Hayden Miedema and Douglas Money

CIS 451

Professor Kurmas

Lab 10: Cache (part 1)

1. Use sim-cache to determine the miss-rates of an 8KB, direct-mapped cache with the following block sizes: 8 bytes, 16 bytes, 32 bytes, and 64 bytes. To do so, use commands that look like this:

~kurmasz/public/Simplescalar/bin/sim-cache -cache:dl1 dl1:*line*:*block*:1:l -redir:prog /dev/null -redir:sim output\_*block* a.out

Where *block* ranges from 8 to 64, and *line* is set such that product of *block* times *line* is 8192.

Hints for running sim-cache:

* + Notice that the end of the cache configuration parameter is the number 1 followed by the letter l (as in "lru").
  + sim-cache is in ~kurmasz/public/CS451/bin
  + It's not difficult to make a script to automatically runs sim-cache with varying block sizes and present the results. The line below gives an example of how to perform arithmetic in a bash script:
  + let num\_lines=8192/$i

After you have run sim-cache for each block size, grep each output file (output\_8, output\_16, etc.) for the line "dl1.miss\_rate". List the miss rate for each block size tested.

|  |  |
| --- | --- |
| Block Size | dl1.miss\_rate |
| 8 Bytes | .1257 |
| 16 Bytes | .0629 |
| 32 Bytes | .03145 |
| 64 Bytes | .0158 |

1. Based on your observations, determine a formula for the miss rate in terms of *block\_size*.
   * 1/block\_size = miss rate
2. Explain your results (i.e., what is happening in the cache during each memory access to produce the results you observed).
   * When the system is entering the first value of the array into the cache it also fills the rest of the block up with the next few values of the array. This way, those next few values will not be a cache miss. The larger the block size, the more of the array can be preloaded. The first value will always be a miss, though, when setting up the cache.
3. Now, write a C program for which the miss rate is considerably higher for a 16 byte block than for an 8 byte block. (The easiest way to do this is to find two array locations that conflict with a 16-byte block, but not with an 8-byte block. If you do this, you will see the cache with the 8-byte blocks have a nearly 0% miss rate while the cache with the 16-byte blocks has nearly a 100% miss rate.) List your source code, all cache parameters used, and the resulting hit rates. Hint: You need not loop through the entire array. Instead, find two addresses that collide in the cache. Remove the inner loop and mke NUM\_LOOPS 1000000.

|  |  |
| --- | --- |
| Block Size | dl1.miss\_rate |
| 8 bytes | 0.0008 |
| 16 bytes | 0.9985 |

for (outer\_loop = 0; outer\_loop < NUM\_LOOPS; outer\_loop++)

{

solution \*= array[0];

solution \*= array[8191];

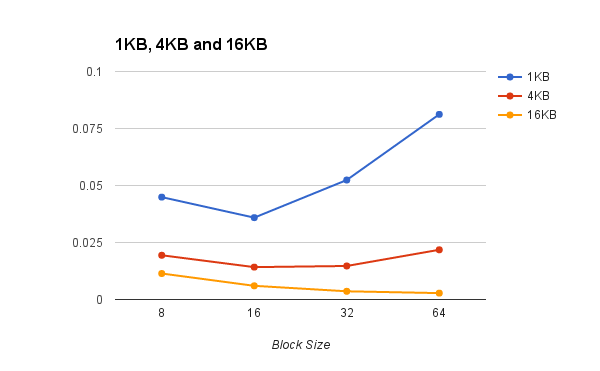
}

return solution;

}

1. Determine the optimal block size for qsort given a 1KB, 4KB, and 16KB cache. Present your results using a graph with block size on the x-axis and the miss rate on the y-axis. Please generate one graph with three lines: One each for 1KB, 4KB, and 16KB. Valid block sizes are powers of 2 from 8 to 64. Your graph should have a form similar to [Figure 8.18](http://www.cis.gvsu.edu/~kurmasz/Teaching/Courses/F16/CS451/Assignments/Cache/effectsOfBlockSize.jpg) in Harris and Harris (2nd edition).

|  |  |  |
| --- | --- | --- |
| Cache Size | Block Size | Miss Rate |
| 1KB | 8 | .0449 |
| 1KB | 16 | .0359 |
| 1KB | 32 | .0524 |
| 1KB | 64 | .0812 |
| 4KB | 8 | .0194 |
| 4KB | 16 | .0142 |
| 4KB | 32 | .0147 |
| 4KB | 64 | .0218 |
| 16KB | 8 | .0114 |
| 16KB | 16 | .0060 |
| 16KB | 32 | .0036 |
| 16KB | 64 | .0028 |



So, optimal block sizes are as follows:

1KB = 16 bytes block size

4KB = 16 byte block size

16KB = 64 byte block size