PERSISTENCE: DISK SCHEDULING

Shivaram Venkataraman
CS 537, Spring 2023
Project 4 grades out. Regrades?

Project 5 – due soon?

Midterm 2 – April 4th, lots of details on Piazza

Thursday - guest lecture
How do you calculate sequential and random tput of a disk?

What algorithms are used to schedule I/O requests?
RECAP
while (STATUS == BUSY) 
    ; // spin
Write data to DATA register
Write command to COMMAND register
while (STATUS == BUSY) 
    ; // spin
Motor connected to spindle *spins* platters

Rate of rotation: RPM

10000 RPM → single rotation is 6 ms
Heads on a moving arm can read from each surface.

- Seek to get to the right track
- Wait for rotation, right sector is below the head
SEEK, ROTATE, TRANSFER

Seek cost: Function of cylinder distance
Not purely linear cost
Must accelerate, coast, decelerate, settle
Settling alone can take 0.5 - 2 ms

Entire seeks often takes 4 - 10 ms
Average seek = 1/3 of max seek

Textbook derivation

Total time = seek + rotation + transfer time

RPM is given by disk manufacturer

Depends on rotations per minute (RPM)
→ 7200 RPM is common, 15000 RPM is high end

Average rotation: Half of time for 1 rotation
on average we wait for half rotation

Pretty fast: depends on RPM and sector density.

100+ MB/s is typical for maximum transfer rate

\[
\text{data size} \rightarrow \frac{4\text{KB}}{} = \frac{\text{Transfer time}}{\text{BW}} = \frac{100\text{ MB/s}}{\text{data size}}
\]
What is the time for 4KB random read with Cheetah?

\[
\text{Total Time} = T_{\text{seek}} + T_{\text{rotate}} + T_{\text{transfer}}
\]

\[
= \frac{4}{6} + \frac{2}{6} + \frac{4 \times 4 \text{KB}}{125 \text{ MB/s} \times 1000}
\]

\[
= 0.032 \text{ ms} + 32 \text{ ms}
\]

\[
= 6 \text{ ms}
\]

**Cheetah 15K.5**

<table>
<thead>
<tr>
<th>Capacity</th>
<th>300 GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPM</td>
<td>15,000</td>
</tr>
<tr>
<td>Average Seek</td>
<td>4 ms</td>
</tr>
<tr>
<td>Max Transfer</td>
<td>125 MB/s</td>
</tr>
<tr>
<td>Platters</td>
<td>4</td>
</tr>
<tr>
<td>Cache</td>
<td>16 MB</td>
</tr>
<tr>
<td>Connects via</td>
<td>SCSI</td>
</tr>
</tbody>
</table>

**Barracuda**

<table>
<thead>
<tr>
<th>Capacity</th>
<th>1 TB</th>
</tr>
</thead>
<tbody>
<tr>
<td>RPM</td>
<td>7,200</td>
</tr>
<tr>
<td>Average Seek</td>
<td>9 ms</td>
</tr>
<tr>
<td>Max Transfer</td>
<td>105 MB/s</td>
</tr>
<tr>
<td>Platters</td>
<td>4</td>
</tr>
<tr>
<td>Cache</td>
<td>16/32 MB</td>
</tr>
<tr>
<td>Connects via</td>
<td>SATA</td>
</tr>
</tbody>
</table>
What is the time for 4KB random read with Barracuda?

\[ T_{\text{total}} = T_{\text{seek}} + T_{\text{rotate}} + T_{\text{transfer}} \]

\[ = 9 \text{ ms} + 4.16 \text{ ms} + 4 \text{ KB} \]

\[ = 9 \text{ ms} + 4.16 \text{ ms} + \frac{4 \text{ KB}}{105 \text{ MB/s}} \]

\[ = 9 \text{ ms} + 4.16 \text{ ms} + 0.038 \text{ ms} \]

\[ = 13.198 \text{ ms} \]

\[ T_{\text{rotate}} = \frac{60 \times 1000 \text{ ms}}{7200} = 8.33 \text{ ms} \times \frac{1}{2} = 4.16 \text{ ms} \]

Barracuda

<table>
<thead>
<tr>
<th>Capacity</th>
<th>300 GB</th>
<th>1 TB</th>
</tr>
</thead>
<tbody>
<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>
So…
- seeks are slow
- rotations are slow
- transfers are fast

How does the kind of workload affect performance?
Sequential: access sectors in order
Random: access sectors arbitrarily

4 - 10 ms

large sequential

1 seek + rotate
sequentially read data
## DISK SPEC

<table>
<thead>
<tr>
<th></th>
<th>Cheetah</th>
<th>Barracuda</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Capacity</strong></td>
<td>300 GB</td>
<td>1 TB</td>
</tr>
<tr>
<td><strong>RPM</strong></td>
<td>15,000</td>
<td>7,200</td>
</tr>
<tr>
<td><strong>Avg Seek</strong></td>
<td>4 ms</td>
<td>9 ms</td>
</tr>
<tr>
<td><strong>Max Transfer</strong></td>
<td>125 MB/s</td>
<td>105 MB/s</td>
</tr>
<tr>
<td><strong>Platters</strong></td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td><strong>Cache</strong></td>
<td>16 MB</td>
<td>32 MB</td>
</tr>
</tbody>
</table>

Sequential read 100 MB: what is throughput for each?

\[
T_{transfer} = \frac{100 \text{ MB}}{125 \text{ MB/s}} \approx 0.8 \text{ s} = 800 \text{ ms}
\]
I/O SCHEDULERS
I/O SCHEDULERS

Given a stream of I/O requests, in what order should they be served?

Much different than CPU scheduling

Position of disk head relative to request position matters more than length of job

Example:

read sector 13 → First  
read sector 24 → After
read sector 14 → Second
read sector 15 → Third
FCFS (First-Come-First-Serve)

Assume seek+rotate = 10 ms for random request

How long (roughly) does the below workload take? Requests are given in sector numbers

\[300001, 700001, 300002, 700002, 300003, 700003\]

= 60 ms

\[300001, 300002, 300003, 700001, 700002, 700003\]

= 20 ms

Reordering can improve performance for this sequence.
**SSTF (SHORTEST SEEK TIME FIRST)**

**Strategy** always choose request that requires least seek time
(approximate total time with seek time) → next request as the one with least seek time

Greedy algorithm (just looks for best NEXT decision)

How to implement in OS?
- How do I know the seek time? Sort by sector number

Disadvantages?
- Starvation: a disk request that is always waiting.
SCAN or Elevator Algorithm:
- Sweep back and forth, from one end of disk other, serving requests as pass that cylinder
- Sorts by cylinder number; ignores rotation delays

C-SCAN (circular scan): Only sweep in one direction

Pros/Cons?
- Trade-off: Scan can lead to some tracks being read more often
- Trade-off: Avoid unfairness across tracks

\[5, 24, 12, 74, 33\]
\[88, 6, 54, 27\]
SPTF (SHORTEST POSITIONING TIME FIRST)

More accurate way of doing scheduling

$\rightarrow$ need to know a lot about disk specification

$\rightarrow$ where is the arm right now.

SATF (SHORTEST ACCESS TIME FIRST)
Disk accesses: 32, 12, 33, 3, 13, 4
Rotation Time = 2ms (non-adjacent reads)
Seek Time (for adjacent track) = 2ms.

What is the time taken when using (FCFS) scheduling?

- Head is at 30
  - 32: 2ms (rotation) = 2ms
  - 12: 2ms (seek) + 2ms rotate = 4ms
  - 33: 2ms (seek) + 2ms rotate = 6ms
  - 3: 4ms (seek) + 2ms rotate = 6ms
  - 13: 2ms + 2ms = 4ms
  - 4: 2ms + 2ms = 4ms
Disk accesses: 32, 12, 33, 3, 13, 4
Rotation Time = 2ms (non-adjacent reads)
Seek Time (for adjacent track) = 2ms.

Order in which requests will be serviced for Shortest Seek Time First (SSTF)?

32, 33, 12, 13, 3, 4

Time Taken:

32: 2 ms
33: 0 ms
12: 2 + 2 = 4 ms < 10 ms
13: 0 ms
3: 4 ms < 10 ms
4: 0 ms
Schedulers

Where should the scheduler go?

OS
- fairness across process
- multiple policies
- upgrade / change

Scheduler

Disk

Knows a lot more
- density, time taken
- where is the arm
WHAT HAPPENS?

Assume 2 processes each calling read() with C-SCAN

```c
void reader(int fd) {
    char buf[1024];
    int rv;
    while((rv = read(fd, buf)) != 0) {
        assert(rv);
        // takes short time, e.g., 1ms
        process(buf, rv);
    }
}
```

File 1          P2: File 2
\[\begin{align*}
P1: & \text{ Read } 300 \\
    & \text{ Read } 301 \\
    & \text{ Read } 700 \\
    & \text{ Read } 701 \\
\end{align*}\]

\[\begin{align*}
P2: & \text{ Read } 300 \\
    & \text{ Read } 302 \\
    & \text{ Read } 303 \\
\end{align*}\]

→ How long do you wait to reorder requests?
Work conserving schedulers always try to do work if there's work to be done.

Sometimes, it's better to wait instead if the system anticipates another request will arrive.

Possible improvements from I/O Merging:
- Contiguous requests get merged.
- Read ahead.
- Never wait or keep disk idle.
SUMMARY

Disks: Specific geometry with platters, spindle, tracks, sector

I/O Time: rotation_time + seek_time + transfer_time
Sequential throughput vs. random throughput

Scheduling approaches: SSTF, SCAN, C-SCAN
Benefits of violating work conservation
Next class: How to achieve resilience against disk errors