Instruction Prefetching Milestone Report

Sarah Allen, Brandon Amos
srallen, bamos

I. Major Changes

As we've been discussing in the private Piazza thread, using LLVM to add instruction prefetching is more difficult than our initial findings indicated because of the following issues:

- The prefetch intrinsic documentation[1] does not make the usage clear.
- It's not clear how to obtain and insert the instruction addresses into the prefetch intrinsic.

Our proposed change is therefore to implement prefetching in the simulator to work around these issues.

II. What you have accomplished so far

A. icache simulator

We first tried getting a project called `cache-pintools`[2] working. Unfortunately, it appears to be several years old and not maintained and was therefore incompatible with the PIN simulator. In our attempts to reconcile the incompatibilities, we found that the PIN simulator contained within it a cache simulator (`tools/Memory/icache.cpp`) which seemed reasonable for our purposes.

In our first attempts to get the benchmarks running on the cache simulator, we noticed that the hit rates for instructions were already quite good. To address this, we created a new benchmark (described below) and we altered the cache simulator to make the instruction cache much smaller.

Finally, we developed a concrete algorithm that we intend to use for the machine learning portion. More specifically, we determined exactly how to obtain training data efficiently and the exact techniques we intend to use, although these may be subject to change as the project progresses.

B. Benchmarks: Polybench

We have identified polybench[3] as a set of benchmarks to evaluate our optimizations. Polybench is widely used in compiler optimization research and provides self-contained files for each benchmark.

We have executed our instruction cache simulator on all of the benchmarks. Table I shows the icache miss ratio, showing that instruction cache optimizations may improve the performance of the benchmarks. Since prefetching will cause more cache misses, we plan to come up with another metric to show how well prefetching works.

We've also created a new benchmark that randomly calls the other polybench files in a loop. This creates about 10% cache misses, but we expected a higher value. We hope for our optimizations to also improve performance on this file.

We intend to either use cross validation on the individual benchmarks or to use the new randomized benchmark to generate training instances. The latter seems more promising in that the number of individual benchmarks is very limited, but we are unsure as to which approach will be better.

2mm	0.01
3mm	0.02
adi	9.71
atax	0.00
bicg	8.82
cholesky	0.08
correlation	0.12
covariance	0.02
doitgen	10.78
durbin	2.03
dynprog	6.92
fdtd-2d	6.72
fdtd-apml	9.84
floyd-warshall	13.63
gemm	0.02
gemver	0.01
gesummv	7.90
gramschmidt	0.03
jacobi-1d-imper	0.04
jacobi-2d-imper	10.02
lu	0.00
ludcmp	0.05
mvt	0.01
reg_detect	3.35
seidel-2d	11.36
symm	11.10
syr2k	10.60
syrk	0.00
trisolv	0.01
trmm	0.01

TABLE I. Polybench icache miss ratios

[1] http://llvm.org/docs/LangRef.html#llvm-prefetch-intrinsic
[2] http://people.csail.mit.edu/rabbah/download/cache-pintools
[3] http://www.cs.ucla.edu/~pouchet/software/polybench/

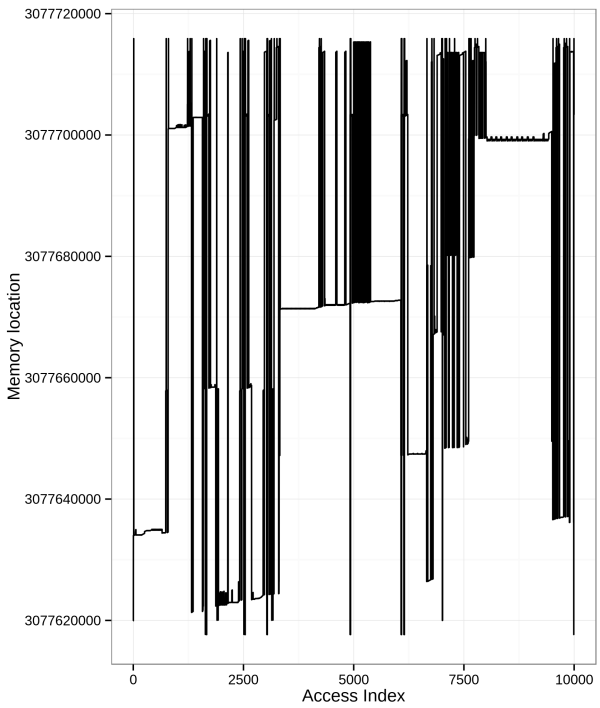

Fig. 1. First 10,000 memory references of floyd-warshall

C. Viewing Instruction Access Patterns

To better understand some of the benchmark access patterns, we have output the instruction access traces from our simulator. Figure 1 shows an example trace from the first 10,000 memory access in floyd-warshall, the worst performance benchmark.

III. Meeting Your Milestone

We did not meet our milestone goal, as implementing the instruction prefetching is not as easy as planned. We do, however, have our simulator up and running and benchmarks to test.

A. Surprises

We were surprised at the difficulty in implementing the instruction prefetching. While our initial reading of the LLVM documentation seemed to suggest that this could be implemented rather simply, this turns out not to be the case.

B. Revised Schedule

We are revising the schedule in that we will need to do a good portion of the implementation in the following two weeks rather than having it already done. This should not impact the overall project goals too much, as we intentionally chose an aggressive midpoint goal in anticipation of possible roadblocks.

C. Resources Needed

None, still working with open source software on the course virtual machine.