Open-source Performance Analysis Tools and SDK for Intel® GPUs: Today and Tomorrow

Zhiqiang Ma and Julia Fedorova

Intel Corporation

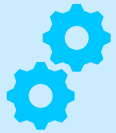
July 27, 2026

Intel Open-source Projects

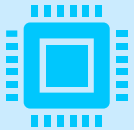
- https://github.com/orgs/intel/repositories?type=all

Agenda

unitrace



PTI SDK



Call for Actions

Acknowledgement

- Without contributions by https://github.com/intel/pti-gpu/graphs/contributors?all=1, these projects would not be possible
- Without customer feedback, these projects would not be what they are today

unitrace

Agenda

- Overview
- Key Features
- Next

Overview

- The unitrace is a light-weight open-source tool for developers to profile, visualize and optimize performance of AI and HPC applications running on Intel® CPUs and GPUs on Linux and Windows
- It tracks CPU and GPU activities and the interactions, profiles GPU hardware utilizations and provides insights into the host and device performance
- It generates timing summary statistics, detailed timelines for both CPU and GPU
- Source release: https://github.com/intel/pti-gpu/tree/master/tools/unitrace

Key Features

- CPU and GPU activity tracking
- GPU utilization profiling
- CPU and GPU activities and GPU utilization visualization
- GPU kernel profiling at instruction level
- Graph profiling
- Scale-up and scale-out profiling
- Focused profiling
- GPU Roofline (experimental and not available for all devices)

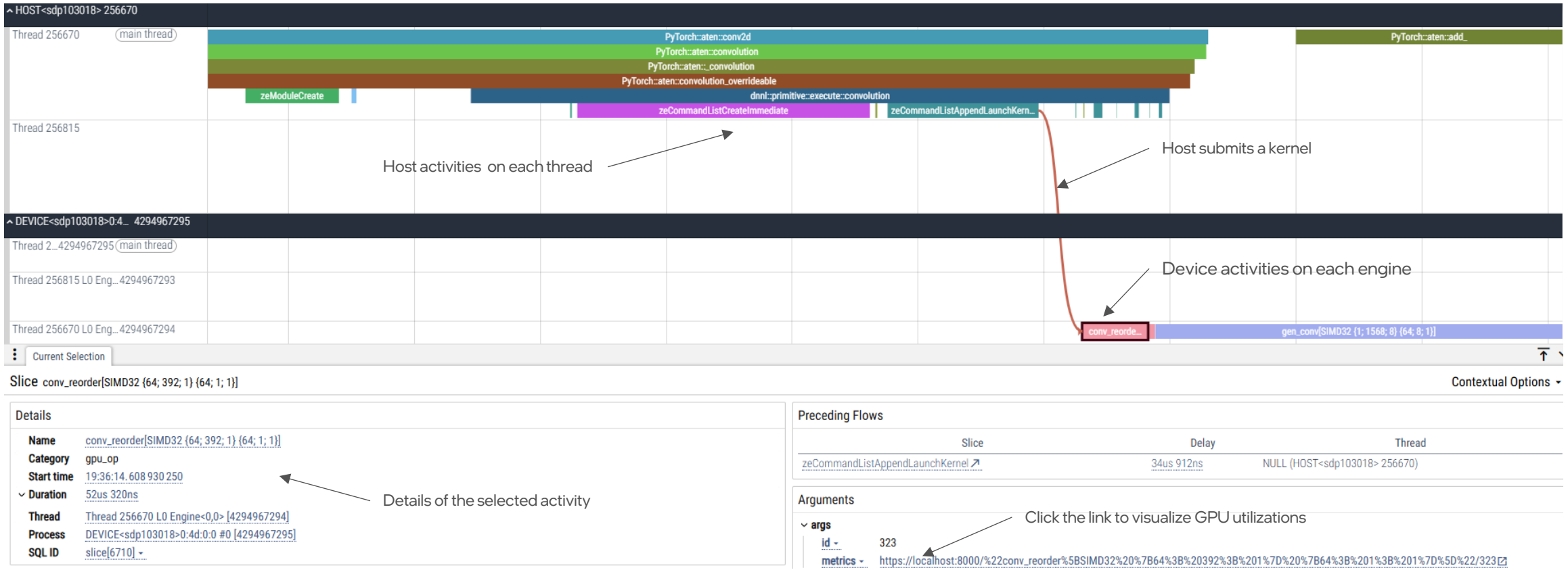
Track CPU and GPU Activities

- Trace host and device activities of the oneAPI software stack layers
  - Kernel mode driver, Level Zero and OpenCL runtime, unified runtime, SYCL runtime, oneMKL, oneDNN, oneCCL, OpenMP, MPI, PyTorch … on CPU
  - Data transfers, SYCL, OpenCL, Triton kernel executions utilizing copy and compute engines on GPU
  - Dependencies between GPU and CPU
- Allow developers of every layer to investigate correlations, dependencies, resource utilizations and performance bottlenecks

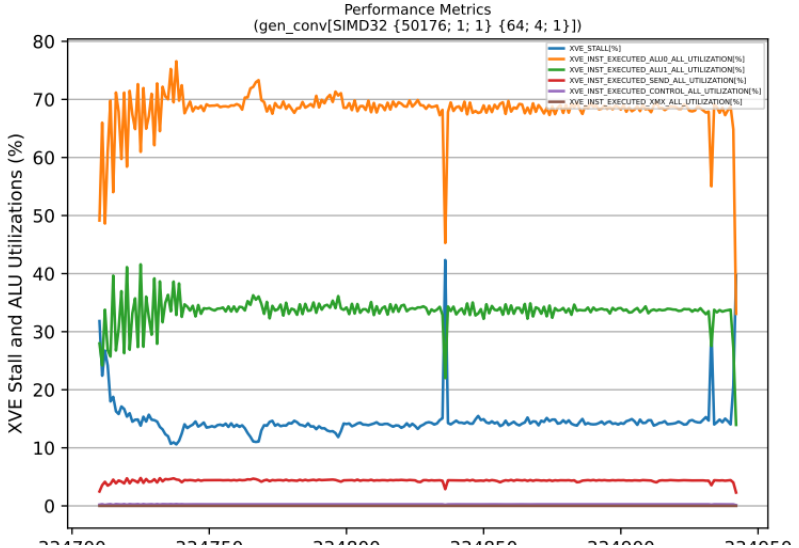
Profile GPU Utilizations

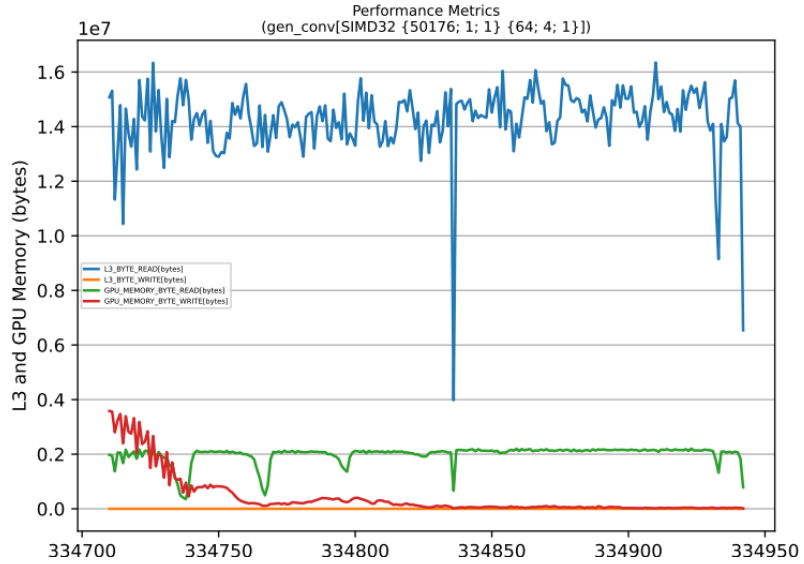
- Profile utilizations of GPU components for each kernel execution and data transfer
  - ALU utilizations, occupancies, L1/SLM/L3/memory traffic and throughput, frequencies, EU stalls, TLB misses …
- Support time-based sampling and event-based sampling
- Allow developers to investigate hardware resource utilizations and kernel performance bottlenecks and gain insights into optimization opportunities

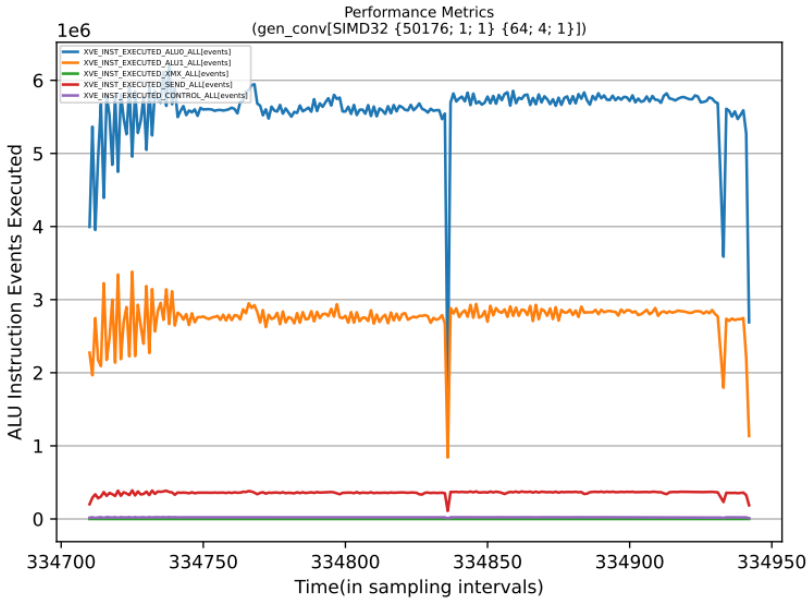
Visualize CPU and GPU Activities and GPU Utilizations on a Unified Timeline (1/2)

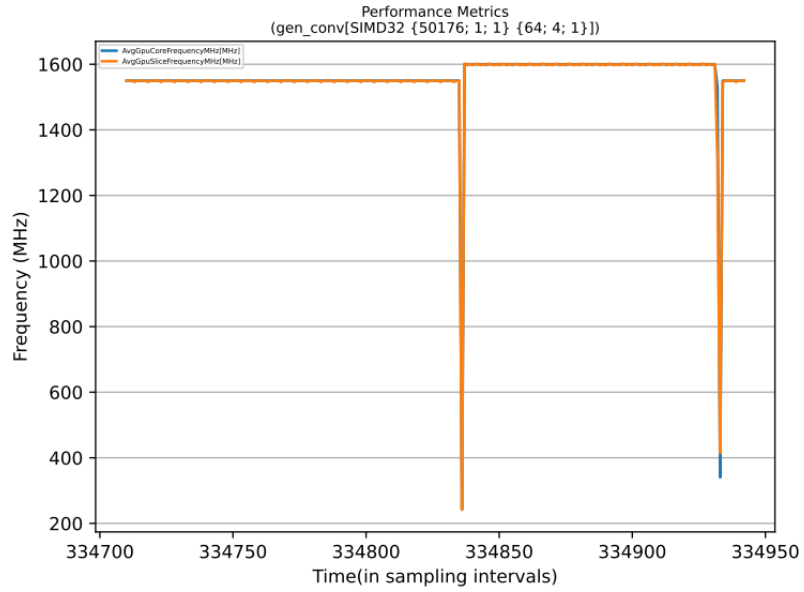


Visualize CPU and GPU Activities and GPU Utilizations on a Unified Timeline (2/2)



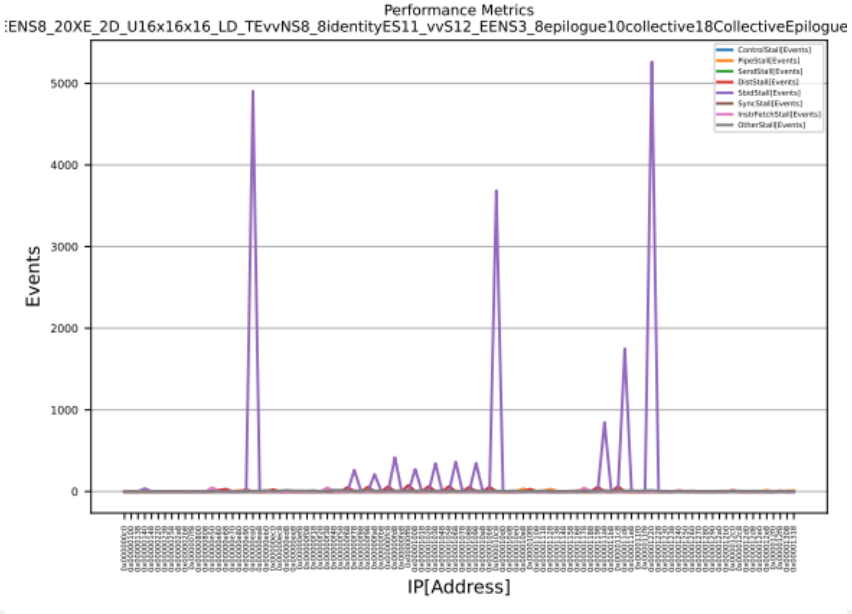






Profile GPU Kernels at Instruction Level

- Pinpoint performance bottlenecks to GPU kernel instructions
- Map instructions to kernel source
- Rank instructions by cost to guide users on prioritizing tasks
- Analyze causes of bottlenecks to provide optimization guidance

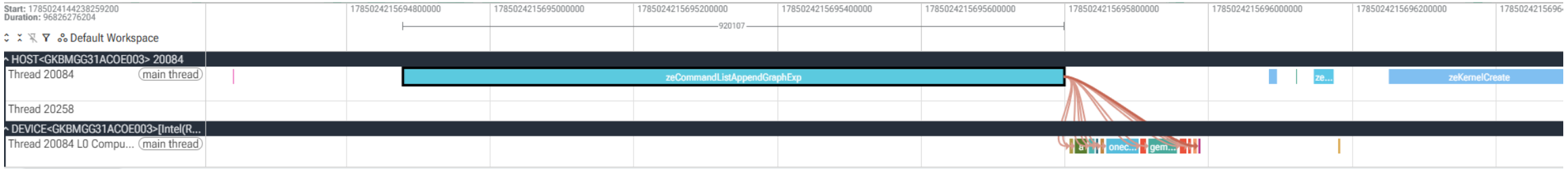


Scale-up and Scale-out Profiling

- Support both single-node-multi-device and multi-node-multi-device workloads
- Profile multiple ranks, processes and devices at same time
- Visualize activities in a single view

Graph Profiling

- Graph are available in both SYCL and Level Zero as experimental extensions to minimize kernel launch overhead



Current Selection

Slice zeCommandListAppendGraphExp

Contextual Options

Details

Name	zeCommandListAppendGraphExp
Category	cpu_op
Start time	1785024215694878720
Duration	920107
Thread	20084
Process	HOST<GKBMGG31ACOE003> [20084]
SQL ID	slice[438315]

Following Flows

Slice	Delay	Thread
triton_poi_fused_add_bitwise_and_bitwise_not_bitwise_or_ge_lt_mul_sub_0[SIMD32 {1; 1; 1} {128; ...	8173	Thread 20084 L0 Compute Engine<0,0> 429
at::native::xpu::IndexKernel<at::native::xpu::IndexKernelConfig<at::xpu::detail::TensorInfo<c10::BFl...	15317	Thread 20084 L0 Compute Engine<0,0> 429
at::native::xpu::ElementwiseGlobalRangeKernel<at::native::xpu::LegacyKernelScalarFunctor<c10::B...	35173	Thread 20084 L0 Compute Engine<0,0> 429
at::native::xpu::VectorizedElementwiseKernel<8, at::native::xpu::CopyScalarFunc<c10::BFloat16>, a...	45173	Thread 20084 L0 Compute Engine<0,0> 429
oneccl_allreduce_counter_pcie<sycl::_V1::ext::oneapi::bfloat16, Rt64_128_PCIE, true, 16, RingTran...	51413	Thread 20084 L0 Compute Engine<0,0> 429
oneccl_allreduce_pcie<sycl::_V1::ext::oneapi::bfloat16, Rt64_128_PCIE, true, 16, RingTransmit>[SI...	58837	Thread 20084 L0 Compute Engine<0,0> 429
vllm::rms_norm_kernel<sycl::_V1::ext::oneapi::bfloat16, 2, 4>[SIMD32 {128; 1; 1} {1024; 1; 1}] ↗	106173	Thread 20084 L0 Compute Engine<0,0> 429
gemm_kernel[SIMD16 {96; 1; 1} {32; 4; 1}] ↗	117717	Thread 20084 L0 Compute Engine<0,0> 429
vllm::rms_norm_kernel<sycl::_V1::ext::oneapi::bfloat16, 3, 4>[SIMD32 {1024; 1; 1} {128; 1; 1}] ↗	161493	Thread 20084 L0 Compute Engine<0,0> 429
vllm::rms_norm_kernel<sycl::_V1::ext::oneapi::bfloat16, 3, 4>[SIMD32 {256; 1; 1} {128; 1; 1}] ↗	173173	Thread 20084 L0 Compute Engine<0,0> 429
vllm::rotary_embedding_kernel<sycl::_V1::ext::oneapi::bfloat16, true>[SIMD32 {128; 1; 1} {512; 1; ...	179669	Thread 20084 L0 Compute Engine<0,0> 429
vllm::reshape_and_cache_flash_kernel<c10::BFloat16, c10::BFloat16, (vllm::Fp8KVCacheDataType)...	186837	Thread 20084 L0 Compute Engine<0,0> 429

Focused Profiling

- Restrict profiling to specific execution of region of interest and/or specific kernels
  - Include kernels and exclude kernels
  - Select devices to profile
  - Select ranks to profile
  - Pause/resume/stop profiling
    - In-target-process control
      - Application source change is required to add __itt_* calls or set/unset an environment variable
    - Out-of-target-process session control
      - No change to source is required
      - Pause/resume/stop profiling in all processes in the same session

Next

- Enhance existing features
- Push new features out
- Reduce overhead
- Support new GPUs

PTI SDK

Agenda

- 1000-foot View
- API Groups
- PyTorch Trace Example
- Next

PTI SDK — an open-source library with C interface for applications, frameworks, … to enable on-demand profiling in the application contex for Intel GPUs.
Already part of PyTorch package and used in its Profiler

1000-foot View

Linux and Windows operating systems

Intel Data Center GPU Max Series (1100, 1550), Intel Data Center GPU Flex Series, Intel Arc Graphics (A-Series and B-Series)

- Easy-to-use agnostic API
  - Internally based on low-level SYCL runtime/Level-Zero APIs
- External Correlation IDs
  - Associate GPU operations with application-level operations & logic
- On-demand profiling
  - Enable a tool to start/stop profiling at arbitrary point; zero overhead outside profiled region(s)
- Buffer Management
  - User controls buffer allocation and delivery
  - Records delivered in batches via registered callbacks
- PyTorch Profiler backend for Intel GPU
  - Expose GPU kernel and memory tracing in PyTorch workflows, correlation with higher-level operations

Available at https://github.com/intel/pti-gpu/tree/master/sdk

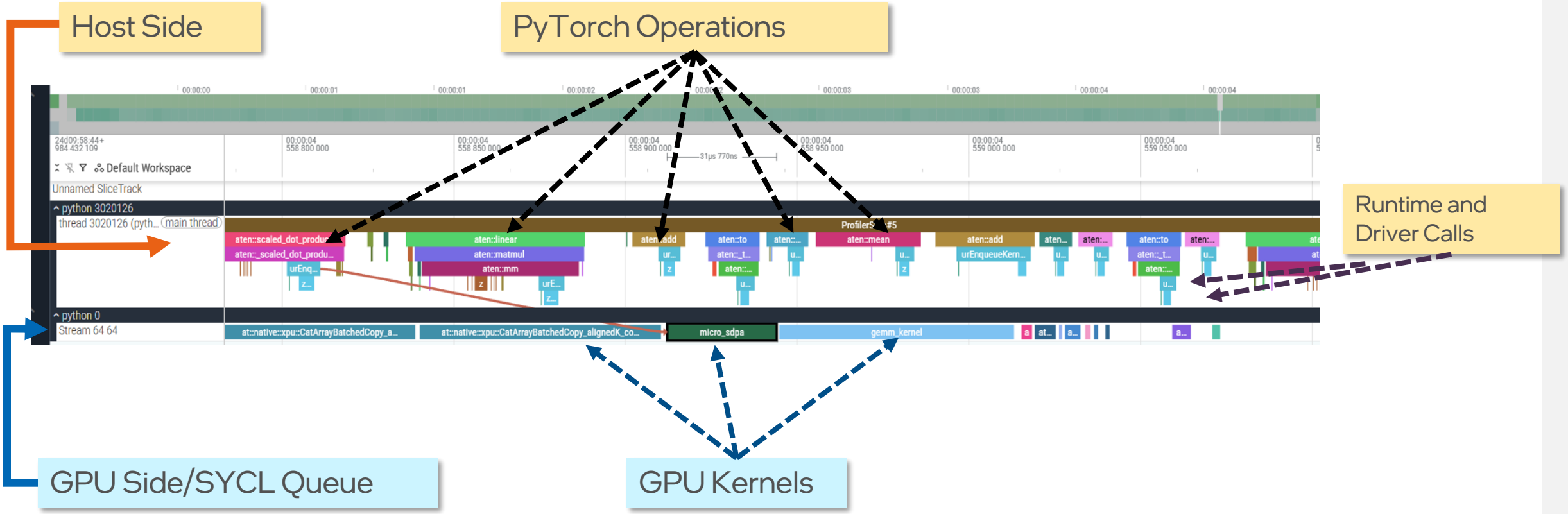
API Groups

API Group	Purpose
ptiView	Trace kernels, memory transfers, host runtime and driver API calls; deliver records via callbacks
ptiMetricsScope	Characterize kernel instances with GPU hardware performance metrics (EU utilization, memory bandwidth, cache statistics)
ptiPcSampling *(experimental)*	Enables fine-grained bottleneck statistical analysis with per-instruction Stall Sampling
ptiMetrics	Discover available GPU hardware metrics and collect them continuously over time
ptiCallback *(experimental)*	Notify synchronously about GPU operation lifecycle events (append, dispatch, complete)

Version 1.0 (released June 2026)

- Stable API and ABI: ptiView, ptiMetricsScope, ptiMetrics
- oneCCL tracing: via ptiView, tracing of Intel oneCCL library operations
- SYCL Graph profiling: via ptiView
  - proof-of-concept for tracing repeatedly executed SYCL graphs
- ptiMetricsScope multi-device
- ptiPcSampling (Experimental)
  - statistical per-instruction stall Sampling of GPU kernels
- ptiCallback (Experimental)
  - synchronous GPU operation lifecycle notification API
- Host-side collection overhead reduction
- Documentation
- Compiler compatibility
  - Intel oneAPI 2026.0 and 2026.1, open-source DPC++ compiler

PyTorch Trace

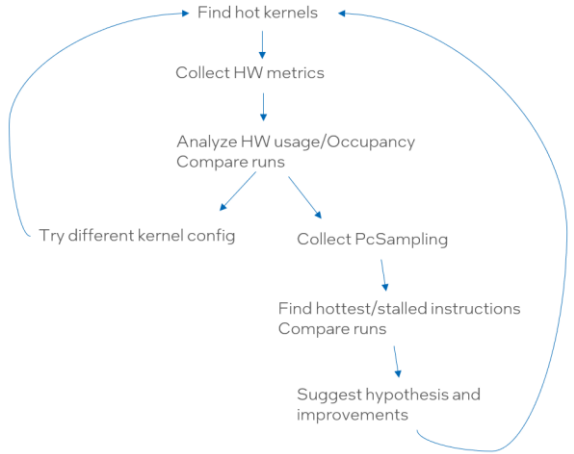


Next

- ~3 months
  - Full Native Graph tracing support — complete support for SYCL and Level-Zero Graph profiling
  - High priority bug fixes

- 3+ Months
  - PC Sampling: address to source line mapping
  - Usability improvements *(e.g. remove required environment variable setup)*
  - Improved initialization interface
  - Utilities for traces manipulations

- Further
  - Performance Analyst on top of PTI libraries and other collectors
    - AI driven performance triage employing PTI collection capabilities: e.g.



Call for Actions

Call for Action

- Install from PyPI
  (pre-built binaries)

```
pip install intel-pti
```

- Build from Source

```
git clone https://github.com/intel/pti-gpu
cd pti-gpu/sdk
cmake --preset linux-icpx-release
cmake --build --preset linux-icpx-release
```

- Run a Sample

```
./build/bin/vector_sq_add     # ptiView tracing sample
./build/bin/dpc_gemm          # comprehensive tracing with correlation IDs
./build/bin/metrics_scope     # per-kernel hardware metrics
```

For more samples and information —
refer to PTI SDK documentation https://intel.github.io/pti-gpu


intel