Profiling AI Applications: Techniques and Best Practices

Giorgos Kosta

Research Engineer

CaSToRC

eurocc.cyi.ac.cy

A Bit About Myself

- Research Engineer at CaSToRC
- Works on DS and ML projects
- Awareness and Communications task leader for EuroCC2
- Email: g.kosta@cyi.ac.cy

Learning Objectives

- Understanding key metrics for AI application profiling.
- Introduction to PyTorch profiling and visualisation using Tensorboard.
- Interpreting profiling results.
- Optimizing your code by implementing changes based on profiling insights.

Why Profile AI Applications?

- Identification of underutilized resources:
  - Correct allocation in HPC environments.
  - Identification of bottlenecks
- It can also be a great when debugging:
  - Make informed decisions on model architecture and training.
- Scaling behavior - training larger datasets.

Key Performance Metrics - GPU metrics

- Core GPU Utilization (SM Activity):
  - Percentage of time GPU streaming multiprocessors are active.
  - Target: >80% during training indicates good compute utilization.
  - Common issues: Low utilization often indicates CPU bottlenecks or inefficient batching .

GPU Summary ⓘ

GPU 0:	
Name	Tesla V100-SXM2-32GB
Memory	31.74 GB
Compute Capability	7.0
GPU Utilization	58.85 %
Est. SM Efficiency	55.17 %
Est. Achieved Occupancy	54.37 %

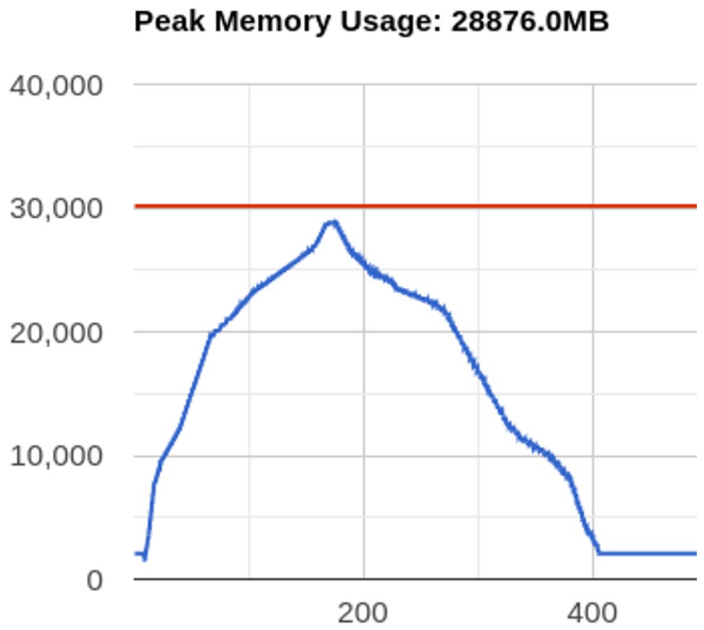
VS

GPU Summary ⓘ

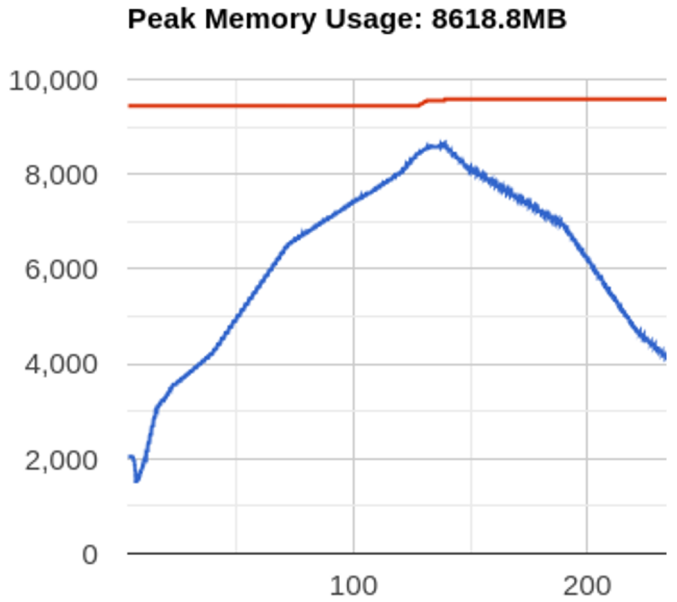
GPU 0:	
Name	Tesla V100-SXM2-32GB
Memory	31.74 GB
Compute Capability	7.0
GPU Utilization	90.99 %
Est. SM Efficiency	86.96 %
Est. Achieved Occupancy	56.72 %

Key Performance Metrics - GPU metrics

- GPU Memory Utilization:
  - Percentage of allocated vs total GPU memory.
  - Optimal range: 80-90% (leaving headroom for spikes).



VS



Key Performance Metrics - GPU metrics

- Relationship between metrics:
  - High utilization + low memory utilisation = compute bound
  - Low utilization + high memory utilisation = memory bound
  - Low both = potential CPU/data loading bottleneck

Peak Memory Usage: 28876.0MB

GPU Summary

GPU 0:	
Name	Tesla V100-SXM2-32GB
Memory	31.74 GB
Compute Capability	7.0
GPU Utilization	90.99 %
Est. SM Efficiency	86.96 %
Est. Achieved Occupancy	56.72 %

VS

Peak Memory Usage: 8618.8MB

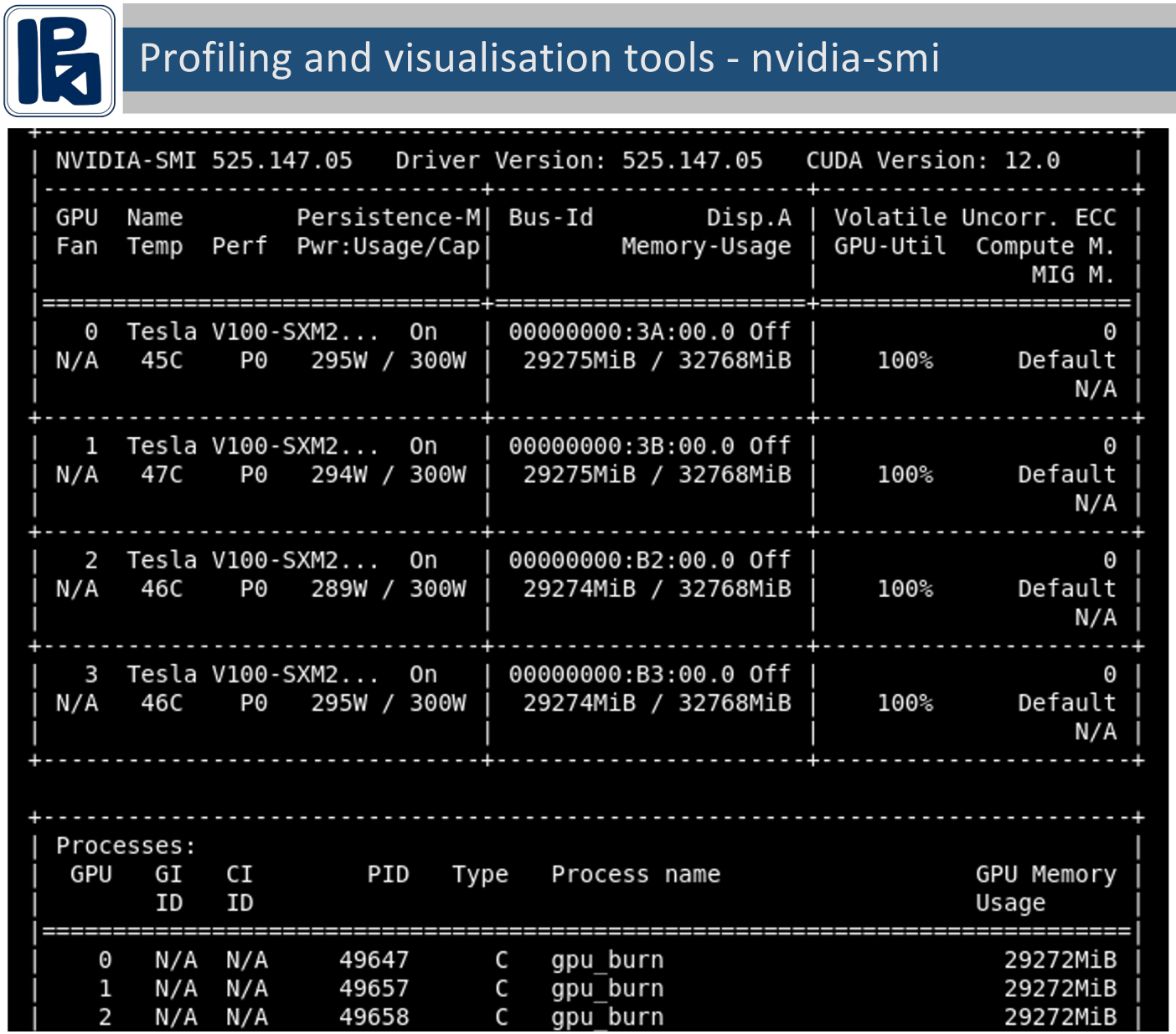
GPU Summary

GPU 0:	
Name	Tesla V100-SXM2-32GB
Memory	31.74 GB
Compute Capability	7.0
GPU Utilization	58.85 %
Est. SM Efficiency	55.17 %
Est. Achieved Occupancy	54.37 %

Key Performance Metrics - CPU metrics

- DataLoader Performance - time per batch retrieval

- Data Transformation Costs
  - Tokenization time (for NLP tasks)
  - Image preprocessing operations
  - Data augmentation computation time



Profiling and visualisation tools - nvidia-smi

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.147.05   Driver Version: 525.147.05   CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000000:3A:00.0 Off |                    0 |
| N/A   45C    P0   295W / 300W |  29275MiB / 32768MiB |    100%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000000:3B:00.0 Off |                    0 |
| N/A   47C    P0   294W / 300W |  29275MiB / 32768MiB |    100%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  Tesla V100-SXM2...  On   | 00000000:B2:00.0 Off |                    0 |
| N/A   46C    P0   289W / 300W |  29274MiB / 32768MiB |    100%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  Tesla V100-SXM2...  On   | 00000000:B3:00.0 Off |                    0 |
| N/A   46C    P0   295W / 300W |  29274MiB / 32768MiB |    100%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A     49647      C   gpu_burn                        29272MiB |
|    1   N/A  N/A     49657      C   gpu_burn                        29272MiB |
|    2   N/A  N/A     49658      C   gpu_burn                        29272MiB |
```

Command:
$ nvidia-smi

Useful usage:
$ watch -n 2 nvidia-smi

Profiling and visualisation tools - torch profiler

Limits trace file size on long running jobs

```
with torch.profiler.profile(
    schedule=torch.profiler.schedule(
        wait=1,     # Number of steps to wait before profiling
        warmup=1,   # Number of warmup steps before recording
        active=3,   # Number of steps to record
        repeat=1    # Number of times to repeat the cycle
    ),
    on_trace_ready=torch.profiler.tensorboard_trace_handler(log_dir),
    record_shapes=True,
    profile_memory=True,
    with_stack=True
) as prof:
    for step, batch_data in enumerate(train_loader):
        loss = train_step(  model=model, data=batch_data, criterion=criterion,
                            optimizer=optimizer, scaler=scaler, device=device)
        prof.step()

        if step >= (1 + 1 + 3)*10:  # wait + warmup + active steps
            break
```

Records stack trace of program

Profiling and visualisation tools - torch profiler

```python
with record_function("batch_processing"):
    # Move data to GPU
    input_ids = batch['input_ids'].cuda()
    attention_mask = batch['attention_mask'].cuda()
    labels = batch['label'].cuda()

    # Simulate some computation
    with record_function("model_computation"):
        time.sleep(0.01)  # Simulate model forward pass

    # Ensure GPU operations are complete
    torch.cuda.synchronize()
```

Average time per batch: 18.07 ms

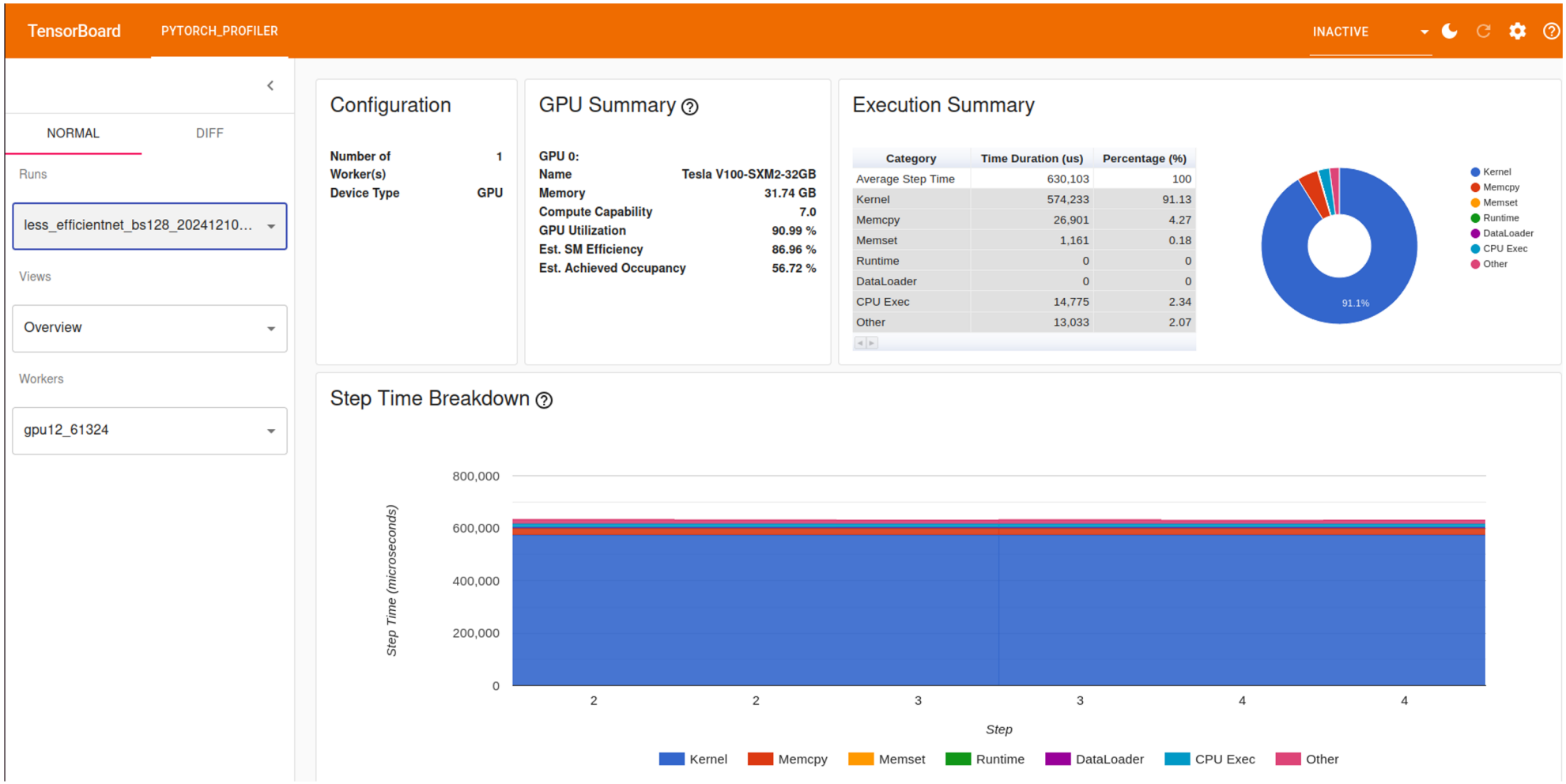
Detailed profile by operation type:

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg
ProfilerStep*	1.80%	606.000us	98.34%	33.187ms	11.062ms
batch_processing	2.54%	856.000us	94.02%	31.727ms	10.576ms
model_computation	90.13%	30.414ms	90.14%	30.417ms	10.139ms
enumerate(DataLoader)#_MultiProcessingDataLoaderIter...	2.28%	769.000us	2.31%	779.000us	259.667us
aten::zeros	1.35%	455.000us	1.73%	583.000us	48.583us
aten::to	0.07%	22.000us	1.25%	421.000us	46.778us
aten::_to_copy	0.32%	108.000us	1.18%	399.000us	44.333us
aten::copy_	0.15%	50.000us	0.73%	248.000us	27.556us
aten::empty	0.50%	170.000us	0.50%	170.000us	7.083us
cudaStreamSynchronize	0.41%	137.000us	0.41%	137.000us	15.222us

Profiling and visualisation tools - torch profiler

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time a
ProfilerStep*	0.19%	416.000us	99.97%	221.120ms	73.707
enumerate(DataLoader)#_MultiProcessingDataLoaderIter...	85.66%	189.461ms	85.72%	189.594ms	63.198
batch_processing	0.12%	264.000us	14.05%	31.082ms	10.361
model_computation	13.65%	30.197ms	13.65%	30.199ms	10.066
aten::to	0.02%	35.000us	0.27%	607.000us	25.292
aten::_to_copy	0.02%	38.000us	0.26%	572.000us	63.556
aten::copy_	0.02%	55.000us	0.22%	484.000us	53.778
cudaMemcpyAsync	0.13%	295.000us	0.13%	295.000us	32.778
cudaStreamSynchronize	0.06%	134.000us	0.06%	134.000us	14.889
aten::empty	0.05%	109.000us	0.05%	109.000us	2.795

Profiling and visualisation tools - Tensorboard - Overview

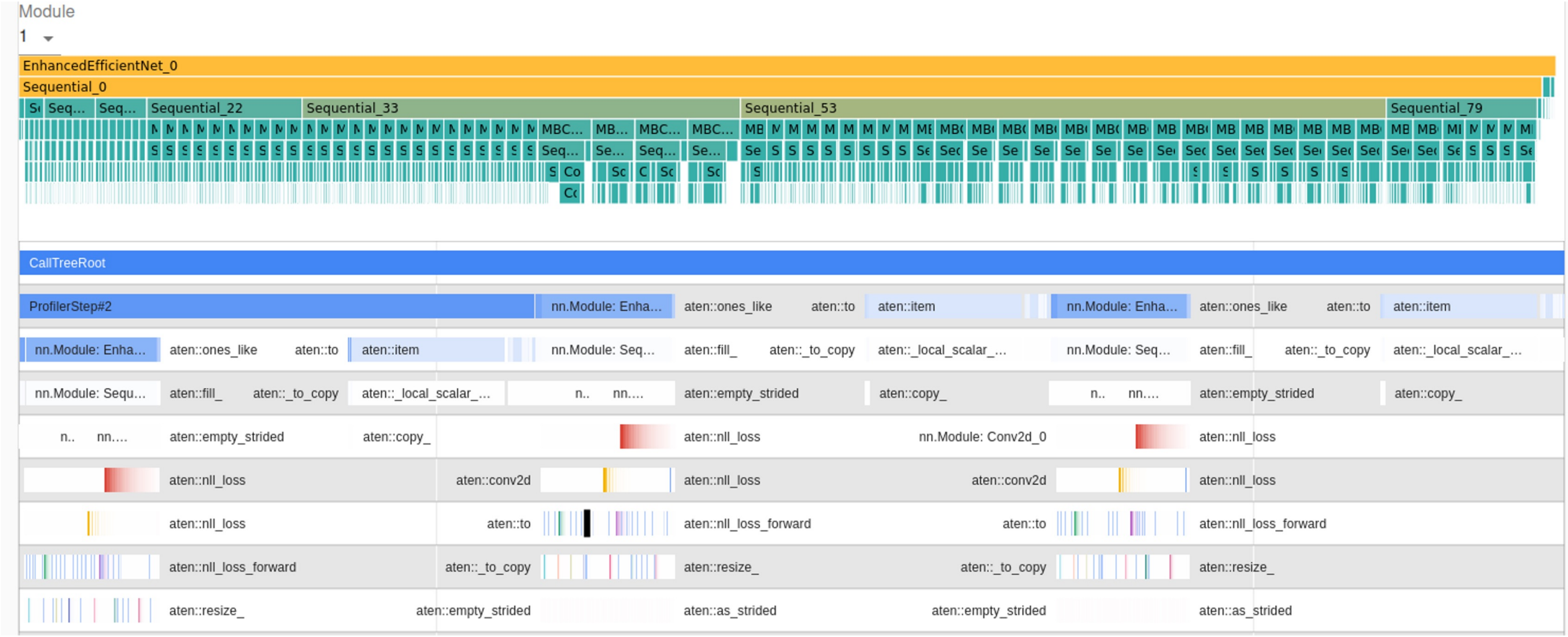


Profiling and visualisation tools - Tensorboard - Module view

Module View

Module Name	Occurences	Operators	Host Total Time	Host Self Time	Device Total Time	Device Self Time
CrossEntropyLoss_0	3	3	444.078125	98.3544921875	0	0
− EnhancedEfficientNet_0	3	9	482772.07421875	236.7734375	0	0
+ Sequential_0	3	0	478405.6484375	130.7431640625	0	0
AdaptiveAvgPool2d_61	3	3	147.2841796875	60.814453125	0	0
− Sequential_87	3	0	2499.5439453125	142.298828125	0	0
Linear_0	3	3	680.4931640625	53.5078125	0	0
ReLU_0	3	3	202.0869140625	53.4443359375	0	0
Dropout_0	3	3	341.5185546875	79.119140625	0	0

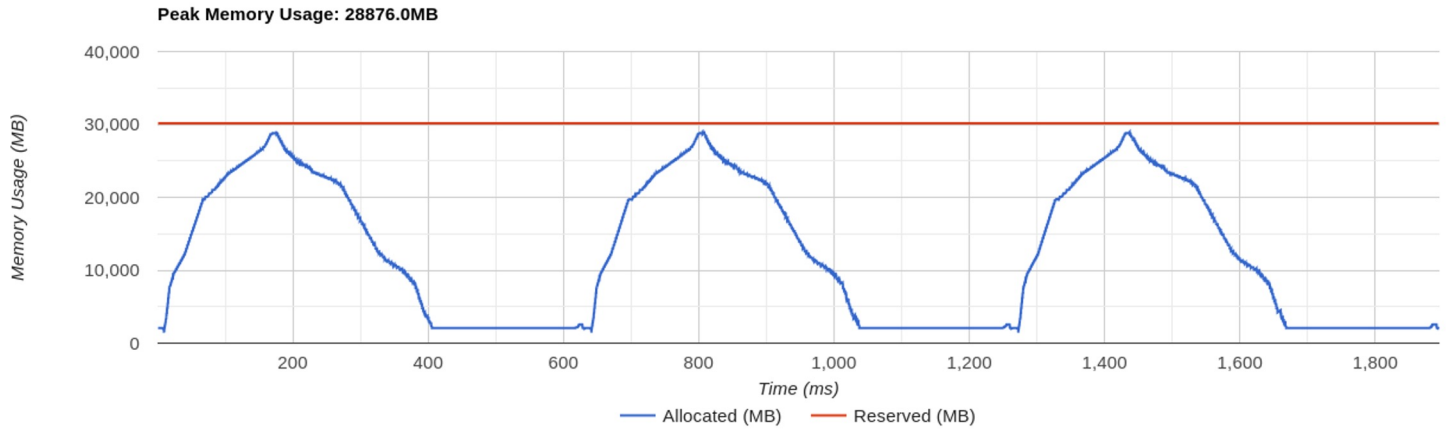
Profiling and visualisation tools - Tensorboard - Module view



Profiling and visualisation tools - Tensorboard - Module view

Memory View

Device
GPU0



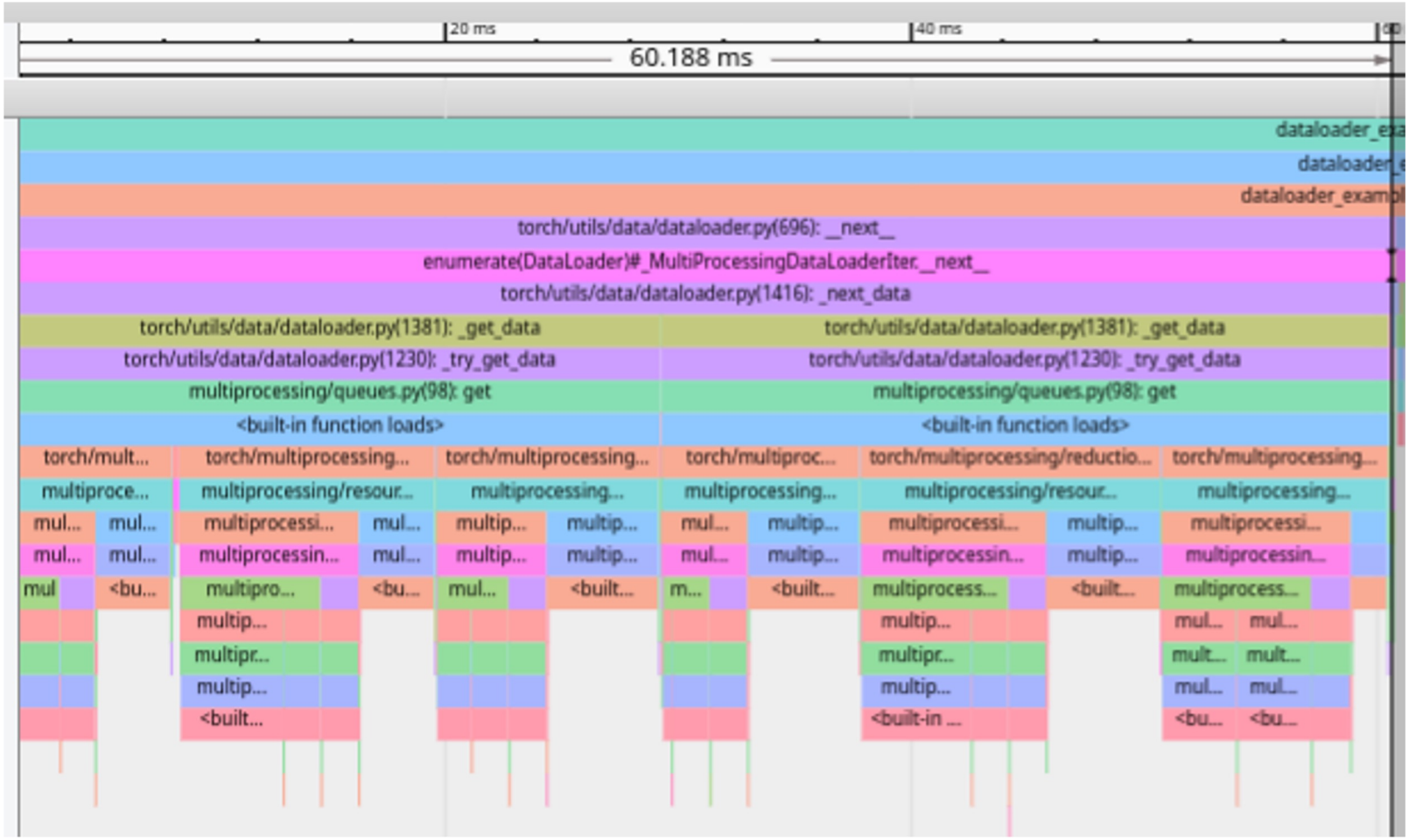
Search by Name

Min Size(KB) 64232.125

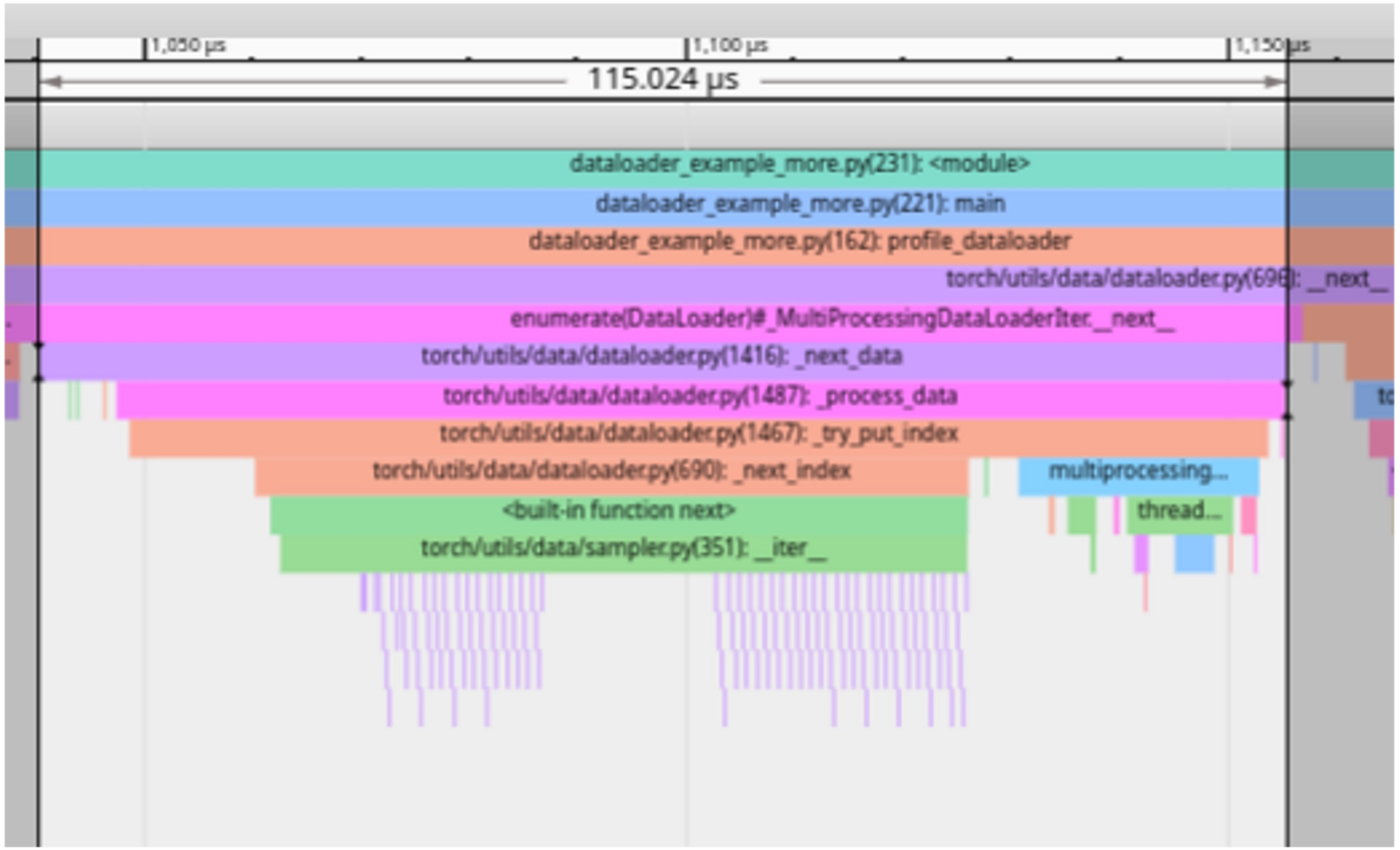
Max Size(KB) 256928.5

Operator	Size (KB)	Allocation Time (ms)	Release Time (ms)	Duration (ms)
aten::cudnn_convolution	100444	10.31	10.37	0.05
aten::cudnn_convolution	100444	10.84	10.88	0.04
aten::empty_like (aten::empty)	100352	10.98	11.27	0.29
aten::cudnn_convolution	100444	11.41	11.45	0.04
aten::empty_like (aten::empty)	100352	11.55	11.81	0.25
aten::cudnn_convolution	100444	11.96	12	0.04

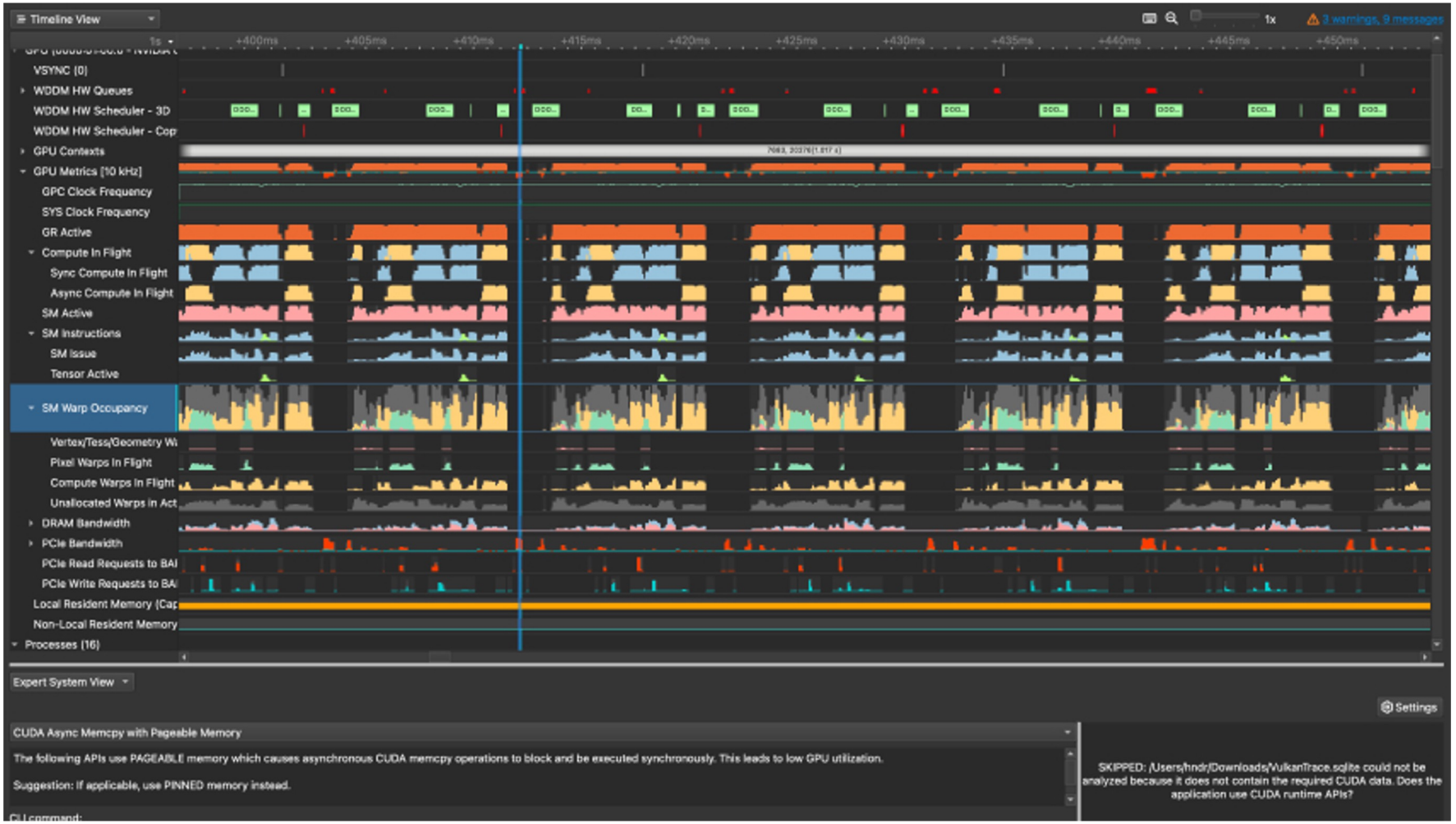
Profiling and visualisation tools - Tensorboard - Trace view (slow)



Profiling and visualisation tools - Tensorboard - Trace view (fast)



Profiling and visualisation tools - NSight



https://developer.nvidia.com/nsight-systems

Thank you for the attention!

More information:

https://castorc.cyi.ac.cy/
https://eurocc.cyi.ac.cy/

Contact us at:
eurocc-contact@cyi.ac.cy

Funded by the European Union. This work has received funding from the European High Performance Computing Joint Undertaking (JU) and Germany, Bulgaria, Austria, Croatia, Cyprus (co-funded by the EU within the framework of the Cohesion Policy Programme "THALIA 2021-2027"), Czech Republic, Denmark, Estonia, Finland, Greece, Hungary, Ireland, Italy, Lithuania, Latvia, Poland, Portugal, Romania, Slovenia, Spain, Sweden, France, Netherlands, Belgium, Luxembourg, Slovakia, Norway, Türkiye, Republic of North Macedonia, Iceland, Montenegro, Serbia under grant agreement No 101101903.

Practical

1. Launch tensorboard:

```
$ cd edu26
$ cp -r logs_pactical1/ ~/
$ ./launch_tensorboard.sh
```

2. Launch python examples on compute nodes.

```
$ cp -r practical1/ ~/
$ cd ../practical1/
$ sbatch launch_dataloader.py
```

https://developer.nvidia.com/nsight-systems