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

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

- To set breakpoints on a cache line, all the instructions in the cache line must be known
- All instructions following the current instruction until the cache line boundary are executed, their addresses, and first bytes are saved in a hash table
- Single-steping the execution of an application, a breakpoint can be placed at an instruction.
- If the execution remains on the same cache line a reuse of 1 is recorded
- 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 execution 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

- Phase C: High Miss-Ratio! can be optimized for code size
- Average can be misleading

- Low Miss-Ratio! code fits in 8kB cache

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