CS 537
Lecture 11
Reliable Storage and RAID

Michael Swift

1

Disk Reliability

- Published failure rate
  - Mean Time To Failure (MTTF) 1,000,000 hours
    - 114 years
- Measured rate in HPC clusters and at Google
  - Annual replacement rate of 2-4%
    - 2-4 out of 100 drives fail
    - MTTF of 25-50 years
- Failures increase linearly with age
  - Little “infant mortality” or death from old age

2

Failure Models

- Fail stop
  - Disk fails and reports failure
- Latent sector error
  - Disk returns corrupt data on read

3

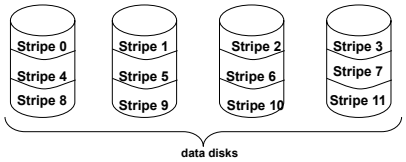
Scaling Challenge

- Disk transfer rates are improving, but much less fast than CPU performance
- How do we:
  - Use multiple disks to improve performance?
  - Use multiple disks to improve capacity?
- Without
  - Sacrificing reliability
- With 2% annual failure rate per disk:
  - With one disk, 98% chance of no failures
  - With two disks: (98%) * (98%) = 96% chance of no failure
  - With 10 disks: $(98\%)^{10}$ = 81% chance of no failure

4

Striping

- Files are *striped* across disks, no redundant info
- Benefits:
  - High read throughput
  - High write throughput Drawbacks:
  - Any disk failure results in data loss
    - 100 disks have 1/100th the MTTF (mean time to failure) of one disk
  - Reliability worse than one big disk



The challenge

- We need striping for performance, but we need something to help with reliability / availability
- To improve reliability, we can add **redundant** data to the disks, in addition to striping
  - Extra copies of data
  - Error-correct codes
- Evaluation criteria:
  - **Capacity**: how much space is wasted with redundancy
  - **Efficiency**: how much time wasted reading/writing redundant data instead of real data

Workloads

- Sequential access: read/write long sequence of consecutive blocks
  - Example: playing a video
  - Time spent: rotating disk to read data and **transfer** to computer
- Random access: read/write a sequence of random blocks
  - Example:
    - database updates
    - mixing sequential access from multiple programs
    - email client
  - Time spent:
    - **Seeking** to track on disk
    - **Rotating** to desired sector

RAID

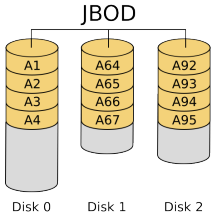
- A RAID is a Redundant Array of Inexpensive Disks
- Disks are small and cheap, so it's easy to put lots of disks (10s to 100s) in one box for increased storage, performance, and availability
- Data plus some redundant information is striped across the disks in some way
- How striping is done is key to performance and reliability

Some RAID tradeoffs

- Granularity
  - fine-grained: stripe each file over all disks
    - high throughput for the file
    - limits transfer to 1 file at a time
  - course-grained: stripe each file over only a few disks
    - limits throughput for 1 file
    - allows concurrent access to multiple files
- Redundancy
  - uniformly distribute redundancy information on disks
    - avoids load-balancing problems
  - concentrate redundancy information on a small number of disks
    - partition the disks into data disks and redundancy disks

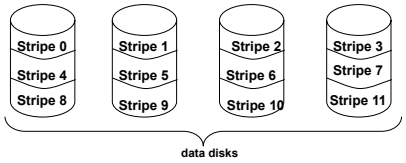
Approach -1: just a bunch of disks (JBOD)

- Combine all the disks into one big disk
  - Add a *pseudo-driver* that converts LBNs in one big address space into LBNs on different disks
- Benefits:
  - Easy, cheap
- Drawbacks:
  - No redundancy for error recovery
  - Sequential blocks are on one disk, so less perf. benefit of multiple disks
  - Disk failure may cause loss of many files



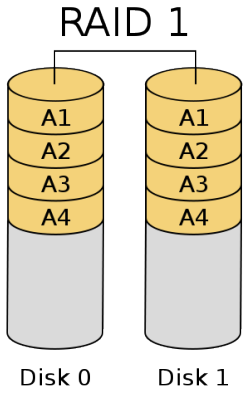
Raid Level 0

- Level 0 is non-redundant disk array
- Files are *striped* across disks, no redundant info
- Benefits:
  - High read throughput
  - Best write throughput (no redundant info to write)
- Drawbacks:
  - Any disk failure results in data loss
  - Reliability worse than one big disk



RAID Level 1

- Mirrored Disks
  - Data is written to two places (data + mirror)
  - On failure, just use surviving disk
  - Combine with JBOD or RAID 0
- On read, choose fastest to read
  - Write performance is same as single drive, read performance is 2x better
- Expensive – 2x space expansion



RAID 1 performance

- Sequential and Random reads:
  - Bandwidth of 2 disks = B*2
    - If can arrange sequential read to hit both disks without skipping blocks
- Sequential Writes:
  - Write to both drives
  - Bandwidth = B
- Random writes:
  - Write to both drives
  - Bandwidth = B

13

Calculating failure rates

- Suppose a disk has a mean-time-to-failure of 3 years (randomly distributed)
  - Failures per year = 1/3
- Stripe data on two disks – both must be available
  - Failures per year = 2* 1/3 = 2/3 -> MTTF = 1.5 years
- Suppose we store the same data on two disks
  - Observe: system works if *either* disk works
    - Assume failures last 1 day
  - Failures per year = 1/3 * probability of second disk failing at the same time
    - Probability = 1 day / 3 years
    - Failures per year = 1/3 * 1/1000 = 1/3000 -> MTTF = 3000 years

14

RAID 4

- RAID levels 4 uses parity disks
  - each byte on the parity disk is a parity function of the corresponding bytes on all the other disks
- Parity Refresher:
  - To each byte, add a bit set so that the total number of 1's is even
    - P = XOR(d[0],d[1],d[2],d[3])
  - Error detection: mismatch parity signals an error
  - Error correction: any single missing bit can be reconstructed

1	0	1	1	0	1	1	0	1

3/12/13

15

Recovering Data with parity

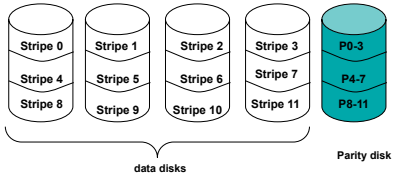
1	0	1	X	0	1	1	0	1

- What is X?
  - Parity of remaining bits = 0
  - Stored parity is 1
  - Missing bit must be 1 = XOR(remaining bits)

16

Raid Level 4

- Block-level parity with stripes
  - Save space of full mirroring
- A read accesses all the data disks – high performance
- A write accesses all data disks plus the parity disk – write speed of single disk (for small writes)

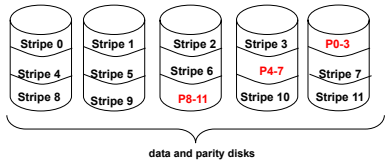


RAID 4 performance

- Sequential and Random reads:
  - Bandwidth of N-1 disks = B*(N-1)
- Sequential Writes:
  - Compute new parity, write to all N disks,
  - Only N-1 get data: Bandwidth = B*(N-1)
- Random writes:
  - All writes must update parity disk
  - Subtractive parity = remove parity of old data
    - Pnew = (D_{old} XOR D_{new}) XOR P_{old}
    - Bandwidth = B/(1 read + 1 write) = B/2
- Single parity disk is a bottleneck

Raid Level 5

- Block Interleaved Distributed Parity
- Like parity scheme, but distribute the parity info over all disks (as well as data over all disks)
  - for each stripe, one disk holds the parity, and the other disks hold the data
- Better read performance, large write performance



RAID 5 performance

- Sequential and Random reads:
  - Bandwidth of N-1 disks = B*(N-1)
- Sequential Writes:
  - Compute new parity, write to all N disks,
  - Only N-1 get data: Bandwidth = B*(N-1)
- Random writes:
  - All writes must update **A** parity disk
  - Can do concurrent updates to different stripes
  - Bandwidth = B*(N/4) (2 reads/ 2 writes)
- Single parity disk is a bottleneck

RAID 5 issues

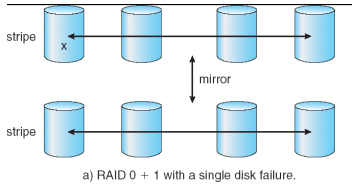
- Small writes (one stripe) :
  - read in the old data block + parity block, then write new data + recomputed parity?
    - Leads to waiting to read data
- Large writes:
  - Compute new parity
  - Write out all blocks + parity block

Raid Level 6

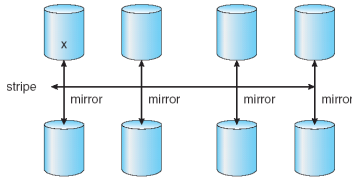
- Level 5 with an extra parity bit
- Can tolerate two failures
  - What are the odds of having two concurrent failures ?
- May outperform Level-5 on reads (more disks), slower on writes (more disks to write 2 parity blocks)

RAID 0+1 and 1+0

- 0+1: strip disks, then mirror the strips
  - Single failure knocks out a stripe
- 1+0: mirror disks, then make stripes
  - Works if one of each mirror pair survives



a) RAID 0 + 1 with a single disk failure.



b) RAID 1 + 0 with a single disk failure.

RAID implementation

- In software, as a layer between the file system and the block device drivers
  - Accepts block request from a large, combined LBN address space, and decides which disks to access and what data to write
  - Benefits: cheap, flexible
  - Drawbacks: CPU overhead for parity, may not support hot-replace of disks
- In hardware, as a disk controller talking to individual disks
  - Benefits: fast, reliable (can have batteries, hot-swap disks)
  - Drawbacks: expensive, inflexible