RAID, Feb 5 2016

Thanks to Greg Ganger and Remzi Arapaci-Dusseau for slides
## Replacement Rates

<table>
<thead>
<tr>
<th>Component</th>
<th>HPC1 %</th>
<th>COM1 %</th>
<th>COM2 %</th>
</tr>
</thead>
<tbody>
<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>Memory</td>
<td>20.1</td>
</tr>
<tr>
<td>Misc/Unk</td>
<td>14.4</td>
<td>Hard drive</td>
<td>18.1</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>Fan</td>
<td>8</td>
</tr>
<tr>
<td>Controller</td>
<td>2.9</td>
<td>CPU</td>
<td>2</td>
</tr>
<tr>
<td>QSW</td>
<td>1.7</td>
<td>SCSI Board</td>
<td>0.6</td>
</tr>
<tr>
<td>Power supply</td>
<td>1.6</td>
<td>NIC Card</td>
<td>1.2</td>
</tr>
<tr>
<td>MLB</td>
<td>1</td>
<td>LV Pwr Board</td>
<td>0.6</td>
</tr>
<tr>
<td>SCSI BP</td>
<td>0.3</td>
<td>CPU heatsink</td>
<td>0.6</td>
</tr>
</tbody>
</table>
Outline

• Using multiple disks
  • Why have multiple disks?
  • problem and approaches

• RAID levels and performance

• Estimating availability
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?
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 files 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
```

```
Stripe
```
Disk striping details

• How disk striping works
  • Break up total space into fixed-size stripe units
  • Distribute the stripe units among disks in round-robin
  • Compute location of block #B as follows
    • disk# = B%N (%=modulo,N = #ofdisks)
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
  - 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
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
- 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**

D  D  D  P

**Fault-Free Write**

D  D  D  P

②  ->  +

③  ↓  +  ④

**Degraded Read**

D  D  D  P

**Degraded Write**

D  D  D  P

②  ← +  ← ③
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
  - Constructed by UC-Berkeley researchers in late 80s (Garth)
- 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-0: Striping

- Evaluate for D disks
- Performance: How much faster than 1 disk?
- Reliability: More or less reliable than 1 disk?
RAID-1: Mirroring

- Motivation: Handle disk failures
- Put copy (mirror or replica) of each chunk on another disk

- Capacity
- Reliability
- Performance
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 1 0 → Parity value?
- **How do you recover from a failed disk?**
  - Example: x 0 0 and parity of 1
  - What is the failed value?

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>XOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
RAID-4: Parity

- Capacity:
- Reliability:
- Performance:
  - Reads
  - Writes: How to update parity block?
    - Two different approaches
      - Small number of disks (or large write):
      - Large number of disks (or small write):
    - Parity disk is the bottleneck
RAID-5: Rotated Parity

- Capacity:
- Reliability:
- Performance:
  - Reads:
  - Writes:
  - Still requires 4 I/Os per write, but not always to same parity disk

Rotate location of parity across all disks
Comparison

<table>
<thead>
<tr>
<th>RAID Type</th>
<th>Capacity</th>
<th>Reliability</th>
<th>Throughput</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAID-0</td>
<td>$N$</td>
<td>0</td>
<td>$N \cdot S$</td>
<td>$D$</td>
</tr>
<tr>
<td>RAID-1</td>
<td>$N/2$</td>
<td>1 (for sure)</td>
<td>$(N/2) \cdot S$</td>
<td>$D$</td>
</tr>
<tr>
<td>RAID-4</td>
<td>$N - 1$</td>
<td>1</td>
<td>$(N - 1) \cdot S$</td>
<td>$D$</td>
</tr>
<tr>
<td>RAID-5</td>
<td>$N - 1$</td>
<td>1</td>
<td>$(N - 1) \cdot S$</td>
<td>$D$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Throughput</th>
<th>RAID-0</th>
<th>RAID-1</th>
<th>RAID-4</th>
<th>RAID-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential Read</td>
<td>$N \cdot S$</td>
<td>$(N/2) \cdot S$</td>
<td>$(N - 1) \cdot S$</td>
<td>$(N - 1) \cdot S$</td>
</tr>
<tr>
<td>Sequential Write</td>
<td>$N \cdot S$</td>
<td>$(N/2) \cdot S$</td>
<td>$(N - 1) \cdot S$</td>
<td>$(N - 1) \cdot S$</td>
</tr>
<tr>
<td>Random Read</td>
<td>$N \cdot R$</td>
<td>$N \cdot R$</td>
<td>$(N - 1) \cdot R$</td>
<td>$N \cdot R$</td>
</tr>
<tr>
<td>Random Write</td>
<td>$N \cdot R$</td>
<td>$(N/2) \cdot R$</td>
<td>$1/2 \cdot R$</td>
<td>$N/4 \cdot R$</td>
</tr>
</tbody>
</table>

Table 38.7: RAID Capacity, Reliability, and Performance
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
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)

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

Installed \quad Fixed \quad TBF_1 \quad TTR_1 \quad TBF_2 \quad TTR_2 \quad TBF_3 \quad TTR_3

Available during these 3 periods of time.
How often are failures?

- **MTBF (Mean Time Between Failures)**
  - \(\text{MTBF}_{\text{disk}} \sim 1,200,00\) hours (~136 years, <1% per year)

- **MTBF_{\text{multi-disk system}}** = mean time to first disk failure
  - which is \(\text{MTBF}_{\text{disk}} / \text{(number of disks)}\)
  - For a striped array of 200 drives
  - \(\text{MTBF}_{\text{array}} = 136\) years / 200 drives = 0.65 years
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