# Feedback Directed Optimization in LLVM

Diego Novillo <dnovillo@google.com>

EuroLLVM 2013

# Feedback Directed Optimization

- Core enabler for peak performance at Google (7-10%)
- Cross-module optimization on top of FDO provides 10-30% (LIPO)
- Traditionally, FDO builds require three steps:
  - Instrumentation build
  - Profiling run
  - Optimization build
- Most benefits from FDO seen in
  - Inlining (particularly cross-module inlining)
  - Code layout
  - Register allocation
- FDO usability is poor
  - Build model is complex
  - Increased build times
  - Representative data sets are frequently hard to produce
  - Google's build environment mitigates some of this difficulty

# Design Principles

- Strive for simplified use model - Auto FDO
    - Ideally, no training runs
        - Requires external profile source
        - Use training runs at first
    - No instrumentation
        - Sample-based plus debug information for location information
        - Tolerate lossy/inaccurate profile data
    - Profile data converted to compiler digestable form
- Allow different sources of profile data
    - Leverage existing external tools
- Enable instrumentation, if needed
    - Peak performance may not be achievable, otherwise

# Initial Plan

1. Target auto and instrumented FDO
2. Profile reading module with support for multiple input formats
3. Connect profile reader to analysis
4. Make sure key transformations are using results from analysis