - What if there aren't enough holes? Write the excess RBs to RAID5, then reclaim the PEG
- Migration: which RBs to demote? Least-recently-**written (not LRU)**
- Balancing: make sure data evenly spread across the disks. (Most important when you add a new disk)

Is this a good paper?

- What were the authors' goals?
- What about the performance metrics?
- Did they convince you that this was a good system?
- Were there any red-flags?
- What mistakes did they make?
- Does the system meet the "Test of Time" challenge?
- How would you review this paper today?

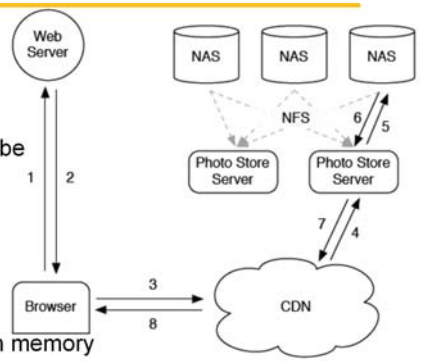
Finding a Needle in Haystack

- This is a *systems level solution:*
  - Takes into account specific application (Photo Sharing)
    - » Large files!, Many files!
    - » 260 Billion images, 20 PetaBytes (10^{15} bytes!)
    - » One billion new photos a week (60 TeraBytes)
    - » Each photo scaled to 4 sizes and replicated (3x)
  - Takes into account environment (Presence of Content Delivery Network, CDN)
    - » High cost for NAS and CDN
  - Takes into account usage patterns:
    - » New photos accessed a lot (caching well)
    - » Old photos accessed little, but likely to be requested at any time ⇒ NEEDLES
- Cumulative graph of accesses as function of age



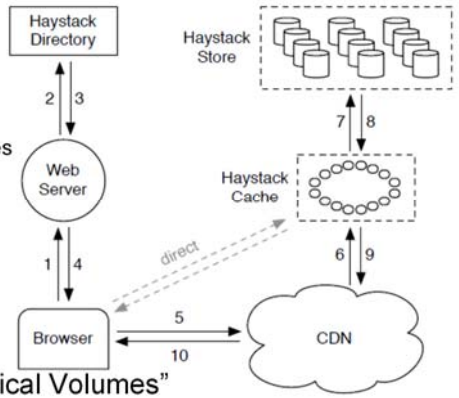
Old Solution: NFS

- Issues with this design?
- Long Tail ⇒ Caching does not work for most photos
  - Every access to back end storage must be *fast* without benefit of caching!
- Linear Directory scheme works badly for many photos/directory
  - Many disk operations to find even a single photo (10 I/Os!)
  - Directory's block map too big to cache in memory
  - "Fixed" by reducing directory size, however still not great (10 ➔ 3 I/Os)
- FFS metadata requires ≥ 3 disk accesses per lookup (dir, inode, pic)
  - Caching all inodes in memory might help, but inodes are big
- Fundamentally, Photo Storage different from other storage:
  - Normal file systems fine for developers, databases, etc.



Solution: Finding a needle (old photo) in Haystack

- Differentiate between old and new photos
  - How? By looking at "Writeable" vs "Read-only" volumes
  - New Photos go to Writeable volumes
- Directory: Help locate photos
  - Name (URL) of photo has embedded volume and photo ID
- Let CDN or Haystack Cache Serve new photos
  - rather than forwarding them to Writeable volumes
- Haystack Store: Multiple "Physical Volumes"
  - Physical volume is large file (100 GB) which stores millions of photos
  - Data Accessed by Volume ID with offset into file
  - Since Physical Volumes are large files, use XFS which is optimized for large files
  - DRAM usage per photo: 40 bytes vs 536 inode
- Cheaper/Faster: ~28% less expensive, ~4x reads/s than NAS



What about these results?

Benchmark	[Config # Operations]	Reads			Writes		
		Throughput (in images/s)	Latency (in ms)		Throughput (in images/s)	Latency (in ms)	
			Avg.	Std. dev.		Avg.	Std. dev.
Random IO	[Only Reads]	902.3	33.2	26.8	–	–	–
Haystress	[A # Only Reads]	770.6	38.9	30.2	–	–	–
Haystress	[B # Only Reads]	877.8	34.2	28.1	–	–	–
Haystress	[C # Only Multi-Writes]	–	–	–	6099.4	4.9	16.0
Haystress	[D # Only Multi-Writes]	–	–	–	7899.7	15.2	15.3
Haystress	[E # Only Multi-Writes]	–	–	–	10843.8	43.9	16.3
Haystress	[F # Reads & Multi-Writes]	718.1	41.6	31.6	232.0	11.9	6.3
Haystress	[G # Reads & Multi-Writes]	692.8	42.8	33.7	440.0	11.9	6.9

- **Workloads:**
  - **A: Random reads to 64KB images – 85% of raw throughput, 17% higher latency**
  - **B: Same as A but 70% of reds are 8KB images**
  - **C, D, E: Write throughput with 1, 4, 16 writes batched (30 and 78% throughput gain)**
  - **F, G: Mixed workloads (98% R/2% MW, 96% R/4% MW of 16 image MW)**
- **Are these good benchmarks? Why or why not?**
- **Are these good results? Why or why not?**

Discussion of Haystack

- Did their design address their goals?
  - Why or why not
- Were they successful?
  - Is this a different question?
- What about the benchmarking?
  - Good performance metrics?
  - Did they convince you that this was a good system?
- Were there any red-flags?
- What mistakes did they make?
- Will this system meet the "Test of Time" challenge?

Is this a good paper?

- What were the authors' goals?
- What about the performance metrics?
- Did they convince you that this was a good system?
- Were there any red-flags?
- What mistakes did they make?
- Does the system meet the "Test of Time" challenge?
- How would you review this paper today?