

1. PROBLEM AND BACKGROUND

No standard support for PrivateUse1 profiling

- Kineto, PyTorch's native profiler backend, compiles CUDA, XPU and ROCm support directly into its own source tree.
- Out-of-tree (OOT) accelerators on the PrivateUse1 dispatch key have lacked kernel-level, device-correlated tracing.
- PyTorch's two independent profiler subsystems (Kineto and ProfilerStubs) expose PrivateUse1 hooks but no implementation; without integration, traces lack device rows or fail outright.
- No standard mechanism or example to provide end to end mapping from framework op dispatch end to end to its hardware execution for OOT backends.

3. ARCHITECTURE

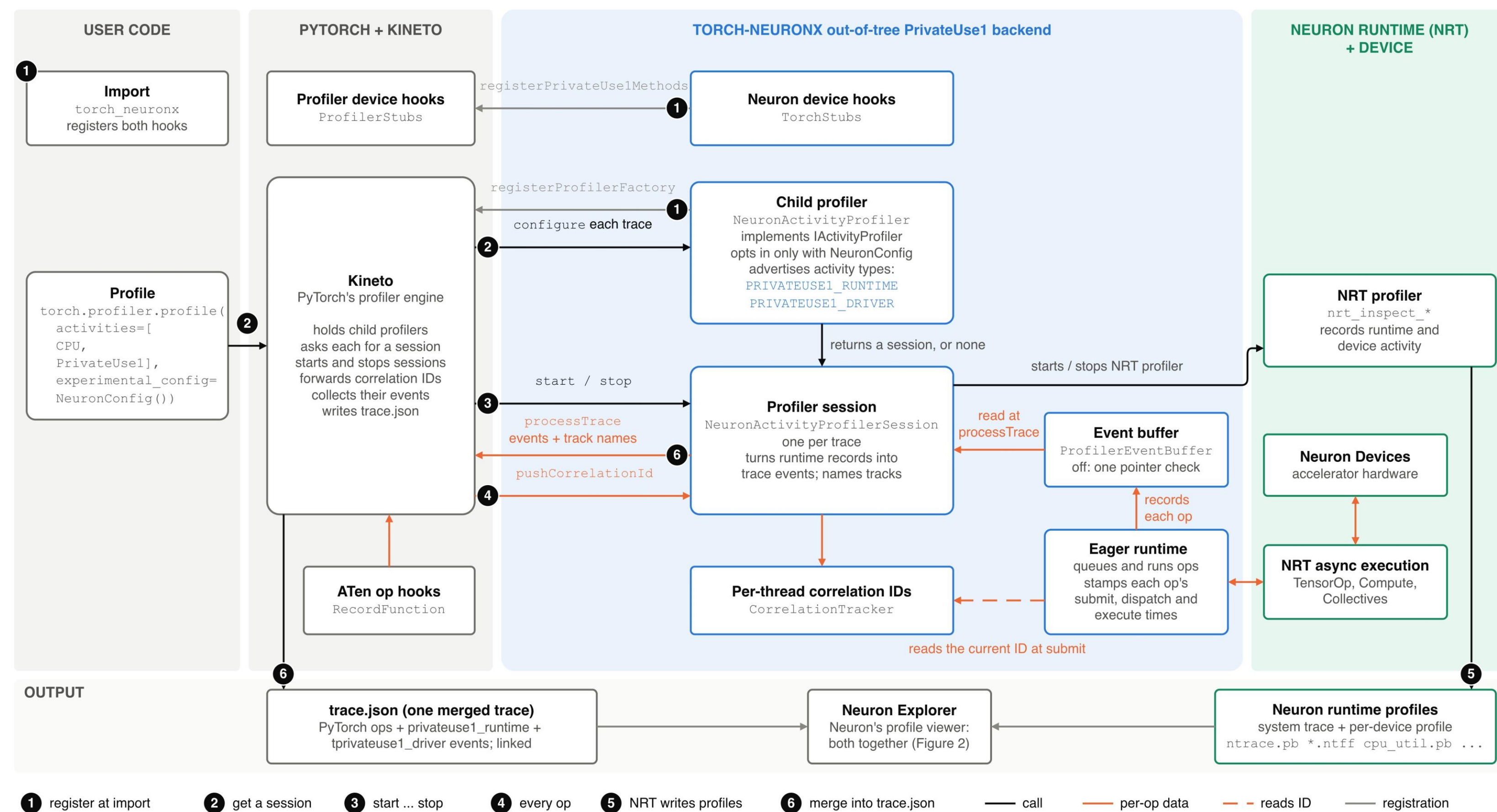


FIGURE 1 Neuron registers as a Kineto child profiler

5. OUTCOMES

- One merged trace shows framework, runtime and device activity on dependency-linked tracks.
- All the device and system events are synced and visualized on the same timeline for E2E performance debugging.
- Pinpoints dispatch latency, communication/compute overlap and host-bound stalls end to end.
- Working reference any PrivateUse1 backend can follow for complete profiler support.

2. IMPLEMENTED SOLUTION

Two registrations: a Kineto child profiler plus PyTorch profiler stubs

- Kineto activity profiler: torch-neuronx implements IActivityProfiler and IActivityProfilerSession outside the Kineto tree. import torch_neuronx registers them through libkineto::api().registerProfilerFactory().
- Profiler stubs: registerPrivateUse1Methods(&stubs) supplies the enabled() gate, current device, CPU timestamps and synchronize.
- Activity types: The profiler advertises PRIVATEUSE1_RUNTIME for host submit and dispatch, and PRIVATEUSE1_DRIVER for the device in-flight window (from enqueue to NRT until the host observes completion).
- Neuron backend runtime APIs emit op queueing, host-device handoff, communication and per-engine device events, tagged with framework correlation IDs.
- Events are post-processed into a common format and aligned and linked with native PyTorch CPU events on one timeline.

4. WHAT THE TRACE SHOWS

TABLE 1 Event classes in the merged trace

Activity	Kineto type (trace category)	Track	Window	What it measures
CPU op (e.g. aten::mm)	CPU_OP (cpu_op)	PyTorch thread	RecordFunction enter to exit	Framework op on the host
torch-neuronx submit	PRIVATEUSE1_RUNTIME (privateuse1_runtime)	PyTorch thread	submit_start to submit_time	Host submit path on the PyTorch thread, up to hand-off to the NeuronStream
torch-neuronx dispatch	PRIVATEUSE1_RUNTIME (privateuse1_runtime)	Stream / XU / Slot (host virtual tid)	dispatch_start to execute_end	RuntimeWorker pickup, scheduling gates, and in-flight time
Device in-flight	PRIVATEUSE1_DRIVER (privateuse1_driver)	Stream / XU (device); kHost ops: host worker thread	execute_start to execute_end	NRT/XU queue wait + device execution + completion detection (host clock)
NRT profile (separate)	Not in trace.json	ntrace.pb, *.ntff	NRT timestamps	NRT system_profile and device_profile; view in Neuron Explorer

FIGURE 2 Neuron Explorer with all traces and dependencies linked

