HP AutoRAID (Lecture 5, cs262a)

Ali Ghodsi and Ion Stoica,
UC Berkeley
January 31, 2018

(based on slide from John Kubiatowicz, UC Berkeley)

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

Redundant Array of Inexpensive Disks (RAID)

Invented at Berkeley 1988:

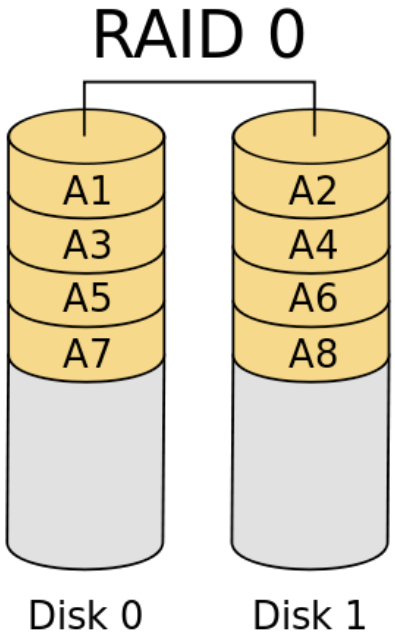
- Dave Patterson
- Garth Gibson (now at CMU,
  CEO of the Vector Institute, Toronto)
- Randy Katz

Patterson, David; Gibson, Garth A.; Katz, Randy (1988). A Case for Redundant Arrays of Inexpensive Disks (RAID), SIGMOD '88

RAID Basics: Levels* of RAID

RAID 0: striping with no parity

- High read/write throughput; theoretically Nx where N is number of disks
- No data redundancy



https://en.wikipedia.org/wiki/Standard_RAID_levels

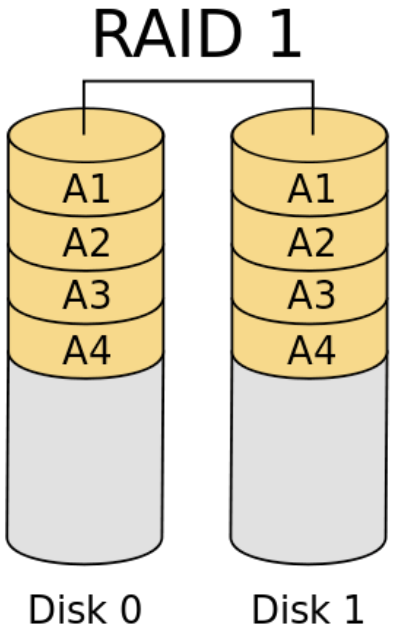
Levels in RED are used in practice

RAID Basics: Levels* of RAID

RAID 0: striping with no parity

RAID 1: Mirroring, simple, fast, but 2x storage

- Each disk is fully duplicated onto its "shadow" → high availability
- Read throughput (2x) assuming N disks
- Writes (1x): two physical writes



https://en.wikipedia.org/wiki/Standard_RAID_levels

Levels in RED are used in practice

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Google*

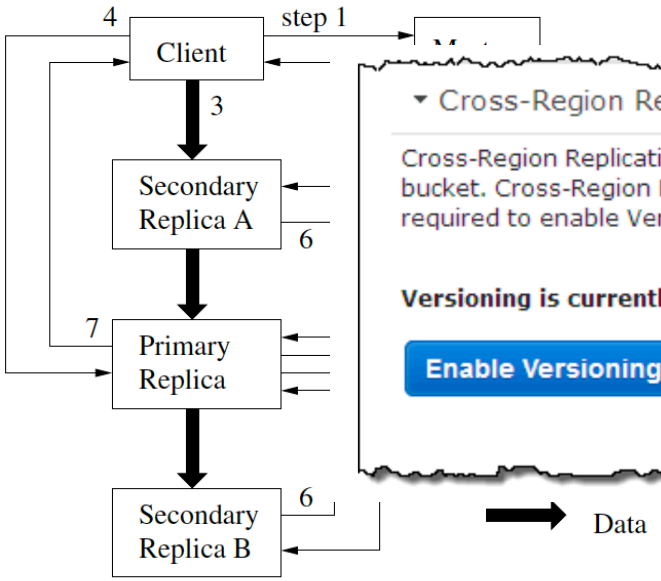


Figure 2: Write Control and Data Flow

Quora Home Answer Notifications Search Quora

Hadoop Distributed File System Apache Hadoop

S?

Cross-Region Replication

Cross-Region Replication replicates every future upload of every object in this bucket to another bucket. Cross-Region Replication is designed for use in conjunction with Versioning. You will be required to enable Versioning on this bucket and the target bucket. Learn More

Versioning is currently not enabled on this bucket.

Enable Versioning

ctures from

11 Answers

Sean Owen, Director, Data Science @ Cloudera

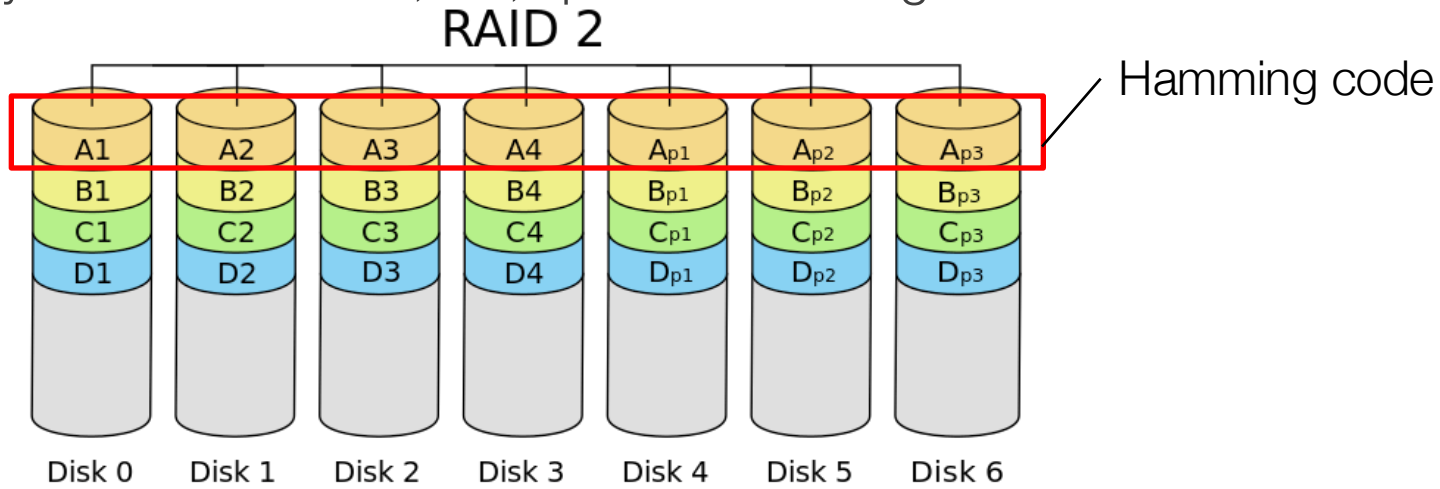
Answered Jan 17, 2014

Yes, it is the default. I don't believe it's magical, although 3x replication is quite

RAID Basics: Levels* of RAID

RAID 2: bit-level interleaving with Hamming error-correcting codes

- d bit Hamming code can detect d-1 errors and correct (d-1)/2
- Need synchronized disks, i.e., spin at same angular rotation



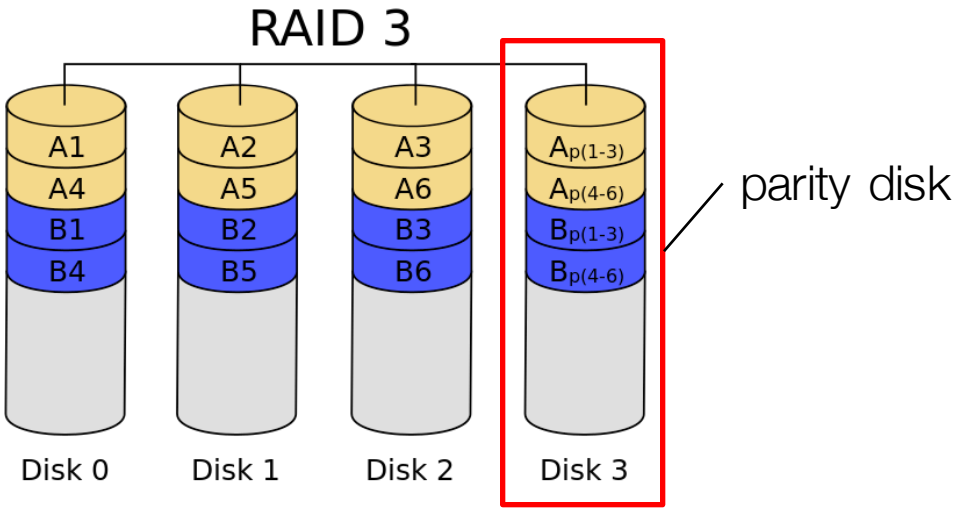
https://en.wikipedia.org/wiki/Standard_RAID_levels

Levels in RED are used in practice

RAID Basics: Levels* of RAID

RAID 3: byte-level striping with dedicated parity disk

- Dedicated parity disk is write bottleneck: every write writes parity



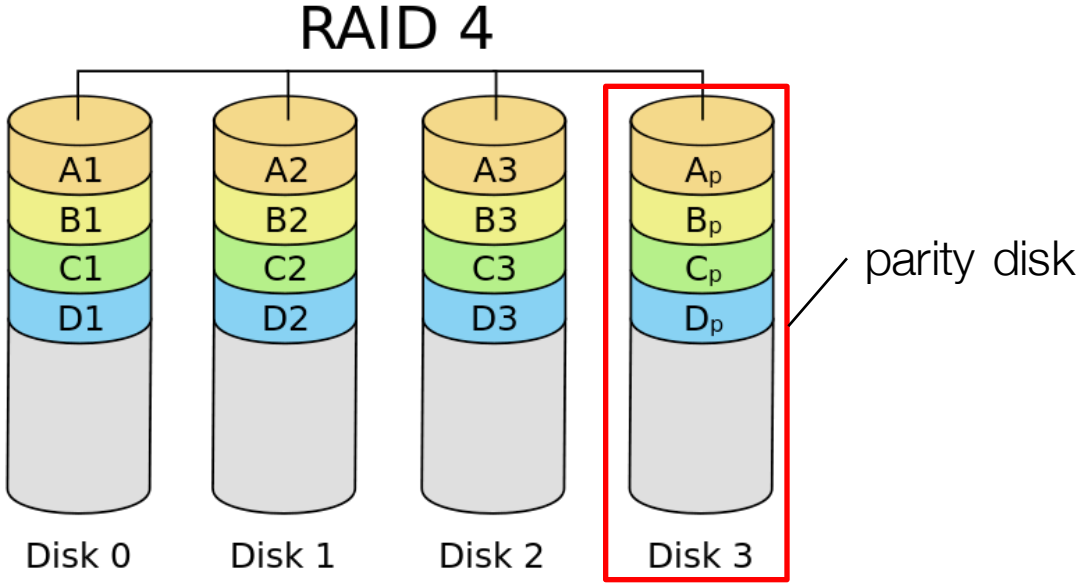
https://en.wikipedia.org/wiki/Standard_RAID_levels

Levels in RED are used in practice

RAID Basics: Levels* of RAID

RAID 4: block-level striping with dedicated parity disk

- Same bottleneck problems as RAID 3



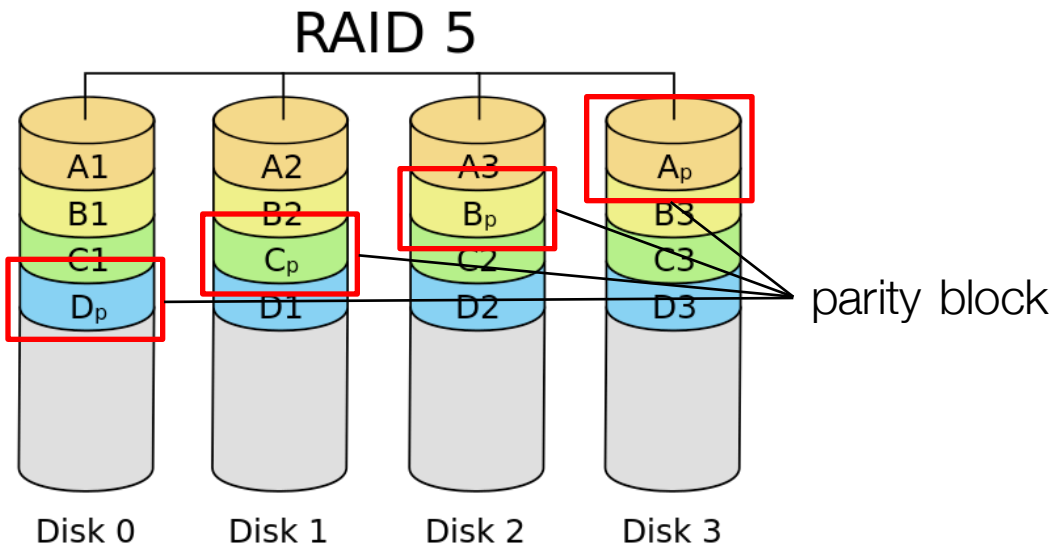
https://en.wikipedia.org/wiki/Standard_RAID_levels

Levels in RED are used in practice

RAID Basics: Levels* of RAID

RAID 5: block-level striping with *rotating* parity disk

- Most popular; spreads out parity load; space 1-1/N, read/write (N-1)x



https://en.wikipedia.org/wiki/Standard_RAID_levels

Levels in RED are used in practice

RAID Basics: Levels* of RAID

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)

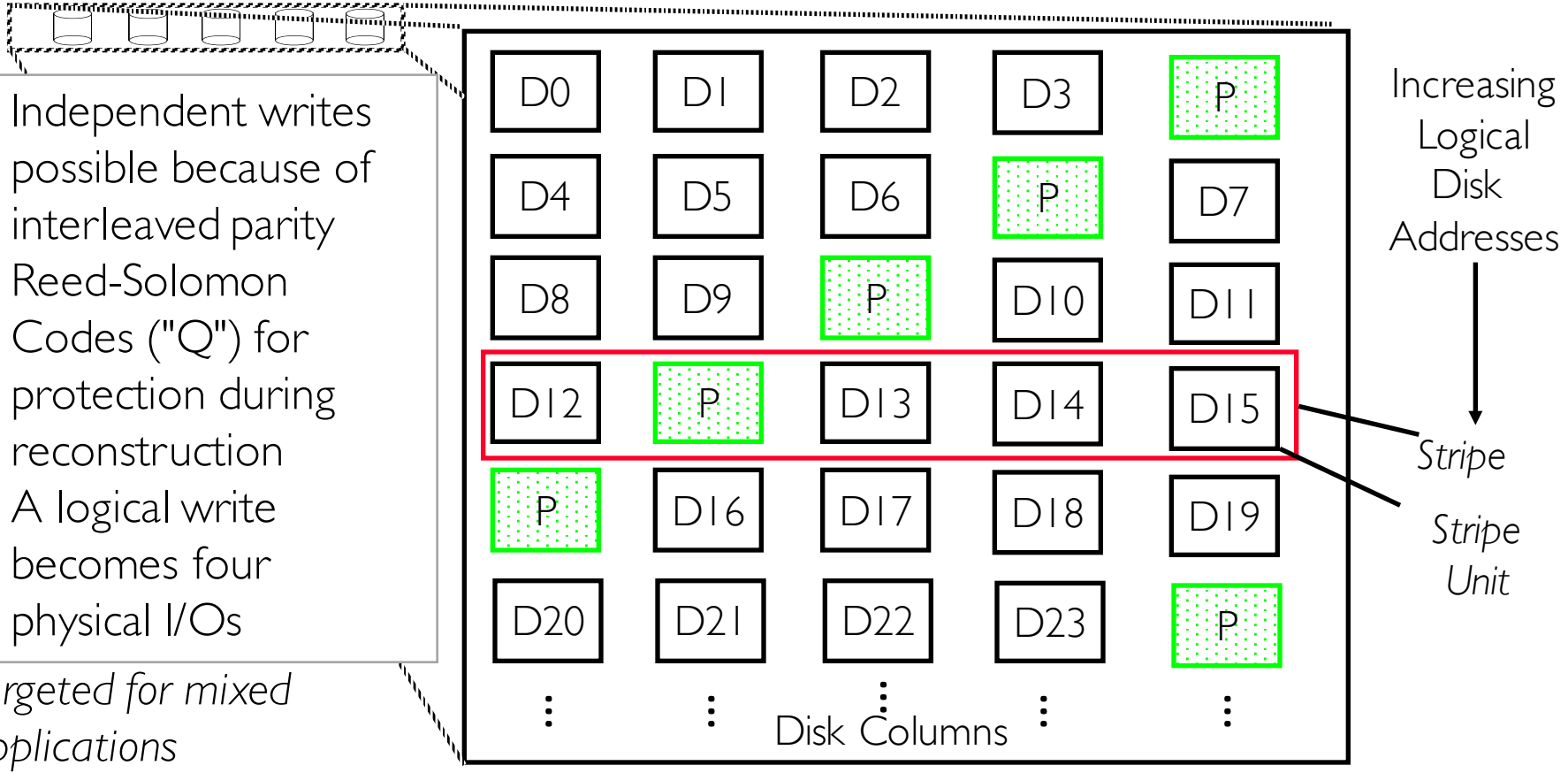
- Use with today's drive sizes! Why?
  - Correlated drive failures (2x expected in 10hr recovery) [Schroeder and Gibson, FAST07]
  - Failures during multi-hour/day rebuild in high-stress environments

Levels in RED are used in practice

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

- Independent writes possible because of interleaved parity
- Reed-Solomon Codes ("Q") for protection during reconstruction
- A logical write becomes four physical I/Os

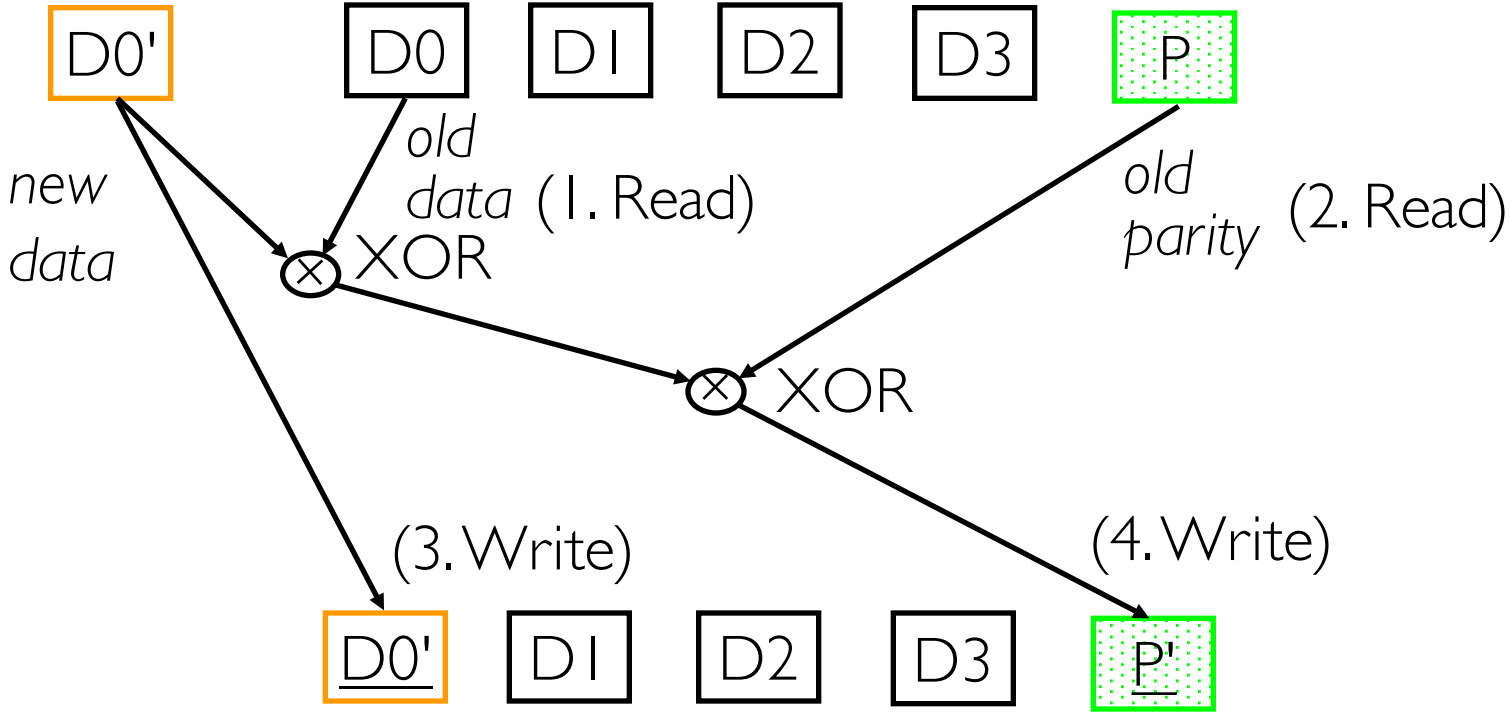
Targeted for mixed applications



Problems of Disk Arrays: Small Writes

RAID-5: Small Write Algorithm

1 Logical Write = 2 Physical Reads + 2 Physical Writes



Introduction to HDFS Erasure Coding in Apache Hadoop

September 23, 2015 | By Zhe Zhang, Andrew Wang, Kai Zheng, Uma Maheswara G., and Vinayakumar B | 23 Comments
Categories: Hadoop HDFS

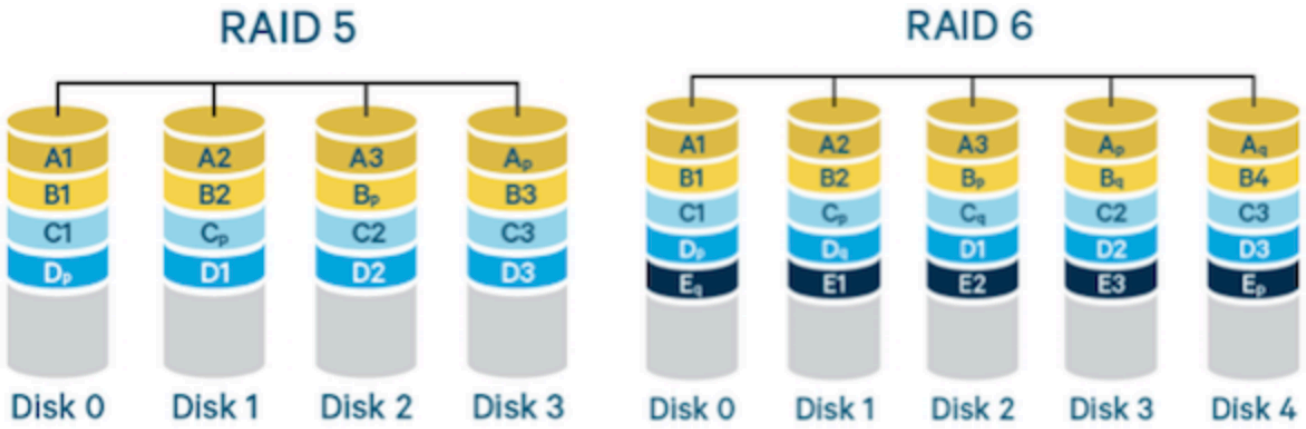
Erasure coding, a new feature in HDFS, can reduce storage overhead by approximately 50% compared to replication while maintaining the same durability guarantees. This post explains how it works.

HDFS by default replicates each block three times. Replication provides a simple and robust form of redundancy to shield against most failure scenarios. It also eases scheduling compute tasks on locally stored data blocks by providing multiple replicas of each block to choose from.

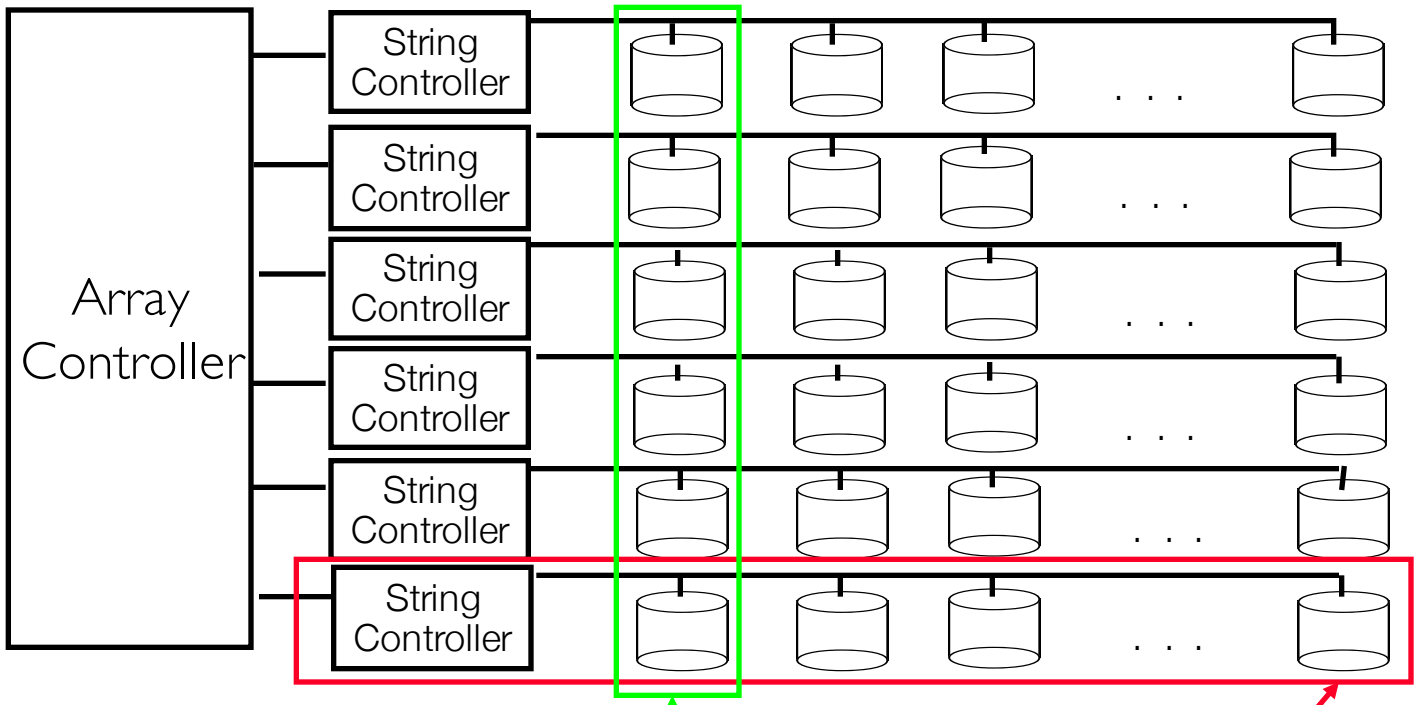
However, replication is expensive: the default 3x replication scheme incurs a 200% overhead in storage space and other resources (e.g., network bandwidth when writing the data). For datasets with relatively low I/O activity, the additional block replicas are rarely accessed during normal operations, but still consume the same amount of storage space.

	Data Durability	Storage Efficiency
Single replica	0	100%
Three-way replication	2	33%
XOR with six data cells	1	86%
RS(6,3)	3	67%
RS(10,4)	4	71%

Table 2: Comparison of replication, XOR, and RS in fault tolerance and storage efficiency

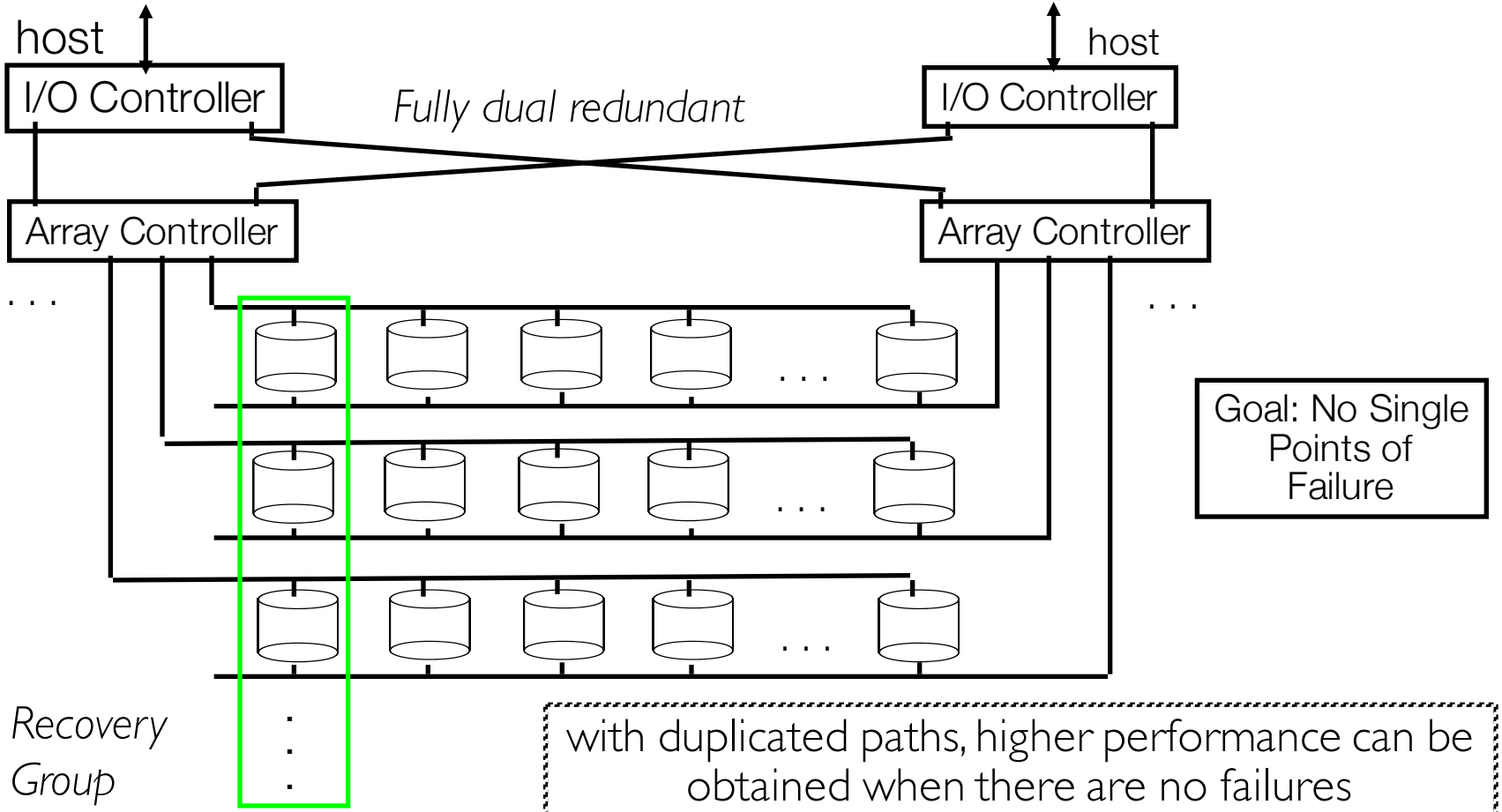


System Availability: Orthogonal RAIDs



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



HP AutoRAID – Motivation

Goals: automate the efficient replication of data in a RAID

- RAIDs are hard to setup and optimize
- Different RAID Levels provide different tradeoffs
- Automate the migration between levels

RAID 1 and 5: what are the tradeoffs?

- RAID 1: excellent read performance, good write performance, performs well under failures, expensive
- RAID 5: cost effective, good read performance, bad write performance, expensive recovery

HP AutoRAID – Motivation

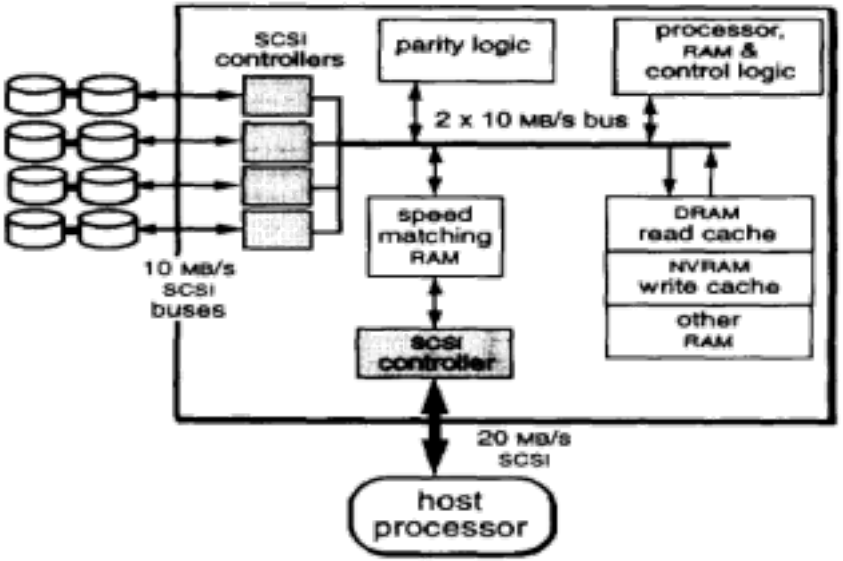
Each kind of replication has a narrow range of workloads for which it is best...

- Mistake ⇒ 1) poor performance, 2) changing layout is expensive and error prone
- Difficult to add storage: new disk ⇒ change layout and rearrange data...
- Difficult to add disks of different sizes

HP AutoRAID – Key Ideas

Key idea: mirror active data (hot), RAID 5 for cold data

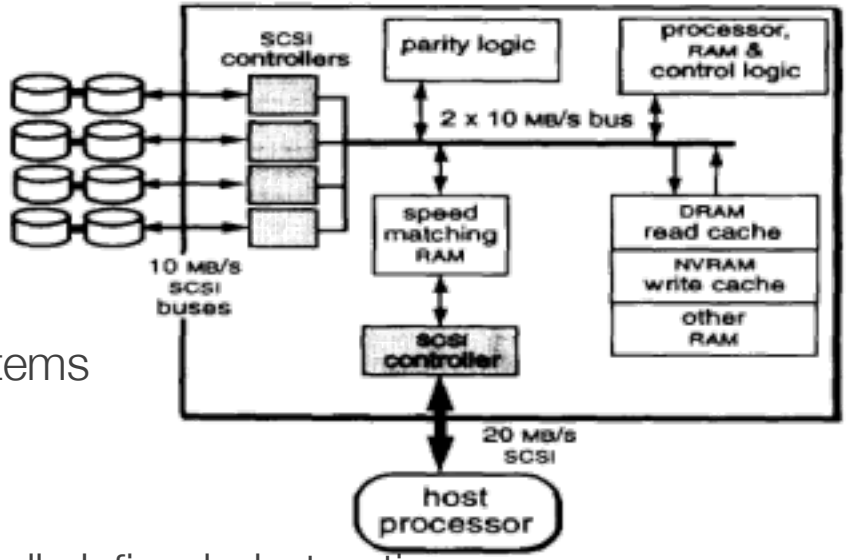
- Assumes only part of data in active use at one time
- Working set changes slowly (to allow migration)



HP AutoRAID – Key Ideas

How to implement this idea?

- Sys-admin
  - make a human move around the files.... BAD. painful and error prone
- File system
  - 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?)



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

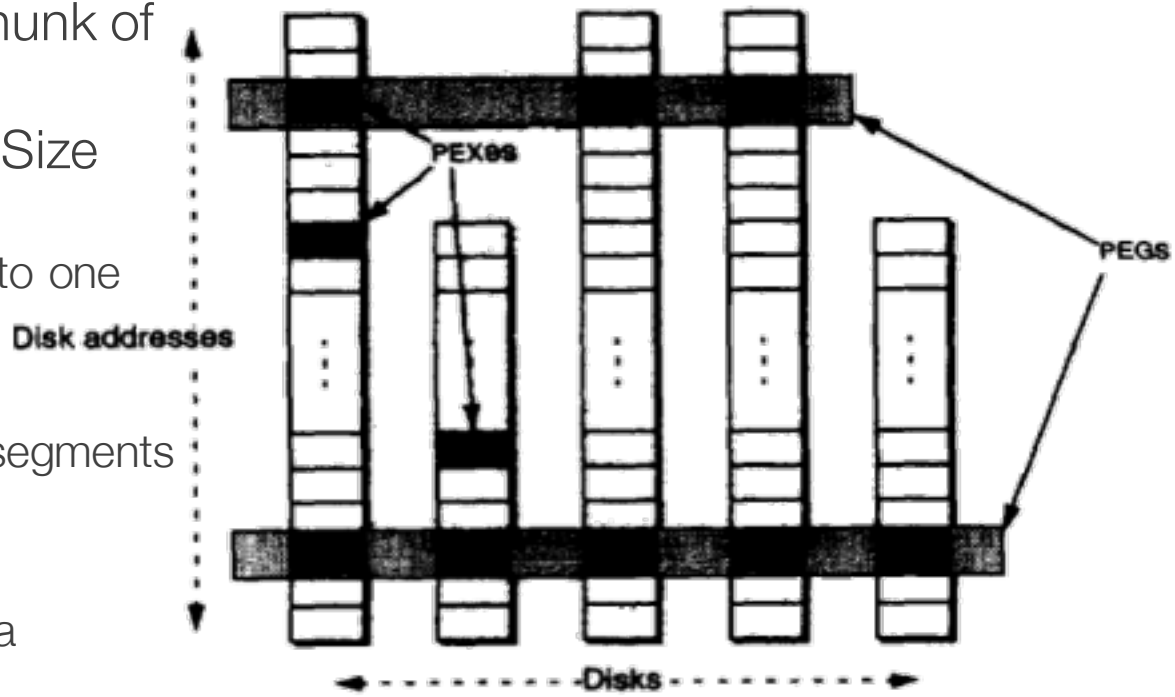
- One row of parity and data segments in a RAID 5 storage class

Segment: 128 KB

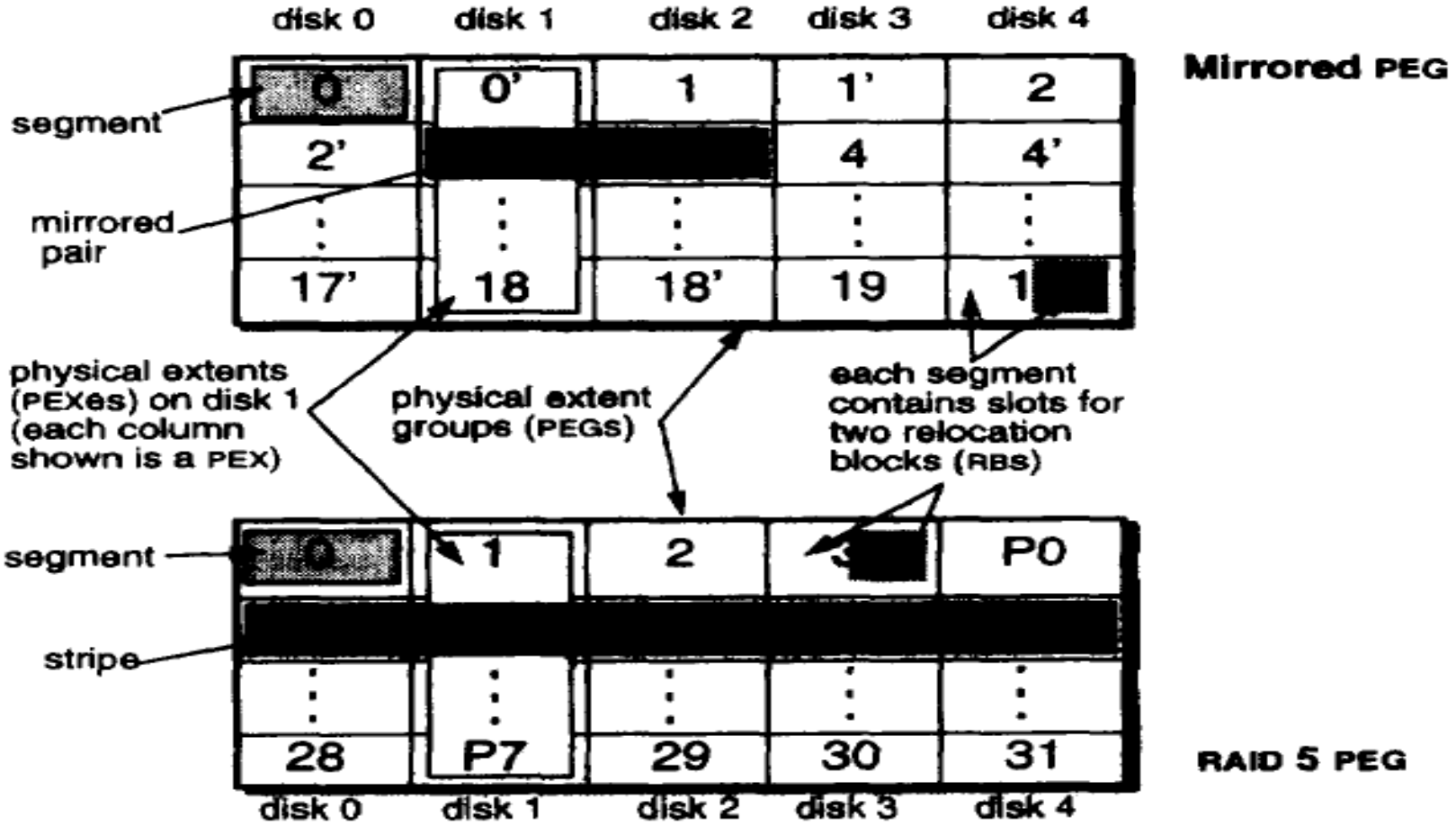
- Strip unit (RAID 5) or half of a mirroring unit

Relocation Block (RB): 64KB

- Client visible space unit



Closer Look


disk 0
disk 1
disk 2
disk 3
disk 4
Mirrored PEG
segment
mirrored pair
0
0'
1
1'
2
2'
4
4'
17'
18
18'
19
1
physical extents (PEXes) on disk 1 (each column shown is a PEX)
physical extent groups (PEGS)
each segment contains slots for two relocation blocks (RBs)
segment
stripe
1
2
P0
28
P7
29
30
31
RAID 5 PEG
disk 0
disk 1
disk 2
disk 3
disk 4

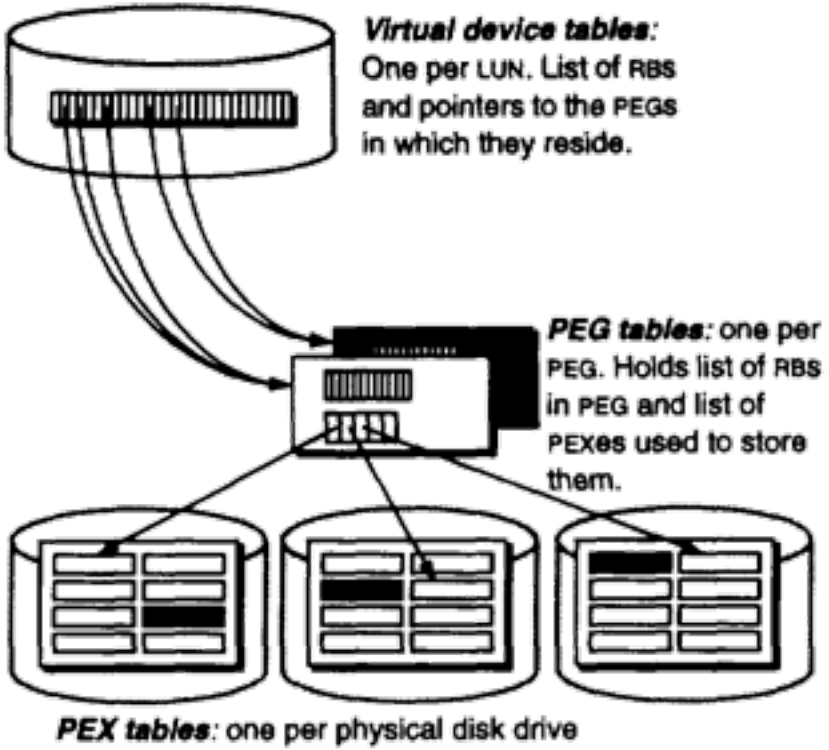
Putting Everything Together

Virtual Device Tables

- Map RBs to PEGs
- RB allocated to a PEG only when written

PEG Tables

- Map PEGs to physical disk addresses
- Unused slots in PEG are marked free until and RB is allocated to them



HP AutoRaid – Features

Promote/demote in 64KB 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

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 reuse
- Demoted RBs are written to the RAID5 log, one write for data, a second for parity

Why log RAID 5 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

How to promote? When a RAID5 block is written...

- Just write it to mirrored and the old version becomes garbage.

Questions

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

Issues

What happens in the background?

- 1) compaction, 2) migration, 3) balancing

Compaction: clean RAID 5 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)