SCAN vs FCFS

COMPARING SCAN AND FCFS ALGORITHMS AS THE HARD DRIVE'S REQUEST SERVING ALGORITHM

Ali Khaleqi Yekta | Implementation of Database Systems | November 2023

Introduction to SCAN (Elevator) and FCFS Disk Scheduling Algorithms

In the realm of disk management and optimization, two fundamental disk scheduling algorithms play a pivotal role: SCAN, also known as the Elevator algorithm, and FCFS, which stands for First-Come-First-Serve. These algorithms are essential components of operating systems and are tasked with efficiently organizing and servicing disk requests. In this comprehensive comparison, we delve into the intricacies of these two algorithms to understand their operation, advantages, and drawbacks.

UNDERSTANDING THE SCAN DISK SCHEDULING ALGORITHM (ELEVATOR ALGORITHM)

The SCAN Disk Scheduling Algorithm, aptly named the Elevator algorithm, employs a unique approach to handling disk requests. This algorithm allows the disk arm to traverse in both directions, demonstrating its metaphorical resemblance to an elevator moving up and down a building. It commences its operation by initiating the disk arm at one end of the disk, subsequently servicing all incoming requests until it traverses the entire length of the disk. Upon reaching the other end, the algorithm seamlessly changes the direction of the head position and proceeds to fulfill further pending requests until it reaches the opposite end once again.

UNVEILING THE FCFS DISK SCHEDULING ALGORITHM (FIRST-COME-FIRST-SERVE)

FCFS, an acronym for First-Come-First-Serve, is an uncomplicated yet fundamental disk scheduling algorithm. It adheres to a straightforward philosophy: the requests are handled in the precise order they are queued in the disk's request queue. Essentially, FCFS mimics the simplicity of a first-come-first-served system, where the process that initially requests access to the processor is the first to receive allocation. This algorithm operates on the principle of a First-In-First-Out (FIFO) queue, ensuring that requests are addressed sequentially, as they appear in the queue.

The Benchmark

Running the attached Kotlin project, which is the comparison software simulating the algorithms, and testing a manual round with some pre-defined inputs gives us the following result:

```
-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-= Disk Scheduling Simulator =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-

1> Elevator & FCFS Algorithm [Manual Input]

2> Elevator & FCFS Algorithm [Random Input]

3> Quit

=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=

1

-=-=-=-=-> Disk State

Current Location: 8000

Count of IO Requests: 6

-=-=-=-=-> Request's Cylinder & First Time Available

Request 1's Cylinder: 8000

Request 1's First Time Available: 0

Added:

X1: 8000.0, T1: 0.0

Request 2's Cylinder: 24000

Request 2's First Time Available: 0

Added:

X2: 24000.0, T2: 0.0

Request 3's Cylinder: 56000

Request 3's First Time Available: 0

Added:

X3: 56000.0, T3: 0.0
```

```
Request 4's Cylinder: 16000

Request 4's First Time Available: 10

Added:

X4: 16000.0, T4: 10.0

Request 5's Cylinder: 64000

Request 5's First Time Available: 20

Added:

X5: 64000.0, T5: 20.0

Request 6's Cylinder: 40000

Request 6's First Time Available: 30

Added:

X6: 40000.0, T6: 30.0

Entrance:

8000.00                          0.00

24000.00                         0.00

56000.00                         0.00

16000.00                        10.00

64000.00                        20.00

40000.00                        30.00

Elevator:

8000.00                          4.30

24000.00                        13.60

56000.00                        26.90

64000.00                        34.20

40000.00                        45.50

16000.00                        56.80

FCFS:

8000.00                          4.30

24000.00                        13.60

56000.00                        26.90

16000.00                        42.20

64000.00                        59.50

40000.00                        70.80
```

SUMMARY AND ANALYSIS

The benchmark results indicate the performance of two disk scheduling algorithms: Elevator (SCAN) and FCFS (First-Come-First-Serve) in servicing a series of IO requests. The initial disk head position was at 8000, and a total of 6 requests were processed.

Elevator Algorithm

The Elevator algorithm exhibits the following key characteristics:

- It starts from the initial head position (8000) and moves in both directions.
- Requests are serviced efficiently, resulting in relatively low seek times.
- The total seek time is 56.80 units.

FCFS Algorithm

The FCFS algorithm, on the other hand, follows a simple first-come-first-served approach:

- It processes requests in the order they are received, without optimizing for proximity.
- As a result, it incurs higher seek times compared to the Elevator algorithm.
- The total seek time in this case is 70.80 units.

In this scenario, the Elevator algorithm outperforms FCFS in terms of seek time optimization. It efficiently services the requests by moving in both directions, which leads to a lower total seek time. However, it's important to note that the choice of the optimal disk scheduling algorithm may vary depending on specific use cases and requirements.

This benchmark comparison provides valuable insights into the behavior of these algorithms in a real-world disk scheduling scenario. It can assist in making informed decisions regarding disk scheduling strategies for specific systems and workloads.

Analysis of Random Input Mode (1000 Requests)

To further assess the performance of the Elevator (SCAN) and FCFS (First-Come-First-Serve) disk scheduling algorithms, we conducted a simulation with a random input mode, involving a total of 1000 requests. The average seek times for each algorithm in this extensive test were as follows:

- Elevator's Average Seek Time: **2655.28 units**
- FCFS's Average Seek Time: **3013.21 units**

In this scenario, the Elevator algorithm continues to exhibit efficiency in optimizing seek times, with an average seek time of **2655.28 units**. This result reinforces its capability to adapt to a wide range of random input patterns by moving in both directions and effectively minimizing seek times. On the other hand, the FCFS algorithm, while still straightforward and reliable, displays a slightly higher average seek time of **3013.21 units**. This indicates that in random input situations, the FCFS algorithm may not always be the most efficient choice, as it processes requests solely based on their order of arrival, potentially leading to higher seek times when compared to the Elevator algorithm.

These findings underscore the importance of selecting the appropriate disk scheduling algorithm based on the specific characteristics of the workload and access patterns. While FCFS remains a straightforward and predictable choice, the Elevator algorithm's adaptability and ability to minimize seek times make it a valuable option in scenarios where random access patterns are prevalent or unpredictable.