Disk I/O Time = seek + rotational delay + transfer.

- Transfer time (bandwidth) time to move the bytes from the disk to memory.
- Rotational delay depends upon how fast the disk spins.
- Seek (latency) time for the sector to rotate under the head.

To read or write a disk block:

Today: Secondary Storage

Approaches to Improving Performance:

- Increase physical memory to reduce amount of time paging and thereby improve CPU utilization.
- Improve CPU utilization.
- Increase the number of devices to reduce contention for a single device and thereby
  reduce interrupt frequency by using DMA controllers.
- Reduce interrupt frequency by using large data transfers.
- Reduce data copying by caching in memory.

Last Class: I/O Systems
If our sector size is too large, we will have lots of internal fragmentation.

- Place commonly-used files where on the disk?
- Lay out data on disk so that related data are on nearby tracks.
- Schedule disk operations to minimize head movement
- Spin disks faster
- Make disks smaller

and rotational latency:

- Key: to get the quickest disk response, time, we must minimize seek time

<table>
<thead>
<tr>
<th>Buffer size</th>
<th>Buffer to host transfer rate</th>
<th>Average rotational latency</th>
<th>Average seek time</th>
<th>Spins per sector</th>
<th>Seconds per track</th>
<th>Tracks per surface</th>
<th>Spinup time</th>
<th>Write cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 MB</td>
<td>200 MB/sec</td>
<td>4.17 ms</td>
<td>8.5 ms</td>
<td>512</td>
<td>134</td>
<td>3832</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>4 MB</td>
<td>470 MB/sec</td>
<td>6.75 ms</td>
<td>12.0 ms</td>
<td>256</td>
<td>132</td>
<td>3840</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>3.5 inches</td>
<td>3.5 inches</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Typical Disk Parameters

- Good PC disk
- High-end disk
When would you expect this algorithm to work well?

- Distance of seeks
- Order of seeks

Example requests: 65, 40, 18, 78

**FCFS**

**Disk Head Scheduling**

1. **FCFS** - Service the requests in the order that they come in.

<table>
<thead>
<tr>
<th>Time (0-100)</th>
<th>Track Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>2</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
</tr>
<tr>
<td>40</td>
<td>4</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
</tr>
<tr>
<td>60</td>
<td>6</td>
</tr>
<tr>
<td>70</td>
<td>7</td>
</tr>
<tr>
<td>80</td>
<td>8</td>
</tr>
<tr>
<td>90</td>
<td>9</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
</tr>
</tbody>
</table>

2. **Shortest Seek Time First (SSTF) - First-Come, First-Served (FCFS)**

3. **SCAN** algorithm (0 to 0, 0 to 100, 100 to 0, 0 to 100, ...)

4. **C-SCAN** circular scan algorithm (0 to 100, 0 to 100, ...)

**Disk Scheduling**

- First: Permute the order of disk requests from the order that they arrive from the users.
- In order to an order that reduces the length and number of seeks.
- To seek the current position and the extreme (0 or N), we don't need to seek there.
SCAN Disk Head Scheduling

- Just as far as the last request,
  - Simple optimization does not go all the way to the edge of the disk each time, but
  - Requiers a sorted list of requests.
  - Distance of seeks:
  - Order of seeks: assuming the head is currently moving to lower numbered blocks:

  \[
  \left(\frac{1}{2}T, \frac{1}{2}S\right), \left(1, 1\right), \left(0, \frac{1}{2}S\right)
  \]

SCAN: Head moves back and forth across the disk (0 to 100, 100 to 0, 0)

STF: always go to the next closest request

- Problems?
  - Is it optimal?
  - Is this efficient enough?
- Can implement this approach by keeping a doubly linked list of requests.
  - Distance of seeks:
  - Order of seeks:
Improving Disk Performance using Disk Interleaving

- More uniform wait times for requests. Why?
- Distance of seeks:
- Order of seeks:

C-SCAN: Circular scan algorithm (0 to 100, 0 to 100, …)
Primary, secondary and tertiary devices form a storage hierarchy.

- Robotic jukeboxes
- Optical disks: Write once read-many (WORM), CD-R, CD-RW
- Jazz and Zip drives
- Tape drives

Used primarily for storing archival data or backups.

Typically slower, larger, cheaper than disks

Lower cost devices than secondary storage (disks)

**Tertiary Storage**

---

Is disk read-ahead any better?

Decided that was difficult to do well since the future is difficult to predict.

We considered pre-fetching virtual pages into physical memory but we

used while you have them under the read.

**Goal:** reduce the number of seeks - read blocks that will probably be

buffer on disk controller.

**Idea:** read blocks from the disk ahead of user's request and place in

**Improving Disk Performance using Read Ahead**
Review Questions:

- Is SCAN or C-SCAN fairer?
- Is SCAN or SSTF fairer?
- Rank the algorithms according to their expected seek time.
- What property of disks can we use to make the insertion, deletion, and access to the
  lists of requests faster?

For I/O systems, and disk, in particular, it is worthwhile to compromise

For most OS features, we are very concerned about efficiency.

Disks are slow devices relative to CPUs.