CMSC611
Homework
Fall 2008

Patrick Trinkle

*Collaboration with Mike Corbin.

Problem 7.12

a) Assume that DMA IO can take place simultaneously with CPU cache hits. Also assume that the operating system can guarantee that there will be no stale-data problem in the cache due to IO. The sector size is 1KB. Assume the cache miss rate is 5%. On average, what percentage of the bus is used for each cache write policy?

I am considering instruction reads as cache reads in this set of problems; although I had considered ignoring them and just focusing on the data cache.

General Information Related to the Problem:
-Instruction Percentages (gcc):

load	25.1%
store	13.2%
other	61.7%

-Loads/Stores take 2 cycles, all other take 1 cycle.
-All instructions require 1 read (from the instruction cache); load has 1 additional read (the load), and store has 1 additional write (the store).
-The CPI based on the above information: (these determine how often we read/write on average, etc).

(0.251*2)+(0.132)+(0.617) = 1.251 reads/instruction

(0.251*2)+(0.132*2)+(0.617) = 1.383 accesses/instruction (CPI, because cycles happens to correspond to number of accesses in this problem).

(0.251*2)+(0.132*2)+(0.617*1) = 1.383 CPI

(0.251*0)+(0.132*1)+(0.617*0) = 0.132 writes/instruction

-How often are we reading/writing?

1.251 (RPI)/1.383 (API) = 0.9045 (90.45%) of all accesses are reads

0.132 (WPI)/1.383 (API) = 0.0954 (9.54%) of all accesses are writes

*I realize this adds to 99.99%, but ignoring that extra bit.

-Write-back Cache
(Although I disagree that you can have a dirty block in your cache if the hit was a miss, the problem description explicitly states this case.)

Since all we are looking for is the bandwidth usage, I consider only the cases requiring the bus (which are misses—because if you hit, then you have it already).

Miss rate: 5%
Dirty block rate: 30% (therefore 70% are clean)
Dirty blocks take 31 cycles; clean take 23 cycles
If you have to read any word in a block, you must read the whole block (see problem description for cache reads).

Two cases to consider, block is dirty and block is clean.
Clean block miss: (0.05*0.70) = 0.035 (3.5%)
Dirty block miss: (0.05*0.30) = 0.015 (1.5%)

On a clean miss, you only need to read in the block, on a dirty miss you write out and then read back in the block (8 words). Therefore the average words transferred over the bus in these cases is:
(0.035*4)+(0.015*8) = 0.26
But there is a latency associated with each, so we must average this:
(0.035*23)+(0.015*31) = 1.27
The capability of the bus is 1 word/cycle, but on average we require 1.27 cycles, and need to transfer 0.26 words; therefore utilizing: 0.26/1.27 = 0.2047 (**20.5%**) of the bandwidth.

-Write-through Cache
Above it was determined that 90.45% of accesses are reads, and we always need to read in a block (on a miss). It was also determined that 9.54% of accesses are writes. When we write a word, it is a single word write, unless it was a miss. If it is a miss we have to read in the block, and then write the new value.
Miss rate: 5%
Write time: 16 cycles
Read time: 23 cycles

Miss cases:
(0.05*0.9045) = 0.045 (4.5%) read
(0.05*0.0954) = 0.00477 (0.48%) writes (note, needs to read, then write)
Hit cases:
(0.95*0.9045) = 0.8593 (85.93%) read
(0.95*0.0954) = 0.09063 (9.06%) writes
Calculate the average word transfers:
(0.045*4)+(0.00477*5)+(0.09063*1) = 0.29448 words
Associated latency:
(0.045*23)+(0.00477*(16+23))+(0.09063*16) = 2.67111 cycles
Bus can only do 1 word/cycle at best, on average we utilize it:
0.29448 (words)/2.67111 (cycles) = 0.1102 (**11.02%**)

b) Start with the same assumptions as in part (a). If the bus can be loaded up to 80% of capacity without suffering sever performance penalties, how much memory bandwidth is available for IO for each cache write policy? The cache miss rate is still 5%.

We know the bandwidth usage from before, so now our maximum is 80%. So, I'm interpreting the question as follows: Given the 80%, and the usage for each cache write policy, how much is left for other IO?

-Write-back
(0.80)-(0.205) = 0.595 (**59.5%**) available
-Write-through
(0.80)-(0.1102) = 0.6898 (**68.98%**) available

c) Start with the same assumptions as in part (a). Assume that a disk sector read takes 1000 clock cycles to initiate a read, 100,000 clock cycles to find the data on the disk, and 1000 clock cycles for the DMA to transfer the data to memory. How many disk reads can occur per million instructions executed for each write policy? How does this change if the cache miss rate is cut in half?

Disk sector read complete: 1,000 + 100,000 + 1,000 = 102,000 cycles
I guess it is 1,000 cycles to DMA, because the disk read should read 1,024 words (assuming 1 word = 1 byte (and not 2 bytes, like in intel and other systems).

Information Calculated above:
1.383 memory accesses / instruction (this includes reading each instruction from instruction cache)
2.67111 cycles / access with write-through cache
1.27 cycles / access with write-back cache

-Write-through
How many cycles (average) is 1 million instructions:
1.383 * 2.67111 * 1,000,000 = 3,694,145.13 cycles
Therefore 3,694,145.13 / 102,000 = **36.217 disk reads per million inst**
Theoretically it can read more if the data sectors to be read are sequential.
How many if the cache miss rate is cut in half:
Just redid the math from earlier with miss rate of 2.5%:
cycles / access: 2.101
1.383 * 2.101 * 1,000,000 = 2,905,683 cycles
Therefore 2,905,683 / 102,000 = **28.487 disk reads per million inst**

-Write-back
How many cycles (average) is 1 million instructions:
1.383 * 1.27 * 1,000,000 = 1,756,410 cycles
Therefore 1,756,410 / 102,000 = **17.219 disk reads per million inst**
How many if the cache miss rate is cut in half:

Just redid the math from earlier with miss rate of 2.5%:

cycles / access: 0.635

1.383 * 0.635 * 1,000,000 = 878,205 cycles

Therefore 878,205 / 102,1000 = **8.61 disk reads per million inst**

d) Start with the same assumptions as in part (c). Now you can have any number of disks. Assuming ideal scheduling of disk accesses, what is the maximum number of sector reads that can occur per million instructions executed?

Assuming normal miss rate of 5%.

Given good scheduling and infinite disks, then the only latency is the time to transfer the data. This is true, beyond the first read, because all other disk reads can be scheduled, data located, etc, while the first read is chugging along—as long as there is DMA bandwidth available.

Okay, so first read takes x cycles, how much can we get out beyond that with the remaining cycles and bandwidth?

Consider that since there is more than one disk you can really send out more than one request simultaneously, and then you run the benefit of not really having to wait for anything and you can just scrape up any available bandwidth. Therefore the latency becomes the 2,000 cycles required to transmit the request and transmit the data—this is assuming the controller uses this bandwidth, which was the assumption in part c).

-Write-through

How much bandwidth is available? 68.98%

How many cycles worth of bandwidth is that?

3,694,145.13 cycles * (0.6898) = 2,548,221.31 cycles

Therefore 2,548,221.31 / 2,000 = **1274.11 disk reads**

-Write-back

How much bandwidth is available? 59.5%

How many cycles worth of bandwidth is that?

1,756,410 cycles * (0.595) = 1,045,063.95 cycles

Therefore 1,756,410 / 2,000 = **878.205 disk reads**