Review of

Auto FDO

Dehao Chen, David Xinliang Li, Tipp Moseley

What is the problem?

Compiler optimizations such as FDOs are the need of the hour in datacenter applications where even small deployments consume lots of hardware resources. The properties of data center applications make FDO adoption difficult though, and there needs to be an optimization process that's easy to maintain.

Summary

AutoFDO collects feedback from production workloads and applies optimizations at the next compile time. AutoFDO can be easily enabled by setting a few flags. Stale profiles can be tolerated for the typical amount of code change between releases. AutoFDO has increased adoption of FDO at Google by 8 times.

Key Insights

- AutoFDO is designed to tolerate source changes and therefore, a relatively imprecise profile from a real workload can act as a good input.
- A system like AutoFDO is easy to implement in the production environment

Strengths

- Since profiles are generated from runs in production, the optimization occurs on real-world input.
- Even 6 month old stale profiles result in 50% performance benefit, so AutoFDO can improve performance for even rapidly increasing code.

Weaknesses

- The sampling might have a bigger overhead in programs as the usage of systems like AutoFDO increases.
- The iterative AutoFDO design does not work well for all cases of optimizations.

Summary of Key Results

- An old profile can still produce up to 50% performance increase.
- AutoFDO has increased the number of FDO users by 8x and doubled the number of cycles spent in FDO-optimized binaries.

Open Questions

- How can further research work on improving profile quality? If it leads to better profiles, then it can improve the stability.