Leveraging NVML GPM for NVIDIA GPU Monitoring

Christian Wassermann[1], Tobias Dollenbacher[1], Christian Terboven[1], Matthias S. Müller[1]

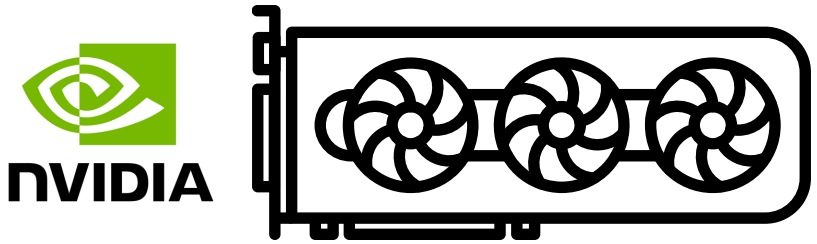
[1] IT Center, RWTH Aachen University, Germany

Sustainable HPC State of the Practice Workshop 2026 in Osaka, Japan

IT Center | RWTH AACHEN UNIVERSITY

Meeting Today's Complexity in HPC with Monitoring for System Observability

- **Challenge**: HPC operators and users face an **increasing complexity in hardware and software**
- **Solution**: **deploy monitoring** throughout the entire system **to enable observability**



nvidia-smi

✅ Ease of use
❌ Limited data

NVML GPM API

✅ Ease of use
✅ More and better data

Data Center GPU Manager (DCGM)

✅ More and better data
❌ Additional SW required

Agenda

- NVML GPM API
  - Available Metrics
  - Usage Example
- Case Study: Energy Consumption Modeling
- Metric Validation
  - Initial Observations
  - Metric Definitions
  - Post-Processing Methodology
  - Comparison to Nsight Compute
- Summary & Outlook

Agenda

- **NVML GPM API**
  - **Available Metrics**
  - **Usage Example**
- Case Study: Energy Consumption Modeling
- Metric Validation
  - Initial Observations
  - Metric Definitions
  - Post-Processing Methodology
  - Comparison to Nsight Compute
- Summary & Outlook

NVML GPM API – Available Metrics

- First released with **NVML v520 in October 2022**
- Supported on **NVIDIA GPUs since Hopper**

NVML GPM
GPU Utilization
SM Utilization
SM Occupancy (warp utilization)
Memory Utilization
Instruction Mix (integer, fp64, fp32, fp16)
Tensor Mix (integer, double, half, any)
PCIe (transmit, receive)
NVLink (transmit, receive)

NVML GPM API – Available Metrics

- First released with **NVML v520 in October 2022**
- Supported on **NVIDIA GPUs since Hopper**

	NVML GPM
also in nvidia-smi	GPU Utilization
	SM Utilization
	SM Occupancy (warp utilization)
also in nvidia-smi	Memory Utilization
	Instruction Mix (integer, fp64, fp32, fp16)
	Tensor Mix (integer, double, half, any)
also in nvidia-smi	PCIe (transmit, receive)
also in nvidia-smi	NVLink (transmit, receive)

NVML GPM API – Available Metrics

- First released with **NVML v520 in October 2022**
- Supported on **NVIDIA GPUs since Hopper**

	NVML GPM
also in nvidia-smi	**GPU Utilization**
	SM Utilization
	SM Occupancy (warp utilization)
also in nvidia-smi	Memory Utilization
	Instruction Mix (integer, fp64, fp32, fp16)
	Tensor Mix (integer, double, half, any)
also in nvidia-smi	PCIe (transmit, receive)
also in nvidia-smi	NVLink (transmit, receive)

NVML GPM API – Usage Example

```
nvmlGpmSampleGet(device, sample_start)

// everything executed here is measured

nvmlGpmSampleGet(device, sample_end)

nvmlGpmMetricsGet(sample_start, sample_end, metrics)
```

- Metric calculation is based on **continuously integrated counter values**
  - "`everything`" indicates that **no load spikes are missed**
  - Unfortunately, the raw counter values cannot be retrieved directly
- ⚠️ **Caveat: minimum time between samples is 100 ms**
- ✅ But still **well suited for monitoring use** (and other coarse-grained data collection)
  - **No interference with other profiling tools** like DCGM or NVIDIA Nsight Systems / Compute observed

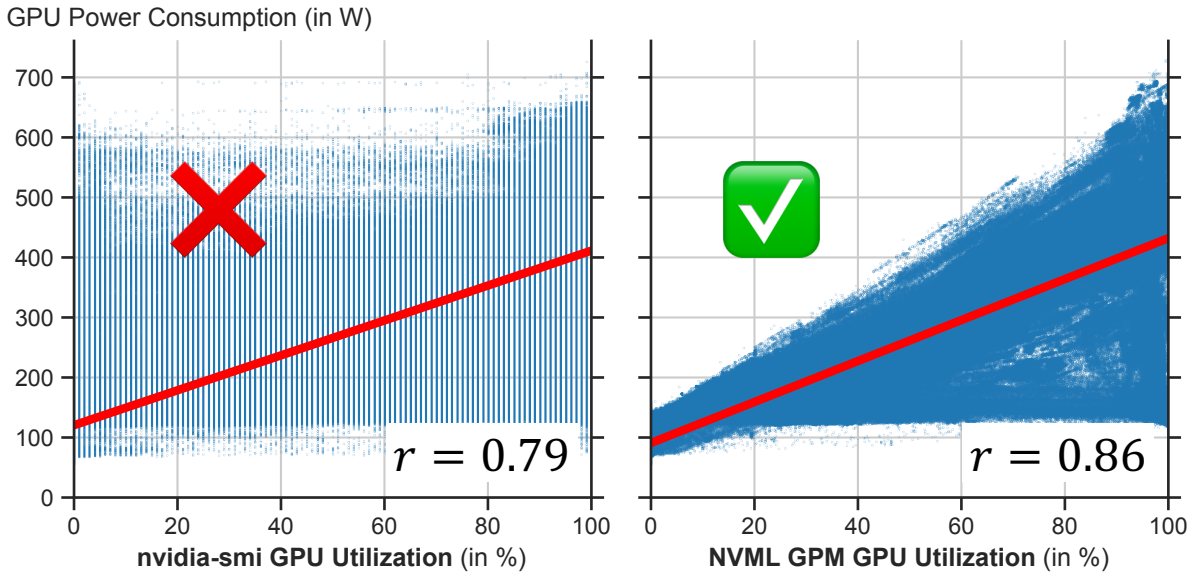
Agenda

- NVML GPM API
  - Available Metrics
  - Usage Example
- **Case Study: Energy Consumption Modeling**
- Metric Validation
  - Initial Observations
  - Metric Definitions
  - Post-Processing Methodology
  - Comparison to Nsight Compute
- Summary & Outlook

Case Study: Energy Consumption Modeling

- Data from **4-weeks in-production monitoring with 1 min sampling frequency**
  - CLAIX-2023 GPU partition with 52 nodes each with 4 NVIDIA H100 GPUs
  - GPU power consumption derived from `nvmlDeviceGetTotalEnergyConsumption` continuously integrated counters

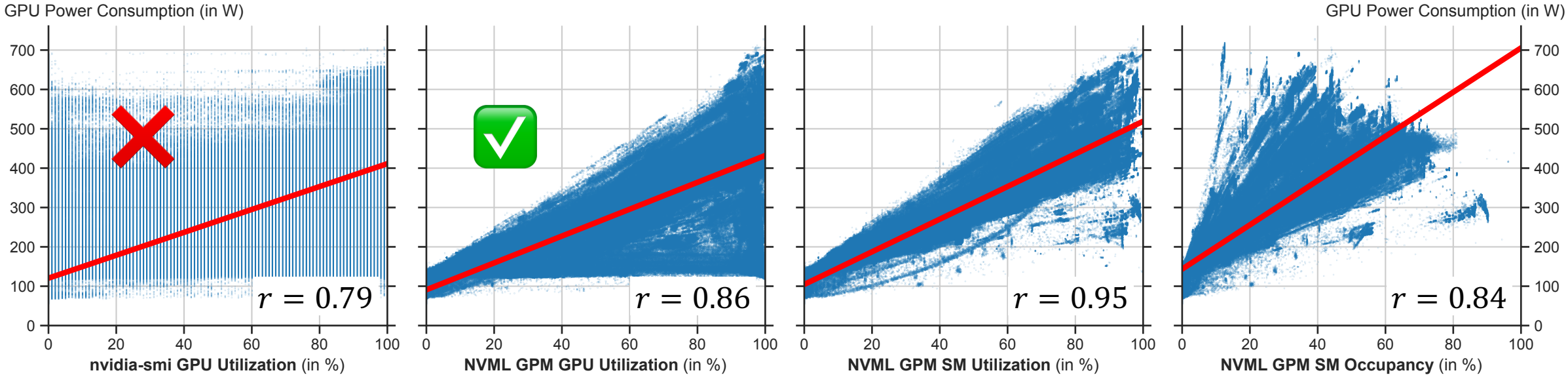
- Observations:
  - **Limited resolution of nvidia-smi** due to integer GPU utilization values
  - **NVML GPM's GPU Utilization with stronger linear correlation** than nvidia-smi (counter-based vs sample-based data)



Case Study: Energy Consumption Modeling

- Data from **4-weeks in-production monitoring with 1 min sampling frequency**
  - CLAIX-2023 GPU partition with 52 nodes each with 4 NVIDIA H100 GPUs
  - GPU power consumption derived from `nvmlDeviceGetTotalEnergyConsumption` continuously integrated counters

- Observations:
  - **Limited resolution of nvidia-smi** due to integer GPU utilization values
  - **NVML GPM's GPU Utilization with stronger linear correlation** than nvidia-smi (counter-based vs sample-based data)
  - **Stronger linear correlation with SM utilization** than with SM occupancy (as also shown by other works [1-4])



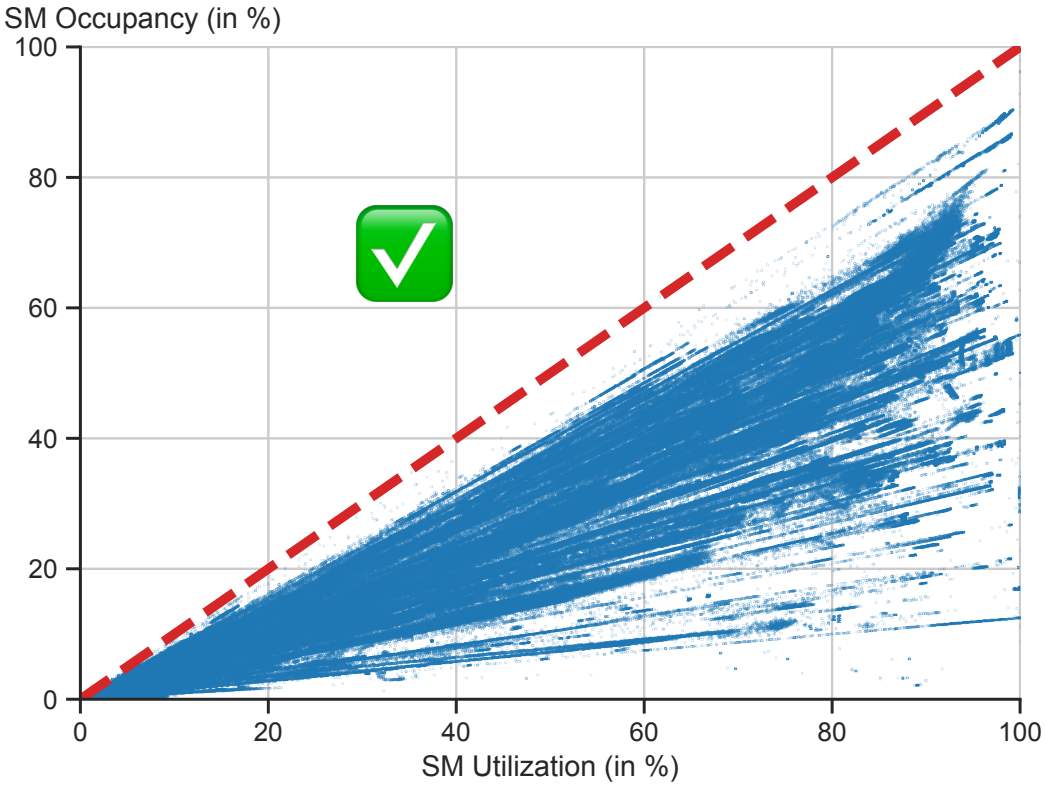
Agenda

- NVML GPM API
  - Available Metrics
  - Usage Example
- Case Study: Energy Consumption Modeling
- **Metric Validation**
  - **Initial Observations**
  - **Metric Definitions**
  - **Post-Processing Methodology**
  - **Comparison to Nsight Compute**
- Summary & Outlook

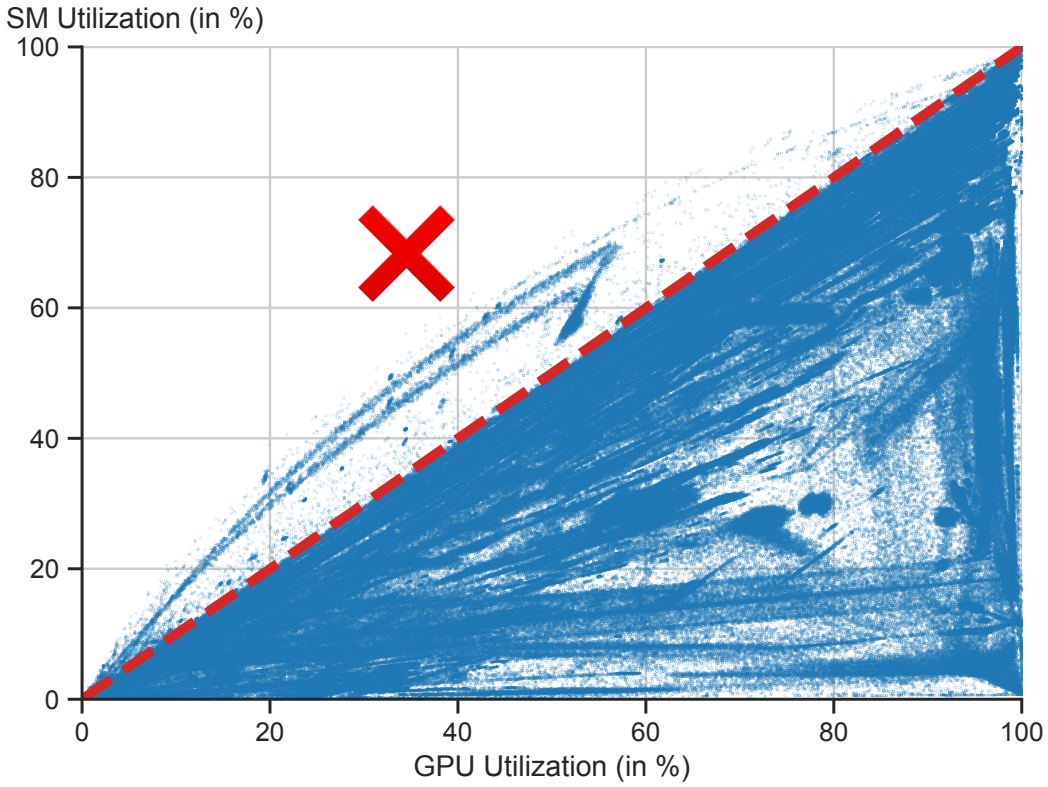
Metric Validation – Initial Observations

- Initial assumption: **SM Occupancy ≤ SM Utilization ≤ GPU Utilization**

The SM occupancy **never exceeds** the SM utilization 🤓👍



The SM utilization **may exceed** the GPU utilization… 🤔



Metric Validation – Metric Definitions

- Checking the documentation of the NVML GPM metrics:

NVML_GPM_METRIC_GRAPHICS_UTIL = 1

Percentage of time any compute/graphics app was active on the GPU. 0.0 - 100.0.

NVML_GPM_METRIC_SM_UTIL = 2

Percentage of SMs that were busy. 0.0 - 100.0.

NVML_GPM_METRIC_SM_OCCUPANCY = 3

Percentage of warps that were active vs theoretical maximum. 0.0 - 100.0.

[5]

DCGM_FI_PROF_GR_ENGINE_ACTIVE 1001

Profiling Fields.

These all start with DCGM_FI_PROF_* Ratio of time the graphics engine is active. The graphics engine is active if a graphics/compute context is bound and the graphics pipe or compute pipe is busy.

DCGM_FI_PROF_SM_ACTIVE 1002

The ratio of cycles an SM has at least 1 warp assigned (computed from the number of cycles and elapsed cycles)

DCGM_FI_PROF_SM_OCCUPANCY 1003

The ratio of number of warps resident on an SM.

(number of resident as a ratio of the theoretical maximum number of warps per elapsed cycle)

[6]

- 💡 **The number of cycles depends on the clock frequency**
  - The clock frequency can be locked with: `nvidia-smi --lock-gpu-clocks=1980,1980`
  - With a locked GPU frequency, the assumption is: SM Utilization ≤ GPU Utilization

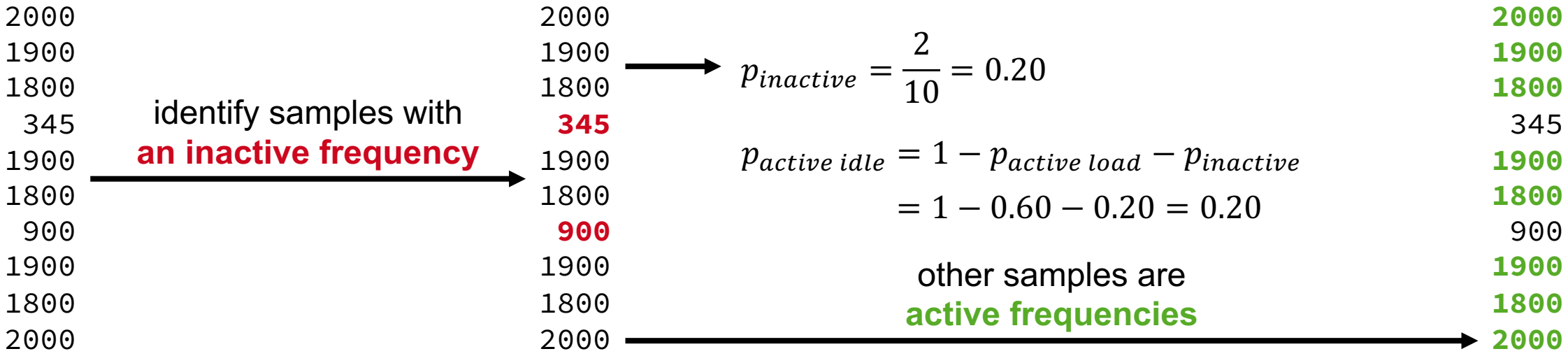
- But **locking the GPU frequency is undesirable** due to a higher energy consumption / degraded performance

Metric Validation – Post-Processing Methodology

- **Apply post-processing to** the collected **raw NVML GPM SM utilization** $util_{raw} = 0.30$
- Use a **simplified clock frequency model** to enable usage in a monitoring context:
  - $f_{active\ idle} = 2000$ chosen as the maximum documented GPU frequency

$$freq_{avg} = freq_{active\ load} \cdot p_{active\ load} + freq_{inactive} \cdot p_{inactive} + freq_{active\ idle} \cdot p_{active\ idle}$$

- Use **NVML GPM GPU utilization** as $p_{active\ load} = 0.60$
- **Clock frequency values from** simultaneously collected **nvidia-smi samples**



Metric Validation – Post-Processing Methodology

- The **active frequencies** are used to estimate $freq_{active\ load}$

$p_{active\ load} = 0.60$
$p_{active\ idle} = 0.20$
$f_{active\ idle} = 2000$

2000
1900
1800
345
1900
1800
900
1900
1800
2000

→ $freq_{active} = 1887.50$ computed as the average of all active frequencies

$$= freq_{active\ load} \cdot \frac{p_{active\ load}}{p_{active\ idle} + p_{active\ load}} + freq_{active\ idle} \cdot \frac{p_{active\ idle}}{p_{active\ idle} + p_{active\ load}}$$

solving this equation gives $freq_{active\ load} = 1850$

- Finally, the post-processed **average SM utilization during kernel execution** $util_{kernel}$ is:

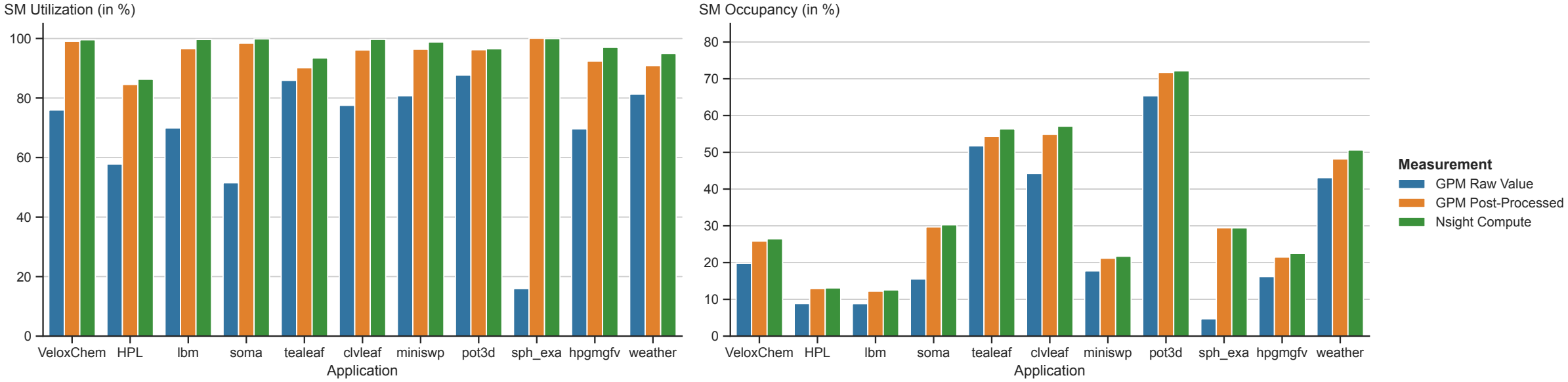
$$util_{kernel} = \frac{freq_{avg} \cdot util_{raw}}{freq_{active\ load} \cdot p_{active\ load}} = \frac{1634.50 \cdot 0.30}{1850 \cdot 0.60} \approx 0.44$$

- This **methodology similarly applies to the SM occupancy** (or any other clock-dependent metric)

Metric Validation – Comparison to Nsight Compute

- Test applications: VeloxChem [5-6], NVIDIA HPL [7], tiny OpenACC SPEChpc2021 [8]
  - Each application was configured to run on a single GPU
  - **VeloxChem** was run once and **took ≈ 27 min**
  - **To guarantee sufficient samples**, all other executions were repeated until **each job ran at least 2 h**

- In all cases, **the post-processing improved the accuracy of the metrics**:
  - GPM raw values differ by 28.35 %pt and 8.74 %pt on average
  - **GPM post-processed values differ by 2.31 %pt and 0.95 %pt on average**



Agenda

- NVML GPM API
  - Available Metrics
  - Usage Example
- Case Study: Energy Consumption Modeling
- Metric Validation
  - Initial Observations
  - Metric Definitions
  - Post-Processing Methodology
  - Comparison to Nsight Compute
- **Summary & Outlook**

Summary & Outlook

- Highlighted the **value of the newly available NVML GPM API**:
  - **Ease of use** without additional installation requirements
  - **High quality data** due to derivation from continuously integrated counters
- Introduced **post-processing methodology for clock-dependent metrics**
- Support NVML GPM data collection with our **open-source NVML-GPM-Collector**
  - Available on GitHub: https://github.com/RWTH-HPC/NVML-GPM-Collector

Outlook

- User-facing monitoring portals can supply **more accurate insights with NVML GPM data**
- Employ **NVML GPM data for application modeling** and system utilization analysis

Thank you for your attention!

References

Acknowledgments:
- We gratefully acknowledge the financial support by the German Federal Ministry of Research, Technology and Space (BMFTR), promotional reference 16ME0614.
- This project receives funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101143931. The JU receives support from the European Union's Horizon Europe research and innovation programme and Spain, Germany, France, Portugal and the Czech Republic.
- Computations were performed with computing resources granted by RWTH Aachen University under project rwth1578.

Publications:
- [1] S. Hong and H. Kim, 'An integrated GPU power and performance model', in Proceedings of the 37th annual international symposium on Computer architecture, in ISCA '10. New York, NY, USA: Association for Computing Machinery, Jun. 2010, pp. 280–289. doi: 10.1145/1815961.1815998.
- [2] L. Lefèvre, A.-C. Orgerie, and D. Boughzala, 'A macroscopic analysis of GPU power consumption', presented at the COMPAS2019 : Conférence d'informatique en Parallélisme, Architecture et Système, Jun. 2019. Accessed: Oct. 27, 2025. [Online]. Available: https://hal.science/hal-04091328
- [3] V. Kandiah et al., 'AccelWattch: A Power Modeling Framework for Modern GPUs', in MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture, in MICRO '21. New York, NY, USA: Association for Computing Machinery, Oct. 2021, pp. 738–753. doi: 10.1145/3466752.3480063.
- [4] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, 'Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits', Proceedings of the IEEE, vol. 91, no. 2, pp. 305–327, Feb. 2003, doi: 10.1109/JPROC.2002.808156.
- [5] Z. Rinkevicius et al., 'VeloxChem: A Python-driven density-functional theory program for spectroscopy simulations in high-performance computing environments', WIREs Computational Molecular Science, vol. 10, no. 5, p. e1457, 2020, doi: 10.1002/wcms.1457.
- [6] X. Li, M. Linares, and P. Norman, 'VeloxChem: GPU-Accelerated Fock Matrix Construction Enabling Complex Polarization Propagator Simulations of Circular Dichroism Spectra of G-Quadruplexes', J. Phys. Chem. A, vol. 129, no. 2, pp. 633–642, Jan. 2025, doi: 10.1021/acs.jpca.4c07510.
- [7] NVIDIA Corporation, NVIDIA HPC-Benchmarks 24.09. (Oct. 15, 2024). Accessed: Oct. 24, 2025. [Online]. Available: https://catalog.ngc.nvidia.com/orgs/nvidia/containers/hpc-benchmarks?version=24.09
- [8] J. Li et al., 'SPEChpc 2021 Benchmark Suites for Modern HPC Systems', in Companion of the 2022 ACM/SPEC International Conference on Performance Engineering, in ICPE '22. New York, NY, USA: Association for Computing Machinery, Jul. 2022, pp. 15–16. doi: 10.1145/3491204.3527498.

References

Links:

- [5] https://docs.nvidia.com/deploy/nvml-api/group__nvmlGpmEnums.html#group__nvmlGpmEnums
- [6] https://docs.nvidia.com/datacenter/dcgm/latest/dcgm-api/dcgm-api-field-ids.html#c.DCGM_FI_PROF_GR_ENGINE_ACTIVE

Images:

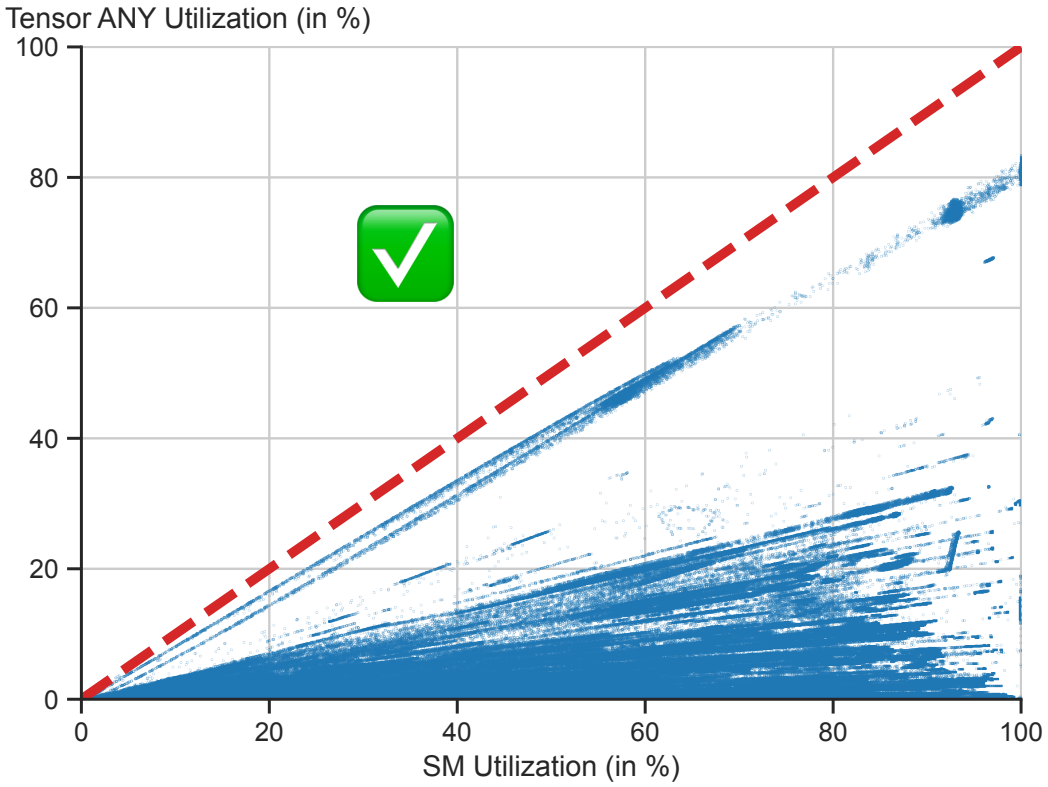
- NVIDIA Logo by Nvidia Corporation from https://nvidianews.nvidia.com/_gallery/get_file/?file_id=692f50553d6332b453bbc5c2 (Logo Guidelines)
- GPU by Misha Petrishchev from https://thenounproject.com/icon/gpu-1132941 (CC BY 3.0 License)

Backup Slides

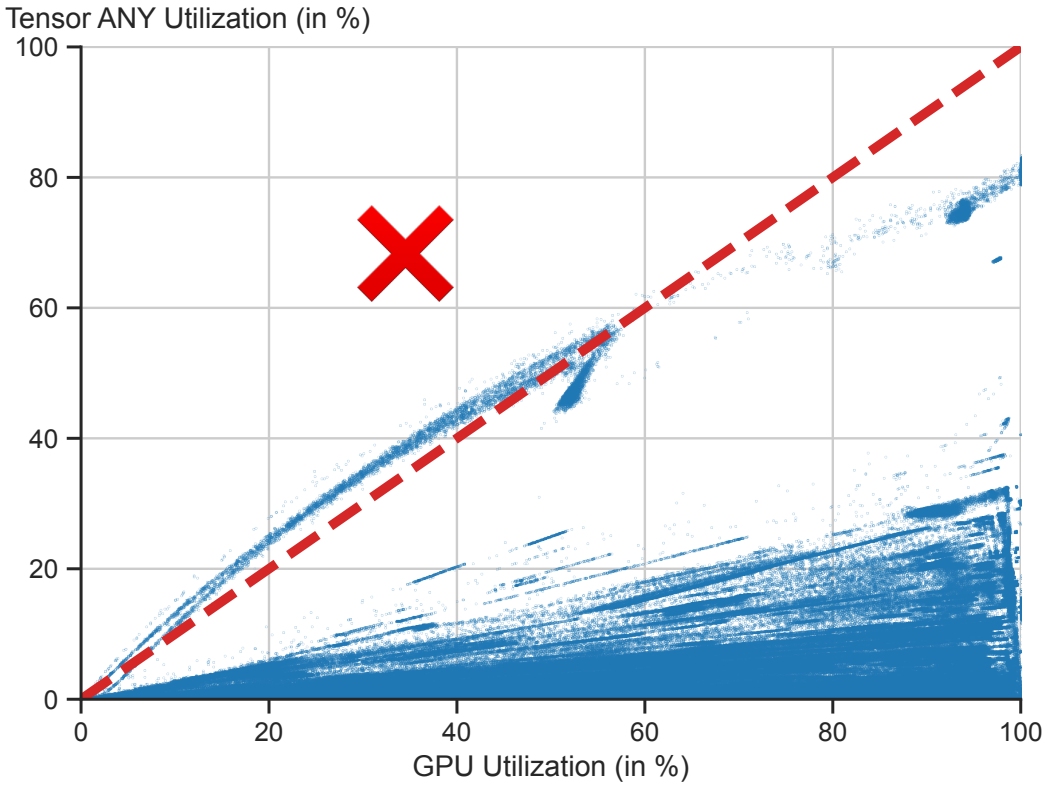
Metric Validation – Initial Observations

- The tensor utilization seems to be clock-dependent as well…

The tensor utilization never exceeds the SM utilization

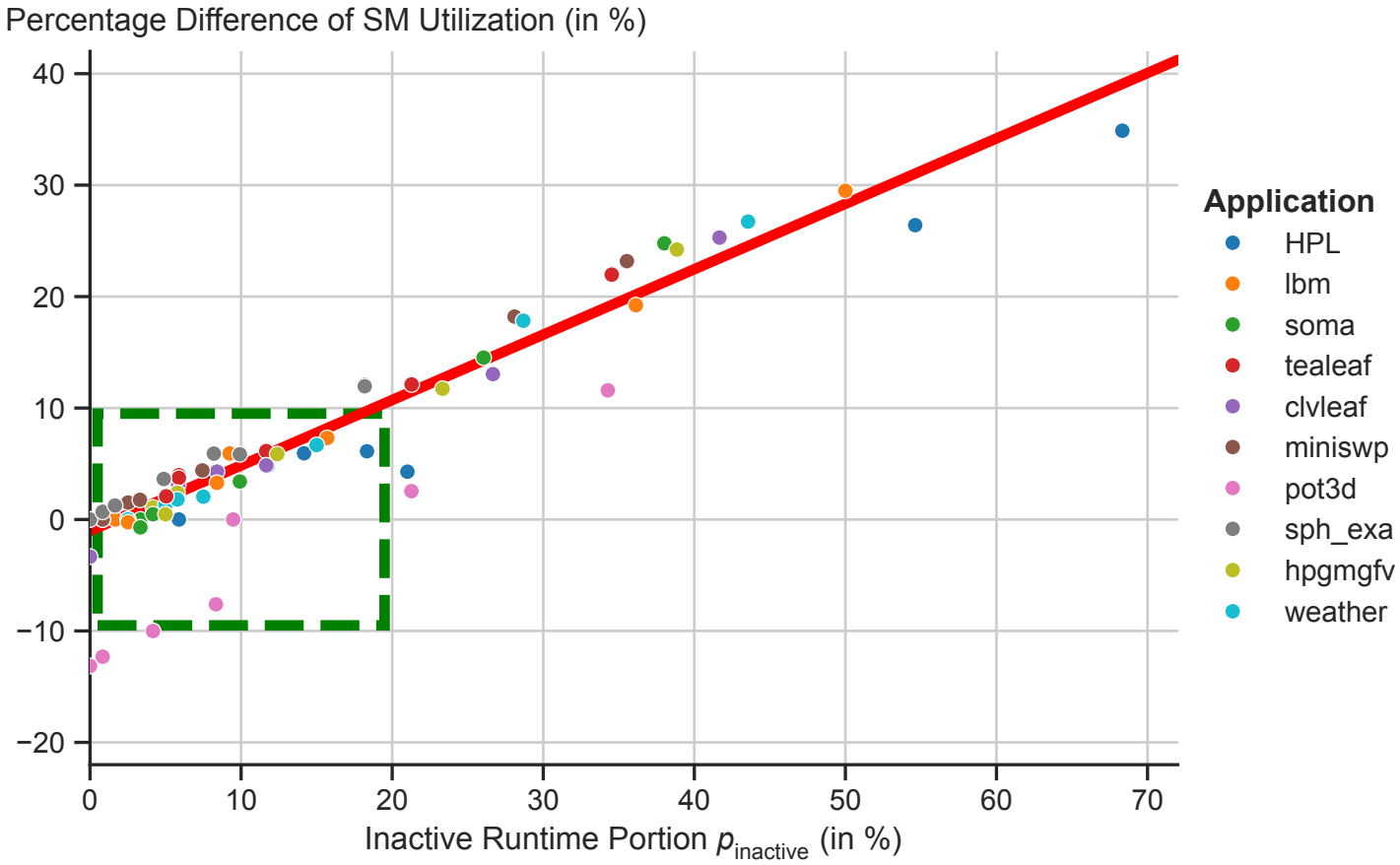


The tensor utilization may exceed the GPU utilization



Metric Validation – Initial Observations

- Minutely monitoring data has sufficient quality for jobs with 2h runtime + at most 20% inactive portion



Theme 2 – GPU Monitoring with NVML GPM

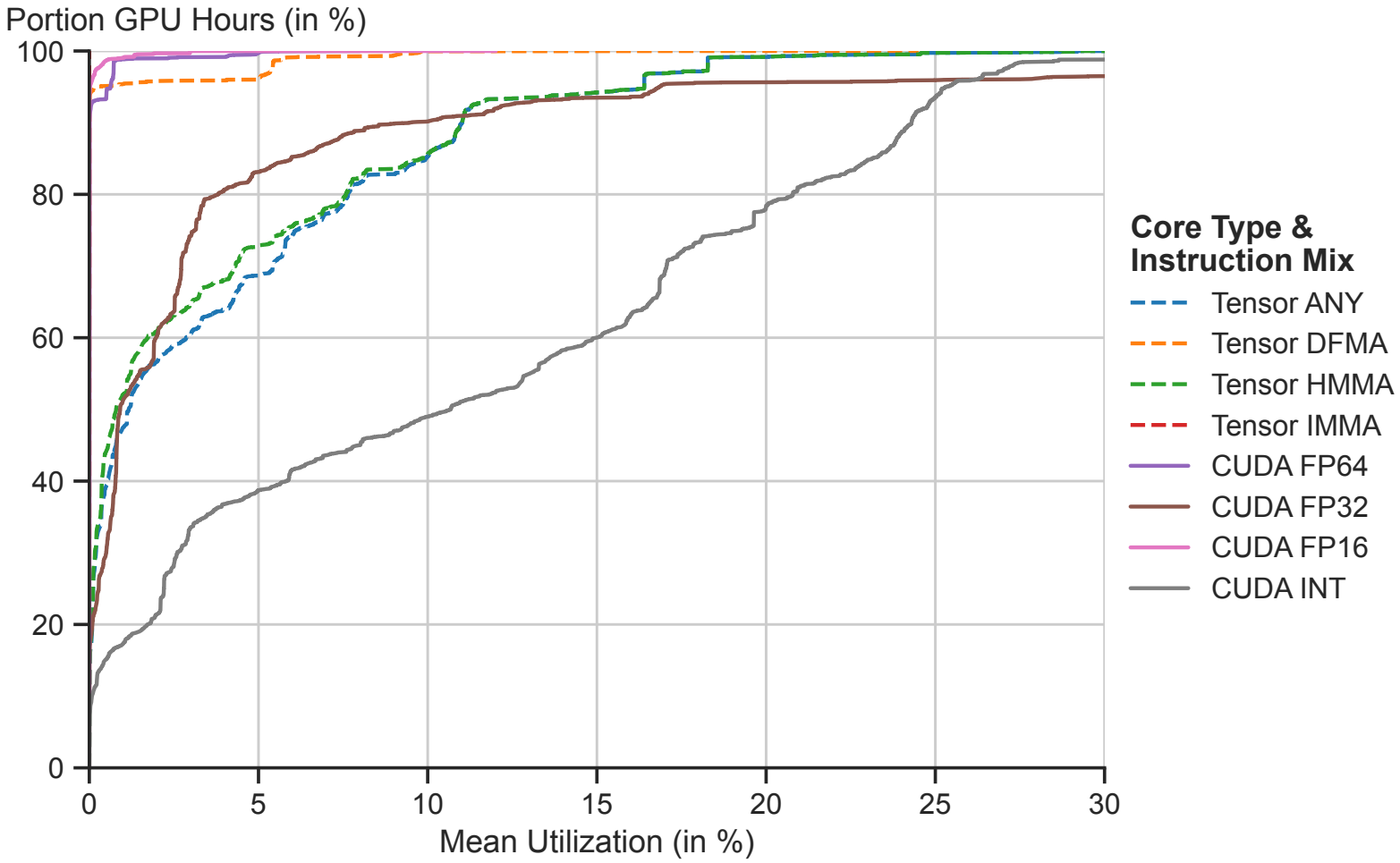
- Theoretically also available with nvidia-smi, but
  - no sub-second time information available
  - data is reported later than expected (to be exact: one delay later)

```
nvidia-smi dmon --delay 2 --count 5 --format csv,nounit --options DT \
    --gpm-metrics 1,2,3,4,5,6,7,9,10,11,12,13,20,21,60,61 --gpm-options dm
// only collection of PCIe metrics here with delay 90
//     [rx|tx]pci: --select t
//     pci[tx|rx]: --gpm-metrics 20,21
#Date, Time, gpu, rxpci, txpci, pcitx, pcirx
 20250814, 17:12:06, 0, 0, 0, -, -
 20250814, 17:13:36, 0, 0, 0, -, -
 20250814, 17:15:06, 0, 0, 0, 0, 0
 20250814, 17:16:36, 0, 0, 0, 0, 0
 20250814, 17:18:06, 0, 0, 0, 0, 0
 20250814, 17:19:36, 0, 0, 1, 0, 0
 20250814, 17:21:07, 0, 0, 0, 13, 22
 20250814, 17:22:37, 0, 0, 0, 0, 0
 20250814, 17:24:07, 0, 0, 0, 0, 0
```

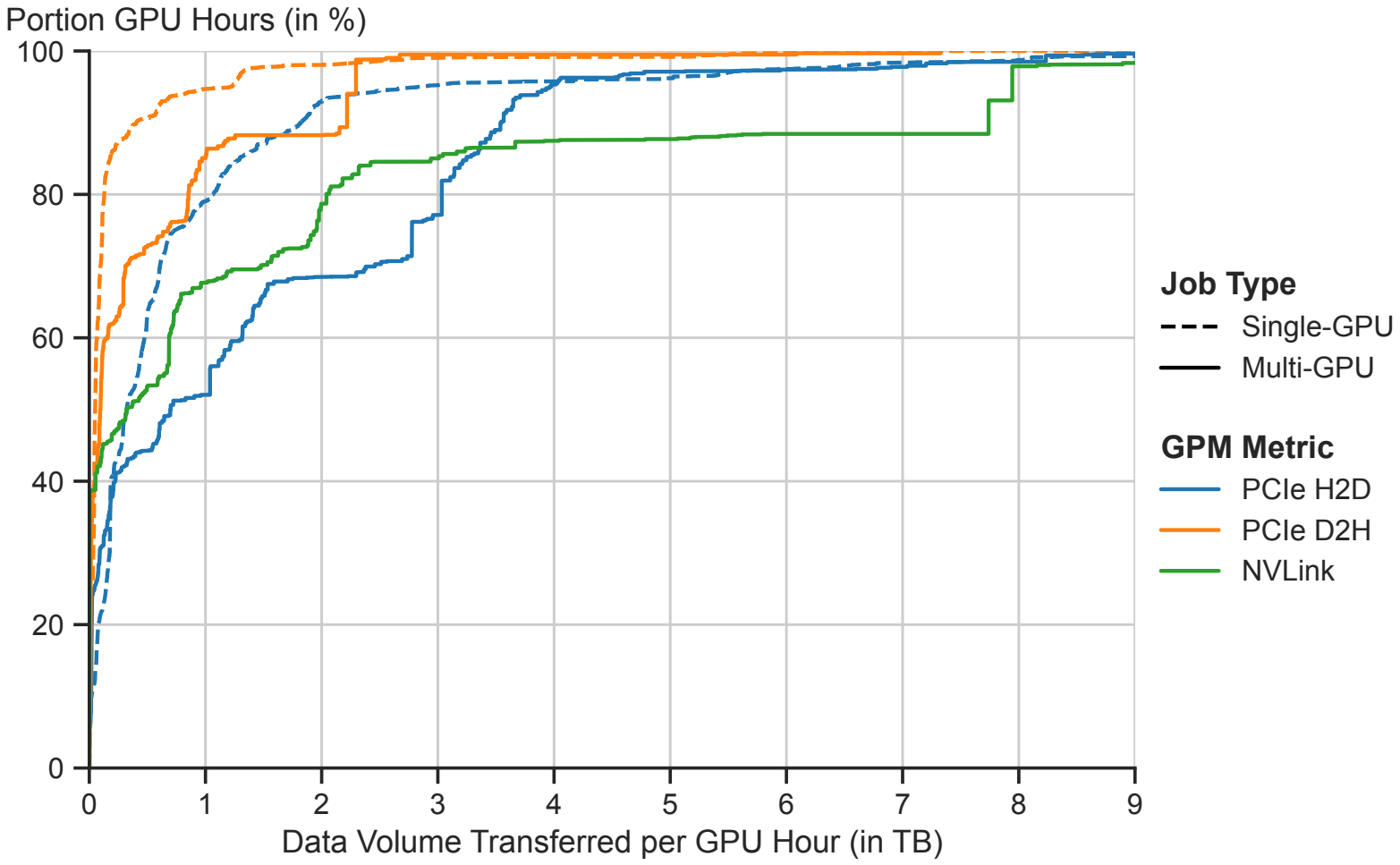
PCIe transfers should already appear here

```
Thu 14 Aug 17:18:12 CEST 2025
// workload executed on GPU 0
Thu 14 Aug 17:18:58 CEST 2025
```

Case Study: Cluster Workload Analysis – Instruction Mix



Case Study: Cluster Workload Analysis – Data Transfers



Details of Job Monitoring Dataset

Table 1: Key characteristics of the GPU partition of the CLAIX-2023 cluster of RWTH Aachen University and of the analyzed batch job dataset.

Characteristic	CLAIX-2023 GPU
Number of Nodes	52
GPUs per Node	4
GPU Architecture	NVIDIA H100 SXM5
GPU Memory	94 GB HBM2e
NVML Version	v570
Time Frame	2025-09-01 to 2025-09-28
Number of Jobs	9468
Minimum Job Runtime	30 min
Sampling Frequency	1 min
The pre-processing removed jobs from this paper's authors. Due to an update of the monitoring system on 2025-09-09, this day's data includes 10 h of incomplete metrics. We do not expect the missing data to notably influence the observations.	