RAID
Redundant Array of Inexpensive Disks

CMPU 334 – Operating Systems
Jason Waterman
I/O Time: Fun With Math

• I/O time ($T_{I/O}$):

\[ T_{I/O} = T_{\text{seek}} + T_{\text{rotation}} + T_{\text{transfer}} \]

• The rate of I/O ($R_{I/O}$):

\[ R_{I/O} = \frac{\text{Size}_{\text{Transfer}}}{T_{I/O}} \]

<table>
<thead>
<tr>
<th></th>
<th>Cheetah 15K.5</th>
<th>Barracuda</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capacity</td>
<td>300 GB</td>
<td>1 TB</td>
</tr>
<tr>
<td>RPM</td>
<td>15,000</td>
<td>7,200</td>
</tr>
<tr>
<td>Average Seek</td>
<td>4 ms</td>
<td>9 ms</td>
</tr>
<tr>
<td>Max Transfer</td>
<td>125 MB/s</td>
<td>105 MB/s</td>
</tr>
<tr>
<td>Platters</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Cache</td>
<td>16 MB</td>
<td>16/32 MB</td>
</tr>
<tr>
<td>Connects Via</td>
<td>SCSI</td>
<td>SATA</td>
</tr>
</tbody>
</table>

**Disk Drive Specs: SCSI Versus SATA**
## I/O Time Example

- **Random workload**: Issue 4KB read to random locations on the disk
- **Sequential workload**: Read 100MB consecutively from the disk

<table>
<thead>
<tr>
<th></th>
<th>Cheetah 15K.5</th>
<th>Barracuda</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{\text{seek}}$</td>
<td>4 ms</td>
<td>9 ms</td>
</tr>
<tr>
<td>$T_{\text{rotation}}$</td>
<td>2 ms</td>
<td>4.2 ms</td>
</tr>
<tr>
<td>$T_{\text{transfer}}$</td>
<td>30 microsecs</td>
<td>38 microsecs</td>
</tr>
<tr>
<td>$T_{\text{I/O}}$</td>
<td>6 ms</td>
<td>13.2 ms</td>
</tr>
<tr>
<td>$R_{\text{I/O}}$</td>
<td>0.66 MB/s</td>
<td>0.31 MB/s</td>
</tr>
</tbody>
</table>

**Random**

<table>
<thead>
<tr>
<th></th>
<th>Cheetah 15K.5</th>
<th>Barracuda</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{\text{transfer}}$</td>
<td>800 ms</td>
<td>950 ms</td>
</tr>
<tr>
<td>$T_{\text{I/O}}$</td>
<td>806 ms</td>
<td>963.2 ms</td>
</tr>
<tr>
<td>$R_{\text{I/O}}$</td>
<td>125 MB/s</td>
<td>105 MB/s</td>
</tr>
</tbody>
</table>

**Sequential**

There is a huge gap in drive performance between **random** and **sequential** workloads.
Disk Scheduling

- **Disk Scheduler** decides *which I/O request* to schedule next
- **SSTF** (Shortest Seek Time First)
  - Order the queue of I/O request by track
  - Pick requests on the nearest track to complete first

![Diagram showing disk scheduling with SSTF algorithm](image)

**SSTF: Scheduling Request 21 and 2**

Issue the request to 21 → issue the request to 2
SSTF is not a panacea

• **Problem 1**: The drive geometry is not available to the host OS
  • Solution: OS can simply implement Nearest-block-first (NBF)

• **Problem 2**: Starvation
  • If there were a steady stream of request to the inner track, request to other tracks would then be ignored completely
Elevator (a.k.a. SCAN or C-SCAN)

• Move across the disk servicing requests in order across the tracks
  • **Sweep**: A single pass across the disk
    • If a request comes for a block on a track that has already been serviced on this sweep of the disk, it is queued until the next sweep

• **F-SCAN**
  • Freeze the queue to be serviced when it is doing a sweep
  • Avoid starvation of far-away requests by nearer by late coming requests

• **C-SCAN** (Circular SCAN)
  • Sweep from outer-to-inner, and then inner-to-outer, etc.
How to account for Disk rotation costs?

- If rotation is faster than seek: request 16 → request 8
- If seek is faster than rotation: request 8 → request 16

On modern drives, both seek and rotation are roughly equivalent: **Thus, SPTF (Shortest Positioning Time First) is useful**
Where is disk scheduling performed?

• Older systems:
  • OS did all the scheduling

• Newer systems:
  • Disks can handle multiple outstanding requests
  • Disks have sophisticated internal schedulers
    • Exact head position is available
    • Can implement SPTF accurately
  • OS receives a small number of disk requests (e.g., 16) and issues them all at once
    • Disk calculates the best possible SPTF order
Other scheduling issues

• I/O Merging
  • Reduce the number of request sent to the disk and lowers overhead
  • E.g., read blocks 33, then 8, then 34:
    • The scheduler merge the request for blocks 33 and 34 into a single two-block request

• How long to wait before issuing an I/O request?
  • Work-conserving – Issue I/O request right away
    • Disk will never be idle if there are requests to serve
  • Non-work-conserving – wait a little bit before issuing I/O request
    • A new and better request might arrive at the disk, increasing efficiency
RAID (Redundant Array of Inexpensive Disks)

- Use multiple disks in concert to build a faster, bigger, and more reliable disk system
  - RAID just looks like one big disk to the host system

- Advantages
  - Performance & Capacity: Using multiple disks in parallel
  - Reliability: RAID can tolerate the loss of a disk

RAIDs provide these advantages transparently to systems that use them
RAID Interface

• When a RAID receives I/O request:
  1. The RAID calculates which disk to access
  2. The RAID issue one or more physical I/Os to do so

• RAID example: A mirrored RAID system
  • Keep two copies of each block (each one on a separate disk)
  • Perform two physical I/Os for every one logical I/O it is issued
RAID Internals

• A microcontroller
  • Run firmware to direct the operation of the RAID

• Volatile memory (such as DRAM)
  • Buffer data blocks

• Non-volatile memory
  • Buffer writes safely

• Specialized logic to perform parity calculation

• In essence, a RAID is a specialized computer system!
Fault Model

- RAIDs are designed to detect and recover from certain kinds of disk faults

- Assume for now a fail-stop fault model
  - A disk can be in one of two states: Working or Failed
    - Working: all blocks can be read or written
    - Failed: the disk is permanently lost
  - RAID controller can immediately observe when a disk has failed

- Worry about other types of failures later
  - Disk corruption
  - Having a single block of the disk fail in a otherwise working drive
How to evaluate a RAID

• **Capacity**
  • How much useful capacity is available to systems?

• **Reliability**
  • How many disk faults can the given design tolerate?

• **Performance**
  • Throughput in MB/s
  • Challenging to evaluate as it heavily depends on workload
  • Talk more in depth later
RAID Designs

• RAID level 0 (striping)
  • No redundancy
  • Gives an upper bound on performance and capacity

• RAID level 1 (mirroring)
  • Tolerates disk failures
  • Keeps one or more copies of each block

• RAID level 4/5 (parity-based redundancy)
  • More complicated than mirroring
  • Provides redundancy with better space utilization than mirroring
RAID Level 0: Striping

- RAID Level 0 is the simplest form as striping blocks
  - Spread the blocks across the disks in a round-robin fashion
  - **No redundancy**
  - Excellent performance and capacity

<table>
<thead>
<tr>
<th></th>
<th>Disk 0</th>
<th>Disk 1</th>
<th>Disk 2</th>
<th>Disk 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td></td>
</tr>
</tbody>
</table>

**RAID-0: Simple Striping**
(Assume here a 4-disk array)
RAID Chunk Size

- Chunk size: number of blocks per disk in a stripe
  - Example: RAID-0 with a bigger chunk size and 4KB blocks
  - Chunk size: 2 blocks (8 KB)
  - A Stripe: 4 chunks (32 KB)

- Chunk size affects performance
  - Small chunk size
    - Files get striped across many disks
    - Increases parallelism for a single file
    - Increases positioning time to access blocks
  - Large chunk size
    - Reduces intra-file parallelism
    - Reduces positioning time

Determining the “best” chunk size is hard to do
Most arrays use larger chunk sizes (e.g., 64 KB)
Evaluating RAID Performance

• Consider two performance metrics
  • Single request latency (time to process a single read request)
  • Steady-state throughput (total bandwidth of many concurrent requests)
    • Main focus of our analysis

• Workload
  • Sequential: e.g., access 1MB of data (block (B) to block (B + 1MB))
  • Random: e.g., access 4KB at random logical address

• A disk can transfer data at
  • $S$ MB/s under a sequential workload
  • $R$ MB/s under a random workload
Sequential vs Performance Calculation

• Sequential ($S$) vs random ($R$)
  - **Sequential** : transfer 10 MB on average as continuous data
  - **Random** : transfer 10 KB on average
  - Average seek time: 7 ms
  - Average rotational delay: 3 ms
  - Transfer rate of disk: 50 MB/s

• Results:
  - $S = \frac{\text{Amount of Data}}{\text{Time to access}} = \frac{10 \text{ MB}}{210 \text{ ms}} = 47.62 \text{ MB/s}$
  - $R = \frac{\text{Amount of Data}}{\text{Time to access}} = \frac{10 \text{ KB}}{10.195 \text{ ms}} = 0.981 \text{ MB/s}$
Evaluating RAID-0 Performance

• Single request latency
  • Similar to that of a single disk
  • RAID-0 simply redirects the request to the proper drive

• Steady-state throughput
  • Expect to get the full bandwidth of the system
  • **Sequential** workload: \( N \cdot S \ \text{MB/s} \)
    • \( S \) is the sequential bandwidth of a single disk
  • **Random** workload: \( N \cdot R \ \text{MB/s} \)
    • \( R \) is the random bandwidth of a single disk

• Represent the upper-bound on RAID performance

\( N \) : the number of disks
RAID Level 0 Analysis

• **Capacity** → RAID-0 is optimal
  • Striping delivers N disks worth of useful capacity

• **Performance** of striping → RAID-0 is excellent
  • All disks are utilized often in parallel

• **Reliability** → RAID-0 is bad
  • Any disk failure will lead to data loss

\( N \): the number of disks
RAID Level 1: Mirroring

- RAID Level 1 tolerates disk failures
  - Copy more than one of each block in the system
  - Block copies are placed on a separate disk

- Two choices
  - RAID-10 (RAID 1+0): mirrored pairs and then stripe (more common, shown on the left)
  - RAID-01 (RAID 0+1): contain two large striping arrays, and then mirrors (shown on the right)
RAID-1 Analysis

• **Capacity**: RAID-1 is Expensive
  • The useful capacity of RAID-1 is N/2

• **Reliability**: RAID-1 does well
  • It can tolerate the failure of any one disk (up to N/2 failures depending on which disks fail)

\( N \) : the number of disks
Performance of RAID-1

- Single request latency
  - Reads have the latency of a single disk
  - Writes need two physical writes to complete
    - It suffers the worst-case seek and rotational delay of the two requests

- Steady-state throughput
  - **Sequential Write**: $\frac{N}{2} \cdot S$ MB/s
    - Each logical write must result in two physical writes
  - **Sequential Read**: $\frac{N}{2} \cdot S$ MB/s
    - Each disk will only deliver half its peak bandwidth
    - This seems counterintuitive, why is it so?
  - **Random Write**: $\frac{N}{2} \cdot R$ MB/s
    - Each logical write must turn into two physical writes
  - **Random Read**: $N \cdot R$ MB/s
    - Distribute the reads across all the disks

<table>
<thead>
<tr>
<th>Disk 0</th>
<th>Disk 1</th>
<th>Disk 2</th>
<th>Disk 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

RAID-1: Sequential read request (0-7)
RAID Level 4: Saving Space With Parity

- Add a single **parity block**
  - A parity block stores the redundant information for that stripe of blocks

<table>
<thead>
<tr>
<th>Disk 0</th>
<th>Disk 1</th>
<th>Disk 2</th>
<th>Disk 3</th>
<th>Disk 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>P0</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>P1</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>P2</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>P3</td>
</tr>
</tbody>
</table>

* P: Parity

Five-disk RAID-4 system layout
RAID Level 4 (Cont.)

• **Compute parity**: the XOR of all of bits

<table>
<thead>
<tr>
<th>C0</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>XOR(0,0,1,1)=0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>XOR(0,1,0,0)=1</td>
</tr>
</tbody>
</table>

• Counting the parity bit, the row will always have an **even number of ones**

• **Recover from parity**
  
  • Imagine the bit of the C2 in the first row is lost
    
    1. Reading the other values in that row: 0, 0, 1
    2. The parity bit is 0 → even number of ones in the row
    3. The missing data must be a 1

• For a block of data
  
  • Compute parity bits for $b_0b_1b_2...b_{511}$
RAID-4 Analysis

• **Capacity**
  • The useful capacity is \((N - 1)\)

• **Reliability**
  • RAID-4 tolerates 1 disk failure and no more

\(N\) : the number of disks
RAID-4 Analysis (Cont.)

• **Performance**
  
  • Steady-state throughput
    
    • Sequential read: \((N - 1) \cdot S\) MB/s
    
    • Sequential write: \((N - 1) \cdot S\) MB/s (using full-stripe write optimization)
      
      • Parity block can be written in parallel with the data blocks

  
  • Random read: \((N - 1) \cdot R\) MB/s
    
    • Reads are spread across the disks, but not the parity disk

<table>
<thead>
<tr>
<th>Disk 0</th>
<th>Disk 1</th>
<th>Disk 2</th>
<th>Disk 3</th>
<th>Disk 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>P0</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>P1</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>P2</td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>P3</td>
</tr>
</tbody>
</table>

**Full-stripe Writes In RAID-4**
Random write performance for RAID-4

• Overwrite a block + update the parity

• **Method 1: additive parity**
  • Read in all of the other data blocks in the stripe (N reads)
  • XOR those blocks with the new block
  • Data and parity writes can happen in parallel (2 writes)
  • **Problem**: larger RAIDs require a high number of reads to compute parity
Random write performance for RAID-4 (Cont.)

• **Method 2: subtractive parity**

  ![Parity Table](image)

  - Update $C_2(\text{old}) \rightarrow C_2(\text{new})$
    1. Read in the old data at $C_2$ ($C_2(\text{old})=1$) and the old parity ($P(\text{old})=0$)
    2. Calculate $P(\text{new})$: $P(\text{new}) = (C_2(\text{old}) \text{ XOR } C_2(\text{new})) \text{ XOR } P(\text{old})$
      - If $C_2(\text{new})==C_2(\text{old}) \rightarrow P(\text{new})==P(\text{old})$
      - If $C_2(\text{new})! = C_2(\text{old}) \rightarrow$ Flip the old parity bit
  - Two reads ($C_2(\text{old})$ and $P(\text{old})$) and two writes ($C_2(\text{new})$ and $P(\text{new})$)
    - Regardless of the size of the array
A I/O latency in RAID-4

• A single read
  • Equivalent to the latency of a single disk request

• A single write (using subtractive parity)
  • Two reads and then two writes
    • Data block + Parity block
    • The reads and writes can happen in parallel
  • Total latency is about twice that of a single disk
Small-write problem

- The parity disk can be a bottleneck
  - Example: update blocks 4 and 13 (marked with *)

<table>
<thead>
<tr>
<th>Disk 0</th>
<th>Disk 1</th>
<th>Disk 2</th>
<th>Disk 3</th>
<th>Disk 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>P0</td>
</tr>
<tr>
<td>*4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>+P1</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>P2</td>
</tr>
<tr>
<td>12</td>
<td>*13</td>
<td>14</td>
<td>15</td>
<td>+P3</td>
</tr>
</tbody>
</table>

Writes To 4, 13 And Respective Parity Blocks

- Disk 0 and Disk 1 can be accessed in parallel
- Disk 4 prevents any parallelism

RAID-4 throughput under random small writes is \( \frac{R}{2} \) MB/s (terrible!)
RAID Level 5: Rotating Parity

- Solves the small write problem
  - Rotate the parity blocks across drives
  - Remove the parity-disk bottleneck for RAID-4

<table>
<thead>
<tr>
<th></th>
<th>Disk 0</th>
<th>Disk 1</th>
<th>Disk 2</th>
<th>Disk 3</th>
<th>Disk 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td>P0</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>6</td>
<td>7</td>
<td>P1</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>11</td>
<td></td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>15</td>
<td></td>
<td>P3</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>P4</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td></td>
</tr>
</tbody>
</table>

RAID-5 With Rotated Parity
RAID-5 Analysis

• **Capacity**
  - The useful capacity for a RAID group is \((N - 1)\)
  - Same as RAID-4

• **Reliability**
  - RAID-5 tolerates 1 disk failure and no more
  - Same as RAID-4

\(N\) : the number of disks
RAID-5 Analysis (Cont.)

- **Performance**
  - Sequential read and write
  - A single read and write request

- Random read: a little better than RAID-4
  - RAID-5 can utilize all of the disks

- Random write: \( \frac{N}{4} \cdot R \) MB/s
  - The factor of four loss due to each write generating 4 total I/O operations

- Performs the same as or better than RAID-4 in all cases
  - RAID-5 has almost completely replaced RAID-4 in the marketplace

\( N : \) the number of disks

Same as RAID-4
RAID Comparison: A Summary

N: the number of disks  
T: the time that a request to a single disk takes  
B: the blocks per disk

<table>
<thead>
<tr>
<th>RAID</th>
<th>Capacity</th>
<th>Reliability</th>
<th>Throughput</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RAID-0</td>
<td>RAID-1</td>
<td>RAID-4</td>
<td>RAID-5</td>
</tr>
<tr>
<td></td>
<td>N · B</td>
<td>(N · B)/2</td>
<td>(N – 1) · B</td>
<td>(N – 1) · B</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1 (for sure)</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N/2 (if lucky)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Throughput**

- **Sequential Read**
  - RAID-0: N · S
  - RAID-1: (N/2) · S
  - RAID-4: (N-1) · S
  - RAID-5: (N-1) · S

- **Sequential Write**
  - RAID-0: N · S
  - RAID-1: (N/2) · S
  - RAID-4: (N-1) · S
  - RAID-5: (N-1) · S

- **Random Read**
  - RAID-0: N · R
  - RAID-1: N · R
  - RAID-4: (N-1) · R
  - RAID-5: N · R

- **Random Write**
  - RAID-0: N · R
  - RAID-1: (N/2) · R
  - RAID-4: 1/2 R
  - RAID-5: N/4 R

**Latency**

- **Read**
  - RAID-0: T
  - RAID-1: T
  - RAID-4: T
  - RAID-5: T

- **Write**
  - RAID-0: T
  - RAID-1: T
  - RAID-4: 2T
  - RAID-5: 2T

**RAID Capacity, Reliability, and Performance**
RAID Comparison: A Summary

• **Performance** and do not care about reliability → RAID-0 (Striping)

• **Random I/O** performance and **Reliability** → RAID-1 (Mirroring)

• **Capacity** and **Reliability** → RAID-5

• **Sequential I/O** and Maximize **Capacity** → RAID-5