Cache Simulation Assignment

November 25, 2007

The assignment is due December 11, 2007.

Note: You may work as a group of two on this assignment. Each group should e-mail one report to the instructor by the due date.

The Dinero Cache Simulator

In this assignment, you will experiment with the design of a cache system for a processor using the Dinero IV cache simulator. Dinero simulates a wide variety of cache configurations, where the various cache parameters (e.g., cache size, block size, associativity level) can be specified on the command line. Dinero reads in a trace file comprising a sequence of memory address references generated by a MIPS CPU simulator running a real program compiled using a version of gcc designed to generate MIPS code. These traces specify whether or not the memory reference is an instruction fetch, a data read (*lw*), or a data write (*sw*). For each memory reference, Dinero simulates the behavior of the type of cache specified, generating hits and misses as appropriate. At the end of the simulation, it produces a set of statistics summarizing the performance of the cache, including the total number of memory references of each type, the percentage of cache misses in each case, and the total amount of memory traffic generated for each memory-reference type, both in absolute terms and as a percentage of the total amount of memory traffic generated by the CPU.

Dinero IV can be downloaded from the course web site as a binary for the Solaris operating system that can be executed on the machine *io*. The source code can be downloaded via anonymous FTP from Wisconsin. Memory reference traces corresponding to a million instructions for three example programs: cc1 (a C compiler), spice (a circuit simulator), and tex (a document formatter) are also available on the course web site.

To run DineroIV, copy the executable and the corresponding trace files from the course web site, or from the following directory

`/export/home/pop-ece/kandasamy/cache_simulator`

into your home directory on *io*. Then, change `dineroIV` as an executable file as

```
[io] % chmod 755 dineroIV
```

You can then execute the cache simulator as

```
[io] % ./dineroIV [cache options] < tracefile
```

You can get command-line help as follows:

```
[io] % ./dineroIV -help
```

A detailed manual page for Dinero is also available on the class web page. Note that the trace files are in "traditional din" format. So, use the command-line option "-informat d" for this assignment. For example, to simulate the behavior of an L1 instruction cache (or I-cache) of size 16K and a cache-block size of 128 bytes, and an L1 data cache (or D-cache) having the same configuration using the *spice.din* trace file, you will provide the following command-line options to Dinero.

```
[io] % ./dineroIV  -l1-isize 16K -l1-dsize 16K -l1-ibsize 128 -l1-dbsize 128
                   -informat d < spice.din
```

Problems

1. **(25 points)** Use Dinero to evaluate different I-cache and D-cache sizes, as well as cache-block sizes for the three different workloads: cc1.din, spice.din, and tex.din. For each workload, simulate I-cache and D-cache sizes of 2K, 4k, 8k and 16K, where the I-cache and D-cache sizes are identical for each simulation run. Use cache-block sizes of 16, 32, 64, 128, and 256 bytes for each I-cache and D-cache combination, for a total of 20 combinations per workload. Use default values provided by Dinero for the other cache parameters. Altogether, this is 3 workloads $\times$ 20 cache combinations for a total of 60 simulation runs.

For each workload, plot the results of your simulations showing the total number of misses and total memory traffic as a function of the block size for different cache sizes. Which configuration incurs the minimum number of misses and the least memory traffic? Provide a concise explanation of why this configuration appears to be the best.

2. **(25 points)** For each of the three workloads, use the cache simulator to evaluate the impact of associativity on cache performance. Use a block size of 32 byte and a unified cache size of 16K in all your simulations. Simulate the cache for set associativities of 1 (direct-mapped), 2, 4, and 8. Use the LRU and random replacement policies for your simulations, for a total of 8 different combinations. Use a write-back cache for all your simulations.

For each workload, plot the results of your simulations showing the number of instruction and data read/write misses, and the total memory traffic for the different cache configurations. Which configuration shows the least number of read/write misses and the least memory traffic? Provide a concise explanation of why this configuration appears to be the best.