DASYA

RAD

IT UNIVERSITY OF COPENHAGEN

www.dasya.dk
@dasyaITU

rad.itu.dk

www.itu.dk

towards transparency in computational footprint of deep learning

Pınar Tözün

Associate Professor, IT University of Copenhagen

pito@itu.dk, pinartozun.com, @pinartozun

INDEPENDENT RESEARCH FUND DENMARK

novo nordisk foundation

ESwML
22/04/2024

background

IT UNIVERSITY OF COPENHAGEN
Copenhagen, Denmark
2018 – present

- Associate Professor
  Resource-aware & -constrained ML

IBM
San Jose, CA, USA
2015 – 2018

- Research staff member
  Building an HTAP system
  – commercialized as IBM Db2 Event Store

EPFL DIAS
Lausanne, Switzerland
2009 – 2014

- PhD student
  Scalable OLTP on Multicores

KOÇ UNIVERSITY
Istanbul, Turkey
2005 – 2009

- BSc student
  Efficient data race detection

challenge#1: unsustainable growth

2012 → **present**

- powerful hardware
- larger datasets
- deep learning frameworks

APIs (python, R, …)

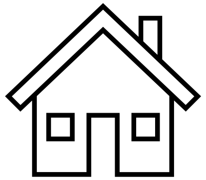
deep learning software

PyTorch TensorFlow

CPU | GPU | TPU | FPGA | …

commodity hardware

- ***>> 5 orders of magnitude increase*** in the ***computational need*** for models.
- estimated carbon footprint for large language model training = ***average yearly energy of several US homes***



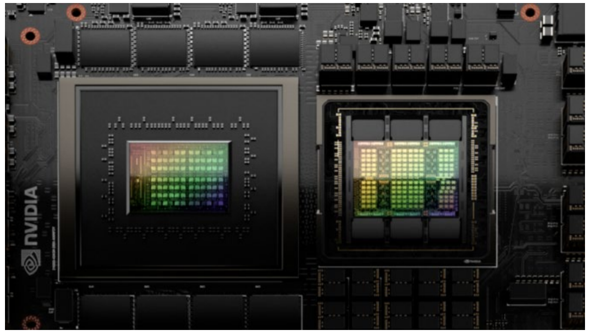
need for higher computational efficiency!

sources: https://openai.com/blog/ai-and-compute/

Dodge et al. "Measuring the Carbon Intensity of AI in Cloud Instances." FAccT 2022

challenge#2: hardware underutilization

NVIDIA H200



141GB GPU memory

50MB L2 cache

4.8TB/s Memory Bandwidth

- ***@ITU***, jobs of data scientists utilize ***less than 50% of GPU resources*** e.g., transfer learning, small models
- ***in real-world*, ~52% GPU utilization*** on average for 100,000 jobs

need for higher computational efficiency!

*Jeon et al. "[Analysis of Large-Scale Multi-Tenant GPU Clusters for DNN Training Workloads]." *ATC* 2019

need for higher computational efficiency

➔ **how to *quantify* computational efficiency?**

- profiling & monitoring tools for GPUs

[EuroMLSys 2023]

➔ **how to make the process *systematic*?**

- resource-aware data science tracker (radT)

[DEEM 2023]

profilers

PyTorch profiler

- framework specific
- runs as part of the training process
- easy to use
  - a few lines of additional code

NVIDIA Nsight Systems (nsys)

- system-wide
- runs as separate process
- more detailed insights to OS & network

NVIDIA Nsight Compute (ncu)

- kernel-level tracing of micro-architectural behavior
- runs as separate process
- finer-grained insights
- intrusive
  - iterates over 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)

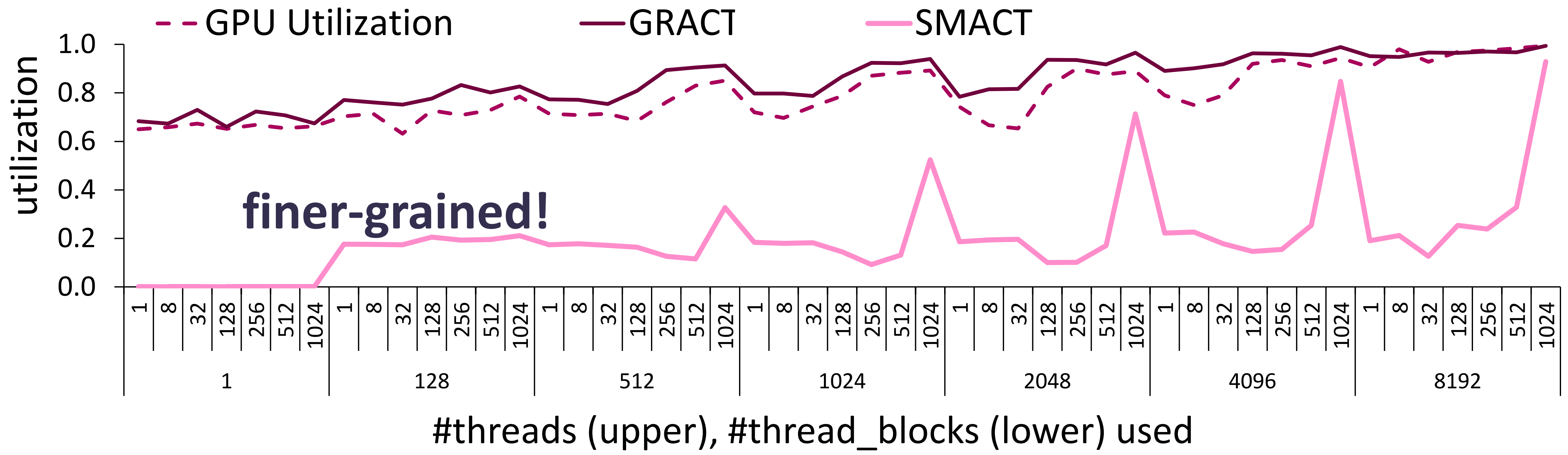
- can create GPU groups
- a wider range of and finer-grained performance metrics for monitoring
  - SM Active (SMACT)
  - SM Occupancy (SMOCC)
  - …
- can monitor MIG instances

*on H100s, unlike on A100s, nvidia-smi can also track additional metrics such as SMACT.

GPU utilization

each thread fetches a data item and takes its square

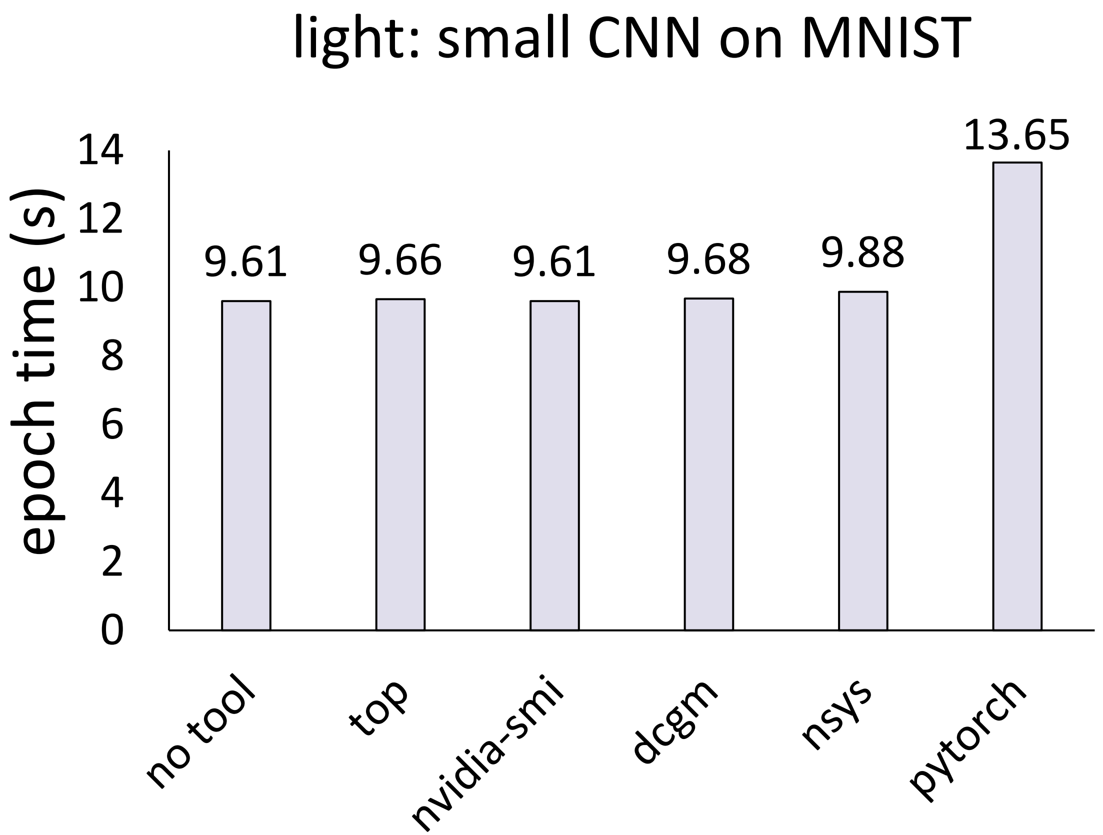
- **GPU utilization**: % of time one or more kernels were executing on the GPU
- **GRACT**: % of time any portion of the graphics or compute engines were active
- **SMACT:** the fraction of active time on an SM, averaged over all SMs = streaming multiprocessor

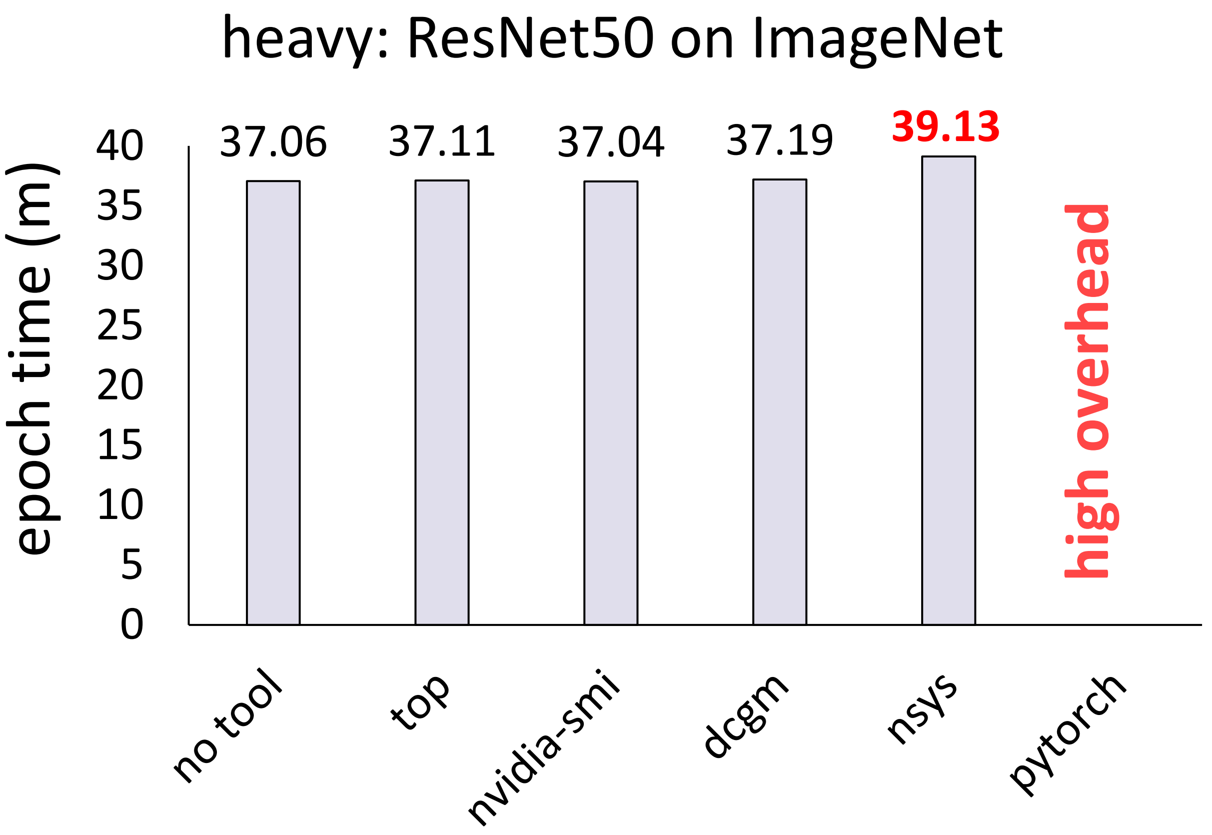


coarse-grained GPU utilization metrics could be misleading!

time overhead of tools

5 epochs on PyTorch 1.31 & DGX A100 Station





→monitoring tools have negligible time overhead.

→profilers' overhead is noticeable.

→profiling just for one iteration might be enough.

space overhead of tools

5 epochs on PyTorch 1.31 & DGX A100 Station

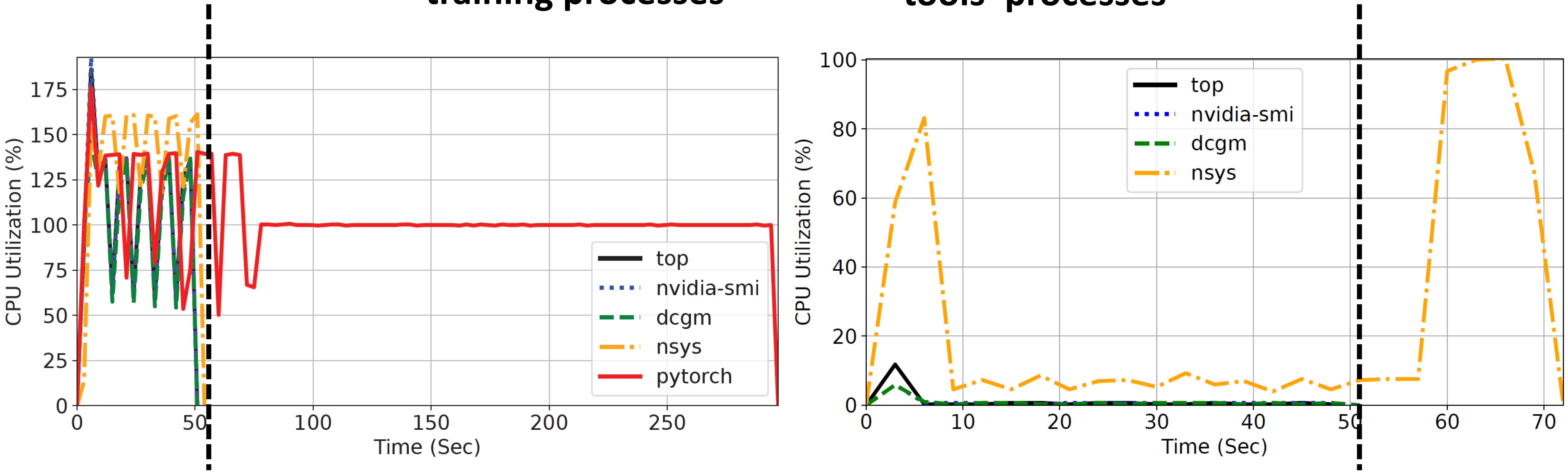
tool	light: small CNN on MNIST	heavy: ResNet50 on ImageNET
top	~20KB	~2MB
nvidia-smi	~20KB	~2MB
dcgm	~85KB	~8MB
nsys	~40MB	~5GB
pytorch	~1.4GB	-

trends for space overhead are similar to time overhead for all tools

CPU overhead

light model: small CNN on MNIST, 5 epochs
on PyTorch & DGX A100 Station

training processes **tools' processes**



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

insights

- for model level optimization → 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
  - an iteration may be enough to show the behavior of training a model
- for online decision making
  → use monitoring tools with representative fine-grained metrics

need for higher computational efficiency

➔ **how to *quantify* computational efficiency?**

- profiling & monitoring tools for GPUs

[EuroMLSys 2023]

➔ **how to make the process *systematic*?**

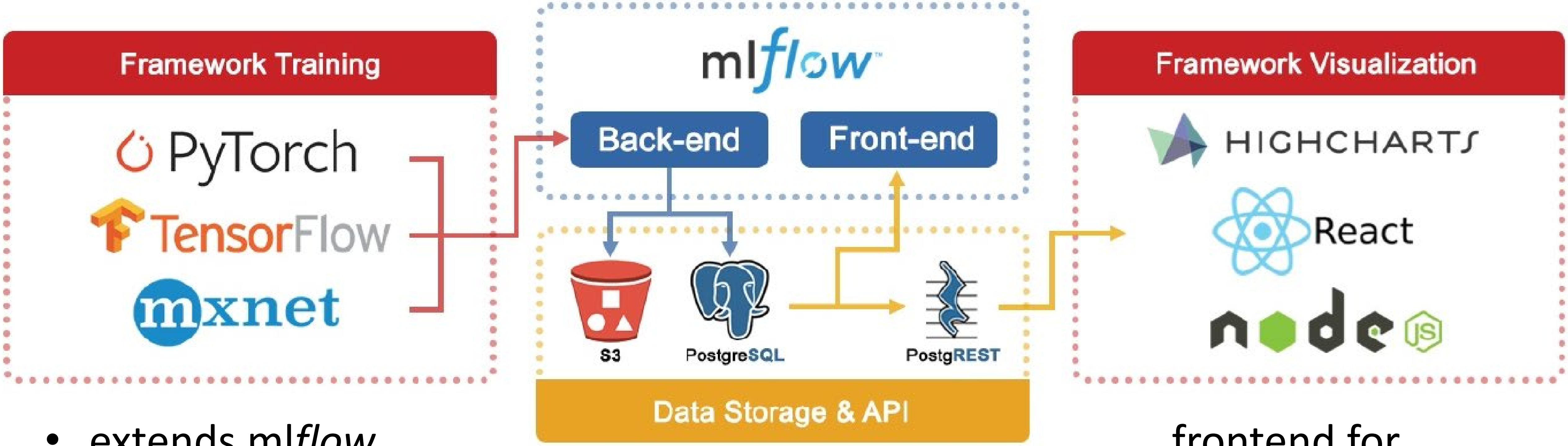
- resource-aware data science tracker (radT)

[DEEM 2023]

requirements

- wide configuration support including collocation on GPUs
- track hardware metrics in addition to software metrics
- handle continuous streams of experimental data
- support efficient visualization for experimental data exploration
- filter large amounts of inconsequential data
- minimal code impact

radT



- extends ml*flow*
- incorporates collocation
- allows easy, extensible, and scalable tracking of hardware metrics on CPUs & GPUs

frontend for data exploration

used by several members of our group, including data scientists, for systematic benchmarking of deep learning

experiment configuration

```
Experiment,Workload,Status,Run,Devices,Collocation,      File,         Listeners,Params
           1,           1,    ,   ,      0,              -,train.py,smi+top+dcgm,--batch-size 128
           1,           1,    ,   ,      1,              -,train.py,smi+top+dcgm,--batch-size 128
           1,           2,    ,   ,      2,        3g.20gb,train.py,smi+top+dcgm,--batch-size 128
           1,           2,    ,   ,      2,        3g.20gb,train.py,smi+top+dcgm,--batch-size 128
           1,           3,    ,   ,      1,            MPS,train.py,smi+top+dcgm,--batch-size 128
           1,           3,    ,   ,      1,            MPS,train.py,smi+top+dcgm,--batch-size 128
```

experiment configuration

```
Experiment,Workload,Status,Run,Devices,Collocation,      File,        Listeners,Params
         1,       1,     ,   ,      0,          -,train.py,smi+top+dcgm,--batch-size 128
         1,       1,     ,   ,      1,          -,train.py,smi+top+dcgm,--batch-size 128
         1,       2,     ,   ,      2,    3g.20gb,train.py,smi+top+dcgm,--batch-size 128
         1,       2,     ,   ,      2,    3g.20gb,train.py,smi+top+dcgm,--batch-size 128
         1,       3,     ,   ,      1,        MPS,train.py,smi+top+dcgm,--batch-size 128
         1,       3,     ,   ,      1,        MPS,train.py,smi+top+dcgm,--batch-size 128
```

experiment configuration

```
Experiment,Workload,Status,Run,Devices,Collocation,File,Listeners,Params
1,1,,,0,-,train.py,smi+top+dcgm,--batch-size 128
1,1,,,1,-,train.py,smi+top+dcgm,--batch-size 128
1,2,,,2,3g.20gb,train.py,smi+top+dcgm,--batch-size 128
1,2,,,2,3g.20gb,train.py,smi+top+dcgm,--batch-size 128
1,3,,,1,MPS,train.py,smi+top+dcgm,--batch-size 128
1,3,,,1,MPS,train.py,smi+top+dcgm,--batch-size 128
```

experiment configuration

Experiment	Workload	Status	Run	Devices	Collocation	File	Listeners	Params
1	1			0	-	train.py	smi+top+dcgm	--batch-size 128
1	1			1	-	train.py	smi+top+dcgm	--batch-size 128
1	2			2	3g.20gb	train.py	smi+top+dcgm	--batch-size 128
1	2			2	3g.20gb	train.py	smi+top+dcgm	--batch-size 128
1	3			1	MPS	train.py	smi+top+dcgm	--batch-size 128
1	3			1	MPS	train.py	smi+top+dcgm	--batch-size 128

code impact

for hardware monitoring, no code changes needed

$ python -m radt model.py --batch-size 256 → single model training

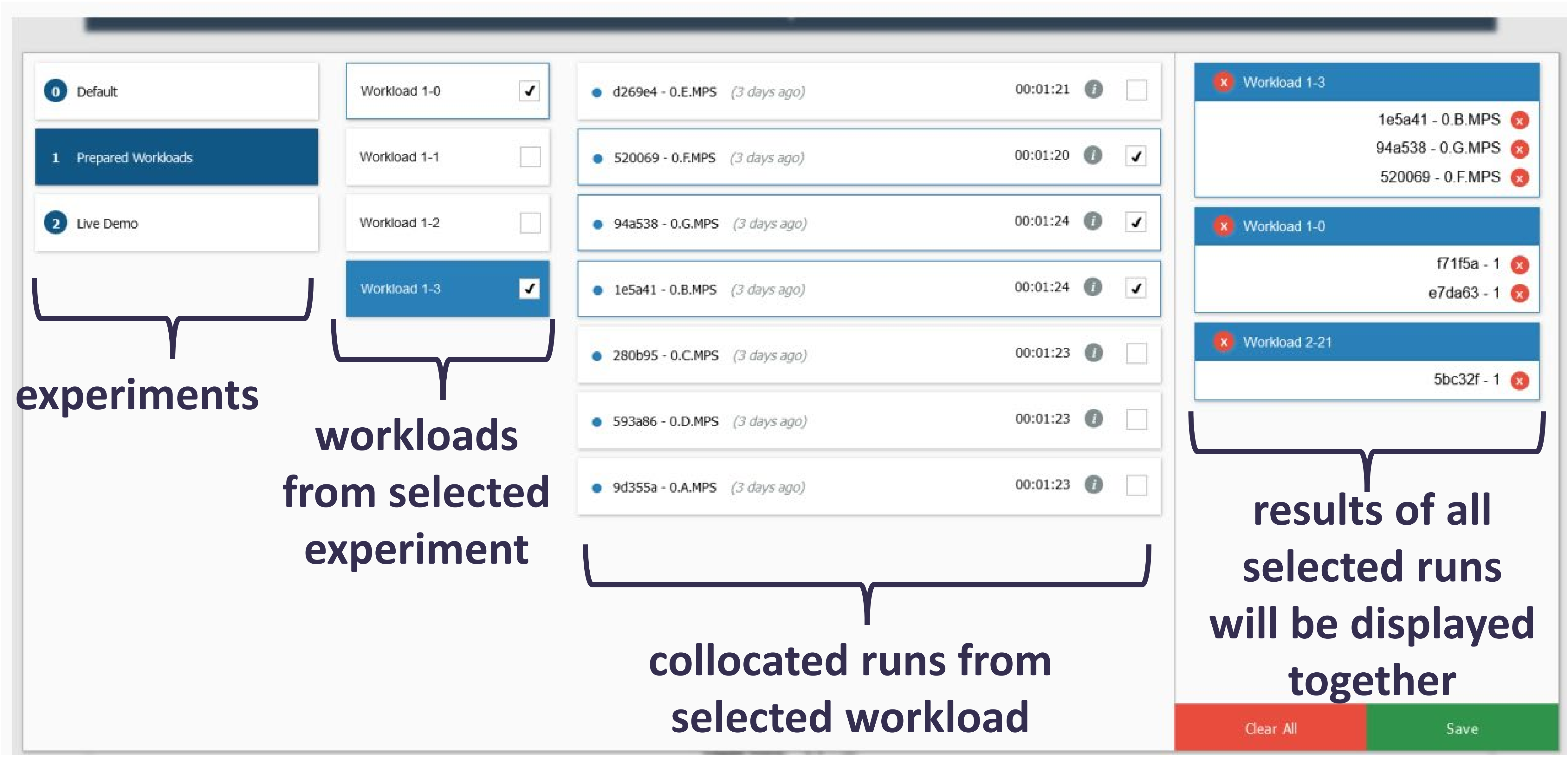
$ python -m radt config.csv → bigger experiment

for customized control over machine learning metrics

```
import radt

with radt.run.RADTBenchmark() as run:
  # training loop
  run.log_metric("Metric A", amount)
  run.log_artifact("artifact.file")
```

frontend


0 Default
1 Prepared Workloads
2 Live Demo
Workload 1-0
Workload 1-1
Workload 1-2
Workload 1-3
d269e4 - 0.E.MPS (3 days ago) 00:01:21
520069 - 0.F.MPS (3 days ago) 00:01:20
94a538 - 0.G.MPS (3 days ago) 00:01:24
1e5a41 - 0.B.MPS (3 days ago) 00:01:24
280b95 - 0.C.MPS (3 days ago) 00:01:23
593a86 - 0.D.MPS (3 days ago) 00:01:23
9d355a - 0.A.MPS (3 days ago) 00:01:23
Workload 1-3
1e5a41 - 0.B.MPS
94a538 - 0.G.MPS
520069 - 0.F.MPS
Workload 1-0
f71f5a - 1
e7da63 - 1
Workload 2-21
5bc32f - 1
experiments
workloads from selected experiment
collocated runs from selected workload
results of all selected runs will be displayed together
Clear All
Save

frontend – demo

variety of visualization options

can download as

- **jpg, png, pdf – to display as image**
- **csv – for drawing it differently**
- **json – to share the exploration results**

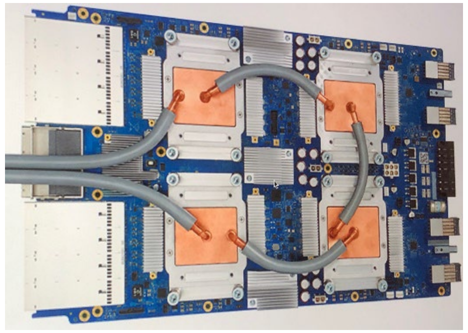
radT for systematic benchmarking

https://github.com/Resource-Aware-Data-systems-RAD/radt

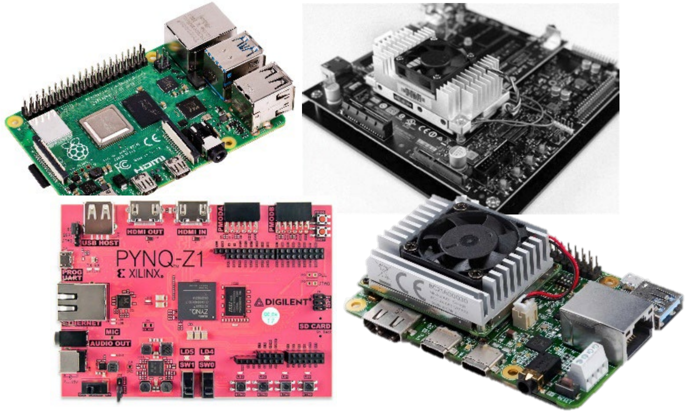
- need for systematic benchmarking of both software and hardware
- track small and large experiments, including collocated ones
- real-time and scalable data tracking and processing
- efficient and effective data exploration

further considerations

- other accelerators and vendors



- resource-constrained hardware
  - tegrastats on NVIDIA Jetsons
  - other platforms
    non-existing or very difficult



- navigating the deep systems stack
  - makes it harder to pinpoint the cause of a performance behavior

APIs (python, R, …)

deep learning software

CPU	GPU	TPU	FPGA	…

teamRAD - resource-aware data systems

rad.itu.dk

Ties
Robroek

Ehsan
Yousefzadeh-Asl-Miandoab

Robert
Bayer

DASYA

Data-Intensive Systems and Applications

www.dasya.dk

@dasyaITU


















need for higher computational efficiency

→ how to *quantify* computational efficiency?

- profiling & monitoring tools for GPUs

[EuroMLSys 2023]

→ how to make the process *systematic*?

- resource-aware data science tracker (radT)

[DEEM 2023]

RAD

rad.itu.dk

thank you!