Pontificia Universidad Javeriana de Cali

Facultad de Ingeniería y Ciencias

Ingeniería en Sistemas y Computación

Disk Scheduling

Sistemas Operativos

Assignment III

Autor:
DAVID ALONSO BAYONA TIMANA
Código: 8968801

Profesor:
Jefferson Amado Peña Torres

Cali, Valle del Cauca
Noviembre 2025

Índice

1. 1. Disk Scheduling Problem

Disk requests come into the disk driver for cylinders 10, 22, 20, 2, 40, 6, and 38, in that order. A seek takes 6 msec per cylinder. How much seek time is needed for:

(a) First-come, first served

(b) Closest cylinder next

(c) Elevator algorithm (initially moving upward)

In all cases, the arm is initially at cylinder 20.

Given:

- Request queue: 10, 22, 20, 2, 40, 6, 38
- Starting position: cylinder 20
- Seek time: 6 msec per cylinder

1.1. (a) First-Come, First-Served (FCFS)

Order served: 10, 22, 20, 2, 40, 6, 38
Head movement:

$$\begin{aligned}
20 \to 10 &= |20 - 10| = 10 \text{ cylinders}\\
10 \to 22 &= |10 - 22| = 12 \text{ cylinders}\\
22 \to 20 &= |22 - 20| = 2 \text{ cylinders}\\
20 \to 2 &= |20 - 2| = 18 \text{ cylinders}\\
2 \to 40 &= |2 - 40| = 38 \text{ cylinders}\\
40 \to 6 &= |40 - 6| = 34 \text{ cylinders}\\
6 \to 38 &= |6 - 38| = 32 \text{ cylinders}\\
\text{Total cylinders} &= 10 + 12 + 2 + 18 + 38 + 34 + 32 = 146 \text{ cylinders}
\end{aligned}$$

Seek time = 146×6 msec = 876 msec

1.2. (b) Closest Cylinder Next (SSTF)

- At 20: {10, 22, 2, 40, 6, 38} Closest is 22. Move: $20 \to 22 = 2$ cylinders
- At 22: {10, 2, 40, 6, 38} Closest is 10. Move: $22 \to 10 = 12$ cylinders
- At 10: {2, 40, 6, 38} Closest is 6. Move: $10 \to 6 = 4$ cylinders
- At 6: {2, 40, 38} Closest is 2. Move: $6 \to 2 = 4$ cylinders
- At 2: {40, 38} Closest is 38. Move: $2 \to 38 = 36$ cylinders
- At 38: {40} Move: $38 \to 40 = 2$ cylinders

Total cylinders moved: $2 + 12 + 4 + 4 + 36 + 2 = 60$ cylinders
Seek time = 60×6 msec = 360 msec

1.3. (c) Elevator Algorithm (SCAN, initially moving upward)

- Arm starts at cylinder 20 (request 20 served immediately)
- Remaining queue: 10, 22, 2, 40, 6, 38

Upward sweep (from 20 to highest needed):

- Path: $20 \rightarrow 22 \rightarrow 38 \rightarrow 40$
- Movement: $20 \rightarrow 22 = 2$, $22 \rightarrow 38 = 16$, $38 \rightarrow 40 = 2$
- Total upward movement $= 2 + 16 + 2 = 20$ cylinders

Downward sweep (from 40 to lowest needed):

- Remaining requests: $\{10, 2, 6\}$
- Path: $40 \rightarrow 10 \rightarrow 6 \rightarrow 2$
- Movement: $40 \rightarrow 10 = 30$, $10 \rightarrow 6 = 4$, $6 \rightarrow 2 = 4$
- Total downward movement $= 30 + 4 + 4 = 38$ cylinders

Total cylinders moved $= 20 + 38 = 58$ cylinders
Seek time $= 58 \times 6$ msec $= 348$ msec
Final order served: 20 (at start), 22, 38, 40, 10, 6, 2

Comparison

- First-Come, First-Served (FCFS) = 146 cylinders
- Closest Cylinder Next (SSTF) = 60 cylinders
- Elevator Algorithm (SCAN, initially moving upward) = 38 cylinders

2. 2. Disk Scheduling Problem

Suppose that a disk drive has 5,000 cylinders, numbered 0 to 4,999. The drive is currently serving a request at cylinder 2,150, and the previous request was at cylinder 1,805. The queue of pending requests, in FIFO order, is: 2,069; 1,212; 2,296; 2,800; 544; 1,618; 356; 1,523; 4,965; 3,681.

Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests for each of the following disk-scheduling algorithms?

- Current position: 2,150
- Previous request: 1,805
- Queue: 2,069; 1,212; 2,296; 2,800; 544; 1,618; 356; 1,523; 4,965; 3,681

2.1. (a) FCFS (First-Come, First-Served)

We serve the requests in the exact order given, starting from cylinder 2,150.

Order served: 2,069 → 1,212 → 2,296 → 2,800 → 544 → 1,618 → 356 → 1,523 → 4,965 → 3,681

Head movement calculation:

$$\begin{aligned}
2,150 \rightarrow 2,069 &= |2,150 - 2,069| = 81 \\
2,069 \rightarrow 1,212 &= |2,069 - 1,212| = 857 \\
1,212 \rightarrow 2,296 &= |1,212 - 2,296| = 1,084 \\
2,296 \rightarrow 2,800 &= |2,296 - 2,800| = 504 \\
2,800 \rightarrow 544 &= |2,800 - 544| = 2,256 \\
544 \rightarrow 1,618 &= |544 - 1,618| = 1,074 \\
1,618 \rightarrow 356 &= |1,618 - 356| = 1,262 \\
356 \rightarrow 1,523 &= |356 - 1,523| = 1,167 \\
1,523 \rightarrow 4,965 &= |1,523 - 4,965| = 3,442 \\
4,965 \rightarrow 3,681 &= |4,965 - 3,681| = 1,284
\end{aligned}$$

Total distance:

$$81 + 857 + 1,084 + 504 + 2,256 + 1,074 + 1,262 + 1,167 + 3,442 + 1,284 = 13,011$$

FCFS total distance = 13,011 cylinders

2.2. (b) SCAN Algorithm

Start at 2,150, moving upward (based on previous request at 1,805). Serve all requests in upward direction until the highest cylinder (4,999), then reverse direction and serve all requests in downward direction until the lowest cylinder (0).

Step 1: Upward sweep (≥ 2,150)

Requests in upward direction: 2,296, 2,800, 4,965, 3,681
Sorted ascending: 2,296, 2,800, 3,681, 4,965
Upward path: 2,150 → 2,296 → 2,800 → 3,681 → 4,965 → 4,999 (end)

Movement:

$$\begin{aligned}
&2,150 \to 2,296 = 146\\
&2,296 \to 2,800 = 504\\
&2,800 \to 3,681 = 881\\
&3,681 \to 4,965 = 1,284\\
&4,965 \to 4,999 = 34
\end{aligned}$$

Total upward movement $= 146 + 504 + 881 + 1,284 + 34 = 2,849$

Step 2: Downward sweep (from 4,999 to 0)

Remaining requests: {2,069, 1,212, 544, 1,618, 356, 1,523}
Sorted descending: 2,069, 1,618, 1,523, 1,212, 544, 356
Downward path: 4,999 → 2,069 → 1,618 → 1,523 → 1,212 → 544 → 356 → 0
Movement:

$$\begin{aligned}
&4,999 \to 2,069 = 2,930\\
&2,069 \to 1,618 = 451\\
&1,618 \to 1,523 = 95\\
&1,523 \to 1,212 = 311\\
&1,212 \to 544 = 668\\
&544 \to 356 = 188\\
&356 \to 0 = 356
\end{aligned}$$

Total downward movement:

$$2,930 + 451 + 95 + 311 + 668 + 188 + 356 = 4,999$$

Step 3: Total distance

Total = Upward movement + Downward movement $= 2,849 + 4,999 = 7,848$ cylinders
SCAN total distance = 7,848 cylinders

2.3. (c) C-SCAN Algorithm

Start at 2,150, moving upward. Serve all requests in upward direction until the highest cylinder (4,999), jump to cylinder 0 (no servicing during return), then continue serving requests in upward direction from 0.

Step 1: First upward sweep (≥ 2,150)

Requests in upward direction: 2,296, 2,800, 4,965, 3,681
Sorted ascending: 2,296, 2,800, 3,681, 4,965
Upward path: 2,150 → 2,296 → 2,800 → 3,681 → 4,965 → 4,999
Movement:

$$\begin{aligned}
&2,150 \to 2,296 = 146\\
&2,296 \to 2,800 = 504\\
&2,800 \to 3,681 = 881\\
&3,681 \to 4,965 = 1,284\\
&4,965 \to 4,999 = 34
\end{aligned}$$

Total upward movement (first part) $= 146 + 504 + 881 + 1,284 + 34 = 2,849$

Step 2: Jump from 4,999 to 0

Movement: $4,999 \to 0 = 4,999$ cylinders (no servicing)

Step 3: Second upward sweep (from 0)

Remaining requests: {2,069, 1,212, 544, 1,618, 356, 1,523}
Sorted ascending: 356, 544, 1,212, 1,523, 1,618, 2,069
Upward path: 0 → 356 → 544 → 1,212 → 1,523 → 1,618 → 2,069
Movement:

$$\begin{aligned}
&0 \to 356 = 356\\
&356 \to 544 = 188\\
&544 \to 1,212 = 668\\
&1,212 \to 1,523 = 311\\
&1,523 \to 1,618 = 95\\
&1,618 \to 2,069 = 451
\end{aligned}$$

Total upward movement (second part):

$$356 + 188 + 668 + 311 + 95 + 451 = 2,069$$

Step 4: Total distance

Total = First upward + Jump to 0 + Second upward = $2,849 + 4,999 + 2,069 = 9,917$ cylinders

C-SCAN total distance = 9,917 cylinders

3. 3. Disk Scheduling problem

A slight modification of the elevator algorithm for scheduling disk requests is to always scan in the same direction. In what respect is this modified algorithm better than the elevator algorithm?

Uniform Wait Times and Better Fairness

The standard elevator algorithm (SCAN) can cause **starvation** for requests that arrive just behind the disk head's current position.
When the head moves in one direction, requests that arrive on the other side of the head must wait for the head to travel to the end, reverse direction, and come back.

The modified always scan in the same direction algorithm (which is essentially C-SCAN) provides:

- **More uniform wait times** - All requests experience approximately the same maximum waiting time
- **No location-based discrimination** - Requests aren't penalized based on their position relative to the head's current location
- **Predictable maximum service time** - The worst-case wait is bounded by one full sweep plus the jump back

As noted in Section 11.2.3 of the textbook:

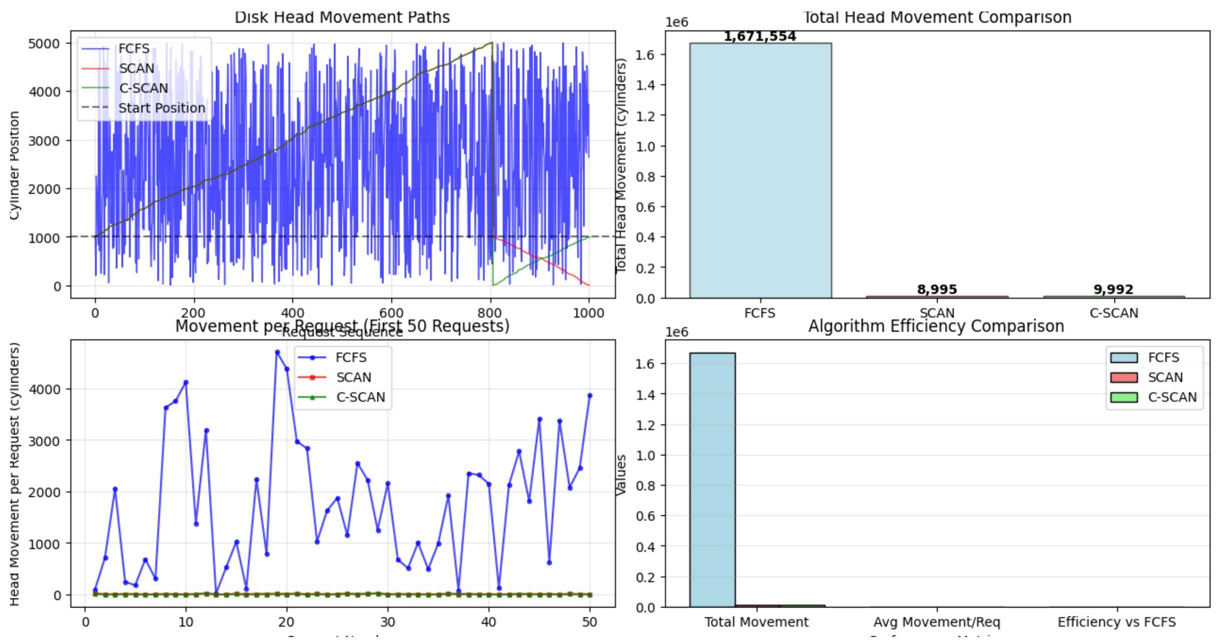
> "C-SCAN scheduling is a variant of SCAN designed to provide a more uniform wait time... The heaviest density of requests is at the other end of the disk. These requests have also waited the longest, so why not go there first? That is the idea of C-SCAN."

This makes C-SCAN particularly suitable for systems where **fairness** and **consistent response times** are more important than absolute minimal seek distance.

Summary of Advantages

Standard SCAN (Elevator)	**C-SCAN (Modified)**
Can cause longer waits for requests just missed by the head	More uniform wait times for all requests
Wait time depends on request location relative to head	Wait time is more consistent regardless of location
Better for minimizing total seek distance	Better for fairness and predictable performance
May starve requests in certain positions	No starvation - all requests served within bounded time

4. Data visualization



This comprehensive visualization clearly demonstrates the performance characteristics of three disk scheduling algorithms.

The FCFS algorithm shows **crazy head movement**, it's the blue line moving up and down. It has the highest total displacement.

While SCAN provides an efficient elevator-like scanning approach, C-SCAN offers **more uniform service times** by consistently moving in one direction and jumping back to the start, making it particularly suitable for **time-sensitive applications**.

The performance metrics confirm that organized scanning strategies dramatically enhance **disk I/O efficiency** and **system responsiveness** compared to the simple but inefficient FCFS approach.