THAPI/Iprof: An Introduction

Thomas Applencourt, Brice Videau and many others (including future you?)

September 25, 2025

Table of Contents

1. Rant: You should profile your code
2. Ok, so now you’re convinced. Back to iprof
3. What kind of Analysis?
4. Conclusion

How to use

Module and Usage:

- `module load thapi`
- `mpirun -n 10000 -- iprof -- ./a.out`

Repo:

- `https://github.com/argonne-lcf/THAPI`
- `https://github.com/argonne-lcf/THAPI-spack`
- `https://docs.alcf.anl.gov/aurora/performance-tools/iprof/`

Rant: You should profile your code

HPC is the art of removing bottlenecks

- You should know your bottleneck. (MPI? GPU?)
- Optimizing GPU kernels is fun! But more or less useless[1]
- Just ask for more nodes, and scale more. We do HPC.

[1] It's obviously an exaggeration, but need to keep this entertaining.

You should have a ”performance model”

- You should know if you are slow or fast
- Meaning you should know how fast you can be
- ”I know I transfer N bytes through the NIC; the bandwidth is X, so it should take Y, but instead it takes Z...”

Everybody should do that

- No matter what programming model you use (oneCCL, MPI, Pytorch, SYCL, Kokkos)
- We are all scientists. If we're here, we should care about performance
- And performance is not an absolute concept, it's relative to the hardware characteristic.

Ok, so now you’re convinced.

Back to iprof

Why a Tracer?

We need to understand what's going on in order to solve bugs[2]

- Why my data-transfer doesn't overlap (H2D + D2H) ?!
- Why OpenMP Mapping take 10 min?!
- Why my SYCL queue in-order have so much submission overhead?!

Or more high level question:

- Am I GPU bound? MPI Bound? Data-transfer bound?
- What is my memory footprint?
- How does my scaling affect my ratio of MPI/GPU times?
- I need a timeline so I can spot any "bubbles" on my GPU execution

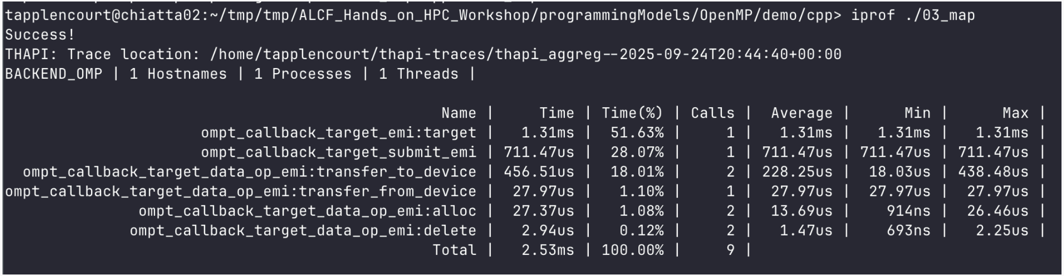
[2] Or to better use the features given us by vendors...

What are we Tracing?

- THAPI: Tracing Heterogeneous API
- We trace library calls (For Aurora: Level Zero, MPI, OpenMP) and analyze them
  - We are ”HPC” First: THAPI/iprof should be able to scale with a low overhead.
  - If you find any bugs, or limitation just slack / email me! Not perfect, but at least we try to improve [3]
- Based on lot a of amazing open-source tools (LTTng, Babeltrace2, Clang, Perfetto, ...)
- Colleen showed you some ”sanity check verification” using the OpenMP backend

[3] ”It is the time you have wasted for your rose that makes your rose so important”

Example openmp



```
tapplencourt@chiatta02:~/tmp/tmp/ALCF_Hands_on_HPC_Workshop/programmingModels/OpenMP/demo/cpp> iprof ./03_map
Success!
THAPI: Trace location: /home/tapplencourt/thapi-traces/thapi_aggreg--2025-09-24T20:44:40+00:00
BACKEND_OMP | 1 Hostnames | 1 Processes | 1 Threads |

                                                 Name |     Time | Time(%) | Calls |  Average |      Min |      Max |
                      ompt_callback_target_emi:target |   1.31ms |  51.63% |     1 |   1.31ms |   1.31ms |   1.31ms |
                      ompt_callback_target_submit_emi | 711.47us |  28.07% |     1 | 711.47us | 711.47us | 711.47us |
  ompt_callback_target_data_op_emi:transfer_to_device | 456.51us |  18.01% |     2 | 228.25us |  18.03us | 438.48us |
ompt_callback_target_data_op_emi:transfer_from_device |  27.97us |   1.10% |     1 |  27.97us |  27.97us |  27.97us |
               ompt_callback_target_data_op_emi:alloc |  27.37us |   1.08% |     2 |  13.69us |    914ns |  26.46us |
              ompt_callback_target_data_op_emi:delete |   2.94us |   0.12% |     2 |   1.47us |    693ns |   2.25us |
                                                Total |   2.53ms | 100.00% |     9 |
```

What kind of Analysis?

What kind of Analysis?

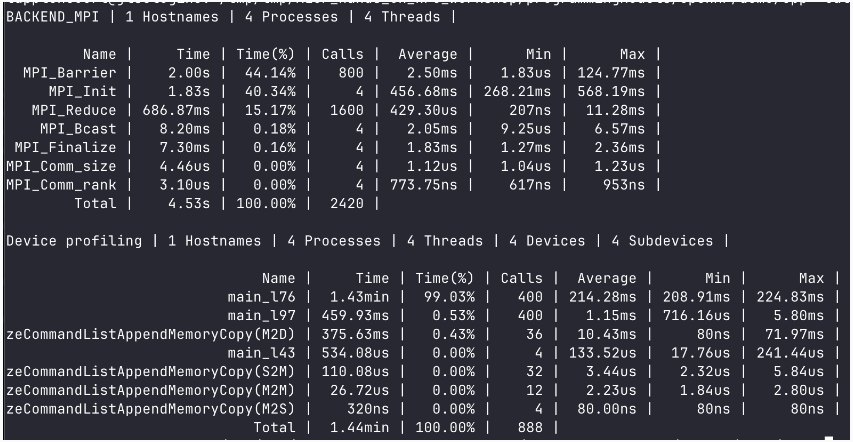
Summary

Summary

- Tally / Summary. Give you overview. You should start with that!
  - `mpirun -n 10000 -- iprof -- ./a.out`
  - This will tell you if you spend more time in MPI or on the GPU[4]

[4] Or not there, meaning on the CPU, in this case you should be ashamed. If I see anyone bounded by the Python interpreter they will be required to take a 10h FORTRAN course

Example of MPI



```
BACKEND_MPI | 1 Hostnames | 4 Processes | 4 Threads |

         Name |      Time | Time(%) | Calls |   Average |       Min |       Max |
  MPI_Barrier |     2.00s |  44.14% |   800 |    2.50ms |    1.83us |  124.77ms |
     MPI_Init |     1.83s |  40.34% |     4 |  456.68ms |  268.21ms |  568.19ms |
   MPI_Reduce |  686.87ms |  15.17% |  1600 |  429.30us |     207ns |   11.28ms |
    MPI_Bcast |    8.20ms |   0.18% |     4 |    2.05ms |    9.25us |    6.57ms |
 MPI_Finalize |    7.30ms |   0.16% |     4 |    1.83ms |    1.27ms |    2.36ms |
MPI_Comm_size |    4.46us |   0.00% |     4 |    1.12us |    1.04us |    1.23us |
MPI_Comm_rank |    3.10us |   0.00% |     4 |  773.75ns |     617ns |     953ns |
        Total |     4.53s | 100.00% |  2420 |

Device profiling | 1 Hostnames | 4 Processes | 4 Threads | 4 Devices | 4 Subdevices |

                                Name |     Time | Time(%) | Calls |  Average |      Min |      Max |
                            main_l76 |  1.43min |  99.03% |   400 | 214.28ms | 208.91ms | 224.83ms |
                            main_l97 | 459.93ms |   0.53% |   400 |   1.15ms | 716.16us |   5.80ms |
zeCommandListAppendMemoryCopy(M2D) | 375.63ms |   0.43% |    36 |  10.43ms |     80ns |  71.97ms |
                            main_l43 | 534.08us |   0.00% |     4 | 133.52us |  17.76us | 241.44us |
zeCommandListAppendMemoryCopy(S2M) | 110.08us |   0.00% |    32 |   3.44us |   2.32us |   5.84us |
zeCommandListAppendMemoryCopy(M2M) |  26.72us |   0.00% |    12 |   2.23us |   1.84us |   2.80us |
zeCommandListAppendMemoryCopy(M2S) |    320ns |   0.00% |     4 |  80.00ns |     80ns |     80ns |
                               Total |  1.44min | 100.00% |   888 |
```

Good, I'm GPU bound! Not lot of data-transfer...Talking about data-transfer

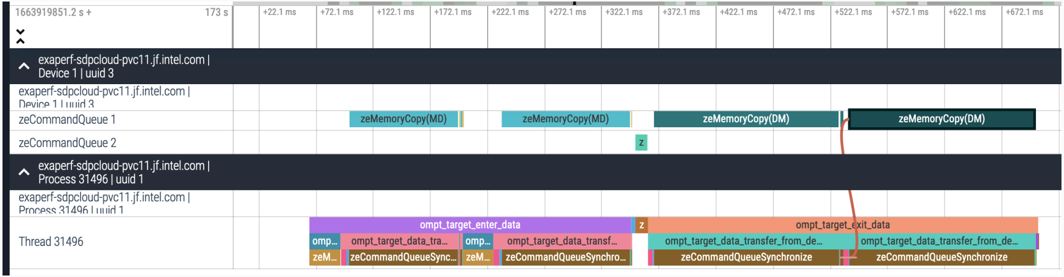
What is slow

- Bandwidth hierarchy: NIC « PCIe « ”Xe-Link” GPU « GPU HBM « GPU Cache
- Data-transfer are slow, especially PCIe.
- D == Device Memory, H == Host Memory (pinned), S == Shared Memory (managed), M == ”Mallocated” memory

What kind of Analysis?

Timeline

Perfetto Timeline: OpenMP on top of Level Zero



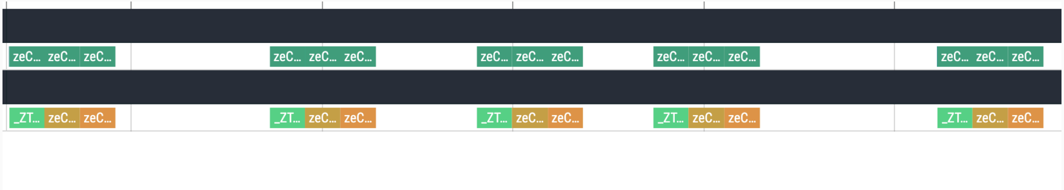
`iprof -l ./a.out` Then open the perfetto in `https://ui.perfetto.dev/`

- Ye showed you some example a few days ago

The timeline backend need a lot of love, if you are interested in visualization just contact me :)

Good and bad code: The bad

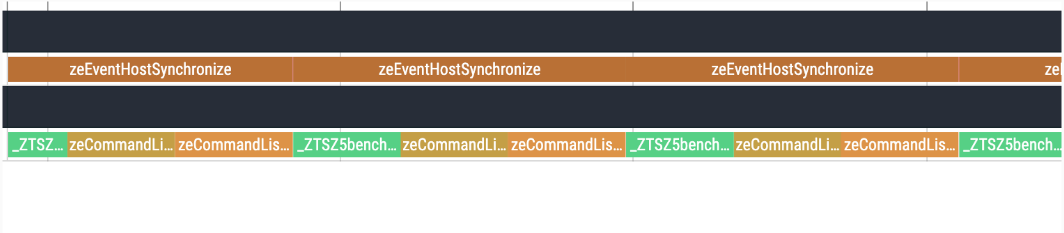
Top line: Host Code, Bottom: GPU Code.



Lot of gaps on the GPU -> Bad.

Good and bad code: The ”Good”

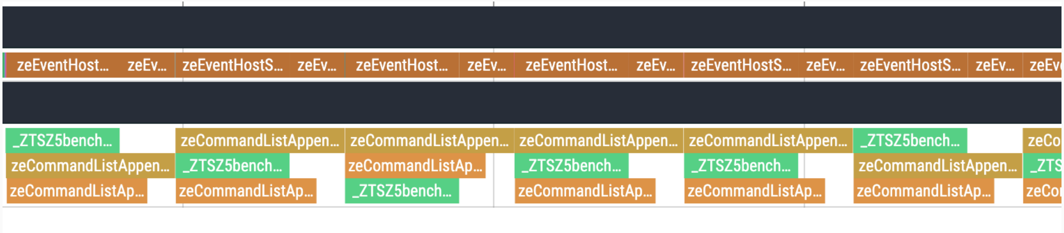
Top line: Host Code, Bottom: GPU Code.



No Gaps... Good?

Good and bad code: The Really Good!

Top line: Host Code, Bottom: GPU Code.



Concurrency! You should always overlap compute and data-transfer.

What kind of Analysis?

Trace

I know some people like to look behind the curtain

Hip on top of Level Zero

```
13:36:02.387547645 - x4204c4s2b0n0 - vpid: 146726, vtid: 146726
- lttng_ust_hip:hipMemset_entry: { dst: 0xff00fffffc4f0000, value: 0, sizeBytes: 12392 }
13:36:02.387550815 - x4204c4s2b0n0 - vpid: 146726, vtid: 146726
- lttng_ust_ze:zeCommandListAppendMemoryFill_entry:
{ hCommandList: 0x0000000004f2da68, ptr: 0xff00fffffc4f0000, pattern: 0x00007fff829294df,
  pattern_size: 1, size: 12392, hSignalEvent: 0x000000001e672818, numWaitEvents: 2,
  phWaitEvents: 0x000000001e673d00,
  pattern_vals: "\x00", phWaitEvents_vals: [ 0x000000001e670658, 0x000000001ed15bd8  ] }
13:36:02.387558470 - x4204c4s2b0n0 - vpid: 146726, vtid: 146726
- lttng_ust_ze:zeCommandListAppendMemoryFill_exit: { zeResult: ZE_RESULT_SUCCESS }
- lttng_ust_hip:hipMemset_exit: { hipError_t: hipSuccess }
[...]
```

```
iprof -t ./a.out
```

Overhead

- The more you ask for, the more overhead[5]
- Some analysis are local (tally of tally is a tally), so should be O(1) respectively of the number of hostname[6]
- For now we do only post-processing, so application performance is not impacted. POC for ”on-the-fly” is available upon request :)

[5] Duh

[6] Working on doing the same for timeline

What kind of Analysis?

What THAPI/iprof is not

- It’s not a full-blown performance analysis framework (Vtune, NSigh, HPC Toolkit, Tau)
- It’s not a line-level profiler [7]

[7] We give you Kernel Time. And we have sampling support of HW counter, but we stop here

Conclusion

Conclusion

- `module load thapi`
- `iprof ./a.out`: so you know where you spend time
- `iprof -t ./a.out`: so you can watch for gap
- `iprof ./a.out`: so you know why

Future work[8]

- Improve timeline visualization
- Improve processing time
- Add new backend (ITT, libfabric)
- Incorporate with other Tracer (python tracer for example)
- Handle multiple binaries

[8] Sound fun, no?