MC48 Lab 4: Studying Cache Auto-Prefetch Effectiveness

Max Hailperin

Due: November 30, 1995

In this lab, you will work together with a partner to gain experience in how hypotheses about memory system performance can be tested using trace-driven cache simulation.

Because there is an odd number of students, exactly one group should be of one or three rather than of two. Please try to find your own partners; if you want to be the group of one or three (with two others), or if you can't find a partner, see me. Each group should submit a single jointly authored lab report.

It is important that you complete the Pre-Lab portion of this assignment before you start the In Lab portion.

Pre-Lab

Rather than waiting for the first access to a block to start fetching that block from main memory to the cache, it is possible to *prefetch* a block likely to be accessed soon. Our textbook mentions one form of prefetching, *compiler-directed prefetching*, in the "Recent Trends" material on page 521. In this lab we will study a different form of prefetching, *hardware prefetching* or *auto-prefetching*, in which the cache hardware automatically chooses when to prefetch a block rather than relying on "hints" from the compiler. Generally this involves a very simple rule such as prefetching the next block after the one currently being accessed.

Auto-prefetch can decrease the miss rate if the prefetched blocks are used, but may also increase the amount of traffic between the main memory and the cache if the blocks go unused. A prefetch strategy is highly *effective* if it causes a substantial decrease in the miss rate with only a minor increase in memory traffic; in the other scenarios, where the miss rate doesn't go down much and/or the traffic goes up a lot, we can term the prefetch strategy *ineffective*, *costly* or *counterproductive*.

The question you are to answer is this: what is the relationship between cache block size and total cache size and the effectiveness of auto-prefetch?

Before starting the lab, you should decide and write down the following two things:

1. What is your hypothesis regarding the answer to the above question? What is the rationale behind your hypothesis?

2. What is your experimental plan for testing your hypothesis? In particular, what values of block size and total cache size are you going to use? What values are you going to use for the other cache parameters that you hold constant? Which of the various auto-prefetch variants supported by the dineroIII simulator are you going to use? Which address trace(s) are you going to use? In order to answer these questions, you will need to read ahead into the In Lab section below.

In Lab

This lab needs to be done on one of the SGI Indy computers, not a NeXT, because the simulations would take too darn long on the NeXTs. (I'd also steer clear of the SGI R3000 Indigos for the same reason; the R4000 Indigo, peanuts, should be ok.) More precisely, the experimental measurements need to be done on an Indy; the analysis and writeup can be done on any machine.

You will use the dineroIII trace-driven cache simulator. A trace-driven cache simulator reads in a trace, that is, a record of the memory references from a particular execution (with each address tagged with whether it was a read, a write, or an instruction fetch). The simulator is parameterizable so that the same trace can be tried with a variety of different parameters for block size, associativity, cache size, prefetch strategy, etc. At the end of the simulation, a variety of statistics are produced, including the miss rate for the cache and the amount of memory traffic. DineroIII is a good quality, flexible simulator often used in such work.

We have available on-line traces from three benchmarks (TEX, gcc, and spice). Each of the traces contains about a million memory references. The traces reside in the directory `~max/MC48`, and have filenames `tex.din.gz`, `cc1.din.gz`, and `spice.din.gz` respectively. The `.gz` suffix indicates that they are compressed using the gzip program (to save disk space); to decompress them at the same time you run dineroIII, you can use the command

`gzcat ~/`*name*`.din.gz | dineroIII` *options*

where *name* is the name of the trace you want to use and *options* is a list of dineroIII parameters as specified in the accompanying manual page for dineroIII. Note that at a bare minimum you need to specify the `-b` option and either the `-u` option or the `-i` and `-d` options. The `-f` option will also be particularly relevant to your investigation.

You should observe the improvement prefetching makes in the miss rate, specifically, the so-called "demand" miss rate, i.e., the miss rates on those blocks actually requested by the CPU, rather than by the prefetching. You should also observe the increase in memory traffic caused by unnecessary prefetches. Further, it would be interesting to see whether instruction and data references are equally suited to auto-prefetching. (DineroIII prints out separate statistics in each category as well as totals.)

Post-Lab

Your lab report should not only clearly summarize the pre-lab and in lab portions of your work, it should also draw the two together by comparing your hypothesis with your observations. Write in clear technical English for an audience familiar with the material in our textbook but unfamiliar with this lab assignment. Be sure to present your experimental plan carefully enough that someone else could repeat your experiments and observe the same output numbers.