CMPSCI 377 Operating Systems **Fall 2005**

Lecture 16: November 29

Lecturer: Emery Berger *Scribe: Sam Farrington*

Today:

In this class we learned about scheduling algorithms and their many advantages and disadvantages.

16.1 Reducing IO Time

IO Time = Seek Time + Rotational Delay + Transfer Time

Rotational delay is already very low.

Transfer time can be reduced.

Latency can be reduced.

Ways to reduce latency:

- Defragment a fragmented drive
- Build smaller disks
- Build faster disks
- Increase sector size
- change the disk layout
- *Improve disk scheduling*

16.2 Why Fragmentation Occurs

Fragmentation occurs because the file system tries to layout data contiguously. As the disk fills up there will be scattered empty chunks. When the disk reaches 50 - 80% capacity these chunks become the only space left to write data to. Some file systems can avoid fragmentation by using a process which is equivalent to garbage collection on the disk.

16.3 Disk scheduling algorithms

The goal of a disk scheduling algorithm is to order disk accesses to reduce the distance the disk head must travel and the total number of disk seeks.

16.3.1 FCFS

This is similar to the other first come first serve algorithms we have used so far. It accesses blocks on the disk in the order that they are requested. As with other FCFS algorithms it is not very efficient.

16.3.2 Shortest Seek Time First

Shortest Seek Time First is a greedy algorithm similar to Shortest Job First. It always moves the disk head to the next closest disk block. Unlike SJF, SSTF is not an optimal scheduling algorithm.

16.3.3 Scan

Scan is a real disk scheduling algorithm. It uses an an algorithm simililar to that of an elevator. The disk head starts in the middle and moves up or down picking up all the requests in that direction as it goes. Scan travels all the way from the center to either edge of the disk on every sweep. This can make the algorithm very slow, however, the slowness tends to not be significant on a disk with uniformly distributed random data.

16.3.4 Scan Look

Scan Look is just like scan but it only goes to the highest or lowest requested block. This cuts down the overhead of using scan.

16.3.5 Circular Scan

Circular Scan sweeps the head across disk from one edge to the other and then resets the head back to the beginning each time. C-Scan can be expensive but usually it is more fair than Scan or Scan Look.

16.3.6 C-Scan Look

C-Scan Look works the same as C-Scan but the disk heads only go between the highest and lowest requested blocks.

16.4 Read Ahead

Read ahead is a great way to increase throughput. However, it is not needed in an software because it is done in hardware by the disk controller.

16.5 RAID

RAID - Redundant Array of Inexpensive Drives

Raid can increase throughput and reliability by spreading data across multiple disks and adding error checking or by duplicating data.

16.5.1 RAID 0

Is not real RAID it just makes multiple disks look like one big disk.

16.5.2 RAID 1

Each disk is mirrored on another disk. This has a very large overhead but is very reliable. Write times don't change but read times can be cut in half.

16.5.3 RAID 2

RAID 2 adds error checking. Every 10 disks requires 4 parity disks. The access times are the same as RAID 1.

16.5.4 RAID 4

RAID 4 uses data striping. Data is spread across multiple disks. This Improves read performance but because the parity disks are not striped it hurts read performance.

16.5.5 RAID 5

RAID 5 striped data and parity information across multiple disks. This removes the bottleneck created by RAID 4.