Beyond nvidia-smi

Tools for real GPU performance metrics

Yash Panchal

SDET III @ Percona

Overview

1. GPU Fundamentals
2. Common GPU Monitoring Mistakes
3. Idle Tensor Core Example
4. Understanding Key Metrics to Measure
5. Workload specific metrics
6. Some cli features for quick alert (Temperature notification)
7. Conclusions

GPU fundamentals

- GPU = Compute + Memory + Decoders
- Compute is mainly executed by Streaming Multiprocessors

Common GPU Monitoring Mistakes

1. Relying too much on nvidia-smi (Myopic)
2. Not identifying workload computation type properly. (Wasted Potential)
3. Not monitoring key metrics like Tensor Cores, SM metrics and DRAM in combination for better understanding

nvidia-smi GPU-Util focus

ssh

```
Every 2.0s: nvidia-smi

Thu Jan 29 14:35:42 2026
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 580.95.05              Driver Version: 580.95.05      CUDA Version: 13.0     |
+-----------------------------------------+------------------------+----------------------+
| 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  NVIDIA H100 80GB HBM3          On  |   00000000:05:00.0 Off |                    Off |
| N/A   58C    P0             675W /  700W |    1651MiB /  81559MiB |      100%      Default |
|                                         |                        |             Disabled |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI              PID   Type   Process name                      GPU Memory |
|        ID   ID                                                             Usage      |
|=========================================================================================|
|    0   N/A  N/A            8993      C   python3                                1642MiB |
+-----------------------------------------------------------------------------------------+
```

Figure A: nvidia-smi showing 100% GPU-Util

nvidia-smi showing 100 %

Is my GPU broken ?

Do I need new GPU for more workload ?

```
ssh

(myenv) root@h100-1:~# python3 fp32.py
PID: 11562
FP32 running for 30 seconds...

ssh

Every 2.0s: nvidia-smi

Thu Jan 29 14:44:41 2026
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 580.95.05              Driver Version: 580.95.05      CUDA Version: 13.0     |
+-----------------------------------------+------------------------+----------------------+
| 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  NVIDIA H100 80GB HBM3          On  |   00000000:05:00.0 Off |                  Off |
| N/A   59C    P0             677W /  700W |    1651MiB /  81559MiB |    100%      Default |
|                                         |                        |             Disabled |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI              PID   Type   Process name                        GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A           11562      C   python3                               1642MiB |
+-----------------------------------------------------------------------------------------+
```

Figure B: nvidia-smi showing 100% for FP32

FP32 uses CUDA Cores

```
ssh

(myenv) root@h100-1:~# vi fp16.py
(myenv) root@h100-1:~# python3 fp16.py
PID: 11173
FP16 running for 30 seconds...

ssh

Every 2.0s: nvidia-smi

Thu Jan 29 14:43:24 2026
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 580.95.05              Driver Version: 580.95.05      CUDA Version: 13.0     |
+-----------------------------------------+------------------------+----------------------+
| 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  NVIDIA H100 80GB HBM3          On  |   00000000:05:00.0 Off |                  Off |
| N/A   60C    P0             697W /  700W |    1139MiB /  81559MiB |    100%      Default |
|                                         |                        |             Disabled |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI              PID   Type   Process name                        GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A           11173      C   python3                               1130MiB |
+-----------------------------------------------------------------------------------------+
```

Figure C: nvidia-smi showing 100% for FP16

FP16 uses Tensor Cores

Idle Tensor Cores

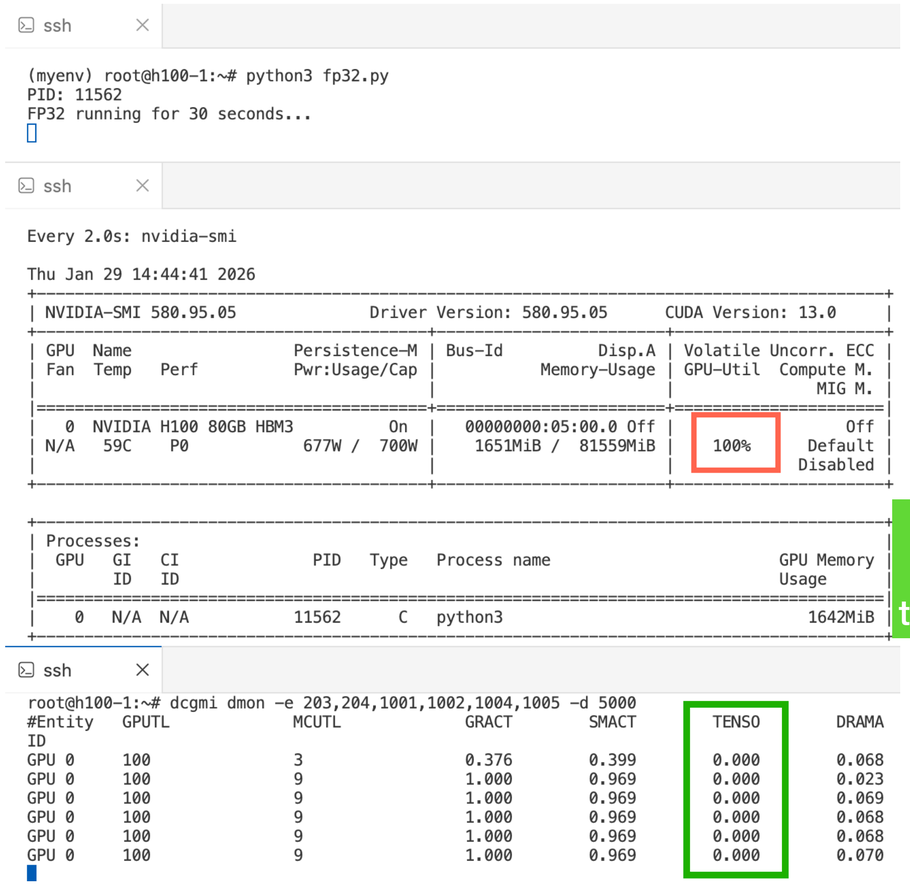


Figure B: nvidia-smi showing 100% GPU-Util 0 Tensor core usage

FP32 uses CUDA Cores

Ok So there is much more that can be observed !

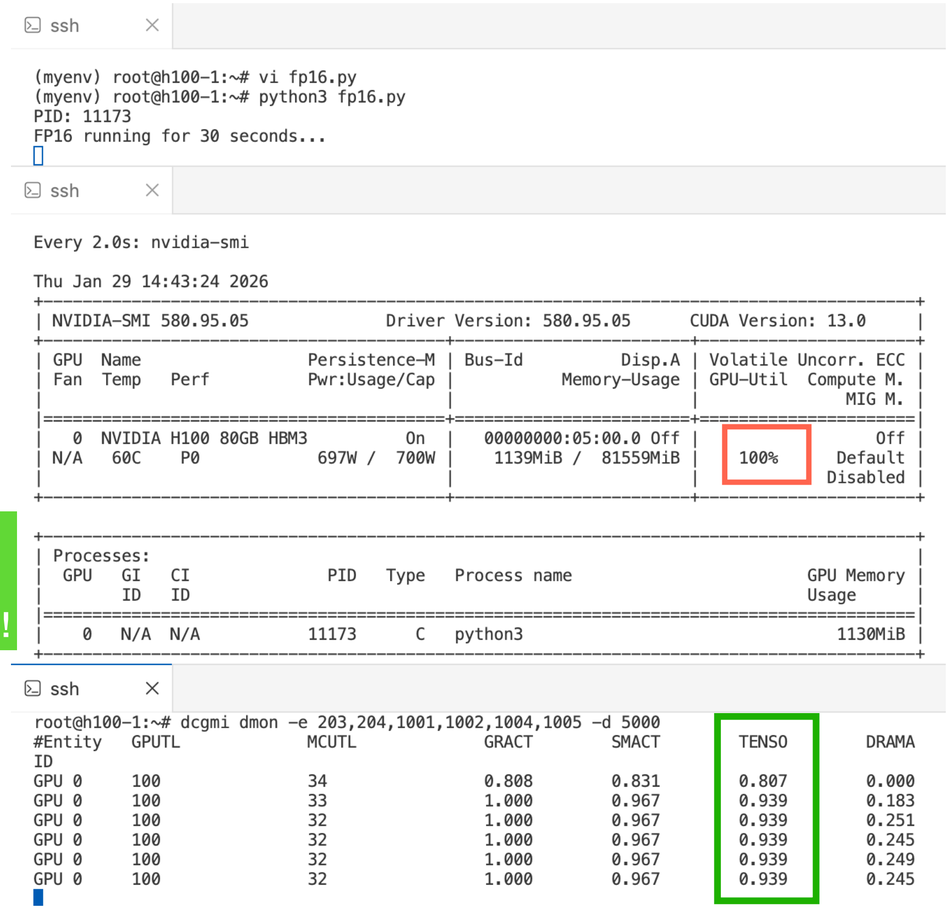


Figure C: nvidia-smi showing 100% GPU-Util 0 Tensor core usage

FP16 uses Tensor Cores

GPU Utilization is not same as GPU Efficiency

- Understanding nature of your workload is Key !
- Modern GPUs have hardware designed for specific workloads
- Tensor Cores: MatMul specific
- Supports most basic AI workloads

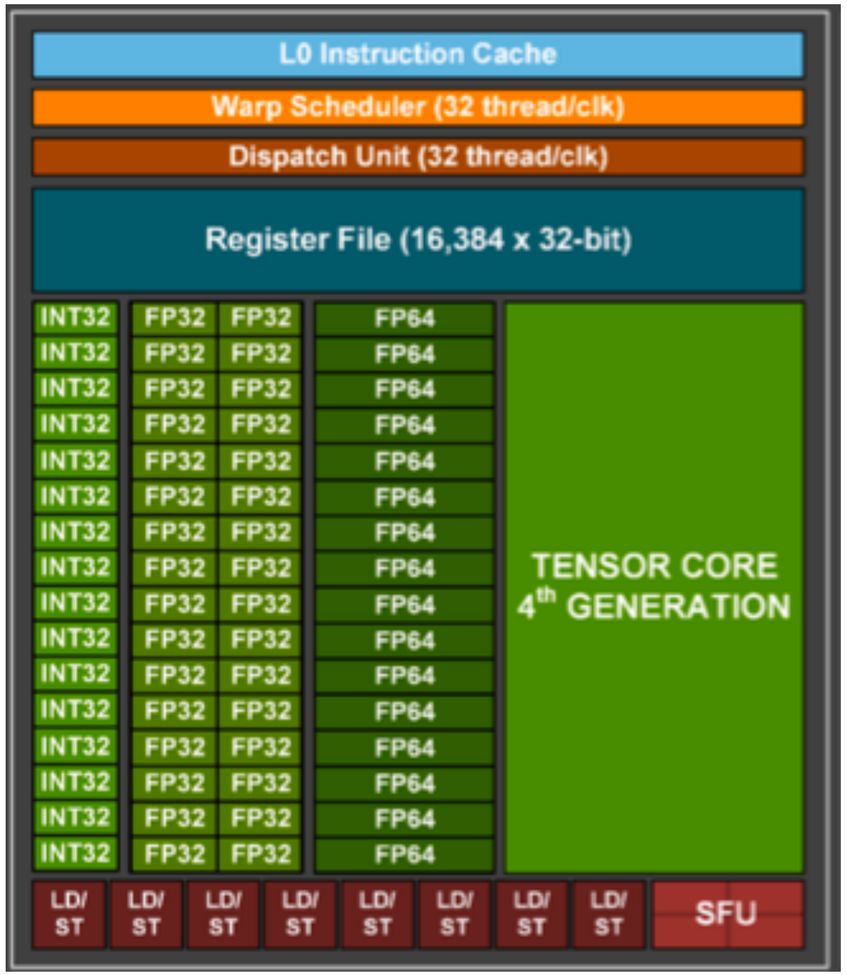


Figure D: Diagram of a Tensor Core

What nvidia-smi 100% GPU-Util means ?

- Are SMs doing any work ?
- Does not measure how much work your SMs are doing.

dcgmi overview

Setup

apt-get install -y datacenter-gpu-manager

sudo systemctl --now enable nvidia-dcgm

dcgmi dmon -e <DCGM_CODES> -d <TIME_DURATION>

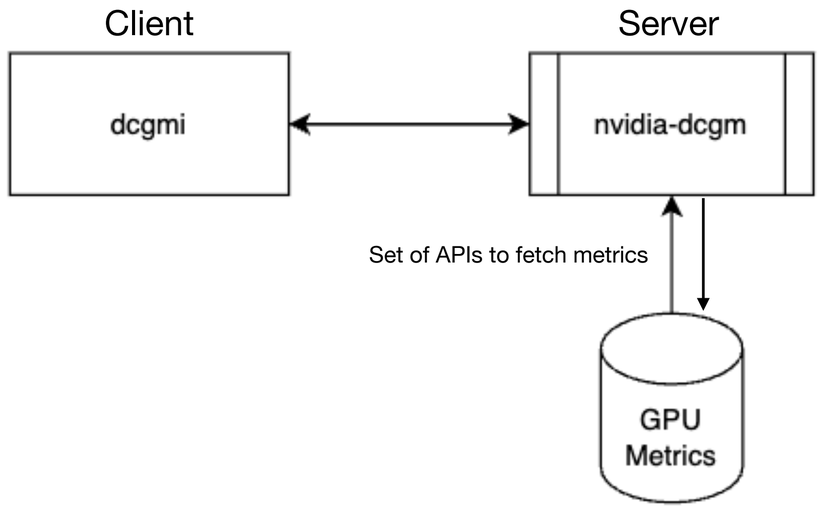


Figure E: Basic dcgm workflow

Key Metrics to measure

- Tensor Core utilization
- Memory Bandwidth
- SM Occupancy
- SM Activity
- Estimated TFLOPS computation

Figure 7. GH100 Streaming Multiprocessor (SM)

Tensor cores

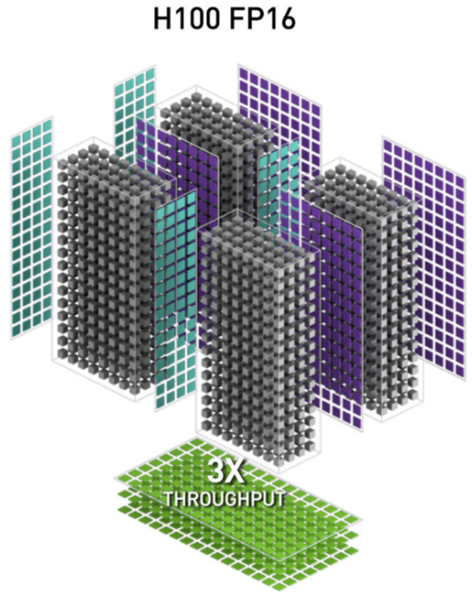


Figure F: Example Matrix Multiplication Accumulator

- Tensor cores are specifically designed to Multiply and Add (Matmul) Stuff
- Around 8-10x faster then CUDA cores

Potential Performance improvement

- Tensor Cores are 8-9 x faster performance for supported tasks
- MMA : Matrix Multiplication Accumulation

Table 1. NVIDIA H100 Tensor Core GPU Preliminary Performance Specs

	NVIDIA H100 SXM5[1]	NVIDIA H100 PCIe[1]
Peak FP64[1]	30 TFLOPS	24 TFLOPS
Peak FP64 Tensor Core[1]	60 TFLOPS	48 TFLOPS
Peak FP32[1]	60 TFLOPS	48 TFLOPS
Peak FP16[1]	120 TFLOPS	96 TFLOPS
Peak BF16[1]	120 TFLOPS	96 TFLOPS
Peak TF32 Tensor Core[1]	500 TFLOPS | 1000 TFLOPS[2]	400 TFLOPS | 800 TFLOPS[2]
Peak FP16 Tensor Core[1]	1000 TFLOPS | 2000 TFLOPS[2]	800 TFLOPS | 1600 TFLOPS[2]
Peak BF16 Tensor Core[1]	1000 TFLOPS | 2000 TFLOPS[2]	800 TFLOPS | 1600 TFLOPS[2]
Peak FP8 Tensor Core[1]	2000 TFLOPS | 4000 TFLOPS[2]	1600 TFLOPS | 3200 TFLOPS[2]
Peak INT8 Tensor Core[1]	2000 TOPS | 4000 TOPS[2]	1600 TOPS | 3200 TOPS[2]

1. **Preliminary performance estimates for H100 based on current expectations and subject to change in the shipping products**
2. **Effective TFLOPS / TOPS using the Sparsity feature**

Figure G: TFLOPS Comparison

Monitor Tensor Core activity

DCGM gives us 4 key metrics for Tensor Core monitoring

Tensor Core Metrics	DCGM Code	Purpose
DCGM_FI_PROF_PIPE_**TENSOR_ACTIVE**	1004	Overall Tensor core active usage
DCGM_FI_PROF_PIPE_**TENSOR_IMMA_ACTIVE**	1013	Int8 Tensor core active usage
DCGM_FI_PROF_PIPE_**TENSOR_HMMA_ACTIVE**	1014	FP16 / BF16 Tensor core active usage
DCGM_FI_PROF_PIPE_**TENSOR_DFMA_ACTIVE**	1015	FP64 Tensor core active usage

MMA: Matrix Multiplication Accumulation

IMMA: Integer (8) Matrix Multiplication Accumulation

HMMA: Half-precision (16) Matrix Multiplication

DFMA: Double Precision (64) Multiply Accumulation

FP16 workload

ssh

```
(myenv) root@h100-server:~# python3 fp16.py
PID: 5217
FP16 running for 30 seconds...
```

ssh

```
root@h100-server:~# dcgmi dmon -e 203,204,1001,1002,1004,1013,1014,1015 -d 2000
#Entity   GPUTL             MCUTL             GRACT        SMACT        TENSO        TIMMA        THMMA        TDFMA
ID
GPU 0     100               33                0.679        0.781        0.759        0.000        0.759        0.000
GPU 0     100               32                1.000        0.967        0.938        0.000        0.938        0.000
GPU 0     100               33                1.000        0.967        0.939        0.000        0.939        0.000
GPU 0     100               32                1.000        0.967        0.938        0.000        0.938        0.000
GPU 0     100               32                1.000        0.967        0.939        0.000        0.939        0.000
GPU 0     100               32                1.000        0.967        0.939        0.000        0.939        0.000
GPU 0     100               32                1.000        0.967        0.939        0.000        0.939        0.000
GPU 0     100               32                1.000        0.967        0.939        0.000        0.939        0.000
```

Overall Tensor usage

Specific FP16/BF16

Figure H: THMMA active for FP16 workload

HMMA: Half-precision (16) Matrix Multiplication

FP64 workload

ssh

```
(myenv) root@h100-server:~# python3 fp64.py
PID: 5833
FP64 running for 30 seconds...
```

ssh

```
root@h100-server:~# dcgmi dmon -e 203,204,1001,1002,1004,1013,1014,1015 -d 2000
#Entity   GPUTL            MCUTL            GRACT        SMACT        TENSO        TIMMA        THMMA        TDFMA
ID
GPU 0     100              20               0.337        0.533        0.525        0.000        0.000        0.525
GPU 0     100              22               1.000        0.986        0.971        0.000        0.000        0.971
GPU 0     100              20               1.000        0.986        0.970        0.000        0.000        0.970
GPU 0     100              19               1.000        0.985        0.970        0.000        0.000        0.970
GPU 0     100              24               1.000        0.986        0.970        0.000        0.000        0.970
GPU 0     100              19               1.000        0.985        0.970        0.000        0.000        0.970
GPU 0     100              23               1.000        0.985        0.970        0.000        0.000        0.970
GPU 0     100              19               1.000        0.986        0.971        0.000        0.000        0.971
GPU 0     100              19               1.000        0.985        0.970        0.000        0.000        0.970
```

Overall Tensor usage

Specific FP64

Figure I: TDFMA active for FP64 workload

DFMA: Double Precision (64) Multiply Accumulation

INT8 Workload

ssh

```
(myenv) root@h100-server:~# python3 int8.py
PID: 5901
Running for 30 seconds...
```

ssh

```
root@h100-server:~# dcgmi dmon -e 203,204,1001,1002,1004,1013,1014,1015 -d 2000
#Entity   GPUTL            MCUTL            GRACT        SMACT        TENSO        TIMMA        THMMA        TDFMA
ID
GPU 0     100              32               0.671        0.793        0.613        0.613        0.000        0.000
GPU 0     100              32               1.000        0.966        0.747        0.747        0.000        0.000
GPU 0     100              31               1.000        0.966        0.747        0.747        0.000        0.000
GPU 0     100              31               1.000        0.966        0.747        0.747        0.000        0.000
GPU 0     100              31               1.000        0.966        0.747        0.747        0.000        0.000
GPU 0     100              31               1.000        0.966        0.747        0.747        0.000        0.000
GPU 0     100              31               1.000        0.966        0.747        0.747        0.000        0.000
GPU 0     100              31               1.000        0.966        0.747        0.747        0.000        0.000
```

Overall Tensor usage

Specific INT8

Figure J: TIMMA active for INT8 workload

IMMA: Integer (8) Matrix Multiplication Accumulation

SM Metrics

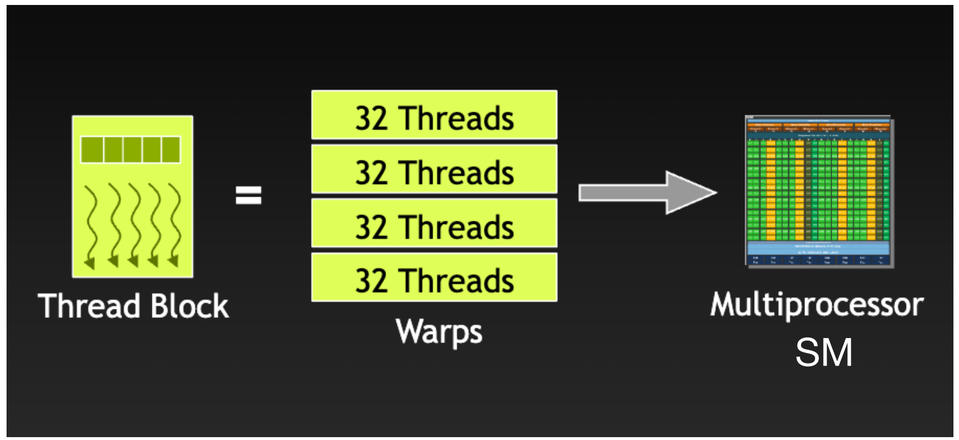


Figure K: Threads, Warps and SM

- SM Occupancy (SMOCC): Warps loaded/occupying SM
- SM Activity (SMACT): Warps doing work on SM

High Occupancy Workload

ssh

```
(myenv) root@h100-server:~# python3 high-occ-example.py
PID: 20225
Conv2D Batch=64 (High Occupancy)
Running for 30 seconds...
Done
(myenv) root@h100-server:~#
```

ssh

```
root@h100-server:~# dcgmi dmon -e 203,204,1001,1002,1003,1004,1013 -d 5000
#Entity   GPUTL            MCUTL            GRACT        SMACT        SMOCC        TENSO        TIMMA
ID
GPU 0     100              65               1.000        0.972        0.699        0.141        0.000
GPU 0     100              65               1.000        0.972        0.699        0.141        0.000
GPU 0     100              65               1.000        0.972        0.699        0.141        0.000
```

Figure L: High occupancy

Low Occupancy workload

ssh

```
(myenv) root@h100-server:~# python3 low-occ-example.py
PID: 20482
Conv2D Batch=1 (Low Occupancy)
Running for 30 seconds...
```

ssh

```
root@h100-server:~# dcgmi dmon -e 203,204,1001,1002,1003,1004,1013 -d 2000
#Entity   GPUTL             MCUTL             GRACT        SMACT        SMOCC        TENSO        TIMMA
ID
GPU 0     100               17                1.000        0.669        0.363        0.095        0.000
GPU 0     100               17                1.000        0.670        0.364        0.095        0.000
GPU 0     100               17                1.000        0.669        0.364        0.095        0.000
GPU 0     100               17                1.000        0.670        0.364        0.095        0.000
GPU 0     100               17                1.000        0.670        0.364        0.095        0.000
GPU 0     100               17                1.000        0.670        0.364        0.095        0.000
```

Figure M: High Active Low occupancy

Estimating TFLOPS using SMs

- Actual TFLOPS are difficult to measure you can estimate.
- SM Metrics are Percentage (SM Occupancy and SM Activity)
- Now for stable workloads this method will work
- For burstable workloads difficult as averages vary with time

Table 1. NVIDIA H100 Tensor Core GPU Preliminary Performance Specs

	NVIDIA H100 SXM5[1]	NVIDIA H100 PCIe[1]
Peak FP64[1]	30 TFLOPS	24 TFLOPS
Peak FP64 Tensor Core[1]	60 TFLOPS	48 TFLOPS
Peak FP32[1]	60 TFLOPS	48 TFLOPS
Peak FP16[1]	120 TFLOPS	96 TFLOPS
Peak BF16[1]	120 TFLOPS	96 TFLOPS
Peak TF32 Tensor Core[1]	500 TFLOPS | 1000 TFLOPS[2]	400 TFLOPS | 800 TFLOPS[2]
Peak FP16 Tensor Core[1]	1000 TFLOPS | 2000 TFLOPS[2]	800 TFLOPS | 1600 TFLOPS[2]
Peak BF16 Tensor Core[1]	1000 TFLOPS | 2000 TFLOPS[2]	800 TFLOPS | 1600 TFLOPS[2]
Peak FP8 Tensor Core[1]	2000 TFLOPS | 4000 TFLOPS[2]	1600 TFLOPS | 3200 TFLOPS[2]
Peak INT8 Tensor Core[1]	2000 TOPS | 4000 TOPS[2]	1600 TOPS | 3200 TOPS[2]

1. **Preliminary performance estimates for H100 based on current expectations and subject to change in the shipping products**
2. **Effective TFLOPS / TOPS using the Sparsity feature**

PEAK TFLOPS X SM ACC X TENSO

Things to keep in mind

- By themselves GPUs are useless CPUs command them
- CPU: Loading, Transfer, Launching Kernels, Retrieve results
- Most inference workloads are Memory bound

High SM Occupancy

+

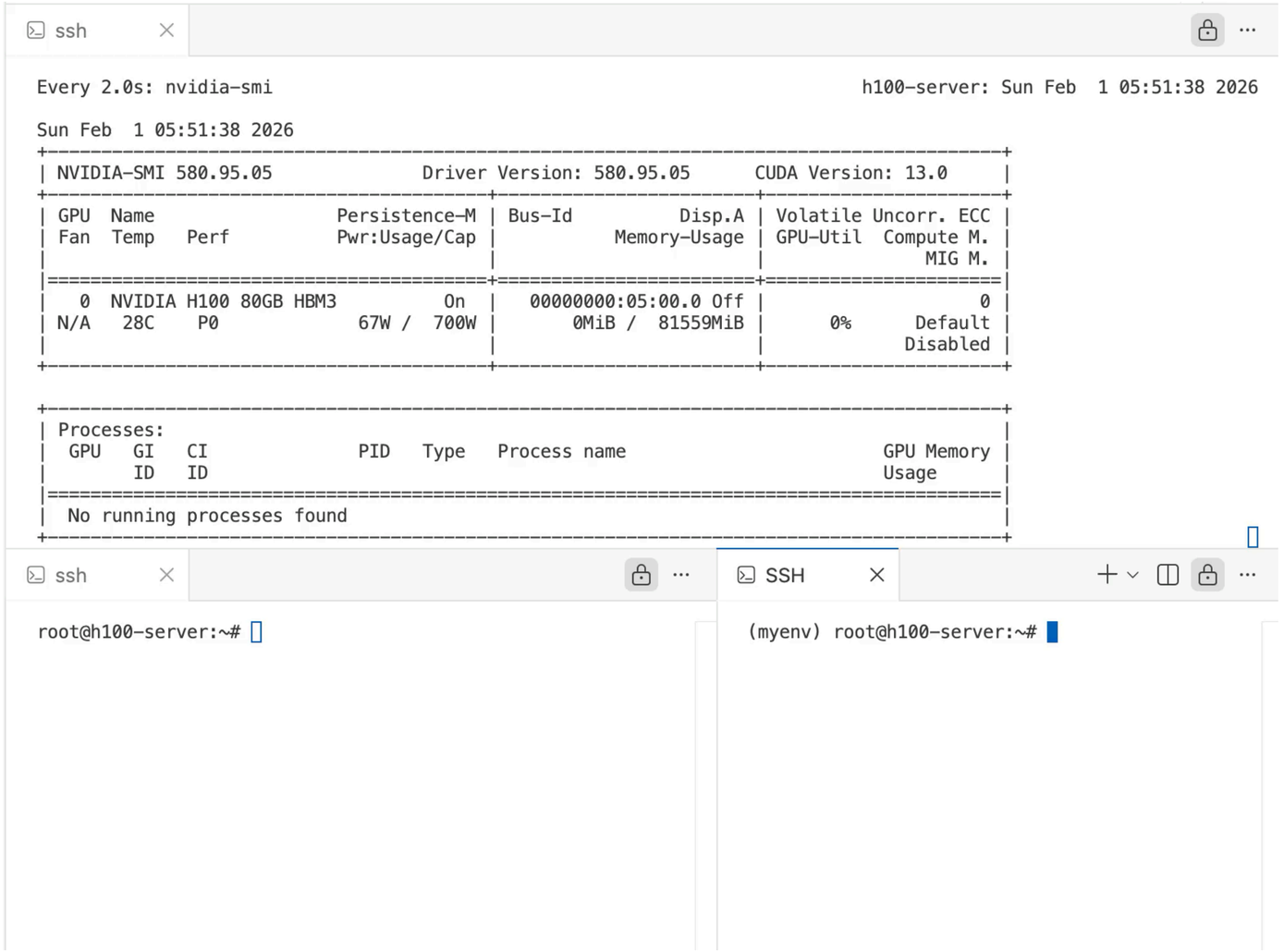
Low SM Activity

+

High VRAM usage

It is ok if this happens, Inference tend to have this behavior

Demo: Quick Temp CLI alert



dcgmi code reference

Tensor Core Metrics	DCGM Code	Purpose
DCGM_FI_PROF_PIPE_**TENSOR_ACTIVE**	1004	Overall Tensor core active usage
DCGM_FI_PROF_PIPE_**TENSOR_IMMA_ACTIVE**	1013	Int8 Tensor core active usage
DCGM_FI_PROF_PIPE_**TENSOR_HMMA_ACTIVE**	1014	FP16 / BF16 Tensor core active usage
DCGM_FI_PROF_PIPE_**TENSOR_DFMA_ACTIVE**	1015	FP64 Tensor core active usage
DCGM_FI_PROF_SM_ACTIVE	1002	Warp Assignment being run
DCGM_FI_PROF_SM_OCCUPANCY	1003	Active warps / Total Warps in SM

Tools to measure GPU performance

	DCGM	**nvidia-smi**	**Nsight Compute**	**Nsight System**
SM	Average SM usage	NA	SM level activity monitoring	Average SM Usage
Tensor Core Usage	Yes (Detailed)	No (very basic)	Yes	Yes

Monitoring solutions

- DCGM Exporter (Combine it with Prometheus and Grafana)
    - Manifests for these are readily available

Identifying bottlenecks

- Datatype based bottle necks : FP32, FP16, FP64
- Why certain configs will not work FP32 example using CUDA FP32 Cores instead of Tensor Cores TFP32
- What are the ways to identify these bottlenecks and supported auto conversions ?

Memory Bandwidth

- Memory Size is essential but Memory Bandwidth is the Key for Speedy inference
- Having Huge VRAM with low bandwidth will simply slow stuff down (Reason why consumer grade GPUs perform poorly)
- Existing Bottleneck in most AI workloads
- Compute is way way more Faster than the speed of data ingestion.

Thank You