Redundant Array of Inexpensive/Independent Disks
RAID

Emin Gun Sirer

Motivation

- **Disks are improving, but not as fast as CPUs**
  - 1970s seek time: 50-100 ms.
  - 2000s seek time: <5 ms.
  - Factor of 20 improvement in 3 decades
- **We can use multiple disks for improving performance**
  - By <u>striping</u> files across multiple disks (placing parts of each file on a different disk), we can use parallel I/O to improve access time
- **Striping reduces reliability -- 100 disks have 1/100th the MTBF (mean time between failures) of one disk**
- **So, 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**

Raid

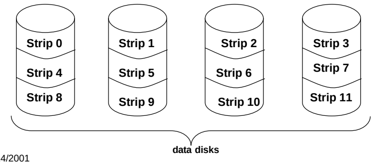
- **A RAID is a Redundant Array of Inexpensive Disks**
  - In industry, “I” is for “Independent”
  - The alternative is SLED, single large expensive disk
- **Disks are small and cheap, so it’s easy to put lots of disks (10s to 100s) in one box for increased storage, performance, and availability**
- **The RAID box with a RAID controller looks just like a SLED to the computer**
- **Data plus some redundant information is striped across the disks in some way**
- **How that striping is done is key to performance and reliability.**

Some Raid Issues

- **Granularity**
  - fine-grained: stripe each file over all disks. This gives high throughput for the file, but limits to transfer of 1 file at a time
  - course-grained: stripe each file over only a few disks. This limits throughput for 1 file but allows more parallel file access
- **Redundancy**
  - uniformly distribute redundancy info on disks: avoids load-balancing problems
  - concentrate redundancy info on a small number of disks: partition the set into data disks and redundant disks

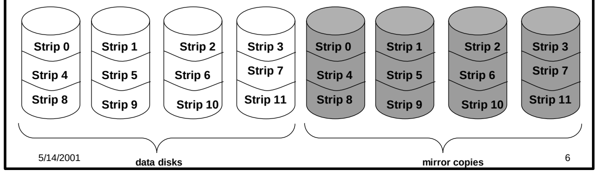
Raid Level 0

- **Level 0 is <u>nonredundant</u> disk array**
- **Files are striped across disks, no redundant info**
- **High read throughput**
- **Best write throughput (no redundant info to write)**
- **Any disk failure results in data loss**
  - Reliability worse than SLED



Raid Level 1

- **Mirrored Disks**
- **Data is written to two places**
  - On failure, just use surviving disk
- **On read, choose fastest to read**
  - Write performance is same as single drive, read performance is 2x better
- **Expensive**



Parity and Hamming Code

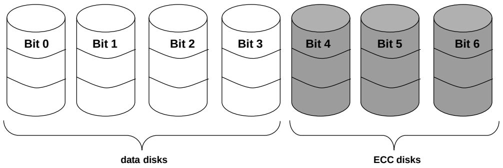
- **What do you need to do in order to detect and correct a one-bit error ?**
  - **Suppose you have a binary number, represented as a collection of bits: <b3, b2, b1, b0>, e.g. 0110**
- **Detection is easy**
- **Parity:**
  - **Count the number of bits that are on, see if it's odd or even**
    - **EVEN parity is 0 if the number of 1 bits is even**
  - **Parity(<b3, b2, b1, b0 >) = P0 = b0 ? b1 ? b2 ? b3**
  - **Parity(<b3, b2, b1, b0,p0>) = 0 if all bits are intact**
  - **Parity(0110) = 0, Parity(01100) = 0**
  - **Parity(11100) = 1 => ERROR!**
  - **Parity can detect a single error, but can't tell you which of the bits got flipped**

Parity and Hamming Code

- **Detection and correction require more work**
- **Hamming codes can detect double bit errors and detect & correct single bit errors**
- **7/4 Hamming Code**
  - **h0 = b0 ? b1 ? b3**
  - **h1 = b0 ? b2 ? b3**
  - **h2 = b1 ? b2 ? b3**
  - **H0(<1101>) = 0**
  - **H1(<1101>) = 1**
  - **H2(<1101>) = 0**
  - **Hamming(<1101>) = <b3, b2, b1, h2, b0, h1, h0> = <1100110>**
  - **If a bit is flipped, e.g. <1110110>**
  - **Hamming(<1111>) = <h2, h1, h0> = <111> compared to <010>, <101> are in error. Error occurred in bit 5.**

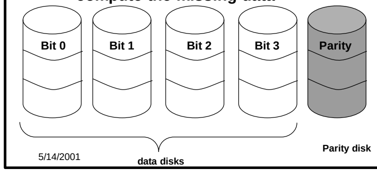
Raid Level 2

- Bit-level striping with Hamming (ECC) codes for error correction
- All 7 disk arms are synchronized and move in unison
- Complicated controller
- Single access at a time
- Tolerates only one error, but with no performance degradation



Raid Level 3

- Use a parity disk
  - Each bit on the parity disk is a parity function of the corresponding bits on all the other disks
- A read accesses all the data disks
- A write accesses all data disks <u>plus</u> the parity disk
- On disk failure, read remaining disks plus parity disk to compute the missing data

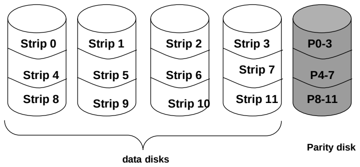


Single parity disk can be used to detect and <u>correct</u> errors

---

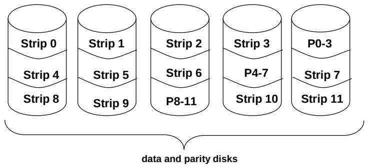
Raid Level 4

- Combines Level 0 and 3 – byte-level parity with stripes
- A read accesses all the data disks
- A write accesses all data disks <u>plus</u> the parity disk
- Heavy load on the parity disk



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)
- Better read performance, large write performance
  - Reads can outperform SLEDs and RAID-0



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, slower on writes**