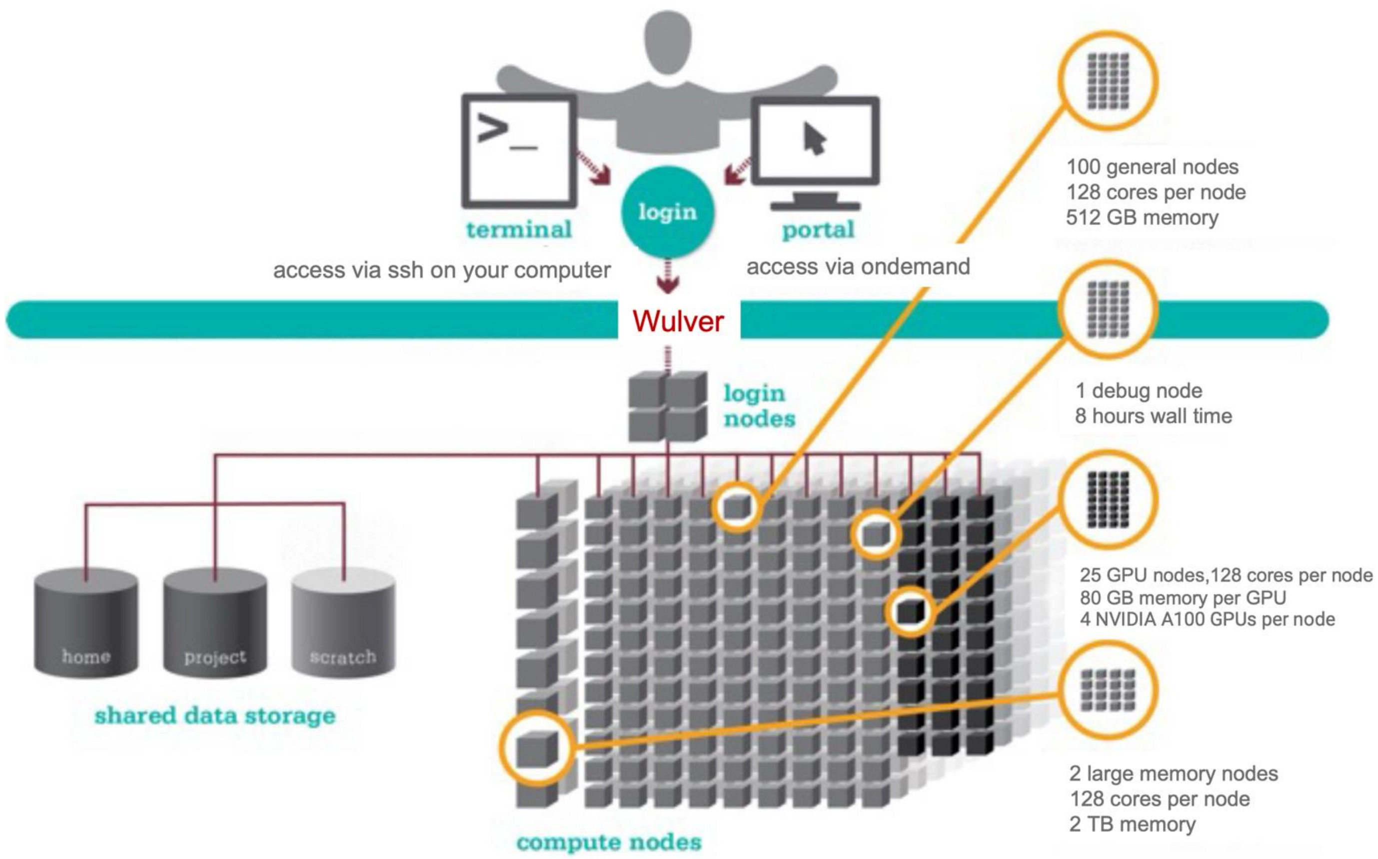
MIG (Multi-Instance GPU) on Wulver

Configuration • Usage • Best Practices

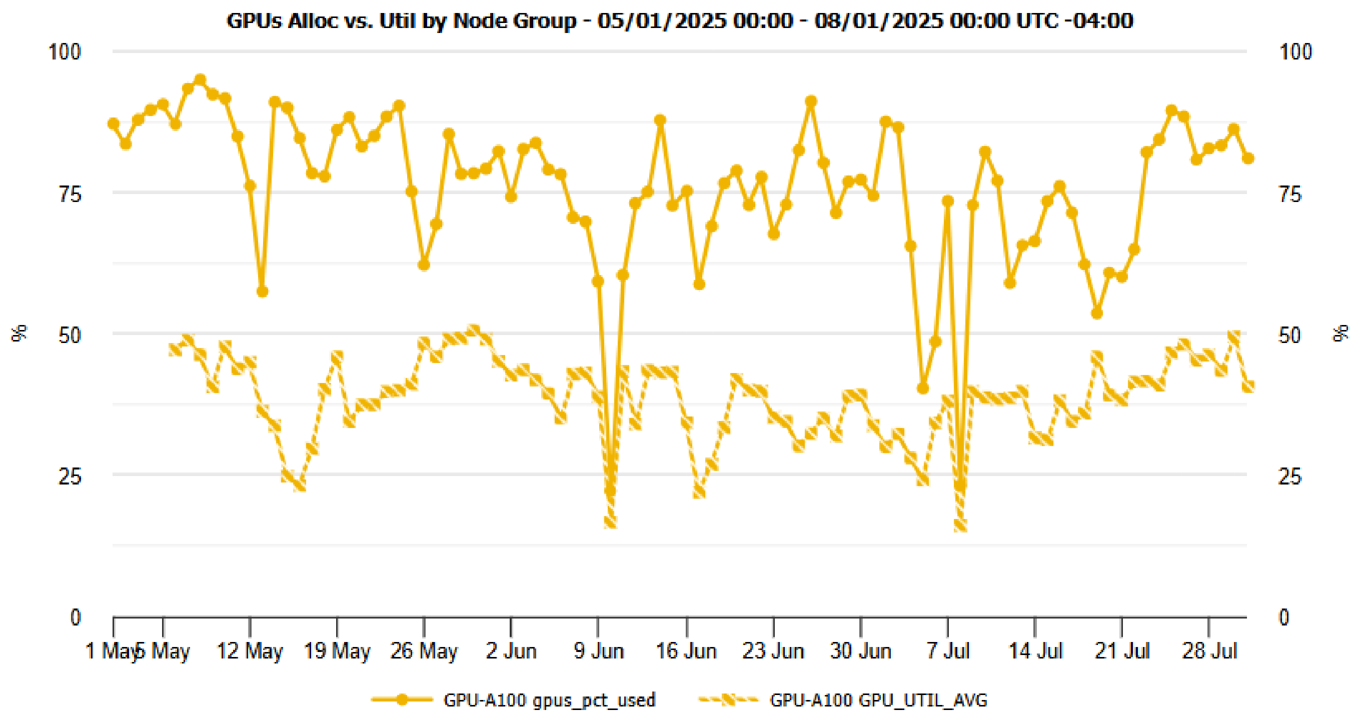
Agenda

- What is MIG?
- Why MIG on Wulver?
- MIG Configuration Example
- Submitting Jobs (srun & sbatch)
- New Billing Model
- Summary & Q&A

Wulver Specifications

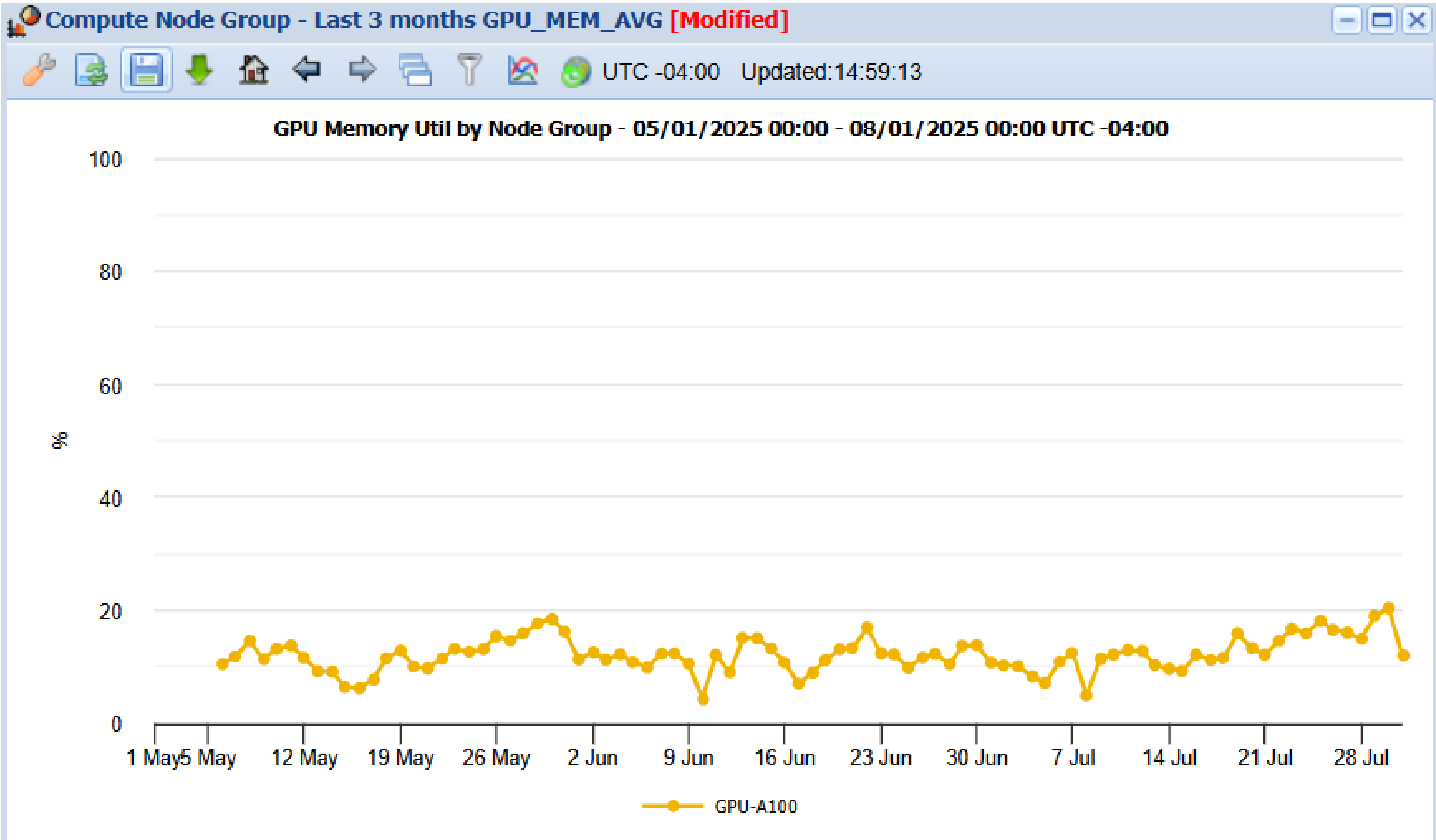


GPU Allocated vs used



GPUs were reserved ~85–95% of the time, while average compute utilization was ~25–50%

GPU Mem Usage



GPUs were **heavily** allocated but **lightly utilized**

GPU Utilization on Wulver

- GPUs were reserved **85–95% of the time**, but:
  - Average compute utilization was only **25–50%**.
  - Memory utilization was mostly **below 20%**.
- This mismatch caused:
  - Long queue times
  - Wasted capacity
  - Full A100 GPUs held by small jobs

What is MIG?

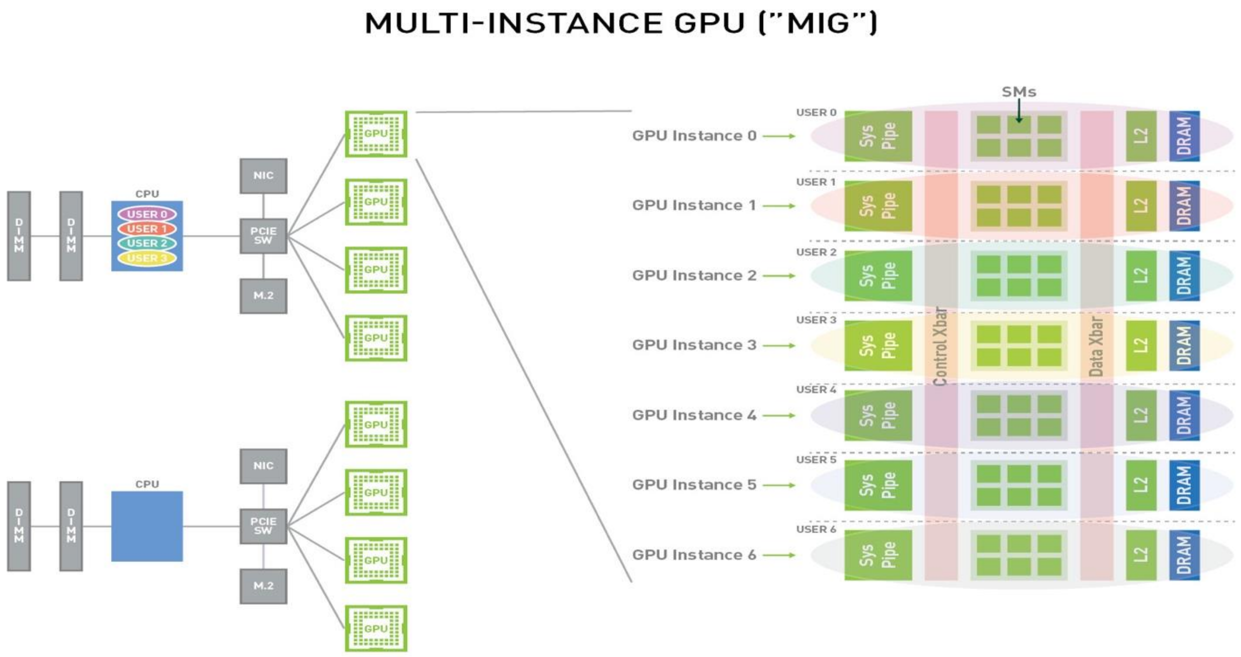
- Partitions an A100 GPU into multiple isolated GPU instances
- Each instance has dedicated memory, SMs(Streaming Multiprocessor), cache, bandwidth
- Appears as a separate GPU to CUDA, PyTorch, TensorFlow
- Up to 7 MIG GPU instances per A100

MIG Profiles on A100 MIG

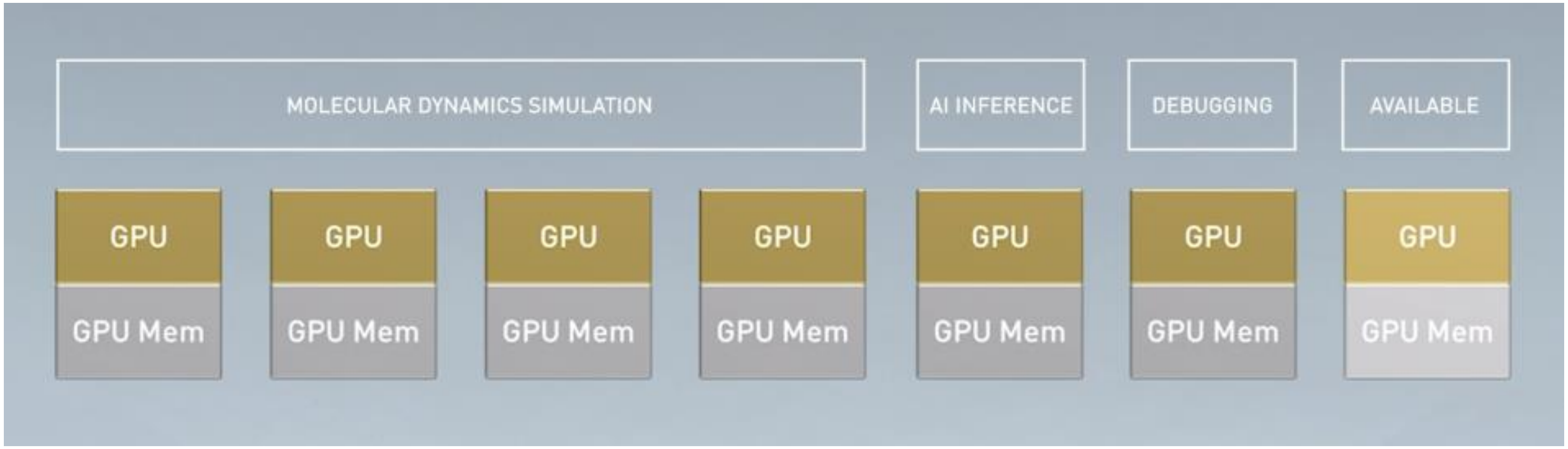
<table>
<tr><th>Config</th><th>GPC Slice #0</th><th>GPC Slice #1</th><th>GPC Slice #2</th><th>GPC Slice #3</th><th>GPC Slice #4</th><th>GPC Slice #5</th><th>GPC Slice #6</th><th>OFA</th><th>NVDEC</th><th>NVJPG</th><th>P2P</th><th>GPU Direct RDMA</th></tr>
<tr><td>1</td><td colspan="7">7</td><td>1</td><td>5</td><td>1</td><td>No</td><td rowspan="19">Supported MemBW proportional to size of the instance</td></tr>
<tr><td>2</td><td colspan="4">4</td><td colspan="3">3</td><td>0</td><td>2+2</td><td>0</td><td>No</td></tr>
<tr><td>3</td><td colspan="4">4</td><td colspan="2">2</td><td>1</td><td>0</td><td>2+1+0</td><td>0</td><td>No</td></tr>
<tr><td>4</td><td colspan="4">4</td><td>1</td><td>1</td><td>1</td><td>0</td><td>2+0+0+0</td><td>0</td><td>No</td></tr>
<tr><td>5</td><td colspan="3">3</td><td colspan="3">3</td><td></td><td>0</td><td>2+2</td><td>0</td><td>No</td></tr>
<tr><td>6</td><td colspan="3">3</td><td colspan="2">2</td><td>1</td><td></td><td>0</td><td>2+1+0</td><td>0</td><td>No</td></tr>
<tr><td>7</td><td colspan="3">3</td><td>1</td><td>1</td><td>1</td><td></td><td>0</td><td>2+0+0+0</td><td>0</td><td>No</td></tr>
<tr><td>8</td><td colspan="2">2</td><td colspan="2">2</td><td colspan="3">3</td><td>0</td><td>1+1+2</td><td>0</td><td>No</td></tr>
<tr><td>9</td><td colspan="2">2</td><td>1</td><td>1</td><td colspan="3">3</td><td>0</td><td>1+0+0+2</td><td>0</td><td>No</td></tr>
<tr><td>10</td><td>1</td><td>1</td><td colspan="2">2</td><td colspan="3">3</td><td>0</td><td>0+0+1+2</td><td>0</td><td>No</td></tr>
<tr><td>11</td><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="3">3</td><td>0</td><td>0+0+0+0+2</td><td>0</td><td>No</td></tr>
<tr><td>12</td><td colspan="2">2</td><td colspan="2">2</td><td colspan="2">2</td><td>1</td><td>0</td><td>1+1+1+0</td><td>0</td><td>No</td></tr>
<tr><td>13</td><td colspan="2">2</td><td>1</td><td>1</td><td colspan="2">2</td><td>1</td><td>0</td><td>1+0+0+1+0</td><td>0</td><td>No</td></tr>
<tr><td>14</td><td>1</td><td>1</td><td colspan="2">2</td><td colspan="2">2</td><td>1</td><td>0</td><td>0+0+1+1+0</td><td>0</td><td>No</td></tr>
<tr><td>15</td><td colspan="2">2</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>1+0+0+0+0</td><td>0</td><td>No</td></tr>
<tr><td>16</td><td>1</td><td>1</td><td colspan="2">2</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0+0+1+0+0+0</td><td>0</td><td>No</td></tr>
<tr><td>17</td><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="2">2</td><td>1</td><td>0</td><td>0+0+0+0+1+0</td><td>0</td><td>No</td></tr>
<tr><td>18</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td colspan="2">2</td><td>0</td><td>0+0+0+0+0+1</td><td>0</td><td>No</td></tr>
<tr><td>19</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>0+0+0+0+0+0+0</td><td>0</td><td>No</td></tr>
</table>

Reference: https://docs.nvidia.com/datacenter/tesla/mig-user-guide/index.html#a100-mig-profiles

MIG Configuration Example



MIG Configuration Example –Cont.



Source: https://developer.nvidia.com/techdemos/video/disc03

Why MIG on Wulver?

- Reduce queue times: smaller slices fit into schedule easier.
- Improve overall cluster utilization.
- Fair, predictable performance via hardware isolation.
- Lower SU cost for small/medium jobs compared to full A100.

MIG Configuration on Wulver

```
$ nvidia-smi -L
GPU 0: NVIDIA A100-SXM4-80GB (UUID: GPU-7e92a539-19c5-2948-67d5-a3b055a816ed)
  MIG 3g.40gb     Device  0: (UUID: MIG-0b586768-e667-5a5c-8298-7cfceacc60fe)
  MIG 2g.20gb     Device  1: (UUID: MIG-0df8d316-2b51-55d5-9d6f-c78f4edc27f4)
  MIG 1g.10gb     Device  2: (UUID: MIG-379a24be-c1b3-5d57-959a-814f18460157)
  MIG 1g.10gb     Device  3: (UUID: MIG-fc813cd3-631f-5a59-9ead-3ac7ed3215db)
GPU 1: NVIDIA A100-SXM4-80GB (UUID: GPU-377e6263-247c-2736-6b78-6439f6164e94)
  MIG 3g.40gb     Device  0: (UUID: MIG-079d02a7-0892-5ea1-88f5-ca22b52ee4f3)
  MIG 2g.20gb     Device  1: (UUID: MIG-978fe824-62ed-5bc7-8f96-c00e09dfecc7)
  MIG 1g.10gb     Device  2: (UUID: MIG-dfdb3d5e-c3f6-5189-b355-7a65c8a8368e)
  MIG 1g.10gb     Device  3: (UUID: MIG-ed0b10c7-3891-5f2e-a02c-ece69c8ce419)
GPU 2: NVIDIA A100-SXM4-80GB (UUID: GPU-c479faea-dea9-fc74-8029-0c5bcd509ec7)
  MIG 3g.40gb     Device  0: (UUID: MIG-ec7618a0-0e1a-5919-a88e-5c9ca61cebad)
  MIG 2g.20gb     Device  1: (UUID: MIG-9ec817d8-06b6-5ac0-94fc-9ae1c233d506)
  MIG 1g.10gb     Device  2: (UUID: MIG-0f10402e-629d-5f98-a838-0b3798fcd842)
  MIG 1g.10gb     Device  3: (UUID: MIG-97b3e81f-d23e-50ee-b432-c1938638f6f4)
GPU 3: NVIDIA A100-SXM4-80GB (UUID: GPU-3d974751-2093-6cf6-111a-3a8c95f68a97)
  MIG 3g.40gb     Device  0: (UUID: MIG-d0128346-7241-502e-99bf-e0e73e96be4d)
  MIG 2g.20gb     Device  1: (UUID: MIG-775a5688-9a5c-54c0-b481-c514414d6149)
  MIG 1g.10gb     Device  2: (UUID: MIG-7a0419b4-6c67-5ba0-aa66-b68778fc8bdc)
  MIG 1g.10gb     Device  3: (UUID: MIG-1d33f90a-1571-5770-88e9-6505a4a6ddf9)
```

MIG Profiles on Wulver

- MIG
  - a100_10g → ~10 GB slice
  - a100_20g → ~20 GB slice
  - a100_40g → ~40 GB slice
- full GPU
  - a100 (--gres=gpu:a100:1) (80 GB)

How MIG Appears in SLURM

- --gres=gpu:a100_10g:1
- --gres=gpu:a100_20g:1
- --gres=gpu:a100_40g:1
- --gres=gpu:a100:1 (full GPU)
- Each MIG slice is treated as a unique GPU device

GPU MIG	Slurm Directive
10G MIG	`--gres=gpu:a100_10g:1`
20G MIG	`--gres=gpu:a100_20g:1`
40G MIG	`--gres=gpu:a100_40g:1`

Submitting Jobs (Batch)

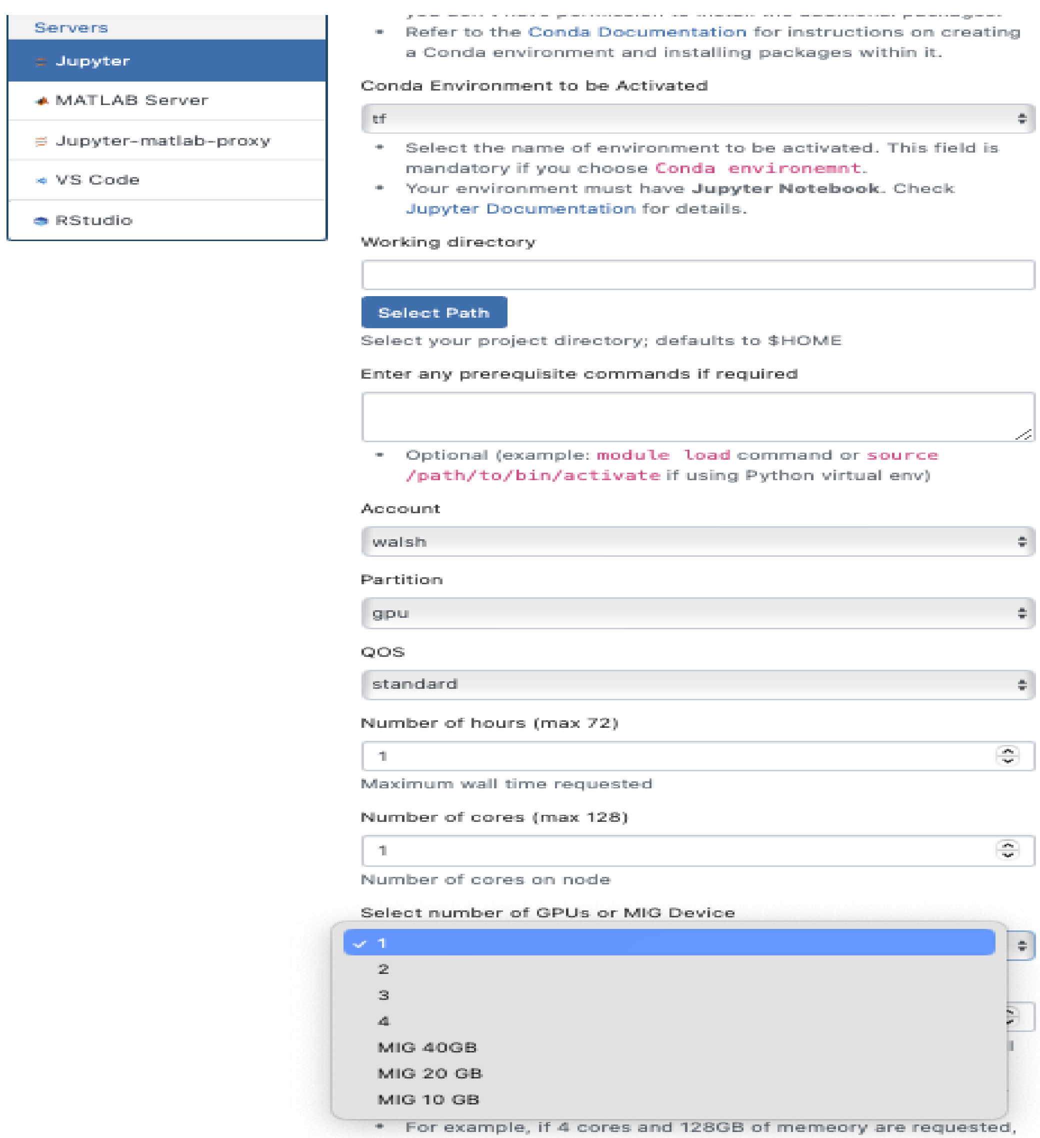
```
#!/bin/bash -l
#SBATCH --job-name=mig_test
#SBATCH --output=%x.%j.out
#SBATCH --error=%x.%j.err
#SBATCH --partition=gpu
#SBATCH --qos=standard
#SBATCH --account=$PI
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem-per-cpu=4000M
#SBATCH --gres=gpu:a100_40g:1
#SBATCH --time=02:00:00

module load Miniforge3
conda activate torch-cuda
srun python torch_tensor.py
```

Submit Jobs (Interactive)

```
$srun --partition=gpu \
      --account=$PI_ucid \
      --qos=standard \
      --gres=gpu:a100_10g:1 \
      --time=00:59:00 \
      --pty bash
```

MIG in Open OnDemand



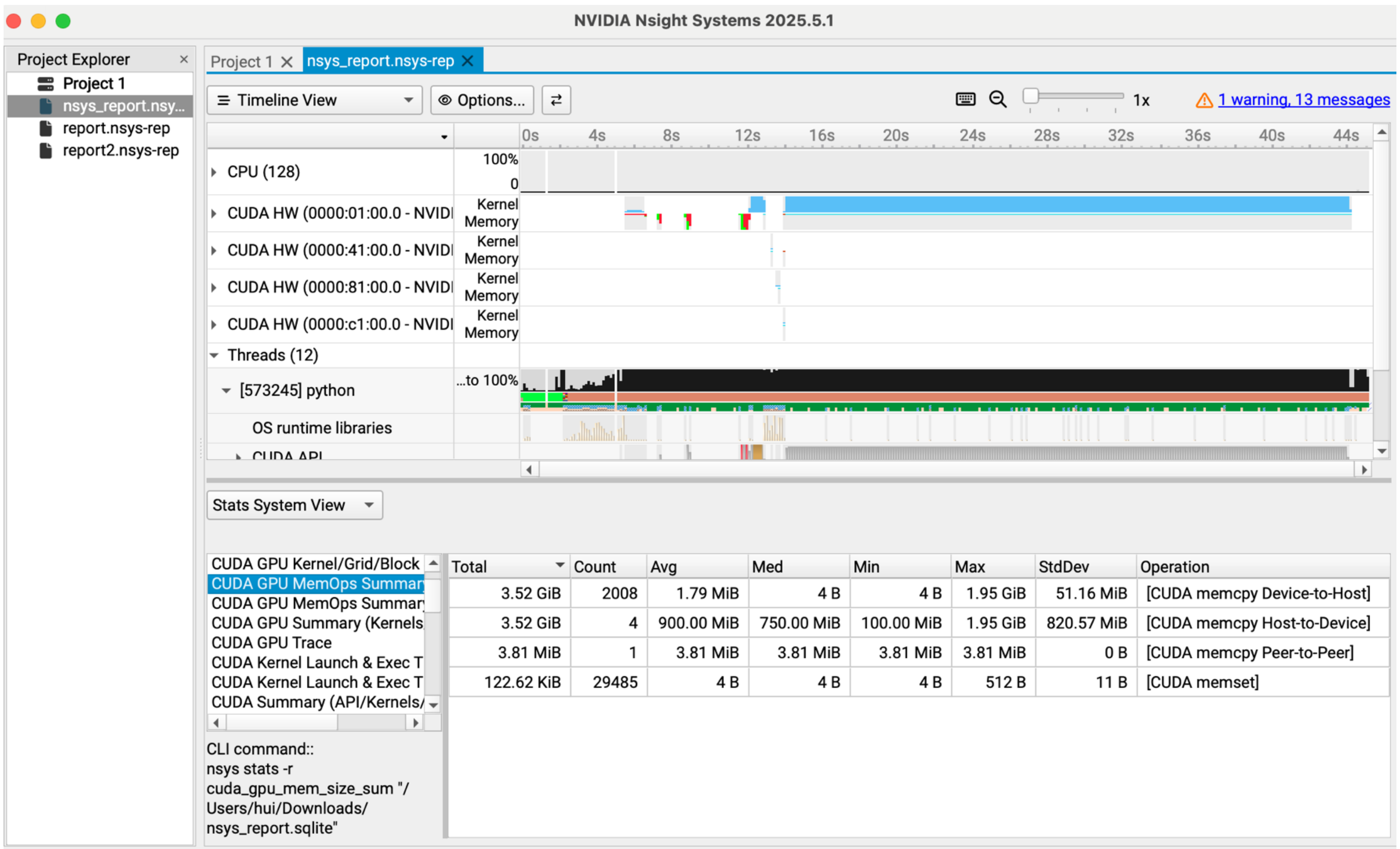
Key Factors for Choosing a MIG Slice

- **Compute requirements** → CUDA cores, Tensor cores, bandwidth
- **Memory needs** → match slice memory to workload footprint
- **Concurrency** → number of simultaneous jobs / services
- **Latency sensitivity** → real-time or interactive workloads
- **Workload profile** → compute-bound vs memory-bound

When to Use Which MIG Slice?

- 10g → inference, debugging, preprocessing
- 20g → medium training, light ML workloads
- 40g → larger models, heavy DL
- 80g (full GPU) → GPU memory-intensive or high-performance workloads

Tools to Profile GPU jobs



Best Practices

- Use smallest MIG slice needed
- Avoid over-allocating CPU or RAM
- Monitor GPU usage with nvidia-smi or other profiling tools
- Track job resource usage for reproducibility

Service Unit (SU) Calculation

$$SU = MAX(\#CPUs, Memory(in\ GB)/4) + 16 \times (GPU\ Memory\ requested / 80GB)$$

- CPU/Memory term: Accounts for CPU cores and RAM usage
- GPU term: Scales with GPU memory requested
- Applies to both MIG slices and full A100 GPUs
- HPC Service Units resource page: https://hpc.njit.edu/Running_jobs/service-units/

How SU Charges Are Applied

- CPU and memory usage are always included in SU calculations
- Full A100 GPU (80 GB) → billed as 16 SU/hr
- MIG slices → billed in fractions, proportional to GPU memory share

Billing Examples (NJIT, 4 CPUs)

Profile	GPU Memory	CPU Alloc	SU/hr
a100_10g	10 GB	4	6
a100_20g	20 GB	4	8
a100_40g	40 GB	4	12
Full A100	80 GB	4	20

Example 1: 4 CPUs + full A100 GPU (80 GB)

SU = MAX(4, 4*4G/4) + 16 × (80/80) = 20 SU/hr

Example 2: 4 CPUs + MIG slice (10 GB)

SU = MAX(4, 4*4G/4) + 16 × (10/80) = 6 SU/hr

Useful Links

- MIG Documentation: https://hpc.njit.edu/MIG/
- Service Units: https://hpc.njit.edu/Running_jobs/service-units/
- Events & Workshops: https://hpc.njit.edu/HPC_Events_and_Workshops/

Summary & Q&A

- MIG enables right-sizing:
  a100_10g / 20g / 40g / full GPU.

- SU charges are now calculated more precisely, incorporating full details for CPU, memory, and GPU requests.

- MIG available in different partition

- Questions?