Argonne NATIONAL LABORATORY

THAPI: iprof

Keep it simple

Thomas Applencourt(apl@anl.gov)

July 31, 2026

Why yet another tool?

- Different Tool, different usage, different overhead
- iprof want to be dead-simple[1]
- iprof is more defined by what it doesn't do
- No call-stack, no 3d visualization, no binary instrumentation, etc
- We just Trace Heterogeneous API

[1] Because I'm not that smart, I want simple too

THAPI

- *https://github.com/argonne-lcf/THAPI*
- We focus on tracing API Layer.
- Main goal: Do not change the application: we trace, THEN we analysis.
- We give you a summary, and some high-level analysis
- Second goal: We dump all the argument

That it, and it's already a lot...

- Need to keep up with the standard, new version, fancy usage, etc

What do we trace

- At the APi layer: Cuda, OpenCL, Level Zero, Hip, MPI
- When programming model give us callback: ITT, OpenMP

On Aurora

- "module load thapi"[2]
- If not available on spack
  https://github.com/argonne-lcf/THAPI-spack/pulls

[2] And don't tell our sysadmin, but in my $HOME too for the latest and greatest, *thapi_devel_clean/build/ici/bin/iprof*

Simple Example

- ‘iprof ./a.out‘ -> Summary
- ‘iprof -l ./a.out‘ -> Perfeto view
- ‘iprof -t ./a.out‘ -> Pretty printing

Live Demo

Some technicality for the people who like this

- We use EfficiOS Technology (used to trace Linux kernel)
- LTTng for tracepoint generation
- Trace in a CTF Format
- Babeltrace2 For reading the trace

Other use because of CTF

- Because we dump all argument into disk
- We have many people using the trace for multiple purpose
- Validation Layer, simulation layer, etc etc

Some technicality for the people who like this

- Most of the the code is auto-generated
- We take the “official header” we parse them into some Intermediate Representation
- From this we generate all the tracing code

Conclusion: As simple as possible

- ‘module load thapi‘
- ‘iprof ./a.out‘ -> Summary
- ‘iprof -l ./a.out‘ -> Perfeto view
- ‘iprof -t ./a.out‘ -> Pretty printing