**For April 16th, 2024**

**To do:**

Updates, questions, concerns

**Workshop date, format, who is going to run it - May 7, 2024 9AM**

**Getting an account** for non MRI folks

**Time limit is infinite - so put a guardrail**

**Ask : limits on how many gups a user can use, how many jobs you can submit**

**Subheadings on how to document**

**HOw to get an account**

**Login to the system**

**Creating directories**

**Copying files to ad from the systems**

**Sbumtiting jobs**

**-a non-gpu job**

**-a gpu job**

**Checkign the stauts of the job**

**To do:**

**Ask : limits on how many gups a user can use, how many jobs you can submit**

**Things to discuss 04/02/2024**

Updates, questions, concerns

Workshop date, format, who is going to run it - status of the manual

**Getting an account**

User requests an account to Dukka with their info

login to the system

creating directories, copying files to and from and the system, submitting jobs, checking the status of the jobs,...

Opening up the cluster - April 5th

**Things to discuss for March 5th**

1. **Updates from each individual**
2. **Problems and issues**
3. **How to organize the tutorials - let’s really think about how to break into 10-15 documents**
4. **When and how to make the cluster available for the whole group - April 1st, workshop around that time**
5. **How to do a workshop such as https://www.depts.ttu.edu/hpcc/about/documents/HPCC\_New\_User\_Training\_P1\_SUM23.pdf**

**Finally, website Something like this**

**https://www.depts.ttu.edu/hpcc/userguides/index.php**

**Issues**

* Cannot use two nodes for training (Tao) - check with IT
* **File system - /scratch, other users have permission issues**

**Data is @ /mnt/mridata (77TB)**

**/mnt/mridatafast (22TB)**

**For the /scratch space**

On the MRI nodes I've created both a SSD and NVME scratch partition available at:

/scratch-ssd

/scratch-nvme

To keep things organized users should add this to their batch files to create a folder for their user on the node and they should work within that folder:

if [ ! -d /scratch-ssd/$USER ]; then

mkdir /scratch-ssd/$USER;

chmod 700 /scratch-ssd/$USER;

fi

if [ ! -d /scratch-nvme/$USER ]; then

mkdir /scratch-nvme/$USER;

chmod 700 /scratch-nvme/$USER;

fi

In my case I'd want to keep scratch files in either /scratch-ssd/jlkarau/ or /scratch-nvme/jlkarau/

It would be a good habit to have users clean those folders after use as well. If they fille it could compromise the node's ability to run.

**Meeting Notes 02/20/2023**

**Python virtual environment create (see other tutorials)**

**pip3 install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118**

[**https://www.psc.edu/resources/bridges-2/user-guide/**](https://www.psc.edu/resources/bridges-2/user-guide/)

**Following**

**Cannot use anaconda**

**How to create various environments?**

**-Should you create virtual environments?**

**https://www.geeksforgeeks.org/creating-python-virtual-environment-windows-linux/**

**Use python**

**Updates from each of you**

1. Next Action items - how can we make readily available to our user base
   1. Beta-testing
   2. Creating sample scripts for sample tasks
   3. Creating a robust how-to documents including linux tutorial
   4. Discussion on required training etc. to bring someone on board
   5. A website

Joshua - how to install the software, - MD simulation, cuda

Ponkshnan- PSC documents, <https://www.psc.edu/resources/bridges-2/user-guide/>

<https://docs.nersc.gov/systems/perlmutter/>

* DNN/ML model

-containers for computer vision - Abel

~~-slack channel - Abel~~

1. Doing a workshop on how to use the Blizzard cluster
2. As may arise

**Questions: how do we handle installation that requires sudo commands? - cuda libraries**

* **Cpu version running fine**

**Any (best) how to tutorials that you found**

**What would be helpful to get started?**

**Meeting Notes 01/23/2023 and to do items**

01/23/2023 MRI testing meeting

Description:

DeepBlizzard consists of 1 head node that has AMD EPYC 7343 3.2GHz processor with 128M cache and 256 GB RDIMM memory. DeepBlizzard also has a storage array that consists of 120TB HDD SAS of hard disk and 30TB of SSD. This cluster consists of 3 high memory (dual CPU nodes with a total of 64 cores and a terabyte of ram) A30 GPU nodes. Each A30 node contains 2 NVIDIA Ampere A30 GPU cards. An A30 GPU contains 3584 CUDA cores, 224 Tensor cores, and 24GB of HBM2 RAM. Additionally, DeepBlizzard cluster also consists of another 2 A30 GPU nodes. This brings the total number of A30 GPUs to 10 A30GPUs.

Furthermore, DeepBlizzard cluster has 7 A100 GPU nodes which contains 4 A100 GPU. An A100 GPU contains 6912 CUDA cores, 432 Tensor cores, and 80GB of HBM2e RAM with a bandwidth of 1555GB/sec. Hence, there are a total of 28 A100 GPUs. In each GPU node, the GPUs are connected through a third-generation NVLink bridge, yielding up to a total bandwidth of 112.5 GB/sec for A40s to 600GB/sec for A100s. The GPUs communicate with the host CPUs through PCIe-4 which delivers a rate of 64GB/s. Each GPU node is hosted by two CPUs, each with 32 physical cores/64 logical cores. Finally, to obtain the best results from these multiple GPU’s a high-performance 40 Port HDR Infiniband interconnect to handle GPU-to-GPU communications.

Largememnodes and GPUs

1. Introduction to the cluster and basic how to
2. Purpose of this meeting
3. Next Action items - how can we make readily available to our user base
   1. Beta-testing
   2. Creating sample scripts for sample tasks
   3. Creating a robust how-to documents including linux tutorial
   4. Discussion on required training etc. to bring someone on board
   5. A website

Joshua - how to install the software, - MD simulation, cuda

Ponkshnan- PSC documents, <https://www.psc.edu/resources/bridges-2/user-guide/>

<https://docs.nersc.gov/systems/perlmutter/>

* DNN/ML model

-containers for computer vision - Abel

~~-slack channel - Abel~~

1. Doing a workshop on how to use the Blizzard cluster
2. As may arise

**How to MRI Cluster**

***Login***

ssh login-mri.research.mtu.edu

partitions are mrigpu, mrilargemem and mrijobs

Softwares are deployed using easybuild but one should be able to install software in your local home directory (?)

***Some helpful commands***

Assigning the GPUs is pretty straight forward:

for example this would show you having access to 1 GPU on a compute node

srun --partition=mrigpu --gres=gpu:1 nvidia-smi

This would show you having access to 2 GPUs on a compute node, and so on

srun --partition=mrigpu --gres=gpu:2 nvidia-smi

software available can be found by:

module use /mnt/it\_software/easybuild/modules/all

Check the available software

cd /mnt/it\_software/easybuild/modules/all/

ml avail

https://www.youtube.com/watch?v=l3Vo9ATA3qQ

Easy Build Tutorial

http://tutorial.easybuild.io/files/EasyBuild-tutorial-ISC22-20220529.pdf

eb

MRI examples

1.Assigning 1 GPU, runs nvidia-smi, lists software available

mnt/it\_research/examples/mri/mri\_gpu1\_test

2. assigns 4 GPUs, runs nvidia-smi, lists software available

/mnt/it\_research/examples/mri/mri\_gpu4\_test

3. using MRI largemem nodes

/mnt/it\_research/examples/mri/mri\_largemem\_test

4. using MRI jobs partition

/mnt/it\_research/examples/mri/mri\_jobs\_test

**Submitting jobs**

sbatch /mnt/it\_research/examples/mri/mri\_gpu1\_test --partition=mrigpu --gres=gpu:1 nvidia-sm

File transfer

Scp

Winscp

Filezilla

**Steps to copy files from local to Blizzard(Tao Liu)**

Linux

rsync -av --ignore-existing /data/taoliu/taoliufile/crop\_rotation\_forcasting taoliu@login-mri.research.mtu.edu:/home/taoliu/data/

Windows

Install Winscp

**Steps to create anaconda environment (Tao Liu)**

Step 1 mount all the available software to the system

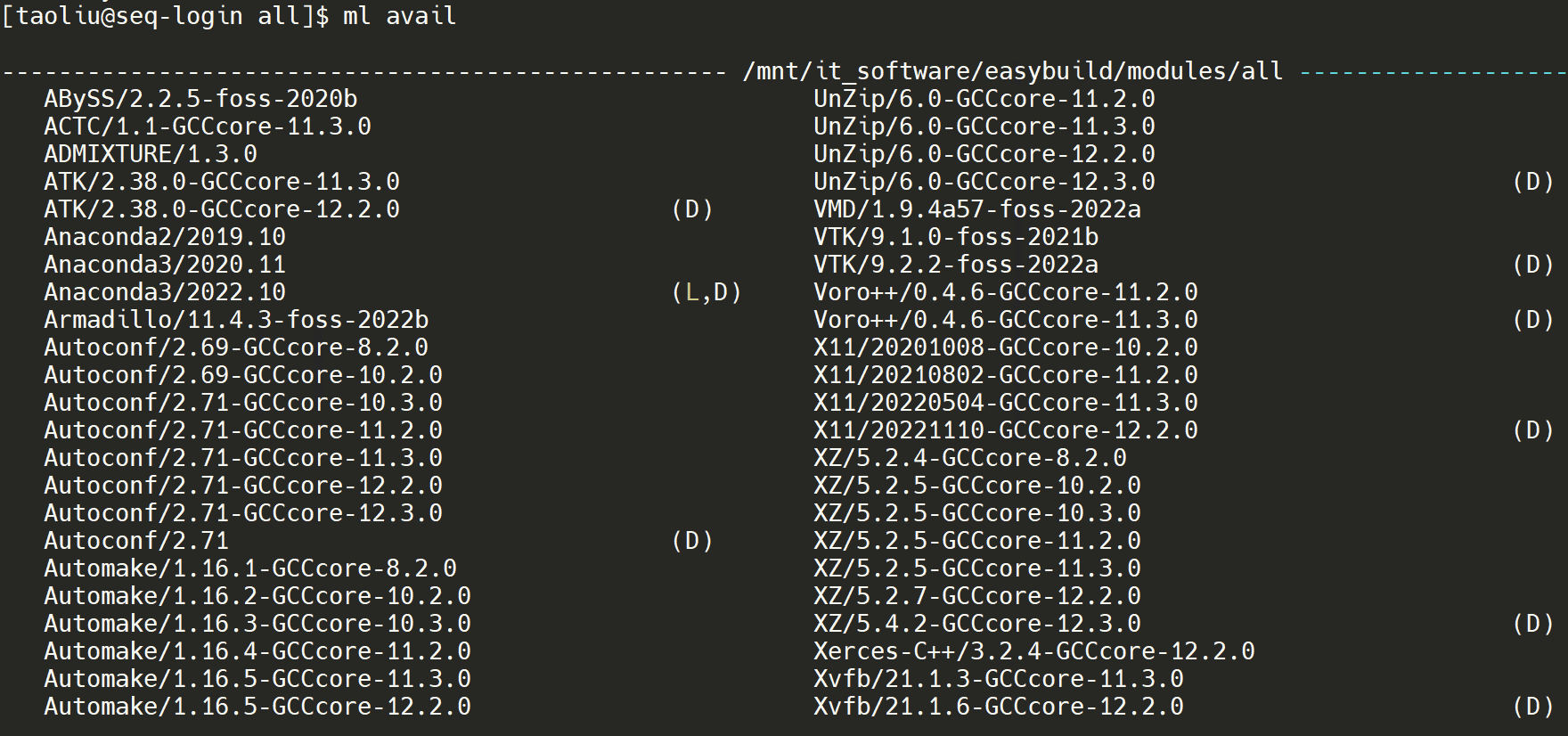
module use /mnt/it\_software/easybuild/modules/all

Step 2 check the full name of Anaconda and that shows three versions of Anaconda are available

ml avail

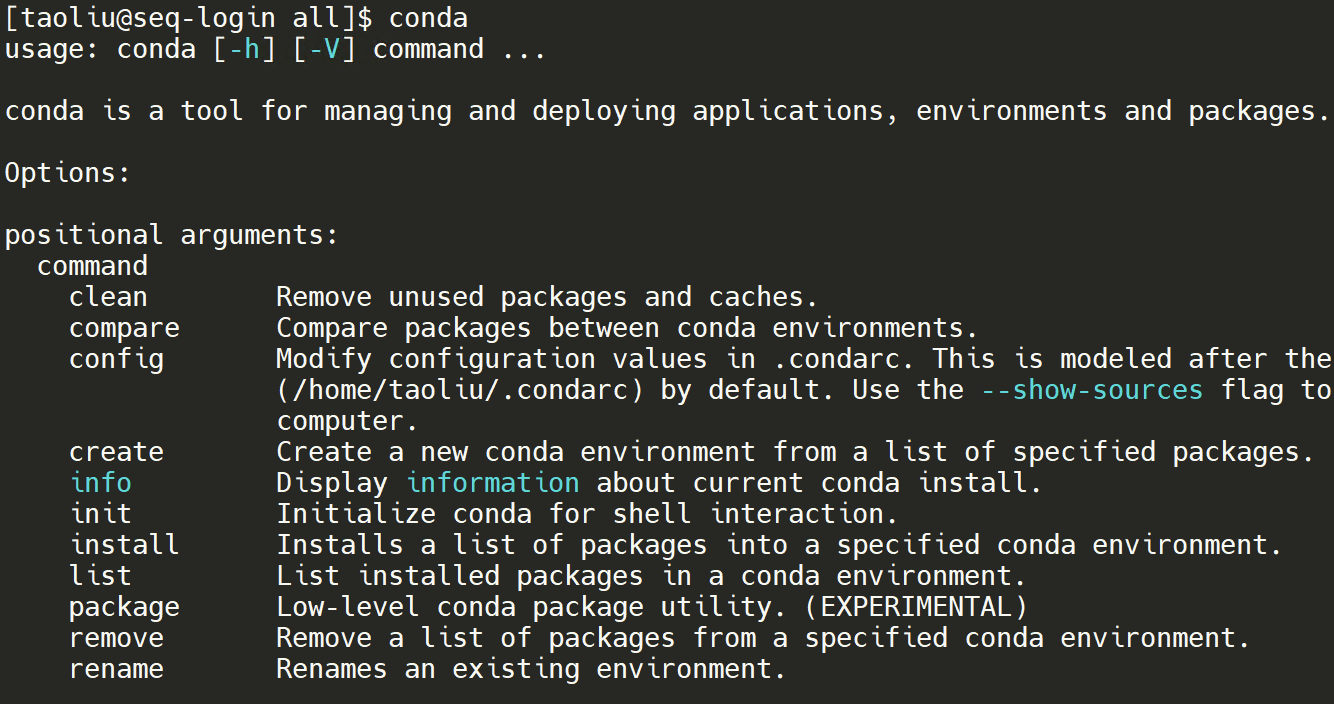
Or make a query of the specific module

ml avail anaconda



Step 3, load one of them, e.g., Anaconda3/2022.10

module load Anaconda3/2022.10

Step 4, Now you can use conda to create environment with conda command

**Steps to activate anaconda environment and install libraries (Shashank Pathrudkar)**

Step 1: Please follow the steps given by Tao Liu above to load anaconda

Step 2: Create conda environment using (Environment name used here: spDL)

$conda create -n spDL

Step 3: Activate the conda environment using

$source activate spDL

(You might have to close the session and login again after this step. Once logged in again activate spDL)

Step 4: Install the required python packages using (I am installing pytorch)

$conda install pytorch

Step 5: Check the installed libraries using

$conda list

**Steps to run a sample Deep Learning job using pytorch (Shashank Pathrudkar, Revanth Mattey - Dr. Ghosh’s Research group)**

Step 1: Create a .sh file with the following contents. Name the file as ‘tryDLjob.sh’.

#!/bin/bash

#SBATCH --partition=mrigpu

#SBATCH --job-name=basic\_slurm\_job

#SBATCH --nodes=1

#SBATCH --ntasks-per-node=1

#SBATCH --time=1:00:00

source activate spDL

python3 tryPytorch.py

(New file can be created using touch tryDLjob.sh)

Step 2: Create a .py file with following contents. Name the file as tryPytorch.py

import torch

import torch.nn as nn

import torch.optim as optim

import numpy as np

# Generate sample data

np.random.seed(0)

X = np.random.rand(1000, 1) # 1000 samples, 1 feature

y = 3 \* X + X \* np.sin(X) + 2 + np.random.randn(1000, 1) \* 0.5 # y = 3X + Xsin(X) + 2 + noise

# Convert numpy arrays to PyTorch tensors

X\_tensor = torch.tensor(X, dtype=torch.float32)

y\_tensor = torch.tensor(y, dtype=torch.float32)

# Define a simple neural network model

class LinearRegression(nn.Module):

def \_\_init\_\_(self, input\_size, output\_size):

super(LinearRegression, self).\_\_init\_\_()

self.linear = nn.Linear(input\_size, output\_size)

def forward(self, x):

return self.linear(x)

# Initialize the model, loss function, and optimizer

model = LinearRegression(input\_size=1, output\_size=1)

criterion = nn.MSELoss()

optimizer = optim.SGD(model.parameters(), lr=0.01)

# Training loop

num\_epochs = 1000

for epoch in range(num\_epochs):

# Forward pass

outputs = model(X\_tensor)

loss = criterion(outputs, y\_tensor)

# Backward pass and optimization

optimizer.zero\_grad()

loss.backward()

optimizer.step()

if (epoch+1) % 100 == 0:

print(f'Epoch [{epoch+1}/{num\_epochs}], Loss: {loss.item():.4f}')

# After training, let's make some predictions

with torch.no\_grad():

predicted = model(X\_tensor).detach().numpy()

# Print the first few predictions

print("Predictions:")

for i in range(5):

print(f"Predicted: {predicted[i][0]:.2f}, Actual: {y[i][0]:.2f}")

Step 3: Submit the job using the following command

$sbatch tryDLjob.sh

Step 4: Check the output in the generated slurm file

$vi slurm{jobid}.out

**goo**

**Steps to install pytorch-GPU (Revanth Mattey)**

**# For Installing the cuda compatible version of pytorch**

**# Just doing conda install pytorch::pytorch will install the latest version of pytorch (‘2.2.0’) which is not compatible with current CUDA version (‘12.2’)**

**# Also make sure to have python version >= ‘3.10’**

conda install pytorch torchvision torchaudio pytorch-cuda=11.8 -c pytorch -c nvidia

**Sample Job submission file for submitting ML jobs**

#!/bin/bash

#SBATCH --partition=mrigpu

#SBATCH --job-name=test

#SBATCH --nodes=1

#SBATCH --gres=gpu:1

#SBATCH --ntasks-per-node=1

#SBATCH --time=1:00:00

**# Move to your appropriate working directory**

cd /home/vmattey/research/

nvidia-smi

**# Load the required modules**

module use /mnt/it\_software/easybuild/modules/all

module load Anaconda3/2022.10

**# Activating the Conda Environment**

source activate pytorch

**# Running the job**

**# If you would like to suppress the output remove the *-u* flag**

python3 -u $FILENAME$

**MODULE command**

module use /mnt/it\_software/easybuild/modules/all

**GDAL module problem(Tao Liu)**

module load GDAL/3.5.0-foss-2022a command will forcefully and automatically switch the Python from the conda environment to /mnt/it\_software/easybuild/software/Python/3.10.4-GCCcore-11.3.0/bin/python, which is undesired, because it will cause various weird errors when you try to use your conda environment. To avoid those errors, you need to avoid using module load GDAL/3.5.0-foss-2022a and install the GDAL in your conda environment instead with conda install gdal. In case gdal is difficult to install, just install geopandas with conda install geopandas, and gdal will be installed automatically as a dependent package of geopandas.

**Monitor the GPU usage corresponding to a job (Tao Liu)**

srun --ntasks-per-node=1 --jobid 123456 nvidia-smi

**Check the status of a job (Tao Liu)**

sacct -j 1033

**Cancel a job (Tao Liu)**

scancel 1082

**Check all the partition status (Tao Liu)**

sinfo

**Use multiple GPUs to train a deep neural network (Tao Liu)**

My test of parallel training using multiple GPUs only works for single node with 4 GPUs at most, as each machine has 4 A100 GPU. The sbatch file and python file are shown below. To make it work, make sure --nproc\_per\_node=4 after torchrun is the same as number of GPUs requested with #SBATCH --gres=gpu:4.

test\_parallel.sh

#!/bin/bash

#SBATCH --partition=mrigpu

#SBATCH --job-name=test

#SBATCH --nodes=1

#SBATCH --gres=gpu:4

#SBATCH --ntasks-per-node=6

#SBATCH --time=50:00:00

# Load the required modules

module use /mnt/it\_software/easybuild/modules/all

source /home/taoliu/anaconda3/etc/profile.d/conda.sh

conda env list

conda activate /home/taoliu/.conda/envs/env4

which python

python --version

echo "Using python from $(which python)"

nvidia-smi

torchrun --node\_rank=0 --nproc\_per\_node=4 --nnodes=1 /home/taoliu/data/field\_boundaries/data/code\_smp/elastic\_ddp.py

elastic\_ddp.py

import torch

import torch.distributed as dist

import torch.nn as nn

import torch.optim as optim

from torch.nn.parallel import DistributedDataParallel as DDP

#torchrun --node\_rank=0 --nproc\_per\_node=2 --nnodes=1 /data/taoliu/taoliufile/gedi\_biomass/code/parallel\_toy.py

class ToyModel(nn.Module):

def \_\_init\_\_(self):

super(ToyModel, self).\_\_init\_\_()

self.net1 = nn.Linear(10, 10)

self.relu = nn.ReLU()

self.net2 = nn.Linear(10, 5)

def forward(self, x):

return self.net2(self.relu(self.net1(x)))

def demo\_basic():

dist.init\_process\_group("nccl")

rank = dist.get\_rank()

device\_count = torch.cuda.device\_count()

device\_id = rank % device\_count

torch.cuda.set\_device(device\_id) # Use device\_id here

print(f"Start running basic DDP example on rank {rank}.")

print(f"use GPU: {torch.cuda.current\_device()}")

print(f"world\_size = {dist.get\_world\_size()}, rank = {dist.get\_rank()}")

print(f"device count={device\_count}")

print(f"device\_id = {device\_id}")

model = ToyModel().cuda(device\_id) # Move model to the correct device

ddp\_model = DDP(model, device\_ids=[device\_id])

loss\_fn = nn.MSELoss()

optimizer = optim.SGD(ddp\_model.parameters(), lr=0.001)

optimizer.zero\_grad()

outputs = ddp\_model(torch.randn(20, 10))

labels = torch.randn(20, 5).to(device\_id)

loss\_fn(outputs, labels).backward()

optimizer.step()

dist.destroy\_process\_group()

if \_\_name\_\_ == "\_\_main\_\_":

demo\_basic()



Submit your job

sbatch /home/taoliu/data/field\_boundaries/data/test\_parallel.sh

**Creating a virtual environment and running a DL job**

module use/mnt/it\_softwares/easybuild/modules/all

ml avail

ml load anaconda3

* Conda becomes available

Create the environment and name it dl-gpu

And then in the shell script e.g. a.sh have the following lines

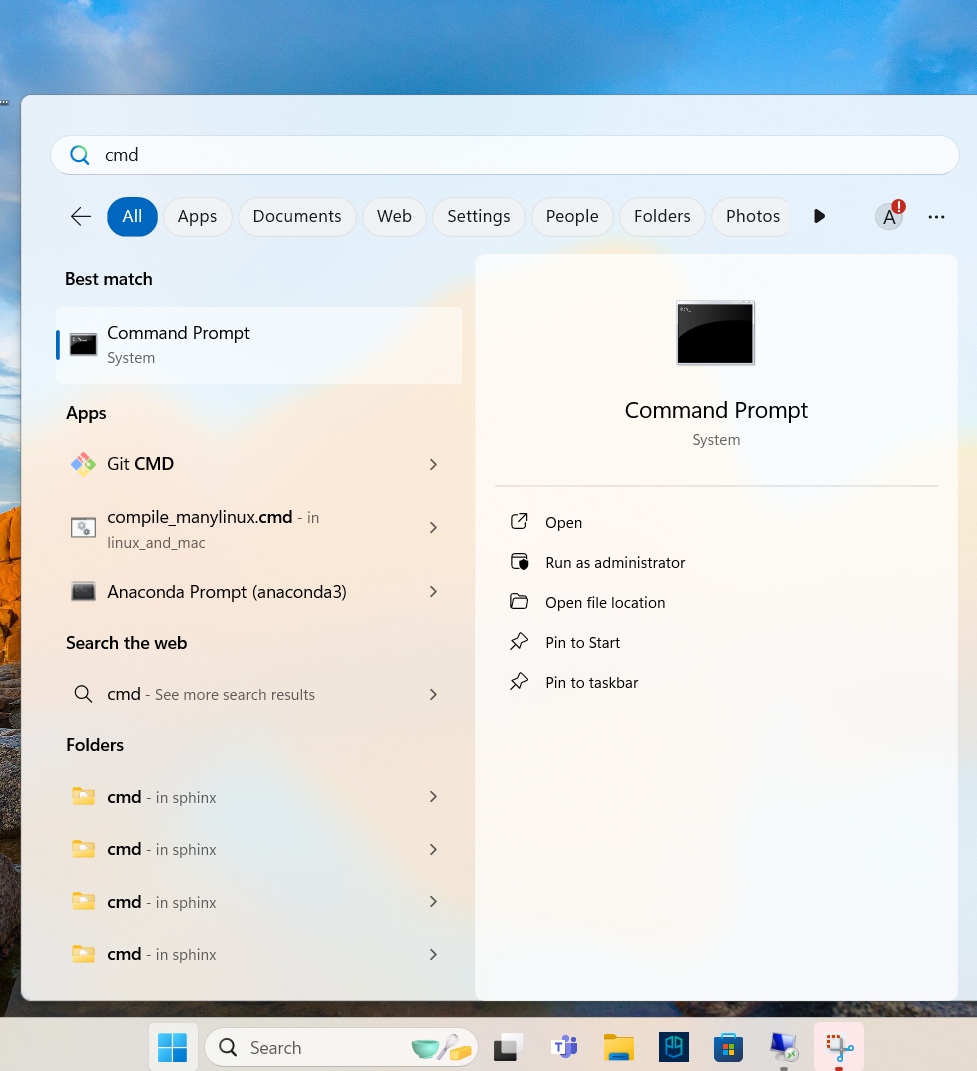
ml load anaconda3

ml load cuda

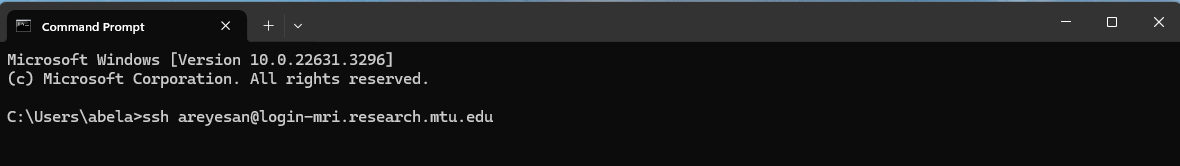
source active dl-gpu # your environment that you created

**Setting up the environment for CV tasks using TensorFlow (AR)**

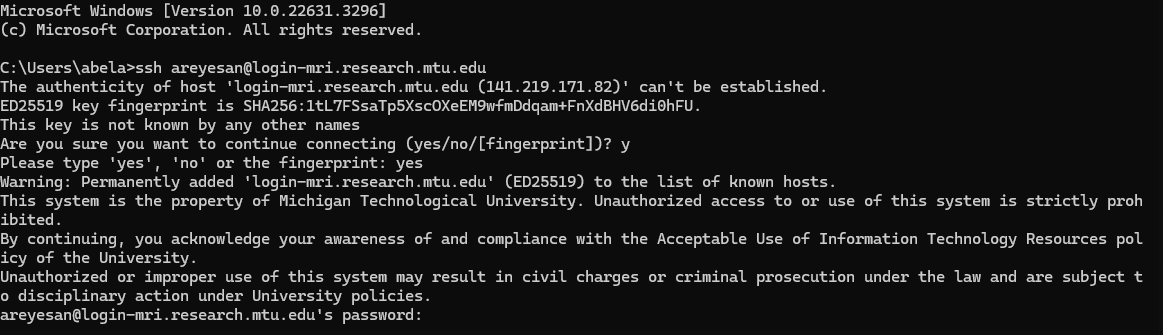
**Login to cluster (SSH)**: Open command prompt

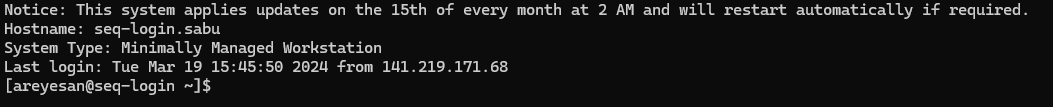


1. ssh user\_name@login-mri.research.mtu.edu

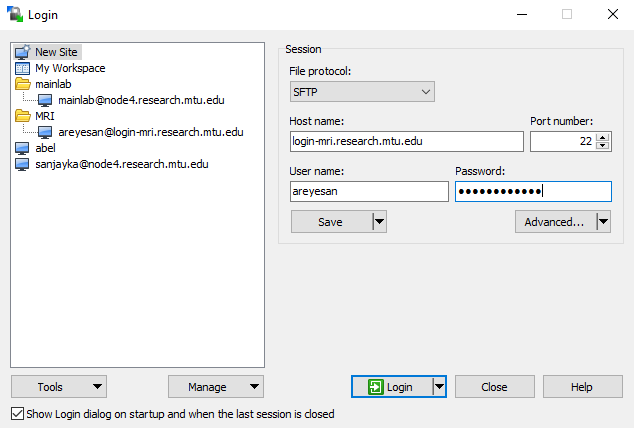


1. Enter your mtu password





Transfer files (WinSCP)



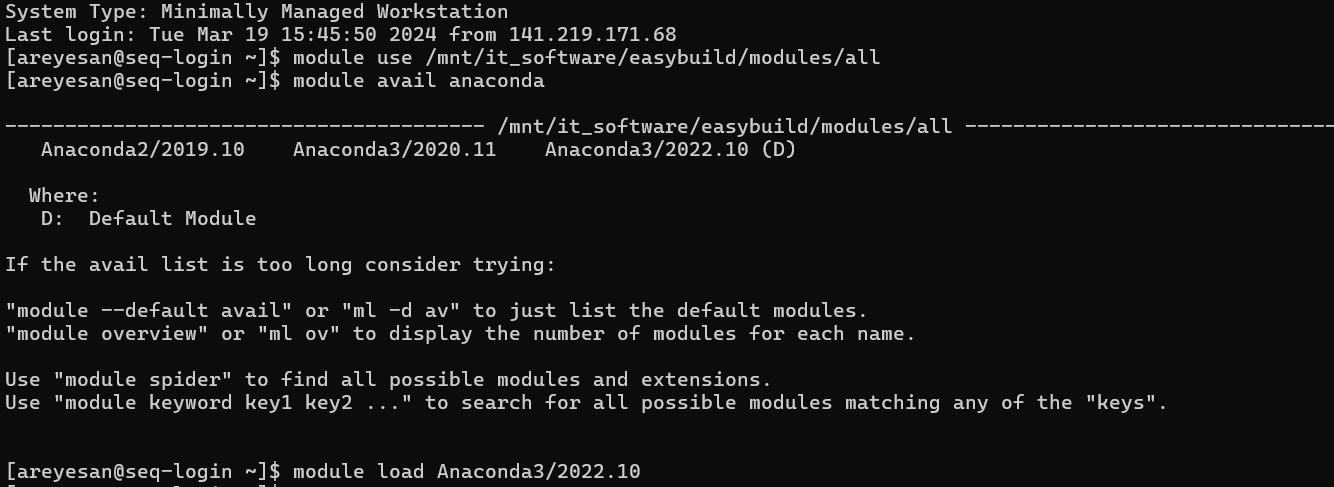
**Creating a new conda environment**

Mount all the available software

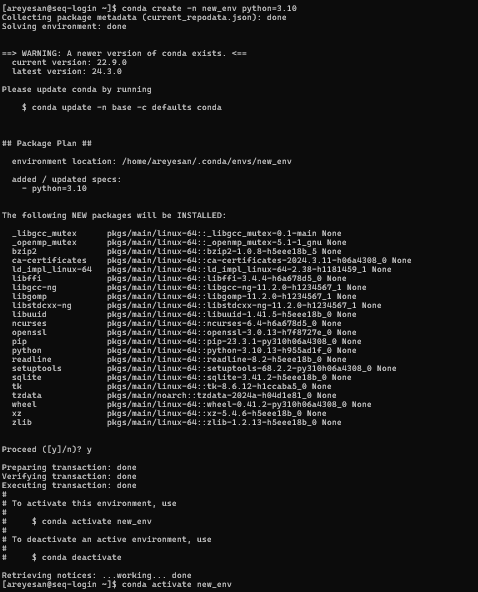
1. module use /mnt/it\_software/easybuild/modules/all

Load one of the anaconda modules

1. module avail anaconda
2. module load Anaconda3/2022.10



Create and activate a new conda environment (Recommended to create an environment with a Python version >=3.10)



1. conda create -n <env\_name> python=3.10
2. conda activate <env\_name>

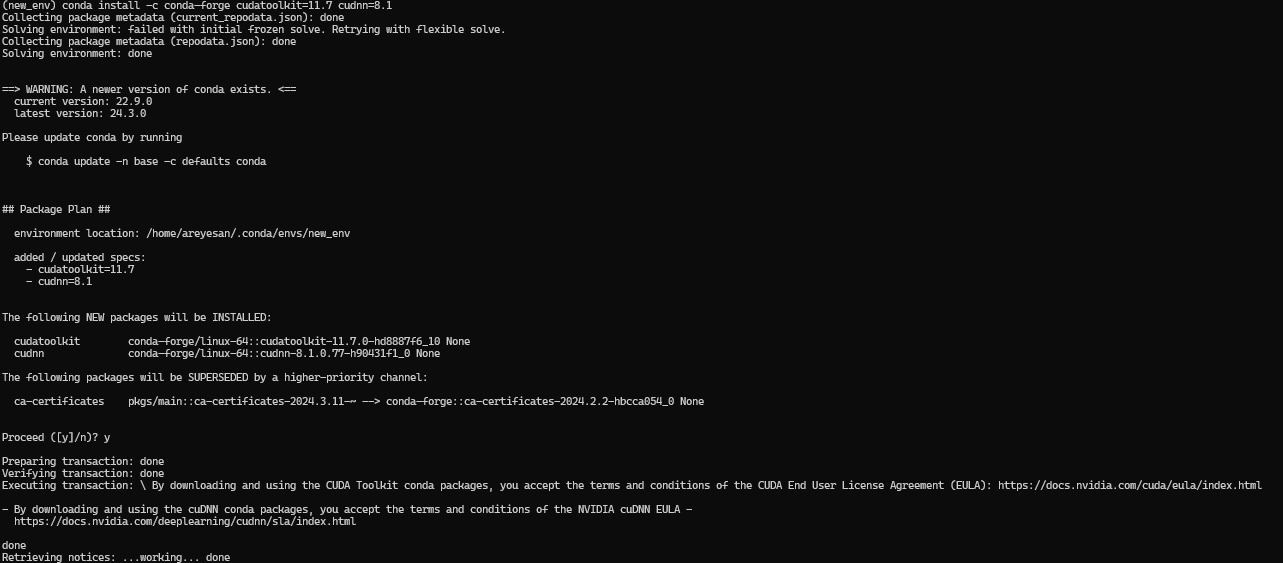
Or

7. source activate <env\_name>



To install cuda and cudnn to use the GPU (This version matches with the cuda module that you could load when a job is sent)

1. conda install -c conda-forge cudatoolkit=11.7 cudnn=8.1

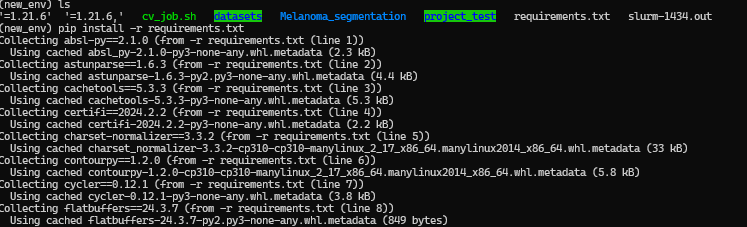


Installing TensorFlow

1. pip install tensorflow==2.11

Further dependencies: you could install further modules individually or by using a requirements.txt file (make sure to have the requirement.txt file in your current pwd)

1. pip install -r requirements.txt



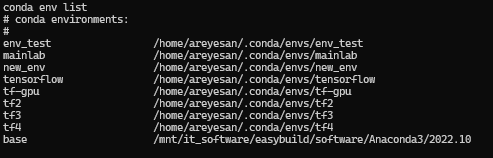
Deactivating the conda environment

1. conda deactivate

**Loading an existing conda environment:**

After step 5, list all your prior conda environments created (optional)

1. conda env list



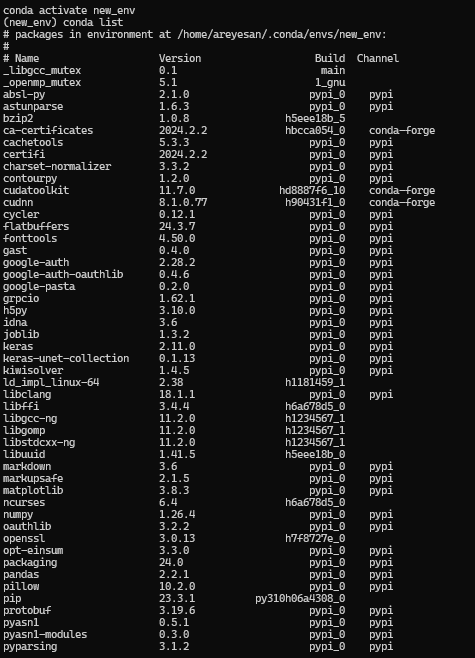
Activate the desired conda environment

1. conda activate <desired\_conda\_env>

The next steps are optional:

To list all the installed libraries in the current environment

1. conda list



To create a requirement.txt file with all the dependencies from your environment

1. pip freeze > requirements.txt

**Sending a batch job (After step 5)**

Creating a bash script to send a job, using one node and one gpu

1. you can create the following file and save it as <job\_name>.sh

#!/bin/bash

#SBATCH --partition=mrigpu

#SBATCH --job-name=<desired\_job\_name>

#SBATCH --nodes=1

#SBATCH --gres=gpu:1

#SBATCH --ntasks-per-node=1

#SBATCH --time=1:00:00

module use /mnt/it\_software/easybuild/module/all ##mounting all the software

module avail anaconda ## checking the anaconda version available

module load Anaconda3/2022.10 ###loading the desired anaconda module

module avail cuda

module load CUDA/11.7.0

conda env list

source activate /home/<user\_name>/.conda/envs/<env\_name>

which python

python --version

echo “using python from $(which python)”

nvidia-smi ##If you want to verify the GPU used

cd <project\_folder> ## moving to your project folder

python3 -u python\_script\_file.py

### ### more commands could be added and will be executed as part of the job



To send the batch job

1. sbatch <job\_name>.sh



A log file will be created in the current directory with the following name: slurm-<job\_id>.out

Read the log file of the job:

1. vi slurm-<job\_id>.out

**Some other useful commands to monitor your jobs**

Status of a job:

sacct -j <job\_id>

Cancel a job

scancel <job\_id>