I/O SCHEDULER

ABSTRACT

Group Members

Aarohi Patel(121006)

Nidhi Mehta(121029)

Rajan Mehta(121043)

Vraj Shah(121063)

Problem Statement

To extend the no-op scheduler to implement the Shortest seek time first (SSTF) algorithm.

Goals

- To minimize time wasted by hard disk seeks.
- To prioritize a certain processes' I/O requests.
- To give a share of the disk bandwidth to each running process.
- To guarantee that certain requests will be issued before a particular deadline.

Description

I/O scheduler:

It is the subsystem of a kernel. It manages requests from a block device. It needs to be fair and give a share of the disk bandwidth to each running process. The I/O scheduler divides the resource of disk I/O among the pending block I/O requests in the system. It does this through the merging and sorting of pending requests in the request queue.

No-op IO scheduler:

The no-op scheduler is one kind of scheduler in Linux. It inserts all incoming IO requests into a simple first in first out (FIFO) queue and implements request merging.

A mechanical devices like hard drive spends a lot of time in disk seeking, finding the right position on a disk plate to perform reading or writing. It does that by moving a mechanical head that has a limited speed. Although that speed seems quite high, flash devices don't require this, and are therefore faster by design. That's why a NOOP scheduler is superior to most schedulers for flash devices.

We will implement SSTF in No-op I/O scheduler.

Why SSTF? :

First Come First Serve (FCFS) serves the request coming first. But it does not provide the fastest service. It is simple to implement. The average head movement in the algorithm is too high. Shortest Seek Time Next (SSTF) selects the request with minimum seek time from the current head position. It gives substantial improvement in comparison to FCFS.

Requests are serviced based on their distance from disk head. Requests are maintained in a queue, current disk head position is identified and the algorithm iterates through the queue to find the request that is nearest to the disk head. Head movement in this algorithm is less as compared to no-op scheduler.