

CS 210 Laboratory on Cache Design

Overview

In this lab you will experiment with parameter tuning in the design of a simple cache system, with the goal of minimizing the average memory access time (AMAT) within some reasonable cost constraints. For this you will use a cache simulator written as an application file within the Matlab program, which runs on CSA.

Experiments

The Matlab program is a full-featured environment for performing mathematical reasoning and displaying the results as graphs or geometrical objects; it also has a built-in programming language with an API for designing GUI interfaces to the code. Our simulator runs as a program (an “M-file”) within Matlab which opens a GUI control panel to the cache simulator. The control panel allows you to set various parameters (cache size, block size, etc.) and also choose certain designs (write-back or write-through, split cache, multi-level cache). For multi-level caches, the parameters for each level of the cache can be set separately. The user then chooses a trace file containing the actual memory references made by a running program, and the simulation will record the behavior of the simulated cache system and draw a graph. The user can select the axes for the graph (e.g., block size vs. hit rate, or cache size vs. average memory access time), draw multiple plots on one graph, insert titles and other information, and print out the graph.

To run the simulator with Matlab, you first need to copy two files:

```
csa% cp /home/fac1/matta/public/cachesim.m .
```

```
csa% cp /home/fac1/matta/public/cachesim.fig .
```

(that’s a dot, indicating the current directory, at the end of the line). Then simply execute the Matlab program:

```
csa% matlab
```

This will bring up the Matlab window, divided into several panes. On the right is the Command Window with a command prompt “>>”. Type the following to see the (limited) help information about the cache simulator:

```
>> more on
```

```
>> help cachesim
```

Don’t read this thoroughly yet, instead run the simulator by typing:

```
>> cachesim
```

This will bring up the simulator control panel. Here is a brief description of the menus:

- **File Menu**

- **Reset Parameters:** This will reset the values of the parameters in the control panel to empty, or the previously-loaded values;
- **Load Parameters:** This will allow you to load a parameter set from a previously-saved parameter set;
- **Save Parameters/Save Parameters As:** This allows you to save the current state of the parameters in a text file to be reloaded later; and
- **Exit.**
- **Plot**
 - **Xaxis, Yaxis:** This allows you to set the parameters along the X and Y axes of the plot;
 - **Clear Plot:** Clear the plot in the current graphing window;
 - **Replot Runs:** Runs of the simulator are recorded in the Status Messages pane; by selecting a run from this pane and then choosing replot, you can redraw the plot from that run.
- **Run:** The only important choice here is Run Simulator, which is better selected using the button near the bottom of the control panel.

Now you should go back and read the help page on cachesim by typing “help cachesim” as before. Read through this carefully, skipping the material on “mapping” (which relates to set-associate caches).

You run experiments by choosing values for the parameters, choosing a trace file and a Limit (the number of lines of the trace file to read---usually you want to look at no more than a couple of thousand), and then hitting Run Simulator. After a while (depending on the Limit value), a graph window will pop open with your plot. You can clear this plot, or run another experiment to place another plot on the same window.

The following should be noted carefully for this lab:

- You should choose only Direct-Mapped caches, and you should use a Hit cost of 10 and a Miss cost of 100 (these are ns, but you should not type in “ns”).
- To load the trace files, you should hit “Browse,” select the text in the Filter box, and type in `/home/fac1/mat t a/publ i c/` and push Filter. Then select **espresso.short.din** from the Files list; finally, type 1000 into the Limit box.
- When entering Block size and cache size parameters, **note that to provide a range of values for the plot, one of these must have two identical values while the other one has a range of values;** for example, to plot the AMAT of caches of size 1MB where the block size changes from 8 bytes to 256 bytes, you would enter 1 MB in both boxes (“1MB to 1MB”) and then enter 8B and 256 B in the block size windows.
- When choosing a parameter to change as just described, reset the Plot->Xaxis value, or else you will get a strange plot.

The Experiment

You must find out which parameters give you the best AMAT for the espresso program. The parameters you will test out will be (1) cache size, and (2) block size. Assume we use a one-level split write-through cache in our design (make sure to select Split and Write through from the simulator window, and 1 for Level and Use levels). You should do this in several steps.

- First find a reasonable size for your cache. Run the simulator with some choice of block size (say, 8 bytes) through a range of cache sizes starting with the block size itself (say, from 16 bytes to 1MB). Observe the graph. It is telling you that the AMAT gets smaller as the cache gets bigger (why?). Of course, you have limited resources, and cannot increase the size of the cache unless it is justified by the reduction in AMAT. Pick a size of the cache where you seem to have reached the point of “diminishing returns,” i.e., making the cache bigger does not give you must improvement in the AMAT, i.e., where the slope seems to be leveling off. (Note: the points on the curve are powers of two, and you can figure out which powers of two by looking at the cache sizes on the X axis).
- Now, close or clear the graph window, and set the parameters so that the cache size is the size you just figured out as optimal. Then, try a range of block sizes (say, 8 bytes to 64 bytes), again in powers of two, and remembering to reset the Xaxis on the Plot menu. Can you explain the curve you see? Write down the optimal block size.
- Finally, go back and repeat from the beginning, to verify that the cache size picked in the first step is still optimal after you change the block size to the optimal one. You may need to iterate this process a couple of times to arrive at the design with the optimal parameters. When you have a stable set of parameters, go through the experiments once more and print out all the graphs to hand in.

What to Hand In

You must provide a written statement of your experiment, with the final values that you determined, what you observed, and, most importantly, your explanation of what you observed (e.g., why does the block size plot have that shape?). Hand this as the final problem of the Memory Homework.

Extra for experts:

After determining the optimal block size and cache size, investigate the use of split vs. unified cache, and the write policy. For this, change this one parameter, and see if there is an improvement at the point in the curve you are interested in. What is your final design for this one-level cache system?

Try playing around with a multilevel cache system, by setting the parameters for each level (specify which Level at the top), and then specifying the number of levels at the bottom with Use Levels. *Warning:* The plots show the effect of changing one parameter

(block size or cache size, but not both), which is given on the X axis. Thus, in a multi-level cache system, you can still *only change one block size or cache size in the whole system*; the rest should have constant values. To see the effect of changing two parameters, draw multiple plots on the same graph space.