Quantifying Profiler Overhead

- Jonathan Madsen
- John Gouwar
- Matin Raayai Ardakani
- Lakhsono Adhianto
- David Boehme
- Holly Wilper

Summary

- Objectives
  - Overhead needs to be kept low for “always-on” profiling
  - Provide recommendations for overhead of various services
  - Get ahead of “feature X is faster than tool Y” complaints before they surface
  - Guidance for AI
  - Detect regressions
- Discussion Results
  - Need to educate that microscopic benchmarking of overhead is problematic
  - Presenting overhead in terms of percentage is tricky, time/measurement can be more useful
  - Two recommended approaches:
    1. Coarse end-to-end difference vs. baseline
    2. Sampling when profiler enabled
  - Consider (optional) report after data collection for estimated overheads
  - Detect inefficient collection situations and warn/disable
  - Suite of apps for quantifying overheads

Objectives

- User complaints about profiling overheads
- How to create reasonable expectations about overheads?
  - Profiling side effects: kernel serialization etc.
- How to deal with regressions/competition?
- Set of recommendations w.r.t. service for tool developers
- Benchmark suite for quantifying this data
- Needs for always-on profiling → need to keep overhead low

How to express/report overhead/costs?

- Caliper uses a microbenchmark and reports cost/event
- Jonathan: AI users do microscopic measurements and complain about overheads in Google microbenchmarks that are negligible in overall context. Need to educate users
- CUPTI: Use gromacs for benchmarking. AI still underexplored.
- Macro view: overhead for whole application
- Micro view: per-event (kernel launch, etc.) overhead
- Need both
- Need to quantify overheads per feature: When does profiling become too expensive? E.g. skip instrumenting small functions
- How does composition of profiling features impact overheads?
- Catalog of test cases (including pathological cases) helps tool developers
- Let AI agents “understand” profiling overhead
- Would a “pre-profiler” help? Tell users overheads for some profiling config on their application
- Profilers could detect inefficient situations (short kernels/functions etc.) and warn

Profiling the profiler

- Jonathan: “time”-like tool capturing all kinds of benchmark metrics (time, context switches, …)
- Keep time of profiling in the profiler? Adds more overhead
- Other profiling perturbations: Cache behavior, memory, initialization/finalization
- Everyone uses Linux perf for profiling the profiler
  - Use sampling profiler to inspect call-stacks exclusive to profiler