# Deep Learning Optimized on Jean Zay

## PyTorch profiler

IDRIS

# PyTorch profiler

# PyTorch profiler

- We use a profiler to monitor an execution.
- It allows us to know the **time** and **memory** consumed by each part of the code.
- The results returned by the profiler point to the weaknesses of our code and tell us which parts we should **optimize** in priority.
- The profiler is a wrapper which records various information during the execution of the code.

This could be slowed down depending on the requested traces.
We usually monitor only **a few training steps**.

```
with prof:
        for epoch in range(0,args.epochs):
                for i, (images, labels) in enumerate(train_loader):
                        [...]
                        prof.step()
```

# PyTorch profiler

```
from torch.profiler import profile, tensorboard_trace_handler, ProfilerActivity, schedule

prof =  profile(activities=[ProfilerActivity.CPU, ProfilerActivity.CUDA],      # 1
                schedule=schedule(wait=1, warmup=1, active=5, repeat=1),        # 2
                on_trace_ready=tensorboard_trace_handler(logname),              # 3
                profile_memory=True,                                            # 4
                record_shapes=False,                                            # 5
                with_stack=False,                                               # 6
                with_flops=False)                                               # 7
```

1. We monitor the activity both on CPUs and GPUs.
2. We ignore the first step (`wait=1`) and we initialize the monitoring tools on one step (`warmup=1`). We activate the monitoring on 5 steps (`active=5`) and repeat the pattern only once (`repeat=1`).
3. We store the traces in a TensorBoard format (.json).
4. We profile the memory usage.
5. We don't record the input shapes of the operators.
6. We don't record call stacks (information about the active subroutines).
7. We don't request the FLOPs estimate of the tensor operations.

# TP2_2: Profiler Overview

Tutorial: https://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html

## Configuration

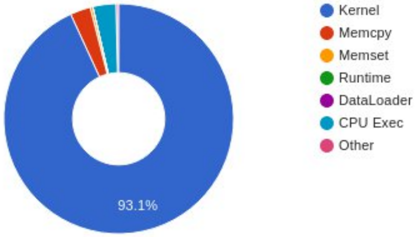
| | |
|---|---|
| **Number of Worker(s)** | **1** |
| **Device Type** | **GPU** |

## GPU Summary ?

| | |
|---|---|
| **GPU 0:** | |
| **Name** | **NVIDIA A100-SXM4-80GB** |
| **Memory** | **79.14 GB** |
| **Compute Capability** | **8.0** |
| **GPU Utilization** | **93.03 %** |
| **Est. SM Efficiency** | **92.18 %** |
| **Est. Achieved Occupancy** | **33.48 %** |

## Execution Summary

| Category | Time Duration (us) | Percentage (%) |
|---|---|---|
| Average Step Time | 449,011 | 100 |
| Kernel | 418,188 | 93.14 |
| Memcpy | 12,849 | 2.86 |
| Memset | 1,620 | 0.36 |
| Runtime | 0 | 0 |
| DataLoader | 0 | 0 |
| CPU Exec | 14,672 | 3.27 |
| Other | 1,682 | 0.37 |




Type and memory capacity of the GPU

% of time spent with an active GPU

% of active SMs

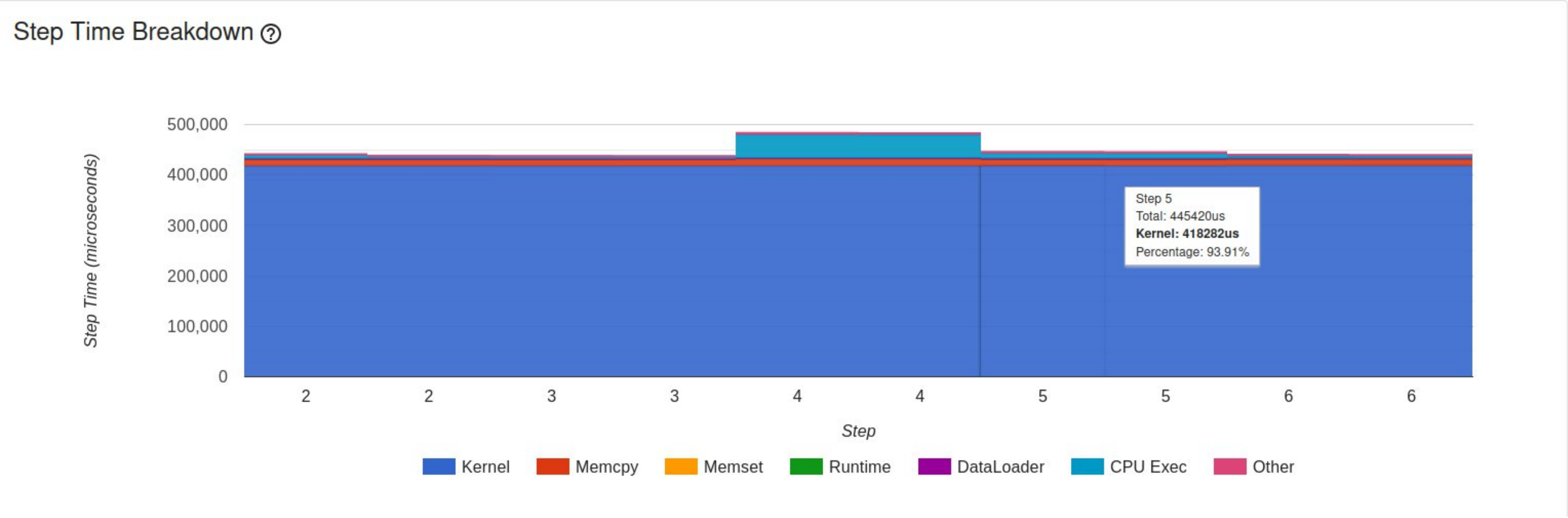
% of active wraps on an SM

A100

Streaming Multiprocessor

Link to image

# TP2_2: Profiler Step Time Breakdown

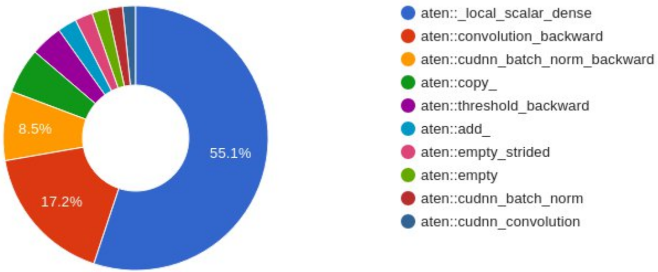


# TP2_2: Profiler Operator View

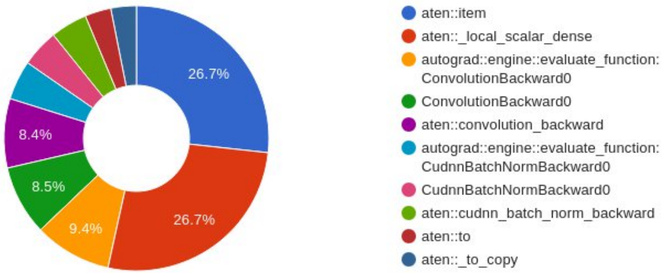
## Operator View

All operators | Top operators to show 10

**Host Self Time (us)**




**Host Total Time (us)**




Group By
Operator

Search by Name

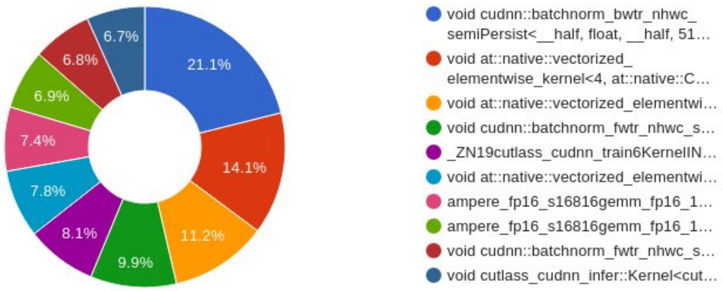
| Name | Calls | Device Self Duration (us) | Device Total Duration (us) | Host Self Duration (us) | Host Total Duration (us) | Tensor Cores Eligible | Tensor Cores Self(%) | Tensor Cores Total(%) | |
|---|---|---|---|---|---|---|---|---|---|
| aten::cudnn_convolution | 775 | 0 | 0 | 31458 | 31458 | Yes | 0 | 0 | View CallStack |
| aten::_convolution | 775 | 0 | 0 | 3146 | 34604 | Yes | 0 | 0 | View CallStack |
| aten::convolution | 775 | 0 | 0 | 4571 | 39175 | Yes | 0 | 0 | View CallStack |
| aten::conv2d | 1550 | 0 | 0 | 3895 | 105907 | Yes | 0 | 0 | View CallStack |
| aten::addmm | 5 | 0 | 0 | 265 | 265 | Yes | 0 | 0 | View CallStack |

# TP2_2: Profiler Kernel View

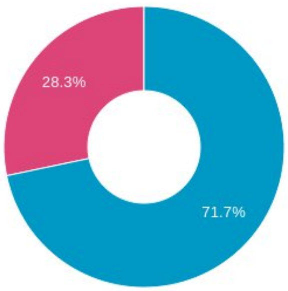
## Kernel View

All kernels ◉ Top kernels to show 10

**Total Time (us)** ?




**Tensor Cores Utilization** ?

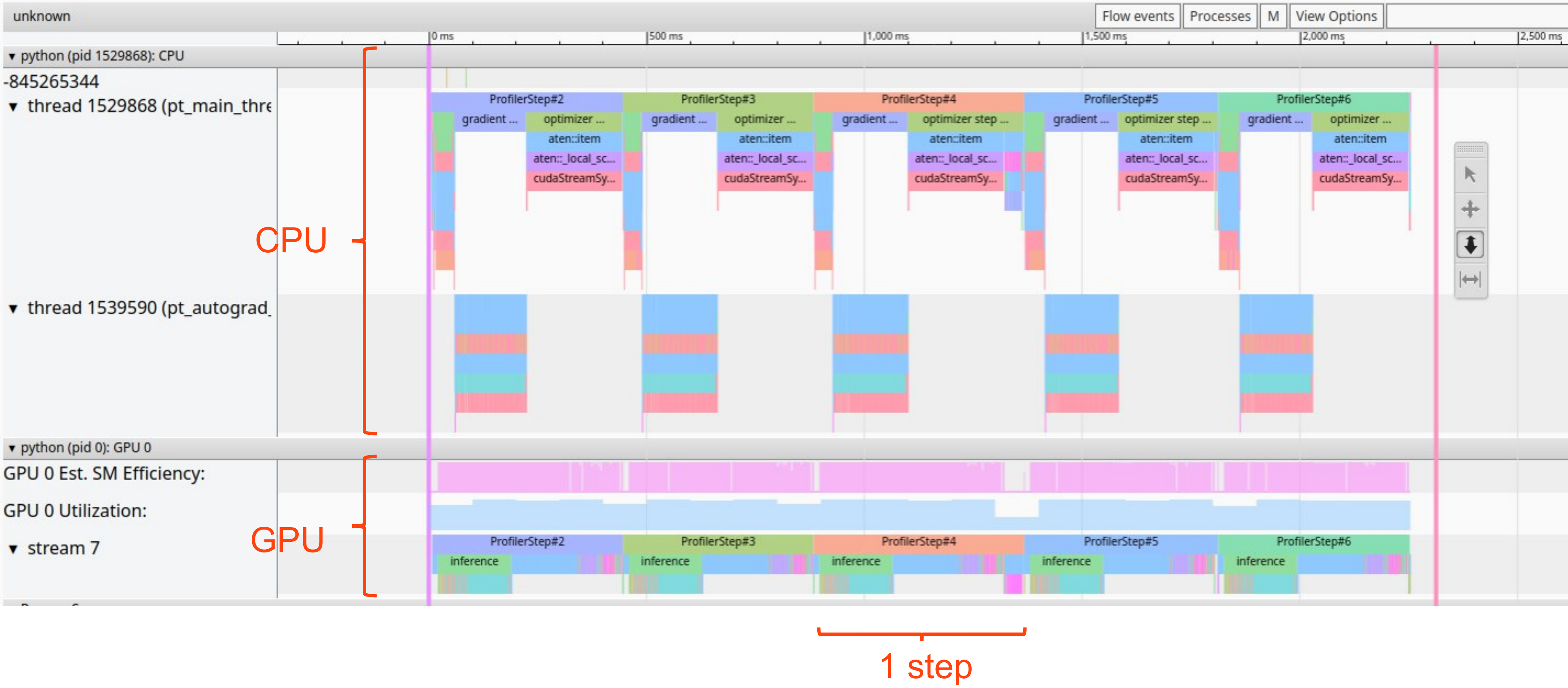



Group By
Kernel Name

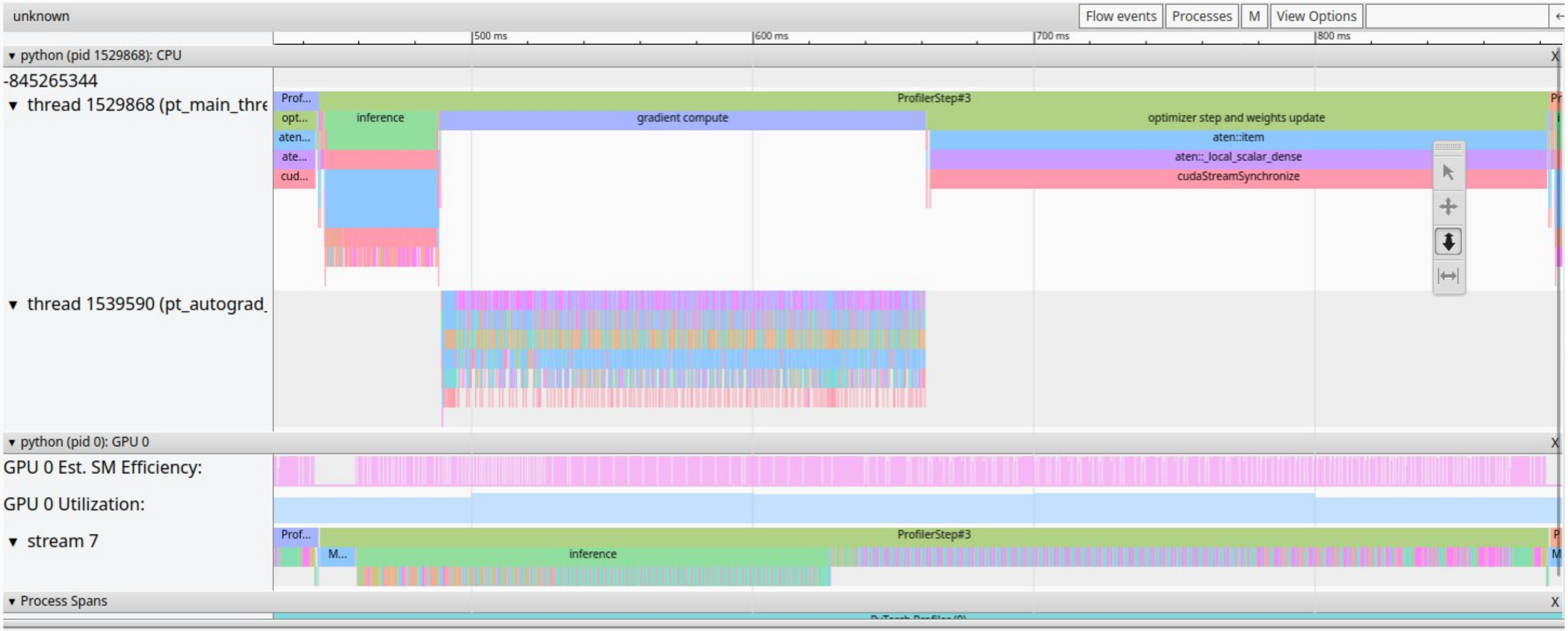
Search by Kernel Name

| Name | Tensor Cores Used | Calls | Total Duration (us) | Mean Duration (us) | Max Duration (us) | Min Duration (us) | Mean Blocks Per SM | Mean Est. Achieved Occupancy (%) |
|---|---|---|---|---|---|---|---|---|
| ampere_fp16_s16816gemm_fp16_128x128_ldg8_f2f_stages_32x5_tn | Yes | 430 | 119562 | 278 | 773 | 223 | 39.41 | 0 |
| ampere_fp16_s16816gemm_fp16_128x128_ldg8_f2f_stages_32x5_nn | Yes | 415 | 111298 | 268 | 562 | 230 | 39.14 | 0 |
| void cutlass_cudnn_infer::Kernel<cutlass_tensorop_f16_s16816dgrad_optimized_f16_128x128_32x4_nhwc_unity_stride_align8 >(cutlass_tensorop_f16_s16816dgrad_optimized_f16_128x128_32x4_nhwc_unity_stride_align8::Params) | Yes | 230 | 108892 | 473 | 758 | 450 | 31.52 | 0 |

# TP2_2: Profiler Trace



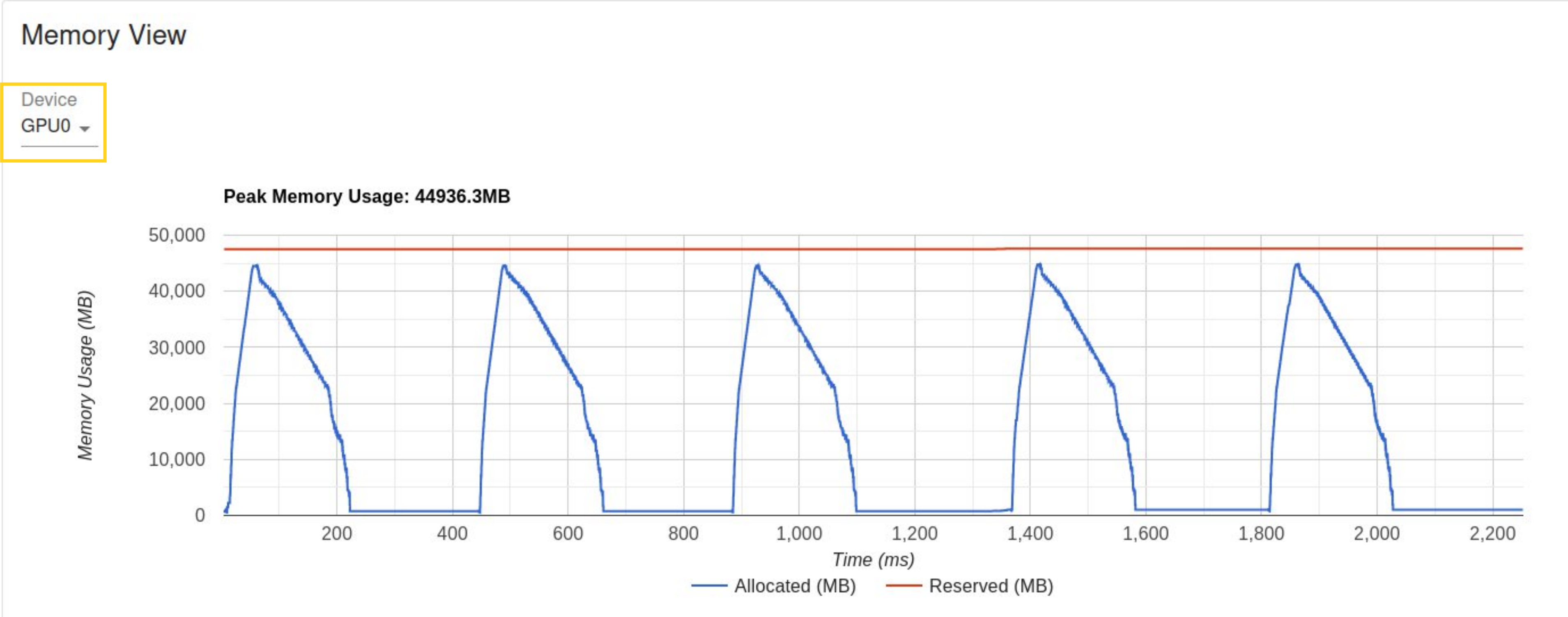
# TP2_2: Profiler Trace (1 step - GPU)



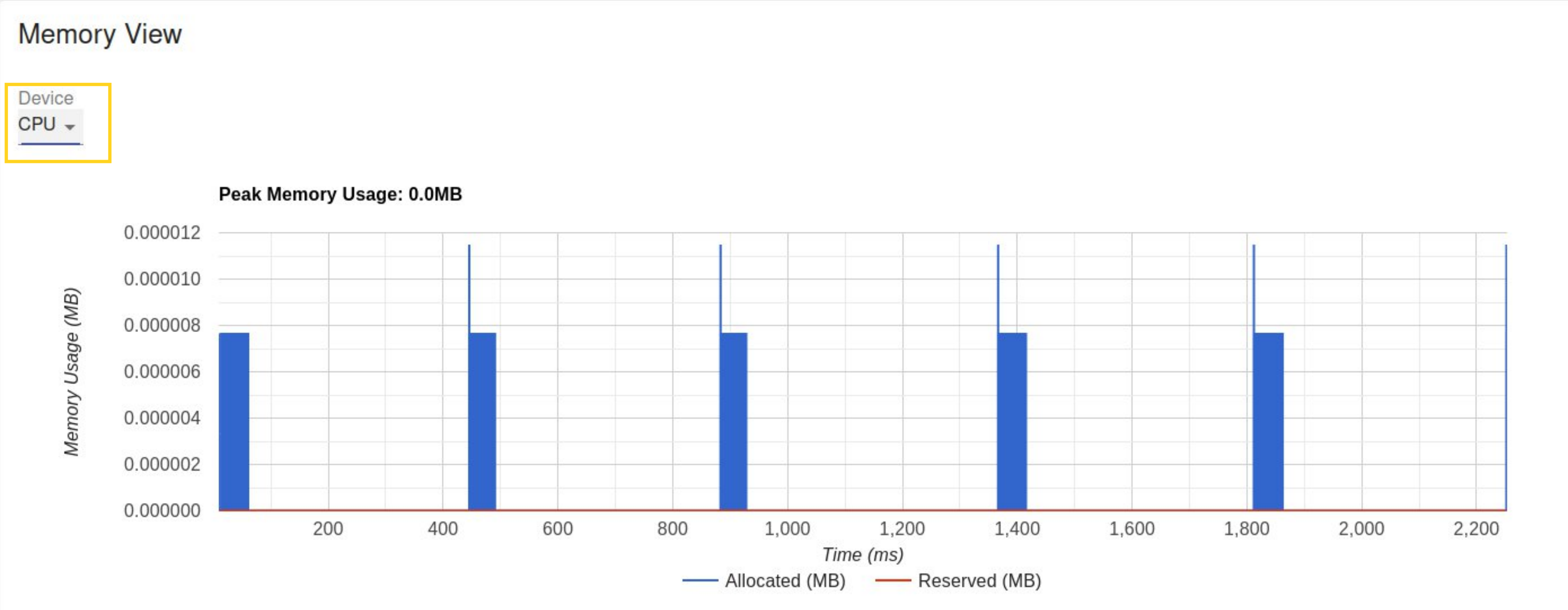

forward

backward

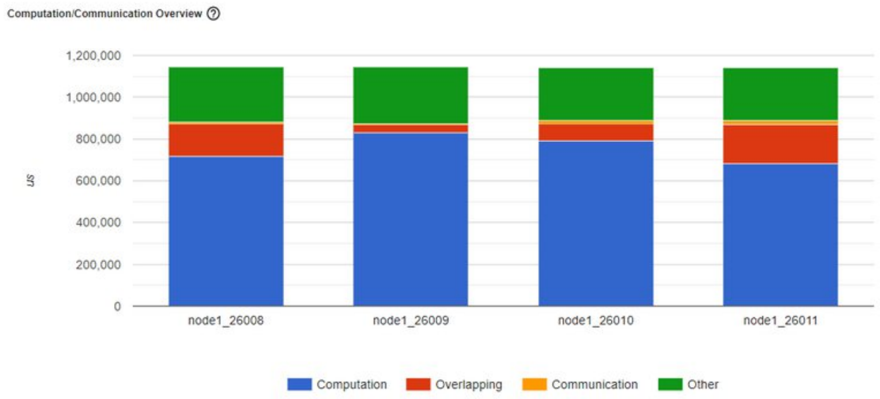
# TP2_2: Profiler Memory View (GPU)



# TP2_2: Profiler Memory View (CPU)



# TP2_2: Profiler Distributed



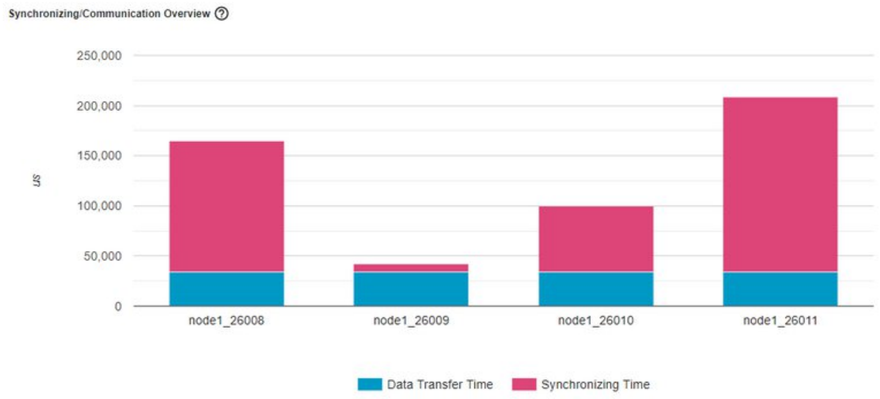




Image from the tutorial: https://pytorch.org/tutorials/intermediate/tensorboard_profiler_tutorial.html

# TP2_2: PyTorch profiler

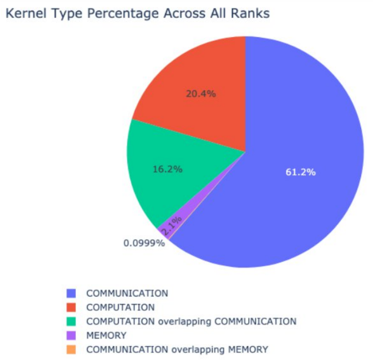
• NOTE

TensorBoard Plugin support has been deprecated, so some of these functions may not work as previously. Please take a look at the replacement, HTA.

**Holistic Trace Analysis**: https://hta.readthedocs.io/en/latest/

- Analyses PyTorch Profiler traces.
- Less user-friendly than TensorBoard Plugin.
- Focus on GPU usage.

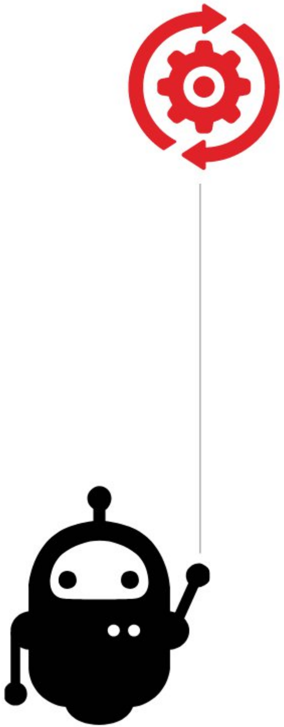
```
analyzer = TraceAnalysis(trace_dir = "/path/to/trace/folder")
kernel_type_metrics_df, kernel_metrics_df = analyzer.get_gpu_kernel_breakdown()
```




```
time_spent_df
```

| | rank | idle_time(ns) | compute_time(ns) | non_compute_time(ns) | kernel_time(ns) | idle_time_pctg | compute_time_pctg | non_compute_time_pctg |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 552069 | 596651 | 884850 | 2033570 | 27.15 | 29.34 | 43.51 |
| **1** | 1 | 431771 | 596759 | 1004227 | 2032757 | 21.24 | 29.36 | 49.40 |
| **2** | 2 | 312107 | 596886 | 1124788 | 2033781 | 15.35 | 29.35 | 55.31 |
| **3** | 3 | 274646 | 604137 | 1154491 | 2033274 | 13.51 | 29.71 | 56.78 |
| **4** | 4 | 418833 | 598040 | 1021824 | 2038697 | 20.54 | 29.33 | 50.12 |
| **5** | 5 | 318972 | 601581 | 1112561 | 2033114 | 15.69 | 29.59 | 54.72 |
| **6** | 6 | 388040 | 598029 | 1047787 | 2033856 | 19.08 | 29.40 | 51.52 |
| **7** | 7 | 454830 | 599358 | 979022 | 2033210 | 22.37 | 29.48 | 48.15 |

# TP2_2: PyTorch profiler



- Implement the PyTorch profiler in **`dlojz.py`**.
- Visualize the trace with TensorBoard and draw conclusions about possible optimizations.