Fast and Accurate Modeling of Instruction Caches Using Instruction Reuse Profiles

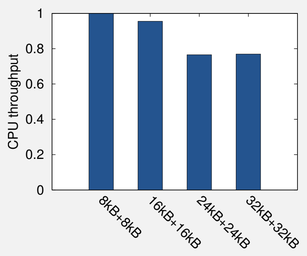
UPMARC
Uppsala Programming for Multicore Architectures Research Center

Muneeb Khan, Andreas Sembrant and Erik Hagersten

UPPSALA UNIVERSITET

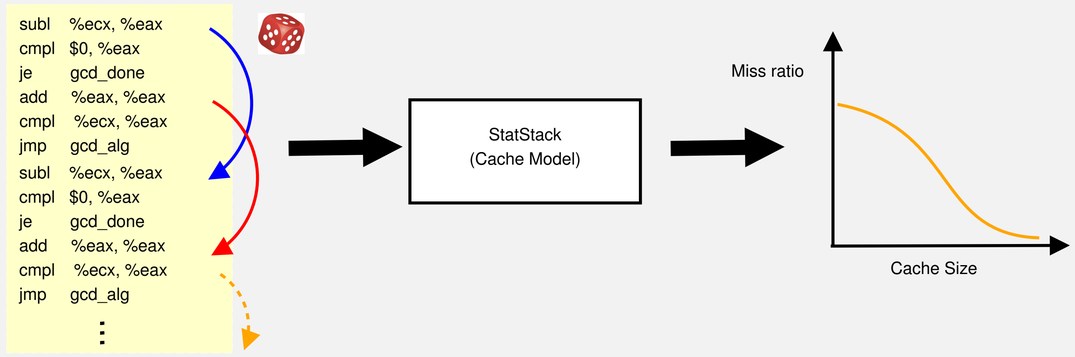
1-Objective

- Instruction cache behavior is important as it can significantly affect performance
- Simulation takes too long
- We need a faster method to model application instruction cache performance
- **Solution:** statistical cache modeling



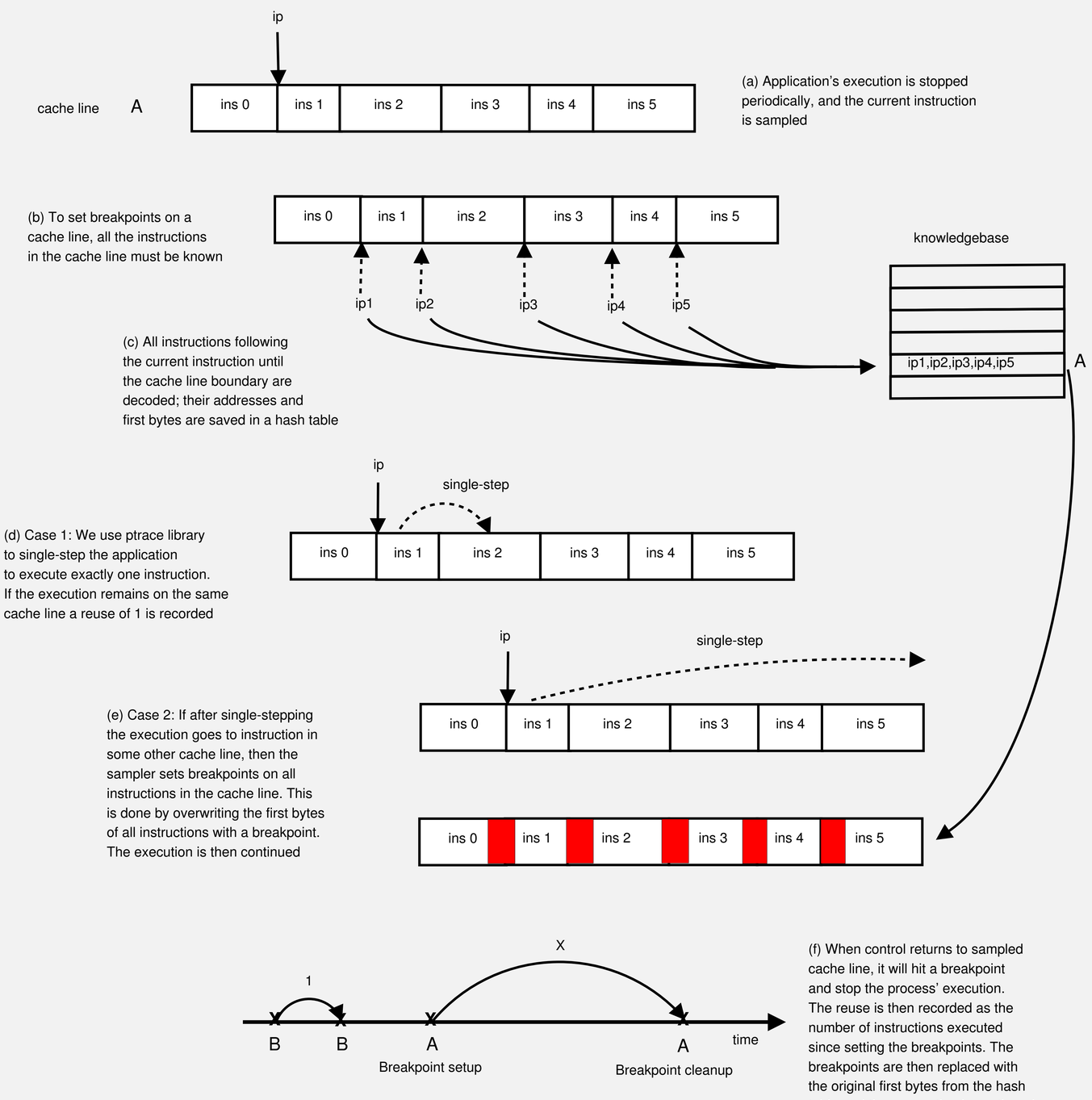
- CPU throughput degrades by more than 20% when two processes contend equally for the instruction cache on 2-way SMT Sandybridge

2-Method



- Instruction reuse distance from the dynamic instruction stream is **sampled** and fed to a Statistical cache model, which outputs Miss ratio for arbitrary cache sizes

3-Reuse Sampling Methodology



(a) Application's execution is stopped periodically, and the current instruction is sampled

(b) To set breakpoints on a cache line, all the instructions in the cache line must be known

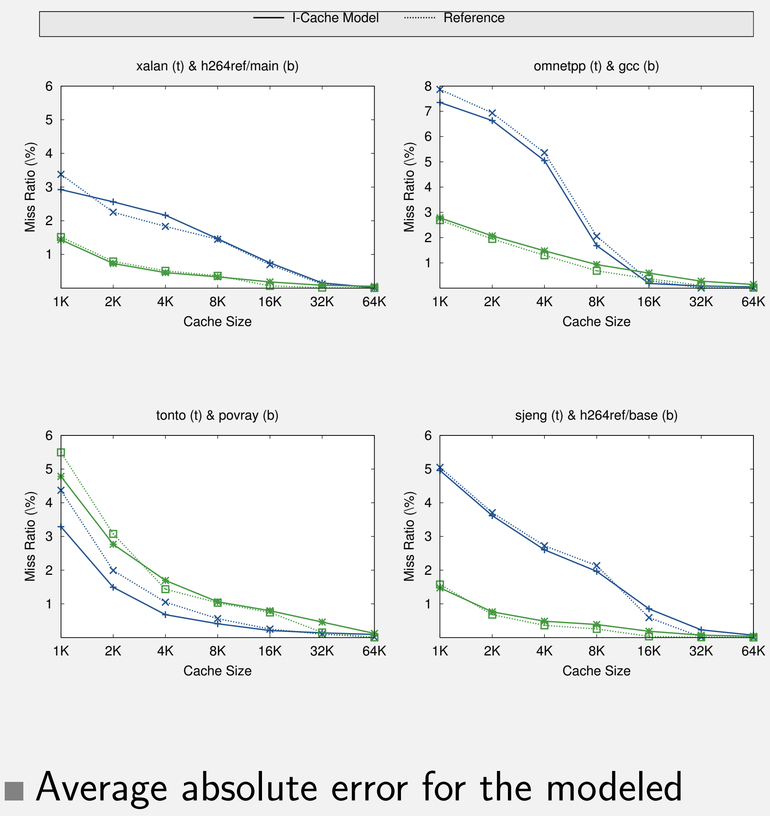
(c) All instructions following the current instruction until the cache line boundary are decoded; their addresses and first bytes are saved in a hash table

(d) Case 1: We use ptrace library to single-step the application to execute exactly one instruction. If the execution remains on the same cache line a reuse of 1 is recorded

(e) Case 2: If after single-stepping the execution goes to instruction in some other cache line, then the sampler sets breakpoints on all instructions in the cache line. This is done by overwriting the first bytes of all instructions with a breakpoint. The execution is then continued

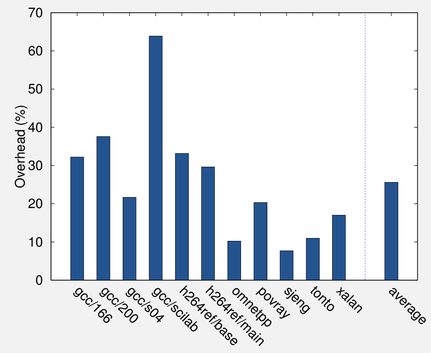
(f) When control returns to sampled cache line, it will hit a breakpoint and stop the process' execution. The reuse is then recorded as the number of instructions executed since setting the breakpoints. The breakpoints are then replaced with the original first bytes from the hash table and then execution is continued

4-Accuracy



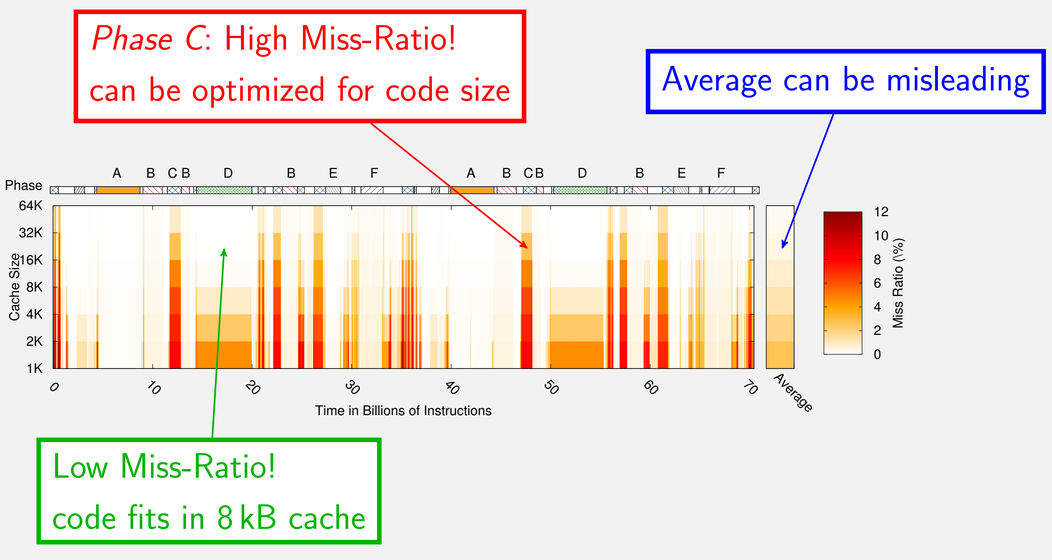
- Average absolute error for the modeled instruction cache miss ratio is 0.2%

5-Overhead



- Average overhead amounts to 24%, whereas simulation takes 1 week

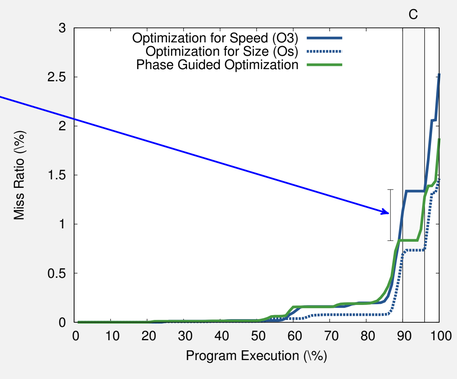
6-Phase-Guided Profiling



- Sampling in context of program phases gives us instruction cache performance over time
- The heatmap shows the instruction cache miss ratio for *gcc* (SPEC CPU2006 benchmark) over time

7-Phase-Guided Optimization

Miss-Ratio reduced from optimizing Phase C



- Compile phase C with -Os
- Compile the rest with -O3

Results in ∼50% lower miss ratio in phase C