Project Announcement

- Grading:
  - Part A: 3/4 of total grade
  - Part B: 1/4 of total grade

- Actual grade formula:
  \[ \frac{3}{4} \times \max(A_1, 0.7 \times A_2) + \frac{1}{4} \times \max(B_{\text{self}}, 0.9 \times B_{\text{with staff code}}) \]

  - A1 is submission for P1a on or before the part A deadline.
  - A2 is version of P1a submitted "any time" before the part B deadline.
  - B_{self} is part B running against your own implementation of LSP
  - 0.9*B_{with staff code} is part B running against the reference LSP

Replacement Rates

<table>
<thead>
<tr>
<th>Component</th>
<th>Component %</th>
<th>Component</th>
<th>Component %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>%</td>
<td>Component</td>
<td>%</td>
</tr>
<tr>
<td>Hard drive</td>
<td>30.6</td>
<td>Power supply</td>
<td>34.8</td>
</tr>
<tr>
<td>Memory</td>
<td>28.5</td>
<td>Hard drive</td>
<td>49.1</td>
</tr>
<tr>
<td>Misc/Unk</td>
<td>14.4</td>
<td>Power supply</td>
<td>10.1</td>
</tr>
<tr>
<td>CPU</td>
<td>12.4</td>
<td>Case</td>
<td>11.4</td>
</tr>
<tr>
<td>Motherboard</td>
<td>4.9</td>
<td>RAID card</td>
<td>10.1</td>
</tr>
<tr>
<td>Fan</td>
<td>3.4</td>
<td>Memory</td>
<td>11.4</td>
</tr>
<tr>
<td>Fan</td>
<td>3.4</td>
<td>Memory</td>
<td>11.4</td>
</tr>
<tr>
<td>Controller</td>
<td>2.9</td>
<td>CPU</td>
<td>2.9</td>
</tr>
<tr>
<td>Controller</td>
<td>2.9</td>
<td>Power supply</td>
<td>2.9</td>
</tr>
<tr>
<td>NIC Card</td>
<td>1.2</td>
<td>CPU</td>
<td>2.2</td>
</tr>
<tr>
<td>MLB</td>
<td>1.2</td>
<td>NIC Card</td>
<td>1.2</td>
</tr>
<tr>
<td>LV Pwr Board</td>
<td>0.6</td>
<td>CD-ROM</td>
<td>0.6</td>
</tr>
<tr>
<td>CD-ROM</td>
<td>0.6</td>
<td>CPU heatsink</td>
<td>0.6</td>
</tr>
<tr>
<td>RAID controller</td>
<td>0.6</td>
<td>RAID controller</td>
<td>0.6</td>
</tr>
</tbody>
</table>

Motivation: Why use multiple disks?

- Capacity
  - More disks allows us to store more data
- Performance
  - Access multiple disks in parallel
  - Each disk can be working on independent read or write
  - Overlap seek and rotational positioning time for all
- Reliability
  - Recover from disk (or single sector) failures
  - Will need to store multiple copies of data to recover
- So, what is the simplest arrangement?

Outline

- Using multiple disks
  - Why have multiple disks?
  - Problem and approaches
- RAID levels and performance
- Estimating availability

Just a bunch of disks (JBOD)

- Yes, it's a goofy name
  - Industry really does sell "JBOD enclosures"
Disk Subsystem Load Balancing

- I/O requests are almost never evenly distributed
  - Some data is requested more than other data
  - Depends on the apps, usage, time, ...
- What is the right data-to-disk assignment policy?
  - Common approach: Fixed data placement
    - Your data is on disk X, period!
    - For good reasons too: you bought it or you’re paying more...
  - Fancy: Dynamic data placement
    - If some of your files are accessed a lot, the admin (or even system) may separate the “hot” files across multiple disks
    - In this scenario, entire file systems or even files are manually moved by the system admin to specific disks
  - Alternative: Disk striping
    - Stripe all of the data across all of the disks

Disk Striping

- Interleave data across multiple disks
  - Large file streaming can enjoy parallel transfers
  - High throughput requests can enjoy thorough load balancing
  - If blocks of hot files equally likely on all disks (really?)

File Foo:
- Stripe unit or block

Now, What If A Disk Fails?

- In a JBOD (independent disk) system
  - one or more file systems lost
- In a striped system
  - a part of each file system lost
- Backups can help, but
  - backing up takes time and effort
  - backup doesn’t help recover data lost during that day
  - Any data loss is a big deal to a bank or stock exchange

Tolerating and Masking Disk Failures

- If a disk fails, it’s data is gone
  - may be recoverable, but may not be
- To keep operating in face of failure
  - must have some kind of data redundancy
- Common forms of data redundancy
  - replication
  - erasure-correcting codes
  - error-correcting codes

Redundancy via Replicas

- Two (or more) copies
  - mirroring, shadowing, duplexing, etc.
- Write both, read either
Mirroring & Striping

- Mirror to 2 virtual drives, where each virtual drive is really a set of striped drives
  - Provides reliability of mirroring
  - Provides striping for performance (with write update costs)

Implementing Disk Mirroring

- Mirroring can be done in either software or hardware
- Software solutions are available in most OS's
  - Windows2000, Linux, Solaris
- Hardware solutions
  - Could be done in Host Bus Adaptor(s)
  - Could be done in Disk Array Controller

Hardware vs. Software RAID

- Hardware RAID
  - Storage box you attach to computer
  - Same interface as single disk, but internally much more
    - Multiple disks
    - More complex controller
    - NVRAM (holding parity blocks)
- Software RAID
  - OS (device driver layer) treats multiple disks like a single disk
  - Software does all extra work
  - Interface for both
    - Linear array of bytes, just like a single disk (but larger)

Lower Cost Data Redundancy

- Single failure protecting codes
  - General single-error-correcting code is overkill
  - General code finds error and fixes it
- Disk failures are self-identifying (a.k.a. erasures)
  - Don’t have to find the error
- Fact: N-error-detecting code is also N-erasure-correcting
  - Error-detecting codes can’t find an error just know it’s there
  - But if you independently know where error is, allows repair
  - Parity is single-disk-failure-correcting code
    - recall that parity is computed via XOR
    - it’s like the low bit of the sum

Simplest approach: Parity Disk

- One extra disk
- All writes update parity disk
  - Potential bottleneck

Updating and using the parity

Fault-Free Read

Fault-Free Write

Degraded Read

Degraded Write
The parity disk bottleneck

- Reads go only to the data disks
  - But, hopefully load balanced across the disks
- All writes go to the parity disk
  - And, worse, usually result in Read-Modify-Write sequence
  - So, parity disk can easily be a bottleneck

Solution: Striping the Parity

- Removes parity disk bottleneck

Outline

- Using multiple disks
  - Why have multiple disks?
    - problem and approaches
- RAID levels and performance
- Estimating availability

RAID Taxonomy

- Redundant Array of Inexpensive Independent Disks
  - RAID 0 – Coarse-grained Striping with no redundancy
  - RAID 1 – Mirroring of independent disks
  - RAID 2 – Fine-grained data striping plus Hamming code disks
    - Uses Hamming codes to detect and correct multiple errors
    - Originally implemented when drives didn’t always detect errors
    - Not used in real systems
  - RAID 3 – Fine-grained data striping plus parity disk
  - RAID 4 – Coarse-grained data striping plus parity disk
  - RAID 5 – Coarse-grained data striping plus striped parity
  - RAID 6 – Coarse-grained data striping plus 2 striped codes

RAID-0: Striping

- Stripe blocks across disks in a “chunk” size
- How to pick a reasonable chunk size?

  How to calculate where chunk # lives?
  Disk:
  Offset within disk:
RAID Comparison: A Summary

We now summarize our simplified comparison of RAID levels in Table 38.7:

<table>
<thead>
<tr>
<th>RAID</th>
<th>Capacity</th>
<th>Reliability</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAID-0</td>
<td>N</td>
<td>0</td>
<td>D</td>
</tr>
<tr>
<td>RAID-1</td>
<td>N/2</td>
<td>1 (for sure)</td>
<td>D</td>
</tr>
<tr>
<td>RAID-4</td>
<td>N-1</td>
<td>1</td>
<td>D</td>
</tr>
<tr>
<td>RAID-5</td>
<td>N-1</td>
<td>1</td>
<td>D</td>
</tr>
</tbody>
</table>

- **RAID-0**: Mostly used as it is slightly simpler to build.
- **RAID-1 (Mirroring)**: Motivation: Handle disk failures. Put copy (mirror or replica) of each chunk on another disk.
  - Capacity: $N$ disks are used.
  - Reliability: Handle disk failures.
  - Performance: Write is the bottleneck. Also, when updating the parity block, the second write of the parity will only result in a seek time.

- **RAID-4 (Parity)**: Motivation: Improve capacity.
  - Idea: Allocate parity block to encode info about blocks.
  - Parity checks all other blocks in stripe across other disks.
  - Parity block = XOR over others (gives "even" parity).
  - Example: $0 \oplus 0 \oplus \text{parity value} = 0$.
  - How do you recover from a failed disk?
    - Example: $x \text{ update } P0$ and parity of 1
    - What is the failed value?

- **RAID-5 (Rotated Parity)**: RAID-5 is the winner; the cost you pay is in small-write performance. To increase reliability, striping is obviously best. If, however, you want to protect against one disk failure, RAID-5 is the winner.
  - Parity disk is the bottleneck.
  - Two different approaches:
    - Small number of disks (or large write): RAID-0.
    - Large number of disks (or small write): RAID-4.
  - Rotation location of parity across all disks.

- **Advanced Issues**:
  - What happens if more than one fault?
    - Example: One disk fails plus "latent sector error" on another.
    - RAID-5 cannot handle two faults.
    - Solution: RAID-6 (e.g., RDP) Add multiple parity blocks.
  - Why is NVRAM useful?
    - Example: What if update 2, don’t update P0 before power failure (or crash), and then disk 1 fails?
    - NVRAM solution: Use to store blocks updated in same stripe.
      - If power failure, can replay all writes in NVRAM.
    - Software RAID solution: Perform parity scrub over entire disk.

---

Table 38.7: RAID Capacity, Reliability, and Performance
RAID 6

- P+Q Redundancy
  - Protects against multiple failures using Reed-Solomon codes
  - Uses 2 “parity” disks
    - P is parity
    - Q is a second code
    - It’s two equations with two unknowns, just make “biggerbits”
      - Group bits into “nibbles” and add different coefficients to each equation (two independent equations in two unknowns)
  - Similar to parity striping
    - De-clusters both sets of parity across all drives
    - For small writes, requires 6 I/Os
      - Read old data, old parity1, old parity2
      - Write new data, new parity1, new parity2

The Disk Array Matrix

<table>
<thead>
<tr>
<th></th>
<th>Independent</th>
<th>Fine Striping</th>
<th>Course Striping</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replication</td>
<td>Mirroring</td>
<td>RAID0</td>
<td>RAID0+1</td>
</tr>
<tr>
<td>Parity Disk</td>
<td>RAID3</td>
<td>RAID4</td>
<td>RAID5</td>
</tr>
<tr>
<td>Striped Parity</td>
<td>Gray90</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Outline

- Using multiple disks
  - Why have multiple disks?
    - problem and approaches
- RAID levels and performance
- Estimating availability

Sidebar: Availability metric

- Fraction of time that server is able to handle requests
  - Computed from MTBF and MTTR (Mean Time To Repair)

\[
\text{Availability} = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}
\]

How often are failures?

- \( \text{MTBF} \) (Mean Time Between Failures)
  - \( \text{MTBF}_{\text{disk}} \approx 1,200,00 \text{ hours} (~136 \text{ years}, <1\% \text{ per year}) \)
  - \( \text{MTBF}_{\text{multi-disk system}} \) = mean time to first disk failure
    - which is \( \text{MTBF}_{\text{disk}} / \) (number of disks)
    - For a striped array of 200 drives
      - \( \text{MTBF}_{\text{array}} = 136 \text{ years} / 200 \text{ drives} = 0.65 \text{ years} \)

Back to Mean Time To Data Loss (MTTDL)

- \( \text{MTBF} \) (Mean Time Between Failures)
  - \( \text{MTBF}_{\text{disk}} \approx 1,200,00 \text{ hours} (~136 \text{ years}, <1\% \text{ per year}) \)
  - \( \text{MTBF}_{\text{multi-disk system}} = \) mean time to first disk failure
    - which is \( \text{MTBF}_{\text{disk}} / \) (number of disks)
    - For a striped array of 200 drives
      - \( \text{MTBF}_{\text{array}} = 136 \text{ years} / 200 \text{ drives} = 0.65 \text{ years} \)
Reliability without rebuild

- 200 data drives with MTBF_{drive}
  \[ \text{MTTDL}_{array} = \frac{\text{MTBF}_{drive}}{200} \]

- Add 200 drives and do mirroring
  \[ \text{MTBF}_{pair} = \left( \frac{\text{MTBF}_{drive}}{2} \right) + \frac{\text{MTBF}_{drive}}{133} \]

- Add 50 drives, each with parity across 4 data disks
  \[ \text{MTBF}_{set} = \left( \frac{\text{MTBF}_{drive}}{5} \right) + \left( \frac{\text{MTBF}_{drive}}{4} \right) = 0.45 \times \text{MTBF}_{drive} \]

Reliability consequences of adding rebuild

- No data loss, if fast enough
  - That is, if first failure fixed before second one happens

- New math is...
  - MTTDL_{array} = \frac{\text{MTBF}_{drive}}{\text{MTTR}_{drive}} \times \left( 1 / \text{prob of 2nd failure before repair} \right)
  - ... which is \( \frac{\text{MTTR}_{drive}}{\text{MTBF}_{seconddrive}} \)

- For mirroring
  \[ \frac{\text{MTBF}_{pair}}{50} = \frac{\text{MTBF}_{drive}}{111} \]

- For 5-disk parity-protected arrays
  \[ \frac{\text{MTBF}_{set}}{50} = \frac{\text{MTBF}_{drive}}{111} \]

Rebuild: restoring redundancy after failure

- After a drive failure
  \[ \text{data is still available for access} \]
  \[ \text{but, a second failure is BAD} \]

- So, should reconstruct the data onto a new drive
  \[ \text{on-line spares are common features of high-end disk arrays} \]
  \[ \text{reduce time to start rebuild} \]
  \[ \text{must balance rebuild rate with foreground performance impact} \]
  - a performance vs. reliability trade-offs

- How data is reconstructed
  - Mirroring: just read good copy
  - Parity: read all remaining drives (including parity) and compute

Three modes of operation

- Normal mode
  - everything working; maximum efficiency

- Degraded mode
  - some disk unavailable
  - must use degraded mode operations

- Rebuild mode
  - reconstructing lost disk’s contents onto spare
  - degraded mode operations plus competition with rebuild

Mechanics of rebuild

- Background process
  - use degraded mode read to reconstruct data
  - then, write it to replacement disk

- Implementation issues
  - Interference with foreground activity and controlling rate
  - Rebuild is important for reliability
  - Foreground activity is important for performance
  - Using the rebuilt disk
  - For rebuilt part, reads can use replacement disk
  - Must balance performance benefit with rebuild interference

Conclusions

- RAID turns multiple disks into a larger, faster, more reliable disk
- RAID-0: Striping
  - Good when performance and capacity really matter, but reliability doesn’t
- RAID-1: Mirroring
  - Good when reliability and write performance matter, but capacity (cost) doesn’t
- RAID-5: Rotating Parity
  - Good when capacity and cost matter or workload is read-mostly
  - Good compromise choice