RAID: (Redundant?) Arrays of Inexpensive Disks

CSCI 333
Why RAID?

1. The ideas used by RAID are “big ideas” in systems
   - Striping
   - Replication
   - Parity

2. Fault tolerance is important in practice
   - We must first define a model for how things can fail
   - Then we can design systems to overcome those failures
How to Read

Do not spend your time memorizing RAID levels

• Instead, think about the “big ideas” and their tradeoffs
  ‣ When would you stripe writes? When is striping not worth the work?
  ‣ Should you use replication or parity? How many replicates do you need?

Think about each idea in terms of:

• Performance
• Capacity
• Fault tolerance

Think about how to apply the ideas elsewhere:

• Modern systems comprise many abstract layers. Where can you apply these ideas, and where does abstraction get in the way?
Other Related Topics

Erasure coding

Crash recovery/consistency

“Byzantine” fault tolerance

Deduplication (perhaps the inverse of replication…)

RAID: (Redundant?) Arrays of Inexpensive Disks

CSCI 333
Redundant Arrays of Inexpensive Disks

- Three “techniques”
  - Striping
  - Mirroring
  - Parity

- Three evaluation criteria
  - Performance
  - Reliability
  - Capacity

- Failure Model
  - Fail-stop

- RAID Levels
RAID from the user’s view

Hardware RAID is transparent to the user

• An array of disks are connected to the computer, and all the computer sees is a single logical disk
• Nothing about the RAID setup is externally visible: it’s just a single LBA space that appears and acts as one device
Why?

Why masquerade a set of $N > 1$ disks as a single volume of storage? What types of things might we want to improve?

• **Capacity**
  ▶ we may just want to store more data than fits on a single disk, but not change our software to manage multiple physical devices

• **Performance (parallelism and/or choice)**
  ▶ a single disk has one disk arm, so it can read from one location at a time. $N$ disks have $N$ disk arms. We can parallelize some operations

• **Recovery**
  ▶ if all of our data is on a single disk, we are extremely vulnerable to any disk failures
  ▶ if our data is on $N$ disks, we may not lose everything if we lose 1 disk
Suppose we have an array of 2 disks, each capable of storing $L$ logical blocks.

- Let’s partition the LBAs as follows:
  
  $0)-(L-1)$
  $L)-(2L-1)$

- What is the capacity?
- What is the performance?
- How many disk failures can we survive?
Suppose we have an array of 2 disks, each capable of storing $L$ logical blocks.

- Let’s partition the LBAs as follows:
  
  - What is the capacity?
  - What is the performance?
  - How many disk failures can we survive?

Striping adds parallelism to *sequential writes*.
Aside: Chunks

How to best “stripe” the data?

• Previous slide has chunk size of 1
• What are the tradeoffs of increasing the chunk size (the number of consecutive LBAs per disk in a stripe)?

Chunk size affects parallelism:

• With a small chunk size, it is more likely that a write will be striped across many disks, increasing parallelism
• With a large chunk size, some writes may directed to fewer disks
  ▶ The system can still get parallelism from making multiple independent requests
In addition to performance, we may use extra disks to increase the reliability of our storage

- Disks fail for a variety of reasons
- We want to be able to undergo one (or more) disk failures without losing data
- If possible, we also want to preserve/improve performance
How Can Disks Fail?

**RAID assumes disks are fail-stop**

- If there is an error, we can detect the error immediately
- Assume a simple state machine: either the entire disk works, or the entire disk has failed
What other classes of errors could possibly exist?

• Failures can be transient
  ‣ e.g., a temporary error that fixes itself

• Failures can be unreliable
  ‣ e.g., sometimes an error is returned, sometimes the correct answer

• Failures can be partial
  ‣ e.g., a single sector or range of sectors become unusable

Disk losses may be correlated

• If your power supply goes, it may take all disks with it
• Flood/fire?
• Theft?

RAID doesn’t attempt to handle these errors
Redundancy: Mirroring

Suppose we have an array of 2 disks, each capable of storing $L$ logical blocks.

- Let’s partition the LBAs as follows:

![Diagram of two disks with partitions: 0—(L-1) and 0—(L-1)]

- What is the capacity?
- What is the performance?
- How many disk errors can we survive?
Bitwise XOR

Rule: Count the number of 1s, and
- If the number of 1s is odd, the parity bit is 1
- If the number of 1s is even, the parity bit is 0

To extend the idea to disk blocks:
- bitwise XOR the i\textsuperscript{th} bit of each block; result is the i\textsuperscript{th} bit of the parity block

Example:

<table>
<thead>
<tr>
<th>2 Bits:</th>
<th>3 Bits:</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR(1,1) = 0</td>
<td>XOR(1,1,1) = 1</td>
</tr>
<tr>
<td>XOR(0,1) = 1</td>
<td>XOR(1,1,0) = 0</td>
</tr>
<tr>
<td>XOR(1,0) = 1</td>
<td>XOR(1,0,1) = 0</td>
</tr>
<tr>
<td>XOR(0,0) = 0</td>
<td>XOR(0,1,1) = 0</td>
</tr>
<tr>
<td>XOR(1,0,0) = 1</td>
<td>XOR(1,0,0) = 1</td>
</tr>
<tr>
<td>XOR(0,0,1) = 1</td>
<td>XOR(0,1,0) = 1</td>
</tr>
<tr>
<td>XOR(0,0,0) = 0</td>
<td>XOR(0,0,0) = 0</td>
</tr>
</tbody>
</table>
Suppose we have an array of 3 disks, each capable of storing $L$ logical blocks.

- Let’s partition the LBAs as follows:

  - $0, 2, 4, ..., (2L-2)$
  - $1, 3, 5, ..., (2L-1)$
  - $P_0, P_1, ..., P(L-1)$

- What is the capacity?
- What is the performance?
- How many disk errors can we survive?
You can combine some RAID levels in fun ways

**RAID 10 vs. RAID 01**

<table>
<thead>
<tr>
<th>Mirror</th>
<th>Stripe</th>
<th>Mirror</th>
<th>Stripe</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1</td>
<td>2 2</td>
<td>1 2</td>
<td>3 4</td>
</tr>
<tr>
<td>3 3</td>
<td>4 4</td>
<td>3 4</td>
<td>5 6</td>
</tr>
<tr>
<td>5 5</td>
<td>6 6</td>
<td>5 6</td>
<td>5 6</td>
</tr>
</tbody>
</table>
You can combine some RAID ideas in fun ways

**RAID 4 vs. RAID 5**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>P0</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>P1</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>P2</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>12</td>
<td>P3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>P0</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>P1</td>
</tr>
<tr>
<td>7</td>
<td>P2</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>P3</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>
Why RAID

It is a relatively straightforward concept, but practical and very useful

The ideas can be applied to distributed storage and other environments (e.g., think of “nodes” as disks)

I felt bad letting you leave without knowing about RAID; I’ve seen it as an interview question

- But most of the concepts can be reasoned about on the fly
  - You should remember mirroring, striping, and parity, not Level 0, Level 1, Level 4, Level 5.
  - (Remind your interviewers that you are there to think not to memorize)
- Other levels are less common, but common sense. Explore!