

CS162 – Section # 10, 04/08/2003

Rodrigo Fonseca

Announcements

Proj. 3 Initial Design due on Tuesday, 4/15 (1 week from today!)

I/O – Chapters 12, 13, 14

Important Concepts for an I/O operation:

Overhead: CPU time to initiate I/O
Latency: time to initiate one byte operation
Bandwidth: rate of transfer, once initiated

Disk performance:

1. Seek time: time to position head on right cylinder (order of 10 msec)
2. Rotational Latency: wait until sector is under head (7000rpm -> 116rps -> 8msec)
3. Transfer time: time to transfer the bytes (4 Mb/sec)

1 and 2 are Latency, 3 is bandwidth

Key: minimize seek and rotational latency

- Data placement
- Request scheduling
- Data organization
- Buffering

Disk Management

Compare the following file allocation schemes. Briefly, you should mention

- Time for sequential access
- Time for random access
- Fragmentation (internal/external)
- File expansion

Contiguous allocation, linked files, indexed files, Multilevel indexed

- Contiguous allocation
 - +fast sequential access
 - +easy random access
 - fragmentation (external)
 - hard to grow files

- Linked Files
 - +Easy to grow

- +Free space managed by linked list of blocks also
- One seek per block!
- Unreliable

Indexed Files: maintain index block
 +easy to grow up to max_size
 +random access good
 - Lots of seeks for sequential access

Multilevel indexed
 Header has 13 pointers (inode)
 10 pointers to data blocks
 11th is pointer to an index block
 12th is pointer to a doubly indirect block
 13th is pointer to a triply indirect block

- +Can accommodate large files, and small files are cheap
- +Files can grow
- Large files have to read lots of index blocks
- Many seeks

DISK ARM SCHEDULING:

Say we have pending I/O requests at the following tracks:

12 31 7 40

Also assume the disk has 128 tracks (numbered 0-127). *Say that the disk head starts out at track 15 (THIS IS DIFFERENT FROM WHAT I HAD IN SECTION 101).* What is the order and seek time (measured in number of tracks seeked) for each of the following disk scheduling algorithms?

FIFO:

Fair to requesters, but long seeks potentially

15 12 31 7 40
 (3) (19) (24) (33) = 79 seek time

SSTF:

Shortest seek time first. Small seeks, but may lead to starvation

15 12 7 31 40
 (3) (5) (19) (9) = 36 seek time

SCAN: (Start the disk head at track 15, going up):

Elevator algorithm: no starvation, similar to SSTF, but is biased towards pages in the middle of the disk.

15 31 40 12 7
 (16) (9) (28) (5) = 58 seek time

CSCAN: (Start the disk head at track 15, going up):

Circular SCAN: only goes in one direction, fairer than SCAN

15 31 40 7 12
 (16) (9) (33) (5) = 63 seek time

Sample question: Performance degradation due to thrashing:

Suppose that a program has 5 pages, which are accessed in a circular fashion.

1. What is the best fault rate that you can achieve with this reference pattern, given a memory of size 4?

Using OPT,

1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 1 2 3 4 5 ...

1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5	...	
*	*	*	*	*				*				*				*				*				*				*			
1	1	1	1	1				1				1			2				2				2								
	2	2	2	2				2				3			3				3				3								
		3	3	3				4				4			4				4				4								
			4	5				5				5			5				5				1								

So, on the limit, we will have 1 page fault every 4 references, or 25% page fault rate. It is interesting to note that in this particular reference pattern, LRU will be the worst case, and MRU will be equivalent to OPT.

2. Suppose that each page fault costs 1ms (this is the *total* time to service a page fault), and that each regular access to memory costs 100ns. Determine the slowdown factor between the situation in which you have 4 frames of memory, and the one in which you have 5 frames of memory.

Let us not consider the first 5 references, when both caches would generate misses. This is equivalent, for this reference stream, to consider the limit situation.

We can calculate the effective memory reference time as

$$e = h*th + (1-h)*tm$$

For the case of 4 pages of memory, the average reference time is

$$e_4 = 0.75 * 100ns + 0.25 * 1ms = 75ns + 250,000ns = 250,075ns$$

For the case of 5 pages of memory, the average reference time is

$$e_5 = 1 * 100ns + 0 * 1ms = 100 ns$$

So, the slowdown is $250,075 \div 100 \sim \mathbf{2500.75 !!!}$

The program is going to run 2500 times slower!

3. What would the hit ratio have to be to have a slowdown of 10 %?

$$110ns = h*100ns + (1-h)*10^6ns$$

$$100h - 10^6h$$

$$h = (110 - 10^6) / (100 - 10^6) \sim 99.999 \%$$

$$(90\% \rightarrow 0.9*100 + 0.1*1000000 = 100090ns/100ns \sim 1000)$$

$$99\% \rightarrow 0.99*100 + 0.01*1000000 = 10099ns/100ns \sim 100$$

$$99.9\% \rightarrow 0.999*100 + 0.001*1000000 = 1099.9ns/100ns \sim 11)$$