Introduction to GPU/AI resources on Minerva

Minerva Scientific Computing Environment

https://labs.icahn.mssm.edu/minervalab

Hyung Min Cho, PhD
The Minerva HPC Team

September 26, 2025

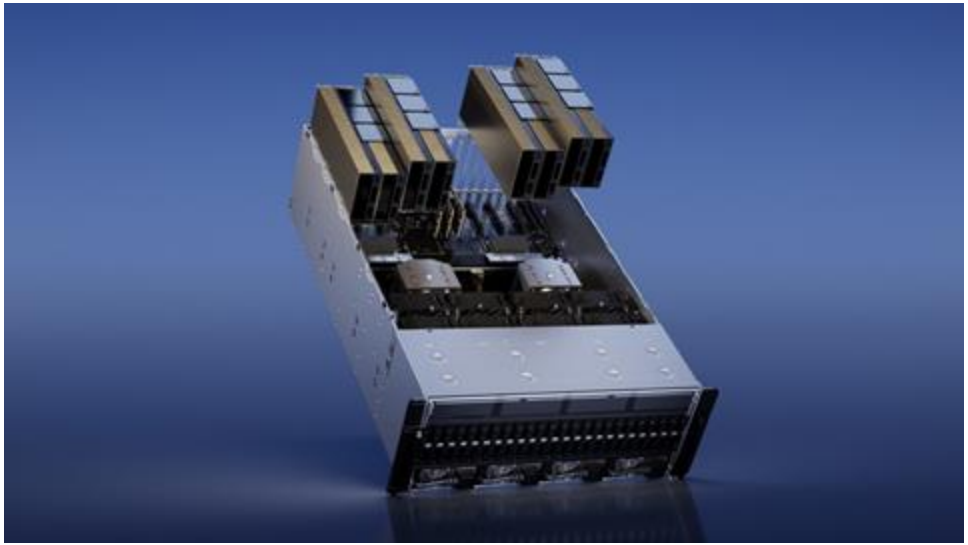
Icahn
School of
Medicine at
Mount
Sinai

Outline

- What is GPU?
- GPU resources on Minerva
- User GPU Software Environment on Minerva
- Run GPU jobs in LSF

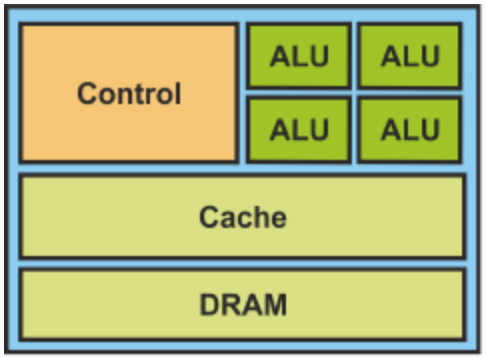
What is GPU?

- A graphics processing unit (GPU) is a specialized electronic circuit initially designed to accelerate computer graphics and image processing.
- GPUs can be used across a wide range of compute-intensive applications:
  - AI/Machine Learning
  - Simulations
  - Professional visualization
  - Gaming

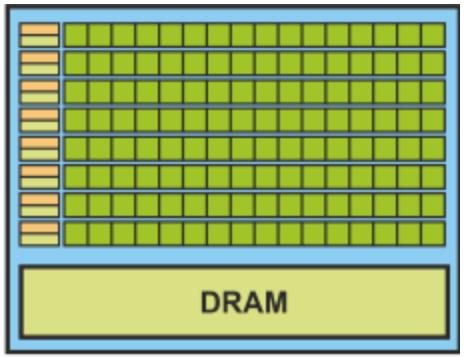


CPU vs GPU

CPU vs GPU		
	CPU	**GPU**
Function	Generalized component that handles main processing functions of a server	Specialized component that excels at parallel computing
Processing	Designed for serial instruction processing	Designed for parallel instruction processing
Design	Fewer, more powerful cores, low latency	More cores than CPUs, but less powerful than CPU cores, high throughput



CPU



GPU

reference

DOI:10.1117/12.892282

GPU resources on Minerva

- GPU resources on Minerva
  - Current:
    - Interactive queue (1 GPU node)
    - gpu queue for batch (74 GPU nodes)
    - 316 GPUs in total

GPU model	V100	A100	A100-80GB	H100	H100-NVlink	L40S
# of nodes	12	8	2	2	47	4
GPU card/node	4	4	4	4	4	8
CPU cores	32	48	64	64	96	96
host memory	384 GB	384 GB	2 TB	512 GB	1.5 TB	1.5 TB
GPU memory	16 GB	40 GB	80 GB	80 GB	80 GB	48 GB

User GPU Software Environment - Major packages

OS: **Rocky 9.4** with **glibc-2.34(GNU C library) available**

- Packages with GPU support:
  - Schrödinger Suite, Amber tools, NAMD, Gromacs, Alpha Fold2, etc.
- AI tools with python/3.12.5
  - CuPy, cuDF, cuML, Numba, scikit-learn, Scanpy, Squidpy, etc.
  - Minerva Python instruction
- AI tools with conda
  - MONAI, Rapids, NVFlare, tensorflow, pytorch, etc.
  - Minerva conda instruction
- AI tools with singularity
  - Holoscan, BioNeMo, Parabricks, DeepVariant, Alpha Fold3, etc.
  - Minerva singularity instruction
  - Minerva Singularity training
- Cuda toolkit versions up to 12.4.0

User Software Environment - Anaconda Distribution

- https://labs.icahn.mssm.edu/minervalab/documentation/conda/
- **To avoid incompatibilities with other python, clear your environment with module purge before loading Anaconda**

  $ml purge

  $ml anaconda3/2024.06

  $conda env list ***# get a list of the env available (Or*** *$conda info --envs)*
- User should install their own envs locally, (see more guide here)
  - ➔ Use option -p PATH, --prefix PATH Full path to environment location (i.e. prefix).

    $conda create python=3.x -p /sc/arion/work/gail01/conda/envs/myenv

    $conda env create -p myenv -f environment.yml
  - ➔ Set envs_dirs and pkgs_dirs in .condarc file, specify directories in which environments and packages are located

    $conda create -n myenv python=3.x
- Set conda base auto-activation false

  conda config --set auto_activate_base false

  More at Conda config guide

```
$ cat ~/.condarc file
envs_dirs:
- /sc/arion/work/gail01/conda/envs
pkgs_dirs:
- /sc/arion/work/gail01/conda/pkgs
conda config --set auto_activate_base false
```

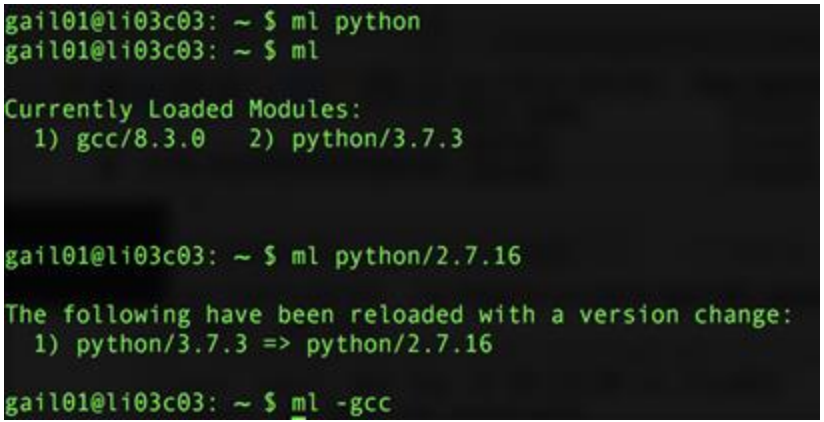
User Software Environment: Lmod

> 1000 modules, and different versions are supported on Minerva

Lmod Software Environment Module system implemented:

- Search for module: **$module avail** or **$ module spider**
  Check all available R versions *$ ml spider R*

  …….R/3.3.1, R/3.4.0-beta, R/3.4.0, R/3.4.1, R/3.4.3_p, R/3.4.3, R/3.5.0, R/3.5.1_p, R/3.5.1, R/3.5.2, R/3.5.3
- To check the detailed PATH setting in module files: **$ml show R**
- Load module: **$ml python** **or $module load python** or **$ml python/3.12.5** (for a specific version)
- Unload module **$ml -gcc or $module unload gcc**
- List loaded modules: **$ml or $module list**
- Purge ALL loaded modules **$ ml purge**
- Autocompletion with tab
- More at:
  - Minerva Lmod guide
  - Lmod user guide



Ollama

- Ollama is a platform that enables users to interact with Large Language Models (LLMs) via an Application Programming Interface (API)

  https://github.com/ollama/ollama
- It is a powerful tool for generating text, answering questions, and performing complex natural language processing tasks. It provides access to various fine-tuned LLMs.
- We provide an Ollama wrapper script that allows you to start an Ollama server on Minerva's compute node and access it from your local machine through an API endpoint:

  https://labs.icahn.mssm.edu/minervalab/documentation/ollama/

Interactive access to GPU resources

- Set up an interactive environment on compute nodes
- Useful for testing and debugging jobs

```
bsub -P acc_hpcstaff -q gpu -n 4 -W 2:00 -R rusage[mem=4000] -R span[hosts=1] -gpu num=1 -R v100 -XF -Is /bin/bash
```

- Interactive, gpu, gpuexpress can be specified for -q.
- -Is: Interactive terminal/shell
- -XF: X11 forwarding
- /bin/bash : the shell to use
- *If GPU model flag is not specified, your job will start on the earliest available GPU nodes.*
- GPU option specification:
  - V100: **-R** v100
  - A100: **-R** a100
  - A100-80G: **-R** a10080g
  - H100: **-R** h10080g
  - NVLinked H100: **-R** h100nvl
  - L40S: **-R** l40s

Minerva LSF queues with GPUs

Queue structure in Minerva	
Queue	**Wall time limit**
interactive (Dedicated to interactive jobs)	12 hours
gpu	6 days/144 hours
gpuexpress	15 hours

bhosts

- bhosts queue_name

```
[choh07@li04e03 ~]$ bhosts gpuexpress
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
lg02e17            ok              -     64      4      4      0      0      0
lg03a02            ok              -     32      0      0      0      0      0
lg03a03            ok              -     32      1      1      0      0      0
lg03a04            ok              -     32     10     10      0      0      0
lg03a05            ok              -     32      4      4      0      0      0
lg03a06            ok              -     32      0      0      0      0      0
lg03a07            ok              -     32      5      5      0      0      0
lg03a08            ok              -     32      5      5      0      0      0
lg03a09            ok              -     32      6      6      0      0      0
lg03a10            ok              -     32      1      1      0      0      0
lg03a11            ok              -     32     10     10      0      0      0
lg03e18            ok              -     64     32     32      0      0      0
lg03e19            ok              -     64     36     36      0      0      0
lg05e01            ok              -     96     16     16      0      0      0
lg05e02            ok              -     96      4      4      0      0      0
lg05e03            ok              -     96      4      4      0      0      0
lg05e04            ok              -     96      4      4      0      0      0
lg05e05            ok              -     96      4      4      0      0      0
lg05e06            ok              -     96      4      4      0      0      0
lg05e07            ok              -     96     67     67      0      0      0
lg05e08            ok              -     96      7      7      0      0      0
lg05e09            ok              -     96     10     10      0      0      0
lg05e10            ok              -     96      4      4      0      0      0
lg05e11            ok              -     96      7      7      0      0      0
```

bhosts

- bhosts -R gpu_model

```
[choh07@li04e03 ~]$ bhosts -R v100
HOST_NAME          STATUS       JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
lg03a01            ok              -     32     12     12      0      0      0
lg03a10            ok              -     32      1      1      0      0      0
lg03a02            ok              -     32      0      0      0      0      0
lg03a06            ok              -     32      0      0      0      0      0
lg03a03            ok              -     32      1      1      0      0      0
lg03a08            ok              -     32      5      5      0      0      0
lg03a05            ok              -     32      4      4      0      0      0
lg03a09            ok              -     32      6      6      0      0      0
lg03a07            ok              -     32      5      5      0      0      0
lg03a04            ok              -     32     10     10      0      0      0
lg03a11            ok              -     32     10     10      0      0      0
```

Batch job submission example

$ cat myfirst.lsf

```
#!/bin/bash
#BSUB -J myfirstjob                          # Job name
#BSUB -P acc_hpcstaff                        # REQUIRED; To get allocation account,
type "mybalance"
#BSUB -q premium                             # queue; default queue is
premium
#BSUB -n 1                                   # number of compute
cores (job slots) needed, 1 by default
#BSUB -W 6:00                                # REQUIRED; walltime in
HH:MM
#BSUB -R rusage[mem=4000]                    # 4000 MB of memory request per "-n";
3000 MB by default
#BSUB -oo %J.stdout                          # output log (%J : JobID)
#BSUB -eo %J.stderr                          # error log
#BSUB -L /bin/bash                           # Initialize the execution
environment

ml gcc                                       # Commands
```

GPGPU - batch jobs

```
#BSUB -q gpu                                  # submit to gpu queue
#BSUB -n Ncpu                                 # Ncpu is 1~48 on A100

#BSUB -gpu num=4                              # request 4 GPUs per node on A100 node
#BSUB -R a100                                 #
#BSUB -R span[hosts=1]                        # request all gpu cards on the same node

module purge
module load anaconda3 ( or 2)                 # to access tensorflow
module load cuda                              # to access the drivers and supporting
source activate tfGPU                         subroutines

python -c "import tensorflow as tf"
```

GPGPU - batch jobs (continue)

- LSF will set CUDA_VISIBLE_DEVICES to the list of GPU cards assigned to the job.

  E.g: 2,1,3 Most standard packages honor these assignments
  - DO NOT MANUALLY CHANGE THE VALUE OF CUDA_VISIBLE_DEVICES.
- Multiple GPU cards can be requested across different GPU nodes

```
#BSUB -q gpu                    # submit to gpu queue
#BSUB -n 8                      # 8 compute cores requested
#BSUB -R span[ptile=2]          # 2 cores per node, so 4 nodes in total requested
#BSUB -R h100nvl                # request specified gpu node h100nvl
#BSUB -gpu num=2                # 2 GPUs requested per node
```

Note that 2 GPU cards will be reserved on each of 4 nodes for your job. If your job cannot /does not run in distributed mode, you will still lock these resources on the nodes that you are not using and prevent others from being dispatched to those node.

CUDA_VISIBLE_DEVICES may be defined differently on each of the nodes allocated to your job.

GPGPU - Local SSD

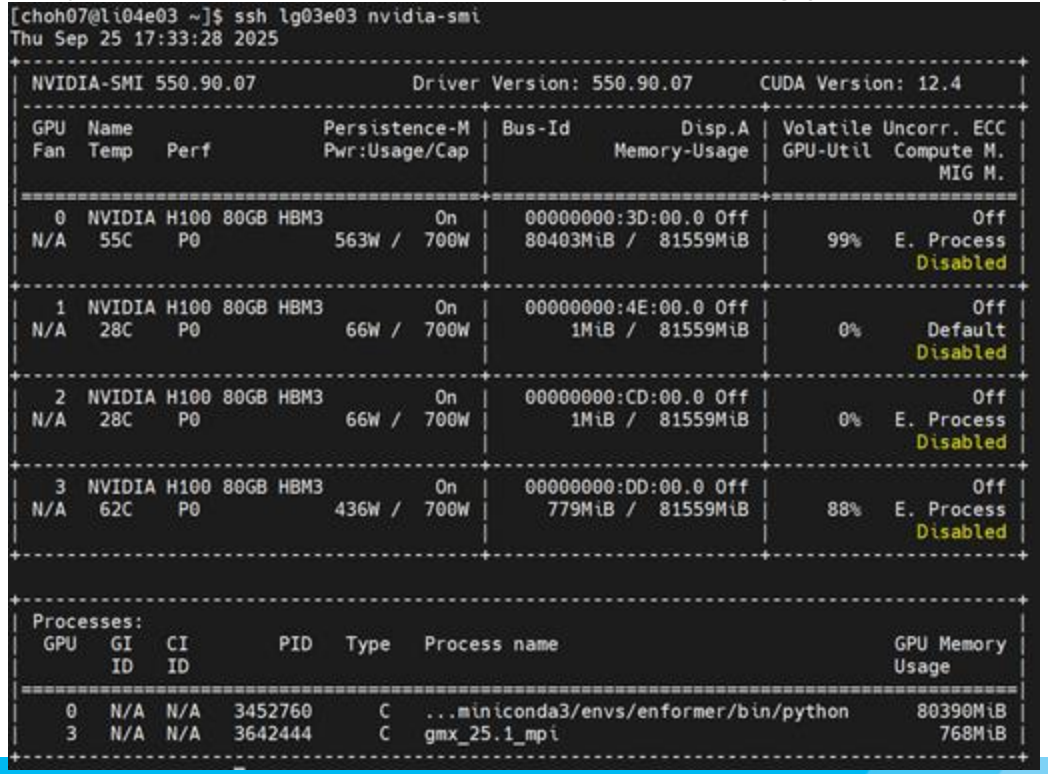
- Make your own directory under /ssd and direct your temporary files there.
- Clean up your temporary files after completion.

A100	**1.8 TB SATA SSD**
A100-80GB	**7.0 TB NVMe PCIe SSD**
H100	**3.84 TB NVMe PCIe SSD**
L40S	**3.84 TB NVMe PCIe SSD**

```
#BSUB -q gpu
#BSUB -gpu num=2
#BSUB -R a100
#BSUB -R span[hosts=1]
#BSUB -E "mkdir /ssd/YourID_$LSB_JOBID"
#BSUB -Ep "rm -rf /ssd/YourID_$LSB_JOBID"
```

nvidia-smi

- A monitoring and management command line utility, nvidia-smi
- Only available on nodes with GPUs during job execution



Minerva Ticket Submission

- Send an email to: hpchelp@hpc.mssm.edu to start a ticket.
- Information to include:
  - The error message or the location of the log file on Minerva.
    - use command “pwd” to check the current working directory.
  - All the commands you used to get the error message.
  - The location of the scripts used and how did you run the script.
  - Which node.
    - yuj25@li04e04
  - The job ID and location of the job script.
    - Job <123456789> is submitted to queue <premium>.

- Please do not include any “<” in the email.

Friendly Reminder

- Never run jobs on login nodes
  - For file management, coding, compilation, etc., purposes only
- Never run jobs outside LSF
  - Fair sharing
  - Scratch disk not backed up, efficient use of limited resources
  - Job temporary dir configured to /local/JOBS instead of /tmp.
- **WE DO NOT BACKUP USER FILES. PLEASE ARCHIVE YOUR IMPORTANT FILES.**
- Follow us by visiting https://labs.icahn.mssm.edu/minervalab
- Acknowledge Scientific Computing at Mount Sinai and NIH funding in your publications with the template https://labs.icahn.mssm.edu/minervalab/policies/acknowledge-scientific-computing-at-mount-sinai/

New Large-Scale Empire AI GPUs available for Mount Sinai Researchers

Empire AI is a consortium of ten New York State institutions with support from New York State, a shared computing facility for artificial intelligence (AI) and high-performance computing (HPC) technologies in New York.

Empire AI is deploying large-scale GPU clusters in phases over 10 years.

The first Alpha system has been in production since Oct. 2024. The second Beta system will be in production in Dec. 2025. **hardware details are described here.**

How to onboard your projects to Empire AI?

1. To access, Mount Sinai PIs must agree no sensitive data will be transferred to Empire AI cluster by submitting the form at https://redcap.link/EAI-DUA
2. After the above form received, you will be contacted with instructions for Empire AI project onboarding

Questions?

1. If you have general questions about this Empire AI resources, please open a ticket at hpchelp@hpc.mssm.edu for now.
2. All future ticket and communication will go via Empire AI ticketing system at support@empireai.edu.
3. We are planning a **Town Hall on Empire AI**, scheduled for **October 10 at 12:00 PM (noon).** Will announce details later.

Acknowledgements

- Supported by the Clinical and Translational Science Awards (CTSA) grant UL1TR004419 from the National Center for Advancing Translational Sciences, National Institutes of Health.

Last but not Least

- Got a problem? Need a program installed? Send an email to:

hpchelp@hpc.mssm.edu

Thank you!