DASYA

www.dasya.dk
@dasyaITU

RAD

https://rad.itu.dk

INDEPENDENT RESEARCH FUND DENMARK

dff.dk

IT UNIVERSITY OF COPENHAGEN

www.itu.dk

Profiling & Monitoring Deep Learning Training Tasks

Ehsan Yousefzadeh-Asl-Miandoab ***(ehyo@itu.dk),***
Ties Robroek (titr@itu.dk), Pınar Tözün (pito@itu.dk)
IT University of Copenhagen

3rd EuroMLSys workshop – EuroSys ’23, Rome, Italy

08/05/2023

GPU Underutilization for ML Workloads

- An analysis of 100,000 jobs run by 100s of users for ~2 months on a real-world cluster shows ~52% GPU utilization on average*
  - Energy-inefficient & waste of hardware resources
- Compute/memory requirements of models don't match with the giant GPUs
  - e.g., transfer learning, small models

Memory: 80GB

L2 cache: 50MB

Memory Bandwidth: 3000 GB/sec

BASIC-L: ~2.44billion parameters

CAIT-M-48-448: ~438million parameters

ResNet50: ~25million parameters

Thus, understanding the profilers and monitoring tools for GPUs is necessary.

* Jeon, Myeongjae, et al. "**Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads**." *USENIX Annual Technical Conference*. 2019.

Profilers

PyTorch Profiler

- Trace-based
- Runs as part of the training process
- Easier to use
  - a few lines of additional code

NVIDIA Nsight Systems (nsys)

- Trace-based system-wide
- Runs as a separate process
- More detailed insights to OS & network
- Doesn’t work when Multi-Instance GPU (MIG) is enabled on the GPU

NVIDIA Nsight Compute (ncu)

- Kernel-level tracing of microarchitectural behavior
- Runs as a separate process
- Intrusive to program behavior
  - Runs the program several times

Monitoring tools

NVIDIA System Management Interface (nvidia-smi)

- Performance configuration (frequency changing, MIG config)
- Tracking a range of high-level performance metrics
  - GPU Utilization
  - Memory Consumption
  - …
- Doesn’t monitor MIG instances

NVIDIA Data Center GPU Manager (dcgm)

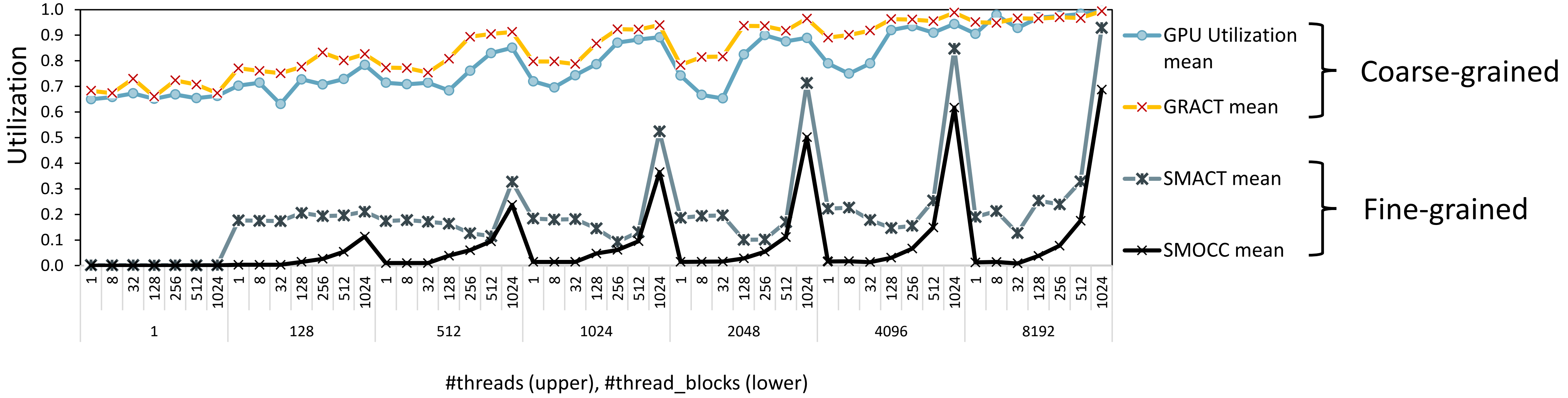
- Easier management by grouping option
- Finer-grained performance metrics for monitoring
  - SM Active (SMACT)
  - SM Occupancy (SMOCC)
  - …
- Can monitor MIG instances

Experimental Setup

- Goal: Understanding GPU utilization metrics; overheads and strengths of the tools
- Experiment 1: A microbenchmark to analyze GPU utilization metrics
- Experiment 2: Model runs to analyze the overheads
  - On PyTorch 1.13.1 with 5 epochs
  - *Light workload*: Small CNN on MNIST
  - *Heavy workload*: ResNet50 on ImageNet, batch-size = 32
- Hardware: NVIDIA DGX A100 Station
  - 4X A100 40 GB
  - 1X EPYC 7742, 64 cores
  - RAM: 512 GB
- Tools
  - Default settings for PyTorch Profiler and Nsight Systems
  - Omitted Nsight Compute due to its intrusive nature

GPU Utilization

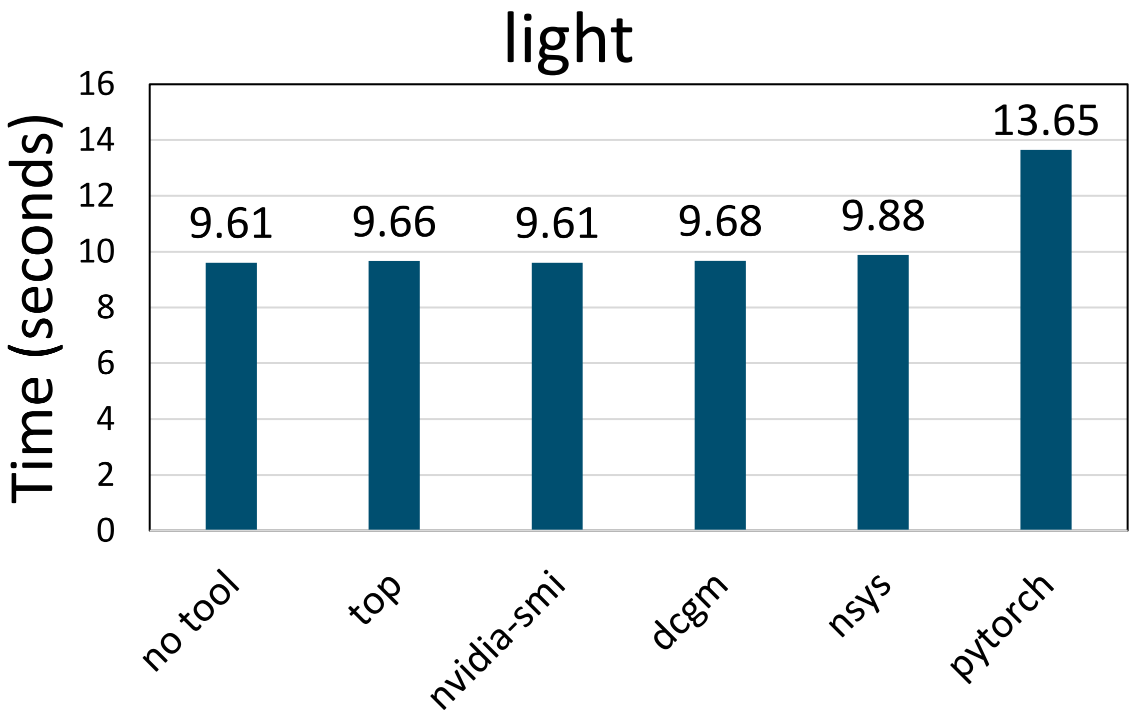
- **GPU Utilization**: % of time one or more kernels were executing on the GPU
- **GRACT**: % of time any portion of the graphics or compute engines were active
- **SMACT:** the fraction of active time on an SM, averaged over all SMs
- **SMOCC**: degree of parallelism / max supported parallelism on SM

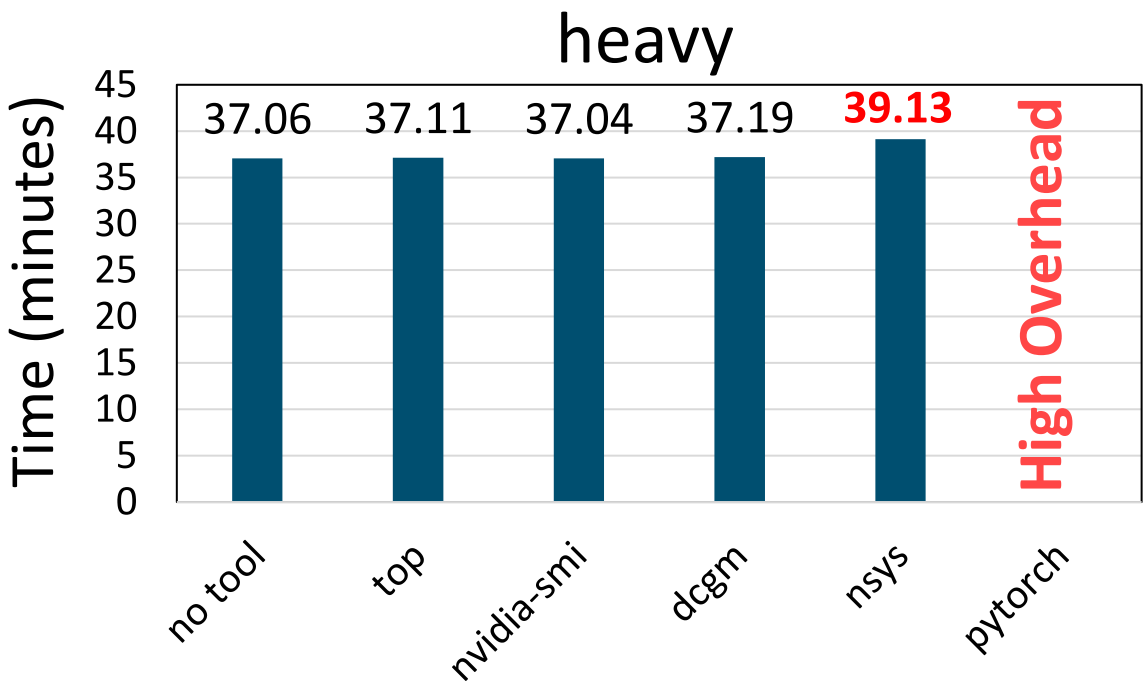


Coarser-grained utilization metrics can be misleading.

Time overhead of tools

Average Epoch Time





➔ **Monitoring tools have negligible time overhead.**
➔ **Profilers' overhead is noticeable.**

➔ **Profiling just for one iteration might be enough.**

Ehsan Yousefzadeh-Asl-Miandoab et al. **"Profiling & Monitoring Deep Learning Training Tasks"**

Space overhead of tools

Tool	Small CNN	ResNet50
top	~20KB	~2MB
nvidia-smi	~20KB	~2MB
dcgm	~85KB	~8MB
nsys	~40MB	~5GB
pytorch	~1.4GB	-

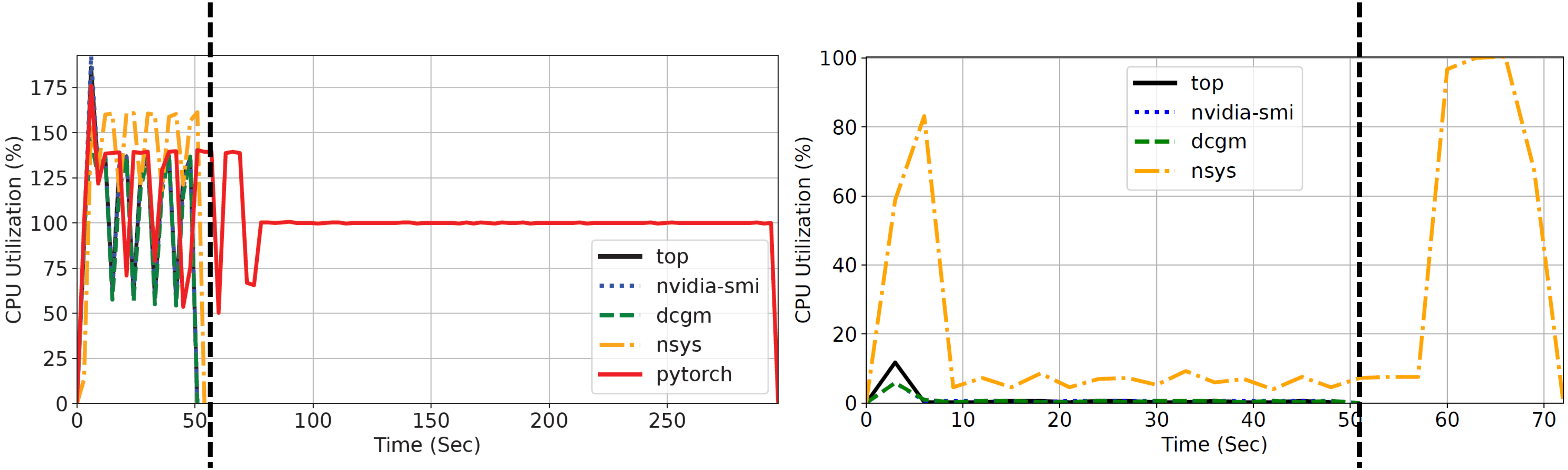
➔ Trends for space overhead are similar to time overhead for all tools.

CPU overhead

Light model

Training processes

Tools' processes



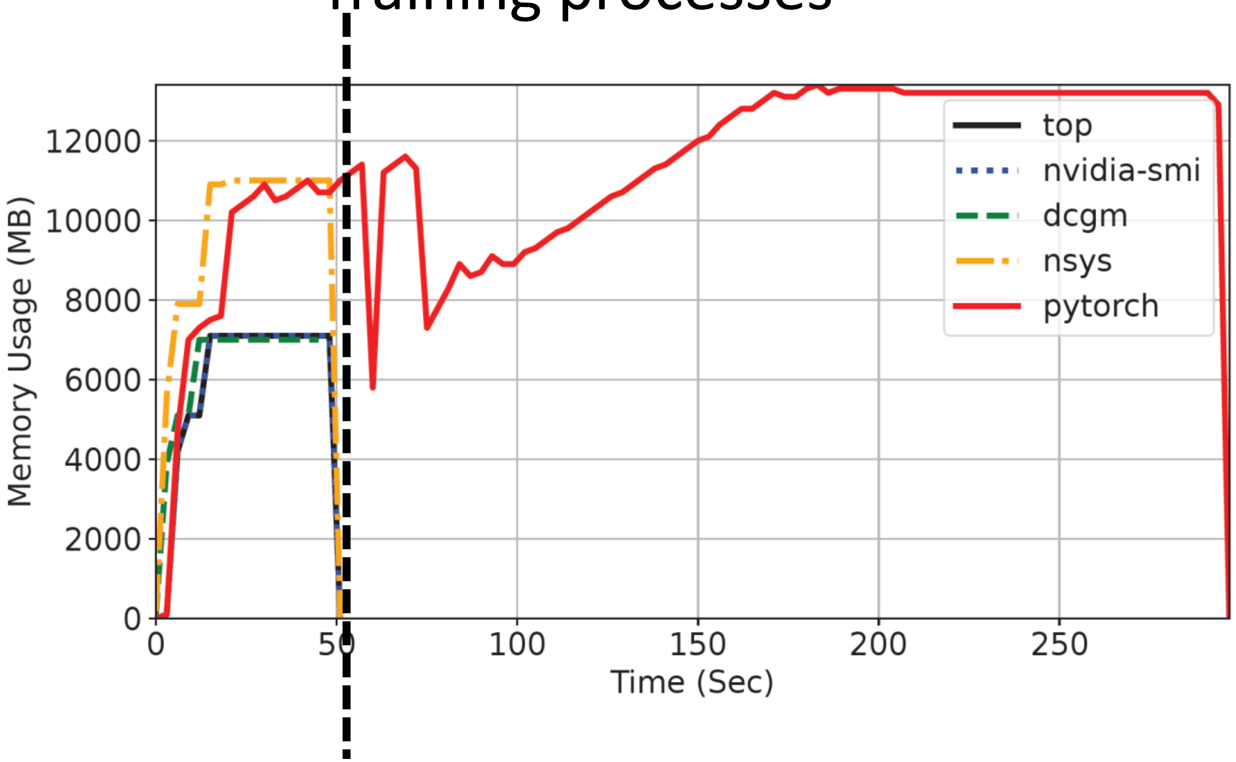
→ CPU usage overhead of profiling tools is higher than monitoring ones.
→ Profiling tools also need time for post-processing of collected traces.

Ehsan Yousefzadeh-Asl-Miandoab et al. **"Profiling & Monitoring Deep Learning Training Tasks"**

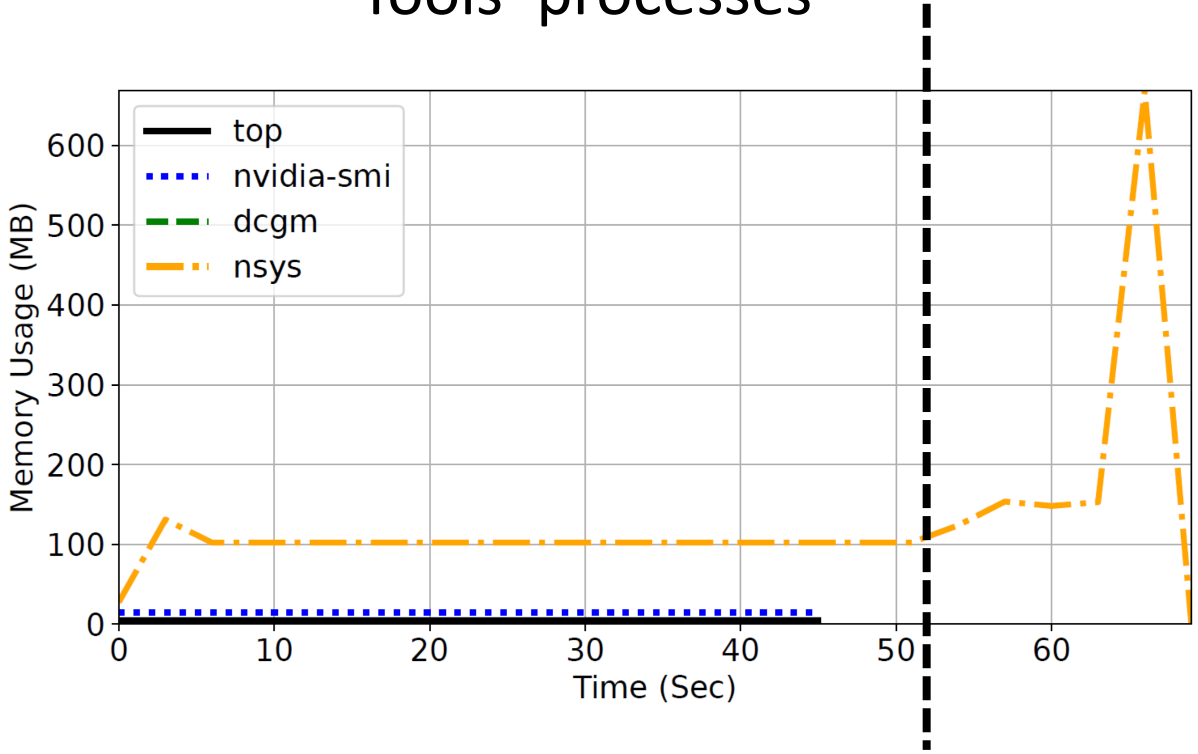
CPU memory overhead

Light model

Training processes



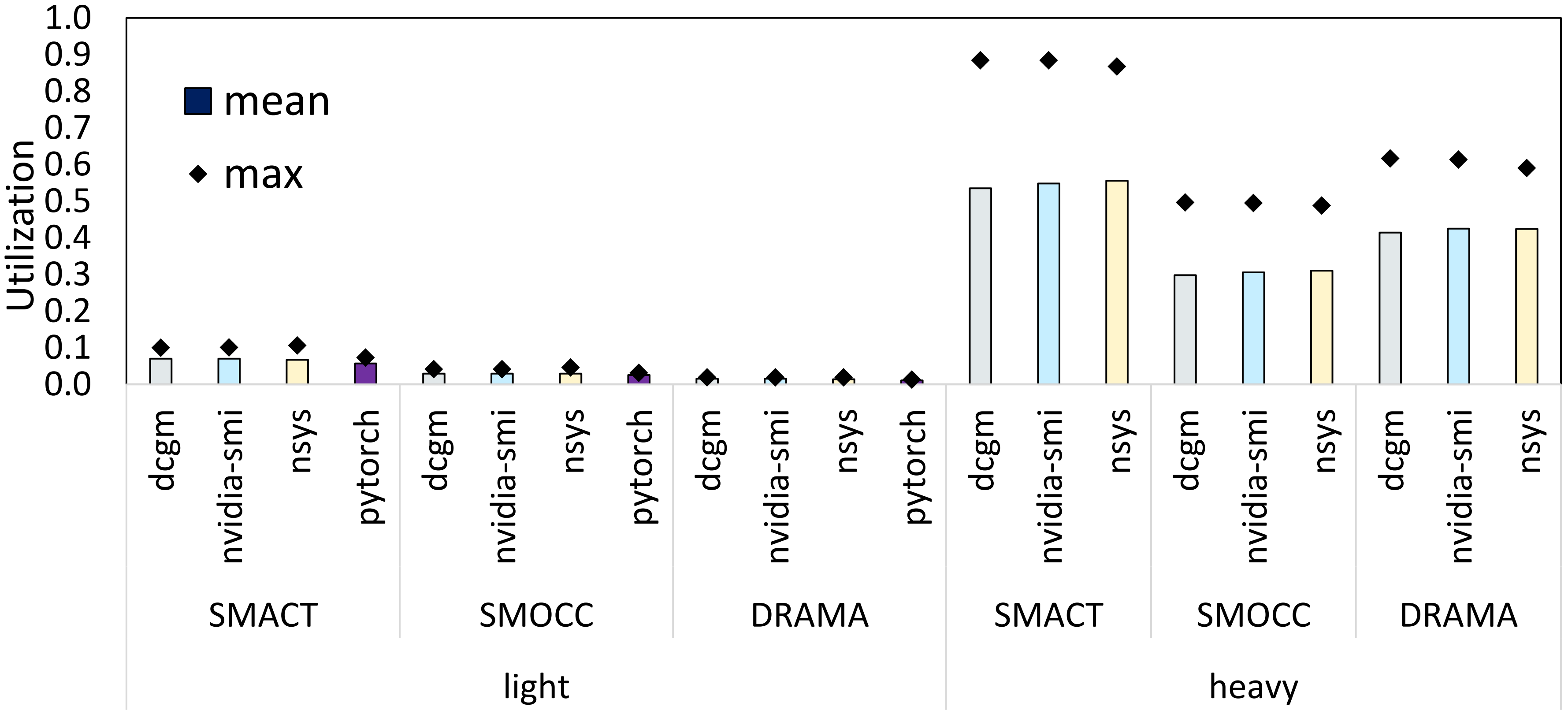
Tools' processes



Memory overhead of profiling tools is also higher than monitoring tools'!

Ehsan Yousefzadeh-Asl-Miandoab et al. **"Profiling & Monitoring Deep Learning Training Tasks"**

GPU overhead



tool, metric, model size from top to bottom

GPU overhead of all the tools is negligible!

Summary – Insights

- For model level optimization purposes
  - Use framework specific profilers
- For digging deeper into OS and system
  - Use Nsight Systems
- For kernel-level optimizations
  - Use Nsight Compute
- Profile the needed amount of code for a reasonable range of time
  - Profiling for an iteration might be enough to show the behavior of training a model
- For online decision-making purposes
  - Use monitoring tools with representative fine-grained metrics

Thanks! ☺