Light-weight profiling tools

Iprof, unitrace, xpu-smi

JaeHyuk Kwack
Perf. Engr., ALCF

Light-weight profiling tools on Aurora

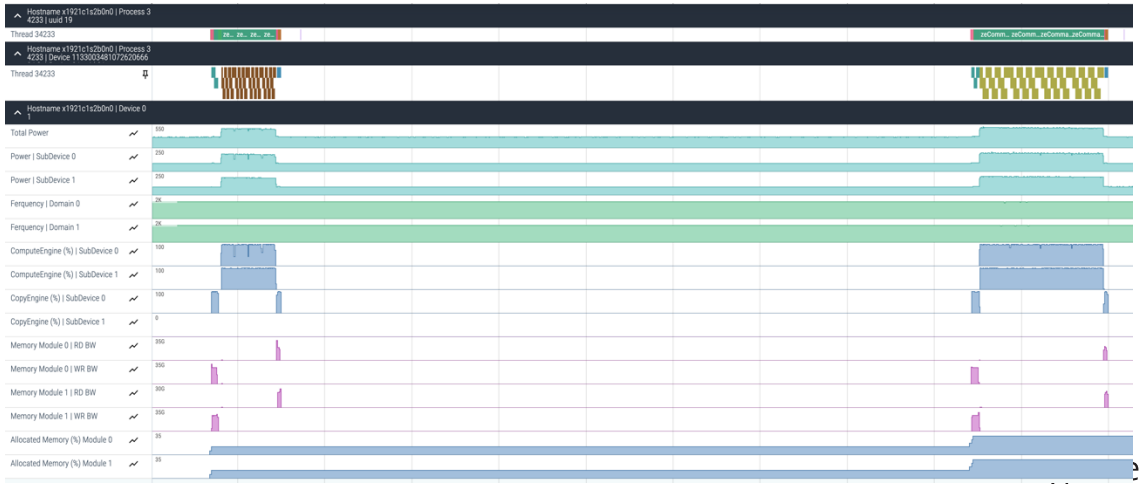
- THAPI/iprof
- Intel unitrace
- Intel xpu-smi

THAPI: Tracing Heterogeneous APIs

A lightweight tool for tracing and sampling

- Overview
  - —THAPI is a portable, programming model-centric tracing framework for heterogeneous systems.
    - OpenCL, L0, Cuda, HIP, OMPT, MPI
  - —Two Components:
    - Tracing Events
      - LTTng based tracing
    - Parsing of the trace
      - Babeltrace2 library and tools

- Device Sampling
  - —Ability to sample device telemetry with API tracing
    - Holistic view of system performance and behavior
    - Help understand H/W behavior in application context
    - Power/energy optimization
    - Resource management
    - Improves Debugging
  - —Power, Fabric and Memory Traffic, Engine Activities
  - —Timeline Visualization
    - Perfetto



THAPI / Iprof on Aurora

```
$ module load thapi
$ mpirun -n 12 -ppn 12 gpu_tile_compact.sh iprof ./Comp_GeoSeries_omp_mpicpp_DP 2048 1000
```

```
THAPI: Trace location: /home/jkwack/thapi-traces/thapi_aggreg--2024-09-25T06:27:15-05:00
BACKEND_MPI | 1 Hostnames | 12 Processes | 12 Threads |

         Name |     Time | Time(%) | Calls |  Average |      Min |     Max |
     MPI_Init |   16.76s |  98.13% |    12 |    1.40s | 434.72ms |   2.48s |
 MPI_Finalize | 219.50ms |   1.29% |    12 |  18.29ms |  17.48ms | 22.41ms |
   MPI_Reduce |  99.94ms |   0.59% |    96 |   1.04ms |    908ns | 19.77ms |
MPI_Comm_rank |  13.49us |   0.00% |    12 |   1.12us |    504ns |  3.27us |
MPI_Comm_size |   7.60us |   0.00% |    12 | 633.67ns |    485ns |   733ns |
        Total |   17.08s | 100.00% |   144 |
……
```

BACKEND_OMP | 1 Hostnames | 12 Processes | 12 Threads |

Name	Time	Time(%)	Calls	Average	Min	Max
ompt_target_exit_data	468.50ms	39.62%	12	39.04ms	31.71ms	45.23ms
ompt_target_data_transfer_from_device	460.39ms	38.93%	12	38.37ms	31.12ms	44.28ms
ompt_target	191.37ms	16.18%	132	1.45ms	1.30ms	9.28ms
ompt_target_enter_data	30.59ms	2.59%	12	2.55ms	1.84ms	3.58ms
ompt_target_data_transfer_to_device	20.24ms	1.71%	12	1.69ms	1.19ms	2.46ms
ompt_target_submit_emi	9.36ms	0.79%	132	70.90us	7.16us	1.13ms
ompt_target_data_alloc	1.33ms	0.11%	24	55.28us	24.86us	93.15us
ompt_target_data_delete	775.82us	0.07%	24	32.33us	5.08us	67.70us
Total	1.18s	100.00%	360			

BACKEND_ZE | 1 Hostnames | 12 Processes | 12 Threads |

Name	Time	Time(%)	Calls	Average	Min	Max	Error
zeModuleCreate	2.72s	77.65%	132	20.61ms	103.89us	224.80ms	0
zeCommandListAppendMemoryCopy	483.38ms	13.80%	180	2.69ms	9.00us	44.28ms	0
zeEventHostSynchronize	197.31ms	5.63%	312	632.41us	108ns	9.25ms	0
zeEventCreate	28.86ms	0.82%	49920	578.08ns	223ns	146.23us	0
zeCommandListCreateImmediate	22.23ms	0.63%	24	926.29us	55.87us	2.90ms	0
zeModuleDestroy	10.31ms	0.29%	132	78.08us	5.86us	478.02us	0
zeEventDestroy	9.04ms	0.26%	49920	181.14ns	108ns	23.10us	0
zeContextMakeMemoryResident	7.99ms	0.23%	84	95.15us	5.18us	610.76us	0
zeCommandListAppendLaunchKernel	7.28ms	0.21%	132	55.12us	6.57us	636.24us	0
zeCommandQueueCreate	3.23ms	0.09%	12	268.84us	232.46us	299.16us	0
zeMemAllocDevice	2.55ms	0.07%	84	30.41us	13.05us	69.45us	0
zeDriverGetExtensionFunctionAddress	2.00ms	0.06%	132	16.69us	289ns	233.68us	12
zeKernelCreate	1.83ms	0.05%	1752	1.04us	684ns	12.13us	0

```
Device profiling | 1 Hostnames | 12 Processes | 12 Threads | 12 Devices | 12 Subdevices |

                              Name |     Time | Time(%) | Calls |  Average |    Min |     Max |
zeCommandListAppendMemoryCopy(D2M) | 203.59ms |  51.65% |    12 |  16.97ms | 6.03ms | 30.29ms |
                       Comp_Geo_l29 | 172.75ms |  43.83% |   132 |   1.31ms | 1.29ms |  1.40ms |
zeCommandListAppendMemoryCopy(M2D) |  17.58ms |   4.46% |    96 | 183.11us |   80ns |  2.10ms |
zeCommandListAppendMemoryCopy(S2M) | 217.76us |   0.06% |    48 |   4.54us | 1.28us | 16.48us |
zeCommandListAppendMemoryCopy(M2M) |  24.40us |   0.01% |    12 |   2.03us | 1.36us |  2.80us |
zeCommandListAppendMemoryCopy(M2S) |    960ns |   0.00% |    12 |  80.00ns |   80ns |    80ns |
                             Total | 394.16ms | 100.00% |   312 |

Explicit memory traffic (BACKEND_MPI) | 1 Hostnames | 12 Processes | 12 Threads |

      Name | Byte | Byte(%) | Calls | Average | Min | Max |
MPI_Reduce | 768B | 100.00% |    96 |   8.00B |  8B |  8B |
     Total | 768B | 100.00% |    96 |

Explicit memory traffic (BACKEND_OMP) | 1 Hostnames | 12 Processes | 12 Threads |

                                Name |     Byte | Byte(%) | Calls | Average |     Min |     Max |
               ompt_target_data_alloc | 805.31MB |  50.00% |    24 | 33.55MB | 33.55MB | 33.55MB |
  ompt_target_data_transfer_to_device | 402.65MB |  25.00% |    12 | 33.55MB | 33.55MB | 33.55MB |
ompt_target_data_transfer_from_device | 402.65MB |  25.00% |    12 | 33.55MB | 33.55MB | 33.55MB |
              ompt_target_data_delete |       0B |   0.00% |    24 |   0.00B |      0B |      0B |
                               Total |   1.61GB | 100.00% |    72 |
```

……

Explicit memory traffic (BACKEND_ZE) | 1 Hostnames | 12 Processes | 12 Threads |

Name	Byte	Byte(%)	Calls	Average	Min	Max
zeContextMakeMemoryResident	845.41MB	51.21%	84	10.06MB	8B	33.55MB
zeCommandListAppendMemoryCopy(M2D)	402.71MB	24.40%	96	4.19MB	4B	33.55MB
zeCommandListAppendMemoryCopy(D2M)	402.65MB	24.39%	12	33.55MB	33.55MB	33.55MB
zeCommandListAppendMemoryCopy(S2M)	2.40kB	0.00%	48	50.00B	8B	144B
zeCommandListAppendMemoryCopy(M2S)	768B	0.00%	12	64.00B	64B	64B
zeCommandListAppendMemoryCopy(M2M)	684B	0.00%	12	57.00B	57B	57B
Total	1.65GB	100.00%	264			

Unified Tracing and Profiling tools (unitrace)

- A performance tools for Intel onaAPI application that traces and profiles host/device activities, interactions and hardware utilizations for Intel GPU applications.
  - https://github.com/intel/pti-gpu/tree/master/tools/unitrace
- Features
  - Level Zero (L0) or Level Zero + OpenCL tracking/profiling
  - Host activities
  - Device and kernel activities
  - Trace and profile layers (e.g., MPI, SYCL, CCL, oneDNN) above L0/OpenCL
  - Categorizing GPU kernels
  - Profile hardware performance metrics

Unified Tracing and Profiling tools (unitrace)

- Run with unitrace

```
$ unitrace [options] <application> [args]
```

The options are as follows:

--device-timing [-d] Report kernels execution time

--ccl-summary-report [-r] Report CCL execution time summary

--device-timeline [-t] Report device timeline

--chrome-mpi-logging Trace MPI

--chrome-sycl-logging Trace SYCL runtime and plugin

--chrome-ccl-logging Trace oneCCL

--chrome-kernel-logging Trace device and host kernel activities

--separate-tiles Trace each tile separately in case of implicit scaling

--output [-o] <filename> Output profiling result to file

--output-dir-path <path> Output directory path for result files

--metric-query [-q] Query hardware metrics for each kernel instance

--metric-sampling [-k] Sample hardware performance metrics for each kernel instance in time-based mode

--group [-g] <metric-group> Hardware metric group (ComputeBasic by default)

--sampling-interval [-i] <interval> Hardware performance metric sampling interval in us (default is 50 us) in time-based mode

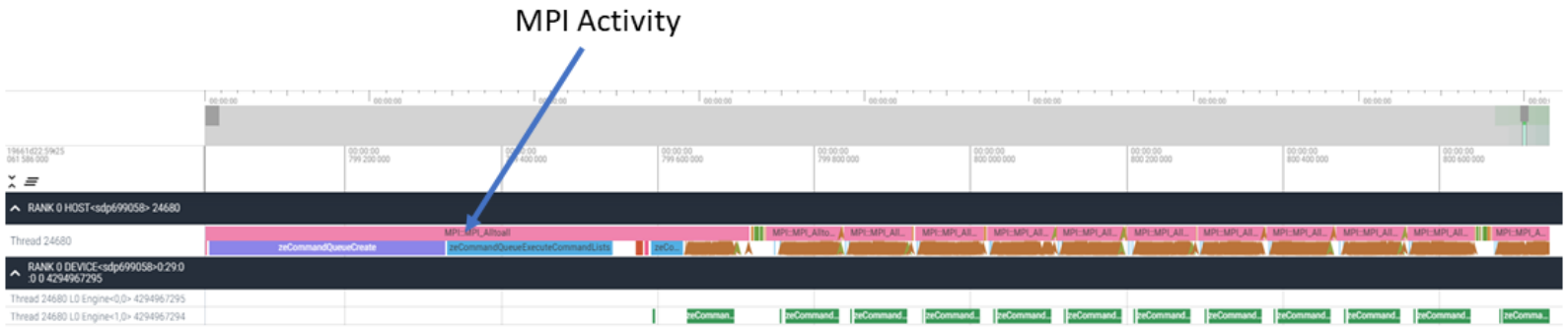
--device-list Print available devices

--metric-list Print available metric groups and metrics

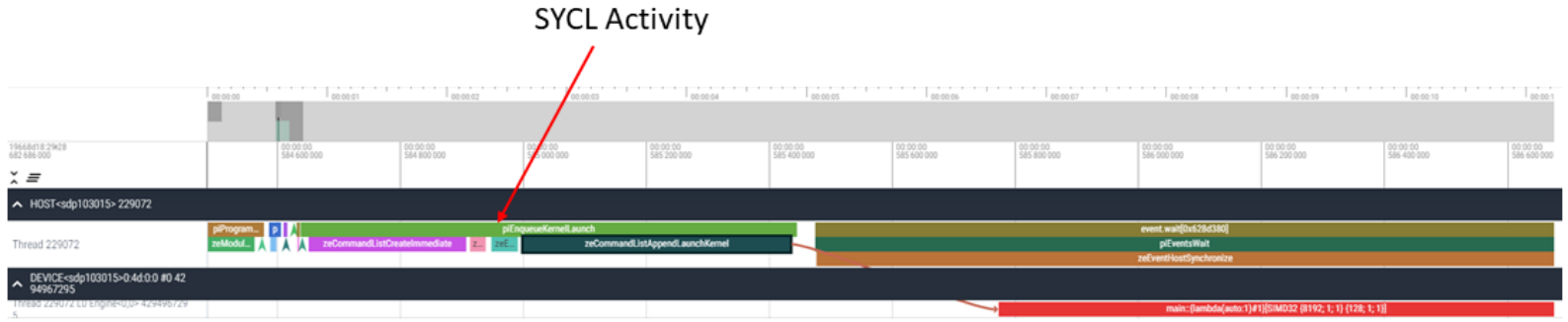
--stall-sampling Sample hardware execution unit stalls. Valid for Intel(R) Data Center GPU Max Series and later GPUs

--ranks-to-sample <ranks> MPI ranks to sample. The argument <ranks> is a list of comma separated MPI ranks

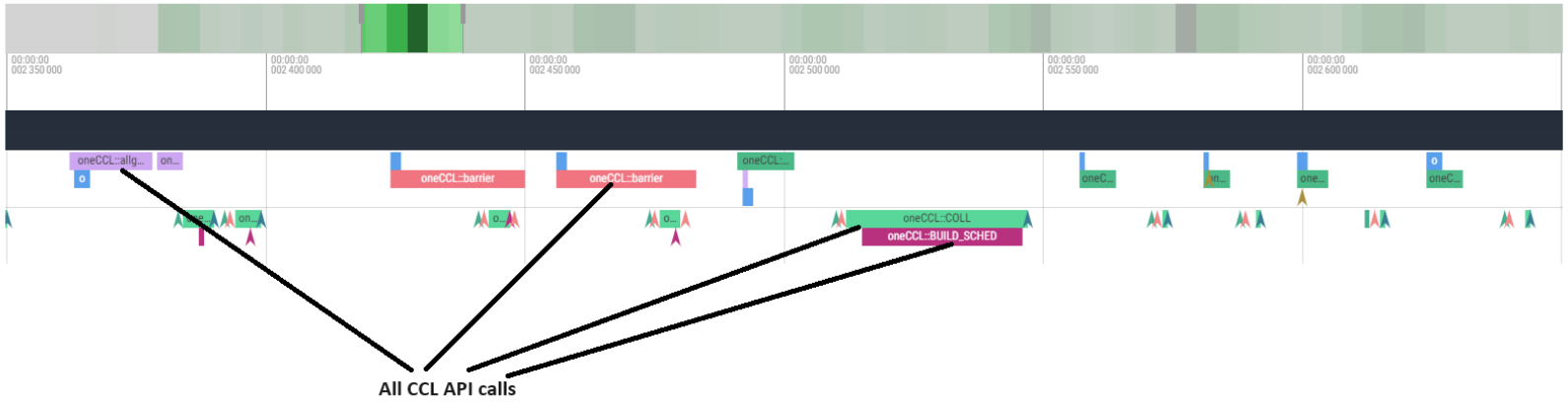
--chrome-mpi-logging



--chrome-sycl-logging



--chrome-ccl-logging



unitrace on Aurora

```
$ module load pti-gpu
$ unitrace –version
2.1.2 (31dd08753125943b26475cec6a489b7c52c064dd)
$ unitrace --device-list
$ unitrace --metric-list | grep Group
```

Intel XPU System Management Interface, xpu-smi

- Are you expecting something like nvidia-smi?
- xpu-smi provides similar features on Intel GPUs
  - https://github.com/intel/xpumanager
- A tool for monitoring and managing Intel data center GPUs
- It is designed to simplify administration, maximize reliability and uptime, and improve utilization.
- Intel(R) XPU System Management Interface (XPU-SMI) is the daemon-less version of XPU Manager and it only provides the local interface.

- xpu-smi on Aurora

```
$ module load xpu-smi
$ xpu-smi --version
CLI:
    Version: 1.2.39.20240906
    Build ID: 11f3c29a
$ xpu-smi discovery
$ xpu-smi dump -d 0 1 2 3 4 5 -m 0 1 2 4 5
```

Demo

Copy the example to your space:

```
$ cp -r /lus/flare/projects/gpu_hack/examples/tools_intel ~/
$ cd ~/tools_intel
```

Build the code:

```
$ make
mpicc -fiopenmp -fopenmp-targets=spir64  -O2 -fdebug-info-for-profiling -gline-tables-only Comp_GeoSeries_omp.c -o Comp_GeoSeries_omp_mpicc_DP
rm -rf *.o *.mod *.dSYM
```

Run the code:

```
$ mpirun -n 12 gpu_tile_compact.sh ./Comp_GeoSeries_omp_mpicc_DP 2048 1000
```

Demo

iprof demo:

```
$ module load thapi
$ which iprof
$ mpirun -n 12 gpu_tile_compact.sh iprof ./Comp_GeoSeries_omp_mpicc_DP 2048 1000
$ mpirun -n 12 gpu_tile_compact.sh iprof -l -- ./Comp_GeoSeries_omp_mpicc_DP 2048 1000
```

unitrace demo:

```
$ module load pti-gpu
$ which unitrace
$ unitrace --device-list
$ unitrace --metric-list | grep Group
$ mpirun -n 11 gpu_tile_compact.sh ./Comp_GeoSeries_omp_mpicc_DP 2048 1000 : -n 1
gpu_tile_compact.sh unitrace ./Comp_GeoSeries_omp_mpicc_DP 2048 1000
$ mpirun -n 12 gpu_tile_compact.sh unitrace --chrome-mpi-logging
./Comp_GeoSeries_omp_mpicc_DP 2048 1000
$ mpirun -n 12 gpu_tile_compact.sh unitrace --chrome-kernel-logging --chrome-mpi-logging
./Comp_GeoSeries_omp_mpicc_DP 2048 1000
```

Demo

xpu-smi demo:

```
$ module load xpu-smi
$ xpu-smi –version
$ xpu-smi -h
$ xpu-smi discovery
$ xpu-smi –h dump
$ xpu-smi dump -d 0 1 2 3 4 5 -m 0 1 2 4 5
```