Tracing Heterogeneous APIs (CUDA, OpenCL, L0, OpenMP)

Thomas Applencourt   Brice Videau

Argonne National Laboratory

25th Oct 2022

Objective

Understand programming models implementation and usages.
Example:

- How programming models are implemented on top of each other?
  - How OpenMP nowait are implemented in LLVM?
- How applications are using programming models?
  - What is the maximum memory allocated by my program on the GPU?

Solution

- Trace as many programming models as possible
  - Trace should capture as much context as possible, and be lightweight as possible
- Develop tools to analyze traces

Programming-Model Centric Debugging / Tracing

```
18:56:59.677295870 - arc03 - vpid: 37040, vtid: 37040
   - lttng_ust_ze:zeKernelSetIndirectAccess_entry:
      { hKernel: 0x0000000002cd2b20, flags: [ ZE_KERNEL_INDIRECT_ACCESS_FLAG_DEVICE ] }
18:56:59.677296042 - arc03 - vpid: 37040, vtid: 37040
   - lttng_ust_ze:zeKernelSetIndirectAccess_exit:
      { zeResult: ZE_RESULT_SUCCESS }
```

- Flexible
  - Fine granularity, you can enable/disable individual events tracing,
  - Trace can be read programmatically (C, Python, Ruby),
  - We provide tools calibrated to our needs as starting-blocks.
- Low/Reasonable overhead

THAPI Consist in 2 bigs components

Open source at: https://github.com/argonne-lcf/THAPI

- The tracing of events
  - Use low level tracing: Linux Tracing Toolkit Next Generation (LTTng):
    - Well maintained and established (used in industry leading data-centers)
    - Binary format, about 0.2us overhead per tracepoint (in our case)
  - Tracepoints are generated from APIs' headers
- The parsing of the trace
  - Use Babeltrace2 library and tools (reference parser implementation of Common Trace Format)
  - Pretty Printer, Tally, Timeline/Flamegraph, . . .

Supported APIs

- OpenCL, Level Zero, Cuda Driver
- OMPT

THAPI Examples: `iprof -t ./a.out`

Wrapping the API entry points to be able to reconstruct the context.

```
> ./iprof -t ./a.out
  { thread_type: ompt_thread_initial, thread_data: 0x00007f5b0cf0ac48 }
ompt_callback_target:
  { kind: ompt_target, endpoint: ompt_scope_end, device_num: 0, task_data: 0x0000000000000000,
    target_id: 1, codeptr_ra: 0x00007f5b26fa47e0 }
[...]
ompt_callback_target_data_op_intel:
  { endpoint: ompt_scope_begin, target_id: 1, host_op_id: 7, optype: ompt_target_data_transfer_to_device,
    src_addr: 0x00007f5b20088280, src_device_num: -10, dest_addr: 0xffffc001ffd80000,
    dest_device_num: 0, bytes: 131072, codeptr_ra: 0x00007f5b26fa47e0 }
clEnqueueMemcpyINTEL_entry:
  { command_queue: 0x181a540, blocking: CL_FALSE,
    dst_ptr: 0xffffc001ffd80000, src_ptr: 0x00007f5b20088280, size: 64, num_events_in_wait_list: 0,
    event_wait_list: 0x0, event: 0x7ffc4ac01378, event_wait_list_vals: [] }
clEnqueueMemcpyINTEL_exit:
  { errcode_ret_val: CL_SUCCESS, event_val: 0x1dffb30 }
ompt_callback_target_data_op_intel:
  { endpoint: ompt_scope_end, target_id: 1, host_op_id: 7, optype: ompt_target_data_transfer_to_device,
    src_addr: 0x00007f5b20088280, src_device_num: -10, dest_addr: 0xffffc001ffd80000,
    dest_device_num: 0, bytes: 131072, codeptr_ra: 0x00007f5b26fa47e0 }
```

THAPI Examples: iprof

```
$iprof ./target_teams_distribute_parallel_do.out # Using Level0 backend
Trace location: /home/tapplencourt/lttng-traces/iprof-20210408-204629
BACKEND_OMP | 1 Hostnames | 1 Processes | 1 Threads |
       Name |   Time | Time(%) | Calls | Average |    Min |    Max |
ompt_target | 3.65ms | 100.00% |     1 |  3.65ms | 3.65ms | 3.65ms |
      Total | 3.65ms | 100.00% |     1 |

BACKEND_OMP_TARGET_OPERATIONS | 1 Hostnames | 1 Processes | 1 Threads |
                                 Name |     Time | Time(%) | Calls |  Average |      Min |      Max |
               ompt_target_data_alloc |   1.97ms |  54.19% |     4 | 491.63us |    847ns |   1.12ms |
  ompt_target_data_transfer_to_device |   1.26ms |  34.63% |     5 | 251.37us | 112.60us | 460.90us |
ompt_target_data_transfer_from_device | 250.76us |   6.91% |     1 | 250.76us | 250.76us | 250.76us |
             ompt_target_submit_intel | 155.04us |   4.27% |     1 | 155.04us | 155.04us | 155.04us |
[...]
                                Total |   3.63ms | 100.00% |    11 |

BACKEND_ZE | 1 Hostnames | 1 Processes | 1 Threads |
                               Name |     Time | Time(%) | Calls |  Average |      Min |      Max |
                     zeModuleCreate | 846.26ms |  96.89% |     1 | 846.26ms | 846.26ms | 846.26ms |
      zeCommandListAppendMemoryCopy |  10.73ms |   1.23% |    12 | 893.82us |  12.96us |   5.33ms |
[...]
                              Total | 873.46ms | 100.00% |   117 |

Device profiling | 1 Hostnames | 1 Processes | 1 Threads | 1 Devices |
                                  Name |     Time | Time(%) | Calls | Average |     Min |     Max |
                      zeMemoryCopy(DM) |  64.48us |   7.14% |     1 | 64.48us | 64.48us | 64.48us |
__omp_offloading_33_7d35e996_MAIN___l9 |  27.84us |   3.08% |     1 | 27.84us | 27.84us | 27.84us |
[...]
                                 Total | 902.72us | 100.00% |    13 |
```

Timeline visualization

Use perfetto/chrome protobuf trace format

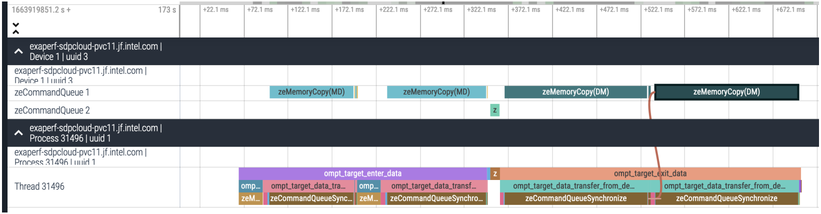

Figure 1: timeline

Iprof is Just a Tool on top of THAPI

- Babeltrace2 is a plugin architecture

```
babeltrace2 --plugin-path=$libdir "$@" \
            --component=filter.zeinterval.interval \
            --component=filter.ompinterval.interval \
            --component=sink.xprof.tally
```

- iprof is just one way of analyzing the trace from THAPI
- Bindings for `babeltrace2` exist in Python, Ruby, . . .
- So users can write their own plugins (e.g. OTF2 convertor, memory footprint tracker, . . .)

Conclusion / Future Work

- Trace all the runtime stack!
- In the process of the `v1.0` release (big refractoring of the internal)
- Deploying it on Polaris
- MPI api / HIP support
- If you want to colaborate, don't hesitate!