

# Profile guided optimization using open source tools

Sebastian Pop and Aditya Kumar

SARC: Samsung Austin R&D Center

September 1, 2015

# Profile guided optimization (Introduction)

- ▶ Iterative approach to program optimization.
  - ▶ Compile  $\rightarrow$  Run + Collect Profile  $\rightarrow$  Process Profile data  $\rightarrow$  Compile with profile info.  $\rightarrow$  Run
- ▶ Intrusive approach (PGO)
- ▶ Non-intrusive approach (AutoFDO)

# Open source tools

- ▶ Compiler: gcc, llvm
- ▶ Gprof
- ▶ Linux Perf
- ▶ Create\_gcov
- ▶ Gooda

# PGO: Compiler instrumentation

- ▶ Compiler inserts probes in the code.
- ▶ Instrumented program collects profile as it executes.
- ▶ Gives very precise profile information.
- ▶ Good for benchmarking, analyzing small programs.
- ▶ gprof/gcov used to process sampled output and display.

# PGO: Compiler instrumentation (Useful Commands)

gprof:

- ▶ `gcc -fprofile-generate test.cpp`
- ▶ `gprof [exe] [gmon.out] > [outfile]`

gcov:

- ▶ `gcc -fprofile-arcs -ftest-coverage test.cpp`
- ▶ `gcov [options] test.cpp`

# AutoFDO: Non Intrusive

- ▶ Linux perf collects profiles (Sample based)
- ▶ No compiler instrumentation, debug-info required while running.
- ▶ Negligible overhead (1-2%) [Google]
- ▶ Run on different machines and at different time intervals.
- ▶ Good for system wide profiling.

# AutoFDO: Non Intrusive (Useful Commands)

- ▶ perf record/report/stat
- ▶ collecting profile: perf record -b [exe] [-o perf.data]
- ▶ report: perf report [perf.data]

## Results [Baptiste Wicht et al.]

PGO Kind	Avg. Run Time Overhead (%)	Performance Improvement (%)
Instrumented	16 (highest 53)	7 to 14
AutoFDO	1 to 2	5 to 10



# Challenges for non-intrusive AutoFDO

- ▶ Value based profiling not available.
- ▶ Only works on recent Intel machines.
- ▶ Tradeoff between accuracy of profile and overhead.
- ▶ Goal is to be as close to the instrumented profile.

# Challenges for non-intrusive AutoFDO...

- ▶ Calculating missing edge/basic-block frequencies based on Equivalence class.
- ▶ Calculating accurate profile of execution paths sharing single line of code based on DWARF discriminator [Baptiste Wicht et al.].

## create\_gcov

- ▶ Creates coverage file from perf.data to be used by the compiler.
- ▶ optimize with profile information: `gcc -fauto-profile=file.gcov -O2 test.cpp`
- ▶ Source: <https://github.com/google/autofdo>

# gooda

- ▶ gooda-analyzer
- ▶ gooda-visualizer
- ▶ Source: <https://github.com/David-Levinthal/gooda>
- ▶ Requires patching linux perf to link against libpfm4 [See: [gooda-analyzer/README](#)]
- ▶ Requires LBR (Available in Intel Sandy Bridge, Ivy Bridge, Westmere)

# Interesting Areas

Tools related:

- ▶ Extend AutoFDO infrastructure in gcc/llvm for non-branchy optimizations.
- ▶ Compiler cost model based on profile to enable demand driven optimization.
- ▶ Online optimizations driven by hardware performance monitoring [Schneider et al.].

Architecture specific:

- ▶ No AutoFDO tool for generating value profiles.
- ▶ Kernel support missing for ETM (ARM).

# References

- ▶ <http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/36575.pdf>
- ▶ Baptiste Wicht et al. “Hardware Counted Profile-Guided Optimization.”
- ▶ <http://www.burningcutlery.com/derek/docs/instant-profiling-CG013.pdf>
- ▶ <https://github.com/David-Levinthal/gooda>
- ▶ <https://github.com/google/autofdo>