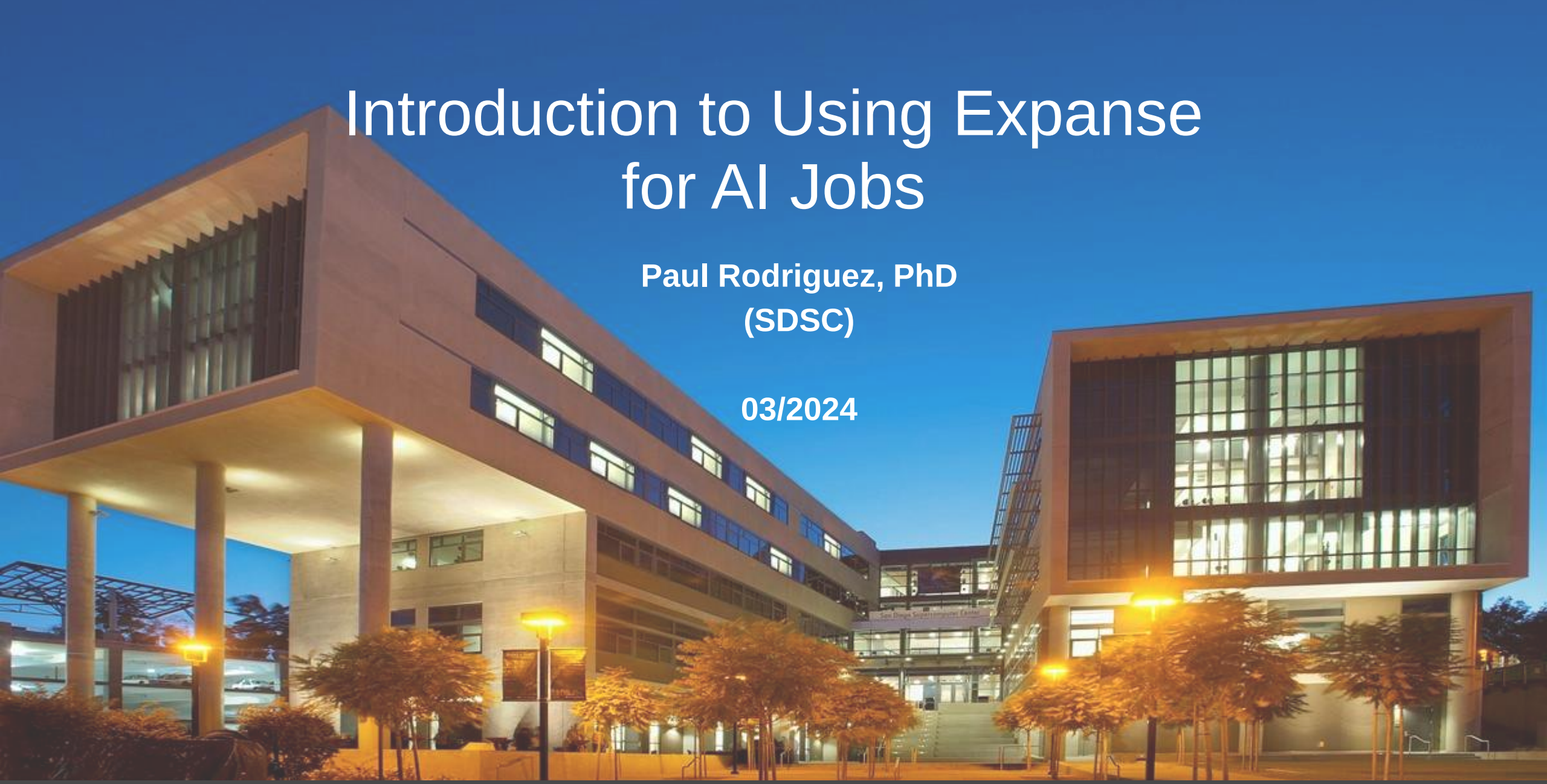
Introduction to Using Expanse for AI Jobs

Paul Rodriguez, PhD
(SDSC)

03/2024



Outline

- Parallelization in Tensorflow
- Expanse Notebooks and Open On Demand
- Starting Jupyter notebook and simple MNIST example
- An example running multinode on Expanse An example running Keras NLP tool in conda environment, w/Juptyer Lab

Things to think about for running a project

- Choosing Hyperparameters – a bit of exploration and exploitation
- Need to figure out efficient Job workflow
- On HPC, CPU work fine for many cases, you will want to use GPUs for ‘large’ models and/or large datasets.
- Model saves and/or checkpoints are available in tensorflow; tensorboard available but needs to be secure (ask for details)

Python Notebook vs Scripts

- **On HPC you may want to run batch jobs on a script not a notebook.**

1 Papermill is one tool

2 Or, you can use "*jupyter nbconvert --to script your-python.ipynb***" in the batch job.**

Also, turnoff plot display, save plots in files, and use a configuration file to pass in parameters

Parallel DL models with multiple nodes/devices

- **The main approach to parallelize training: Data Parallel:**

- **Main tools: Keras/Tensorflow ‘strategy’ or use Horovod MPI wrappers**

Keras/Tensorflow strategy single GPU node

Set up a 'mirror' strategy

```
mirrored_strategy = tf.distribute.MirroredStrategy(["GPU:0", "GPU:1",  "GPU:2",  "GPU:3"])
```

You also need the strategy scope around the model definition so that it can make copies

```
if (n_gpus>0):
    with mirrored_strategy.scope():
      multi_dev_model=build_model()
```

Then train as normal (use batch size multiple of 32)

Keras/Tensorflow strategy multiple GPU node

Keras also has a ‘multiworker’ strategy but it requires setting up config files with IP addresses

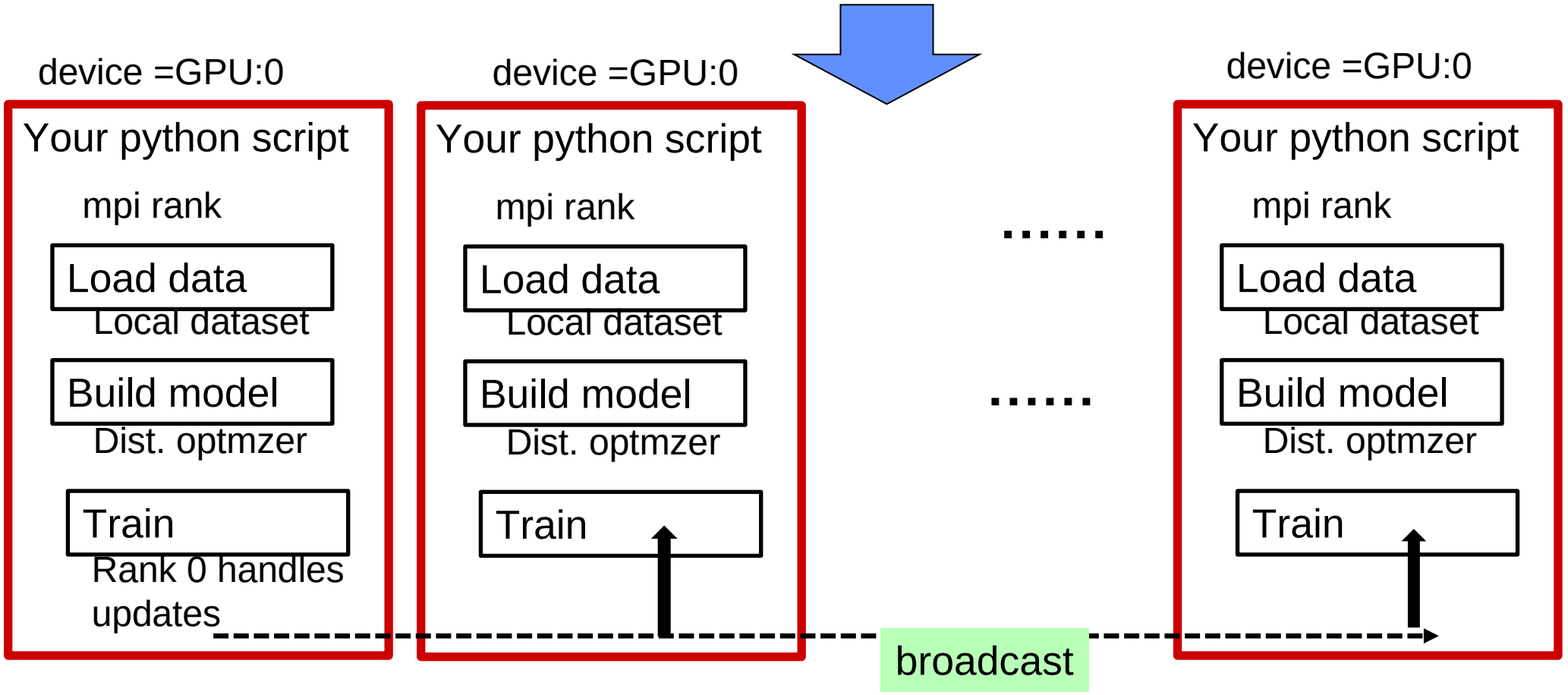
On HPC systems resources are shared so IP addresses are dynamic

Better to use Horovod with MPI and slurm batch job

For each batch: Horovod will aggregate & share weights updates

In slurm batch script:

mpirun –n **number of tasks** singularity → python



Bigger batch size helps, but it uses more memory

Code snippets – Horovod functions

Not many lines of code, but becareful with sharding, batch size,
See https://horovod.readthedocs.io/en/latest/keras.html

Initialize back end communication, get mpi rank

```
import horovod.tensorflow.keras as hvd
hvd.init()
print('INFO, global rank:',hvd.rank(), ' locrank ',hvd.local_rank())
```

'shard' or split data

```
Note: be careful to get batch size in sharding and training
aligned
```

Set up a distributed optimizer

```
 optimizer2use  =
hvd.DistributedOptimizer(Adam(learning_rate=0.001*hvd.size())
```

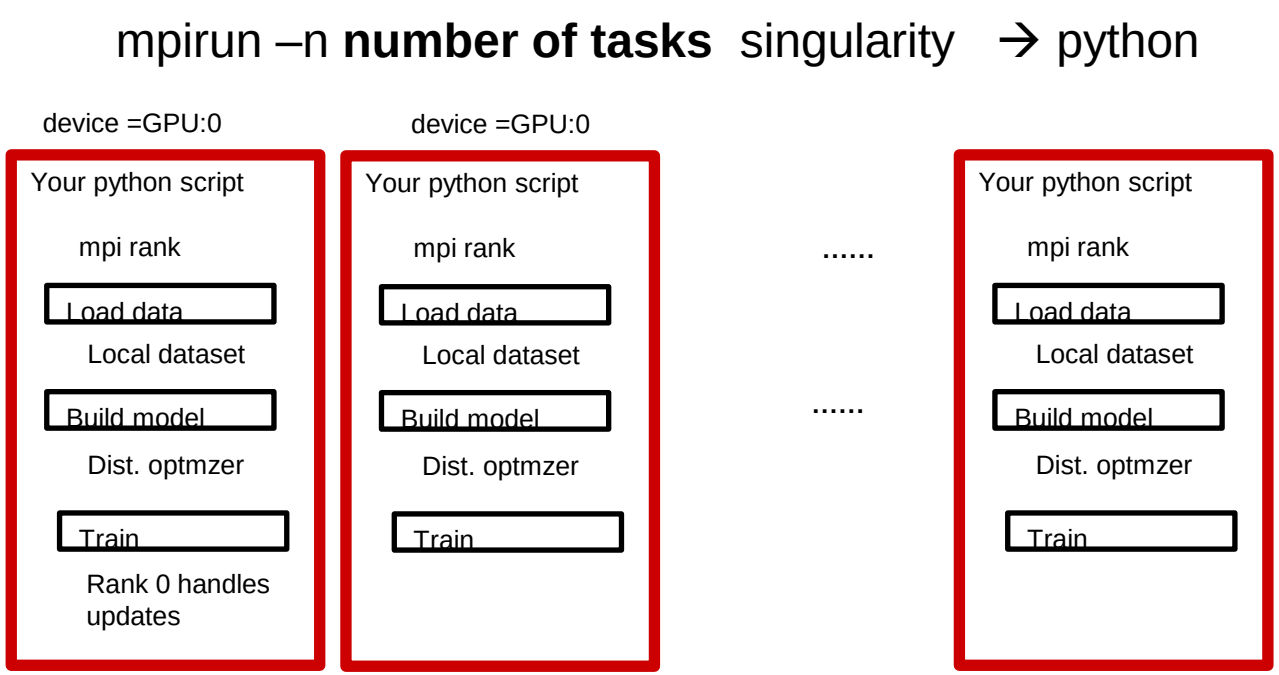
After each batch aggregate weight updates and broadcast new values

```
model.compile(optimizer    =optimizer2use,
 …
experimental_run_tf_function=False)
```

Exercise, multinode MNIST programming and execution

- **Goal: Get familiar with Keras and Horovod coding for multinode execution**
- **Goal: Get familiar with slurm batch script multinode parameters**
- **Let's login and start a notebook**

(see next pages for quick overview)



Accessing Expanse

- **Command line interface to run Slurm jobs**
- **Expanse user portal has web interface to start Jupyter lab/notebooks**
- **Slurm jobs can also start a notebook**

CIP Fellow | CIP Fellow | Your Repo | WKSHOPS | run-hvd-m | CIML23_M | Horovod

pelvis-daisy-faceplate.expanse-user-content.sdsc.edu/edit/WKSHOPS/CIML23/MNODE_wHVD/run-hvd-main-c...

jupyter run-hvd-main-cpu2.sb 15 minutes ago

Logout

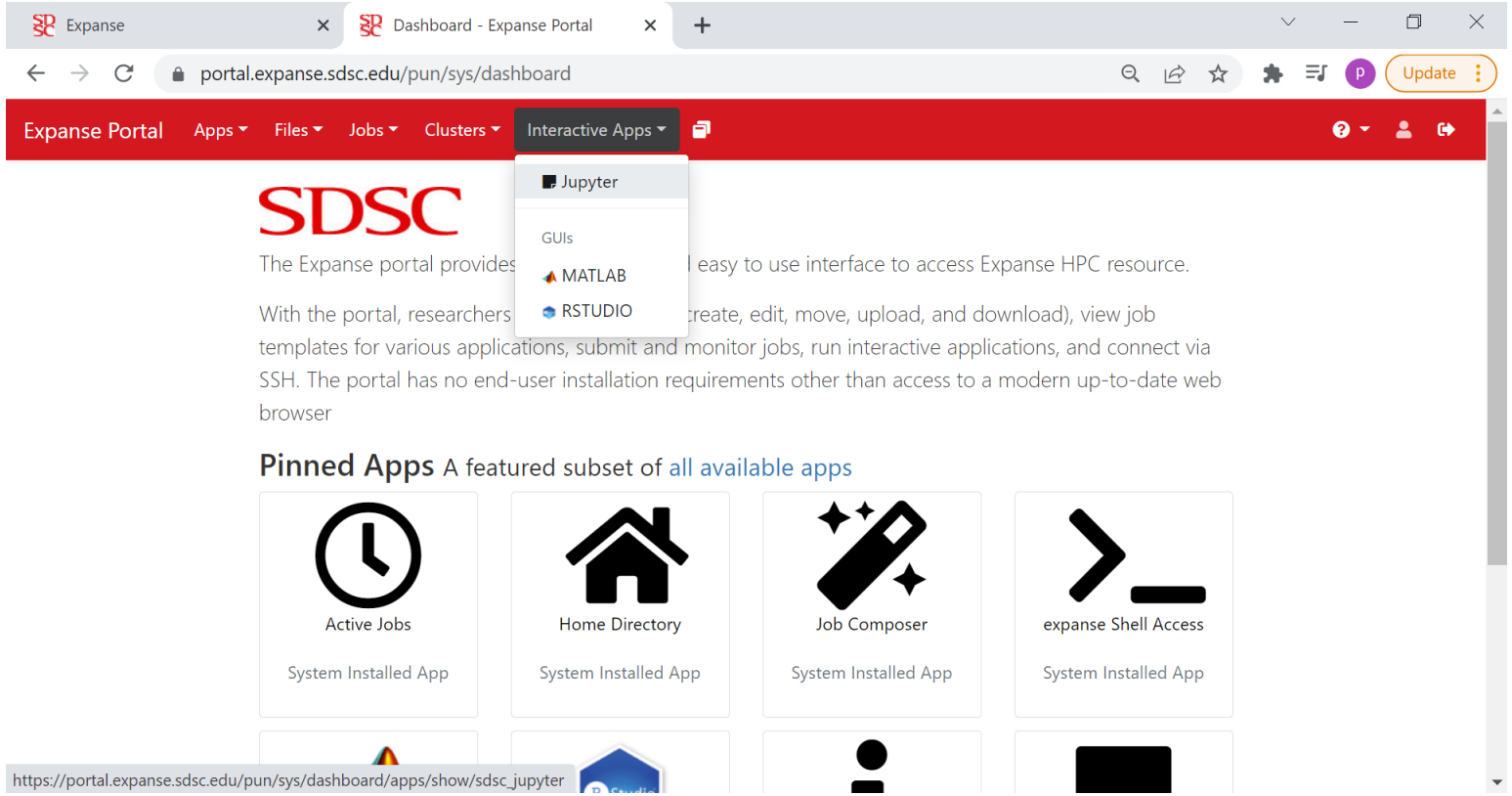
File Edit View Language Plain Text

```
#SBATCH --job-name=tfhvd-cpu
#SBATCH --account=use300
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=16  #<<<<<------ change this to 16 and observe changes in training time
#SBATCH --cpus-per-task=1
#SBATCH --mem=243G
#SBATCH --time=00:15:00
#SBATCH --output=slurm.cpu2.%x.o%j.out

#----------- set up modules ----------------------------
module reset
module load slurm
module load gcc/10.2.0          #compiler, unix
module load openmpi/4.1.3       #open mpi
module load singularitypro/3.9  #container
module list

#----------- set up some environmental settings --------
export OMPI_MCA_btl='self,vader'
export UCX_TLS='shm,rc,ud,dc'
export UCX_NET_DEVICES='mlx5_0:1'
```

```
stdout_*
time: 2.48225  secs
time: 2.31222  secs
[p4rodrig@login01 MNODE_wHVD]$
[p4rodrig@login01 MNODE_wHVD]$
[p4rodrig@login01 MNODE_wHVD]$
[p4rodrig@login01 MNODE_wHVD]$
[p4rodrig@login01 MNODE_wHVD]$
[p4rodrig@login01 MNODE_wHVD]$
[p4rodrig@login01 MNODE_wHVD]$
```

The expanse user portal (open on demand)

https://portal.expanse.sdsc.edu



Login in with ACCESS credentials (NSF CI coordination at https://access-ci.org/)

For local workshops use the assigned “train## ‘ account and go to:

https://portal.expanse.sdsc.edu/training

Galyleo Utility for Jupyter Notebooks

- A tool that launches a Jupyter Lab/Notebook server on a compute node
- Establishes a secured HTTPS connection between that compute node and your web browser (reverse proxy)

For details:

https:/github.com/mkandes/galyleo

https://education.sdsc.edu/training/interactive202112_running_jupyter_notebooks_on_expanse/index.html

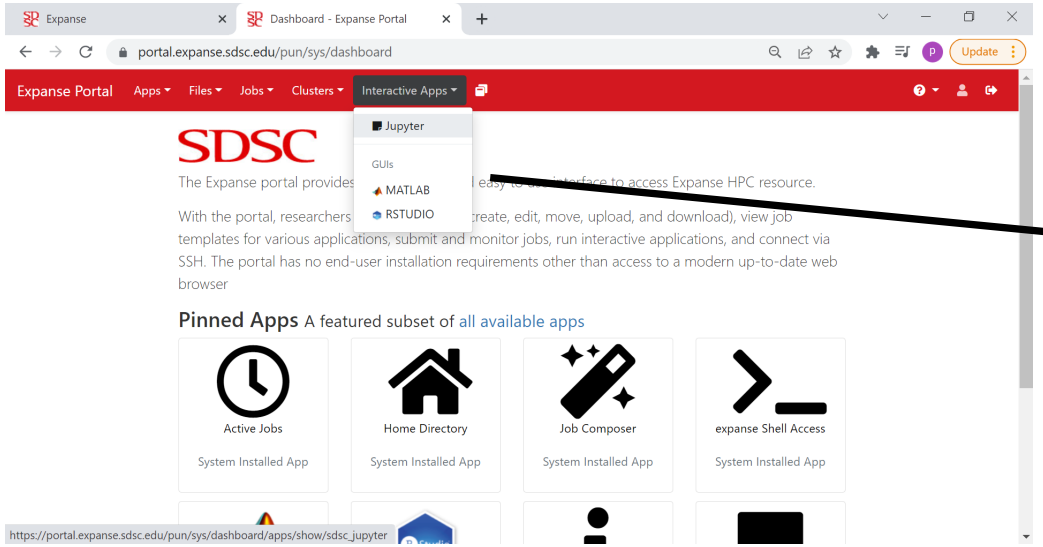
Sample Galyleo scripts

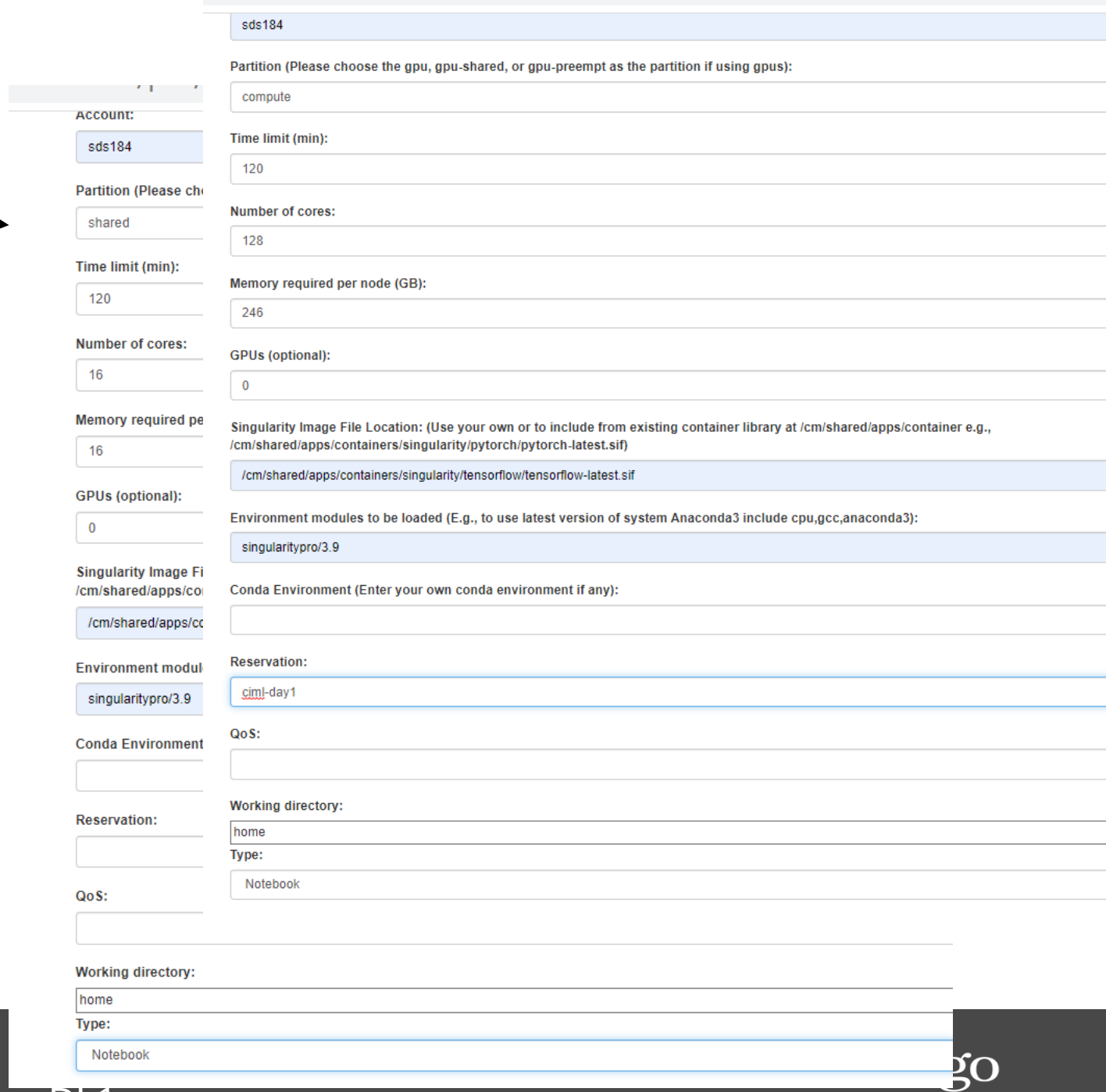
- **A script that loads a conda environment to compute node (cpu)**

```
/cm/shared/apps/sdsc/galyleo/galyleo.sh launch --account gue998
        --partition compute  --nodes 1 --memory 242 --cpus 128 --time-limit 01:00:00
         --conda-env keras-nlp24 --conda-yml keras-nlp24-cpu.yaml --mamba
```

- **A script that loads a singularity image to gpu node**

```
/cm/shared/apps/sdsc/galyleo/galyleo.sh launch -A gue998
     -p gpu -n 10 -M 93 -G 4  -t 00:30:00
     -e singularitypro/3.9 --bind /expanse,/scratch -s
                                /cm/shared/apps/containers/singularity/pytorch/pytorch-latest.sif
```





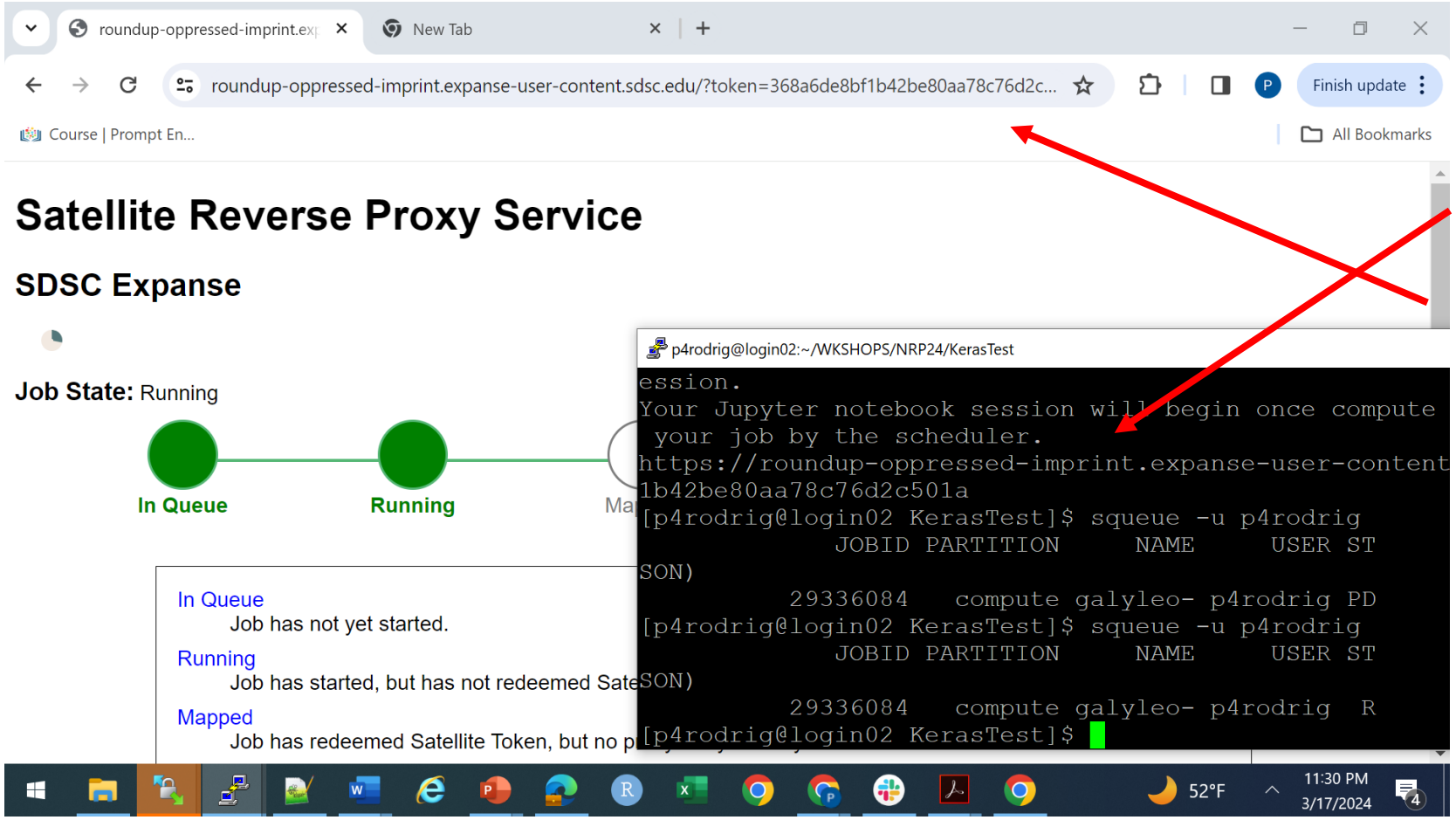
Expanse user portal:
launch a jupyter notebook session

Or

login to Expanse and run:
$ bash jupyter- … -cpures.sh

Simple MNIST exercise

login to Expanse and run from command line:
$ bash jupyter-tflow-cpures.sh

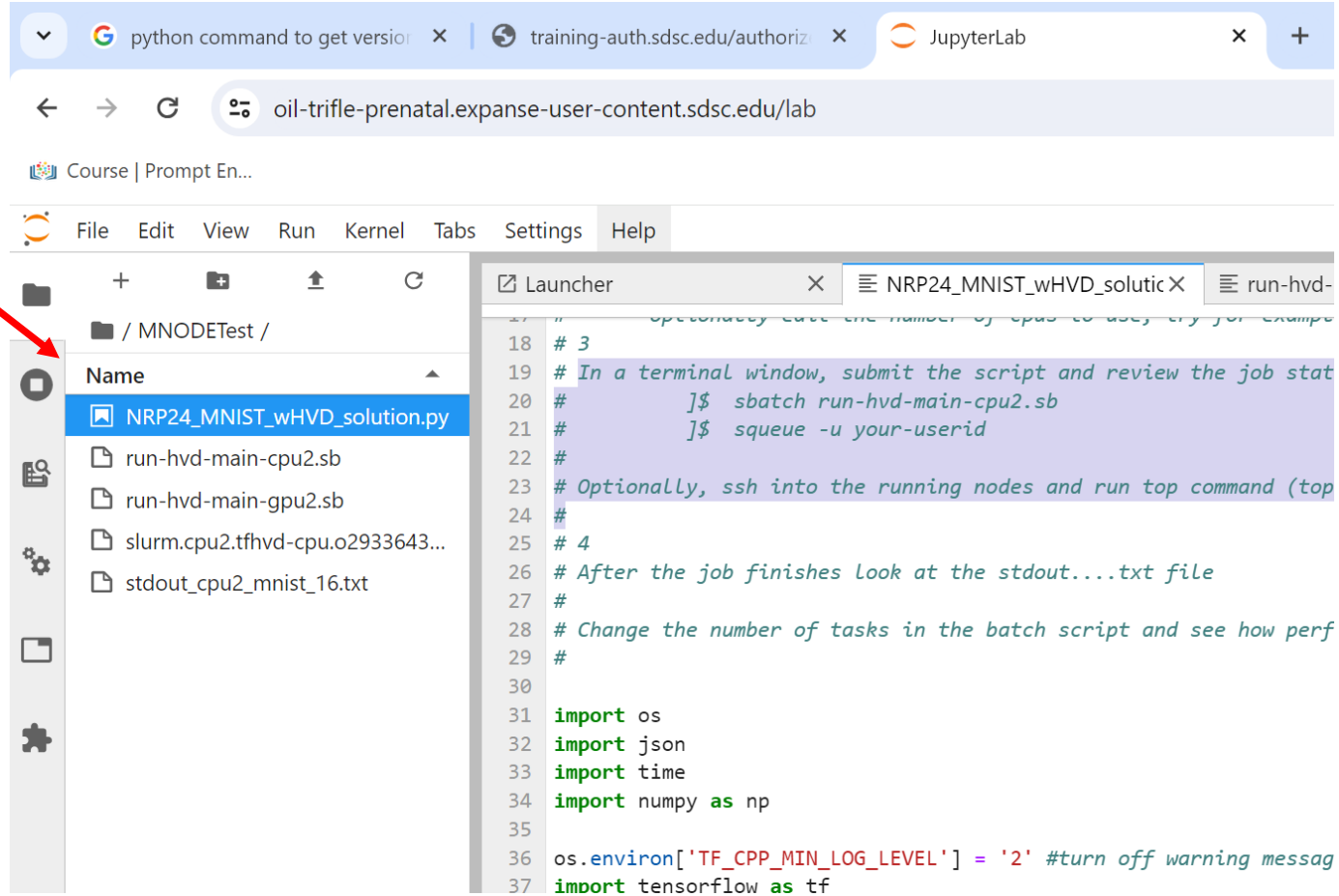


Find url and paste into browser

then wait for Jupyter to load

In jupyter notebook session open the MNIST_Intro notebook

Review notebook or just select
Edit-> clear all cels
Run-> Run all cells



Simple MNIST multinode exercise

```
#SBATCH --job-name=tfhvd-cpu
#SBATCH --account=use300
#SBATCH --partition=compute
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=16  #<<<<<------ change this to 16 and observe changes in training time
#SBATCH --cpus-per-task=1
#SBATCH --mem=243G
#SBATCH --time=00:15:00
#SBATCH --output=slurm.cpu2.%x.o%j.out
```

Launcher | NRP24_MNIST_wHVD_solutic | run-hvd-main-cpu2.sb

```
#----------- set up modules ----------------------------
module reset
module load slurm
module load gcc/10.2.0          #compiler, unix
module load openmpi/4.1.3       #open mpi
module load singularitypro/3.9  #container
module list

#----------- set up some environmental settings --------
export OMPI_MCA_btl='self,vader'
export UCX_TLS='shm,rc,ud,dc'
export UCX_NET_DEVICES='mlx5_0:1'
export UCX_MAX_RNDV_RAILS=1

#cd into the working directory, slurm puts you there

#------ execute the mpirun command to launch container instances ----------
mpirun -n ${SLURM_NTASKS} singularity exec --bind /expanse,/scratch
/cm/shared/apps/containers/singularity/tensorflow/tensorflow-latest.sif python3
./NRP24_MNIST_wHVD_solution.py > stdout_cpu2_mnist_${SLURM_NTASKS}.txt
```

In MNODE_Test directory, see the slurm script: **run-hvd-main-cpu2res.sb**

Many lines for slurm commands and environment set up

Note: mpirun launches instances of ‘singularity exec …’

In MNODE_Test directory

Simple MNIST multinode exercise

Python script: *MNIST_wHVD_exercise* notebook

Slurm script: *run-hvd-main-cpu2.sb*

```
etrain94@login02:~/MNODETest
[etrain94@login02 ~/MNODETest]$ ls
NRP24_MNIST_wHVD_solution.py
run-hvd-main-cpu2.sb
run-hvd-main-gpu2.sb
slurm.cpu2.tfhvd-cpu.o29336438.out
stdout_cpu2_mnist_16.txt
[etrain94@login02 ~/MNODETest]$
[etrain94@login02 ~/MNODETest]$
[etrain94@login02 ~/MNODETest]$
```

In a terminal window, submit the script and review the job status

```
]$  sbatch run-hvd-main-cpu2.sb
]$  squeue -u your-userid
```

Optionally, ssh into the running nodes and run top command (top -u userid)
Look for ‘done’ in the stdout file

```
[p4rodrig@login01 MNODE_wHVD]$
[p4rodrig@login01 MNODE_wHVD]$ grep 'done, rk: 15' st
stdout_cpu2_mnist_32.txt:INFO,done, rk: 15  train tim
stdout_mainhvd_cpu2.txt:INFO,done, rk: 15  train time
[p4rodrig@login01 MNODE_wHVD]$
[p4rodrig@login01 MNODE_wHVD]$
[p4rodrig@login01 MNODE_wHVD]$
```

Note: on Expanse we have gpu-debug queue (30 min limit, 1 device), and gpu-shared queue (1 device) and gpu queue (4 devices per GPU node)

You can run:

$ squeue –u userid to see nodes running your job

$ ssh exp-XX-YY to login to node

$ top –u userid to see processing on a CPU job

```
p4rodrig@exp-1-35:~
top - 14:40:05 up 315 days,  2:34,  1 user,  load average: 78.65, 76.25, 76.07
Threads: 2499 total,  90 running, 2309 sleeping,   0 stopped, 100 zombie
%Cpu(s): 69.0 us,  0.8 sy,  0.0 ni, 29.8 id,  0.0 wa,  0.4 hi,  0.0 si,  0.0 st
MiB Mem : 257509.7 total, 142734.5 free,  40306.1 used,  74469.1 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used. 213120.2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND                P
2460166 p4rodrig  20   0 2967760 362936 181296 R  99.4   0.1   0:11.58 python3             43
2460163 p4rodrig  20   0 2967760 363028 181380 R  99.2   0.1   0:11.15 python3             46
2460161 p4rodrig  20   0 2967764 363072 181420 R  99.0   0.1   0:11.24 python3             34
2460153 p4rodrig  20   0 2967760 363688 181204 R  98.5   0.1   0:12.33 python3             35
2460167 p4rodrig  20   0 2967760 362904 181260 R  98.3   0.1   0:11.55 python3             47
2460155 p4rodrig  20   0 2967744 362908 181280 R  98.1   0.1   0:11.92 python3             39
2460158 p4rodrig  20   0 2967760 362880 181240 R  97.9   0.1   0:10.39 python3             32
2460164 p4rodrig  20   0 2967764 363004 181360 R  97.9   0.1   0:10.01 python3             45
2460162 p4rodrig  20   0 2541772 363332 181692 R  97.9   0.1   0:11.78 python3             38
2460154 p4rodrig  20   0 2967760 362992 181352 R  97.3   0.1   0:11.32 python3             44
2460156 p4rodrig  20   0 2967756 362972 181332 R  92.8   0.1   0:08.37 python3             42
2460160 p4rodrig  20   0 2902224 363032 181388 R  89.1   0.1   0:11.38 python3             36
2460169 p4rodrig  20   0 2967760 362868 181224 S  82.9   0.1   0:10.25 python3             33
2460157 p4rodrig  20   0 2967760 362912 181268 R  76.6   0.1   0:10.16 python3             40
2460159 p4rodrig  20   0 2967760 362932 181292 R  75.4   0.1   0:08.42 python3             37
2460171 p4rodrig  20   0 2967760 362844 181204 S  63.4   0.1   0:10.22 python3             45
2460428 p4rodrig  20   0   67360   7088   3484 R   1.0   0.0   0:00.26 top                122
2458820 p4rodrig  20   0   89628   9512   7896 S   0.0   0.0   0:00.06 systemd             55
2458836 p4rodrig  20   0  326556  12812      0 S   0.0   0.0   0:00.00 (sd-pam)           122
```

Or run: $ nvidia-smi to see usage on GPU devices

```
p4rodrig@login02:/expanse/lustre/projects/sds164/p4rodrig/TFwHVDtests
[p4rodrig@login02 TFwHVDtests]$ squeue -u p4rodrig
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REA
SON)
          21782434       gpu tfhvd-gp p4rodrig  R       0:01      1 exp-10-57
[p4rodrig@login02 TFwHVDtests]$ ssh exp-10-57
[p4rodrig@exp-10-57 ~]$ nvidia-smi
Wed Apr 19 17:50:57 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 510.39.01    Driver Version: 510.39.01    CUDA Version: 11.6     |
|-------------------------------+----------------------+----------------------+
| 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  Tesla V100-SXM2...  On   | 00000000:18:00.0 Off |                    0 |
| N/A   37C    P0    55W / 300W |   1005MiB / 32768MiB |     11%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000000:3B:00.0 Off |                    0 |
| N/A   37C    P0    55W / 300W |   1005MiB / 32768MiB |      6%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
```

KerasNLP

- **KerasNLP is an extension to Keras**

- **KerasNLP has several pre-trained LLMs (large language models). Each model comes with related modules, for example:**

  - GPT2Backbone the model without task specific output layers
  - GPT2CausalLM the model with output predictions
  - GPT2CausalLMPreprocessor the preprocessor that feeds model.fit

KerasNLP

- **KerasNLP is an extension to Keras**

- **KerasNLP has several pre-trained LLMs (large language models). Each model comes with related modules, for example:**

  - GPT2Backbone the model without task specific output layers
  - GPT2CausalLM the model with output predictions
  - GPT2CausalLMPreprocessor the preprocessor that feeds model.fit

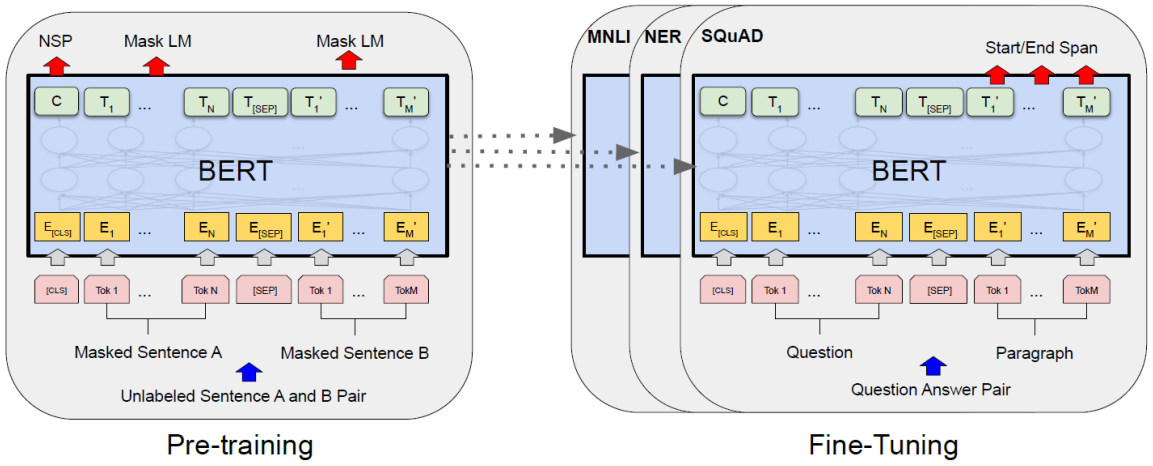
Exercise, use pre-trained BERT and compare different BERT versions

BERT
(Bidirectional Encoder Representations from Transformers)

1 Pretrain on:

- fill-in-the-blank

- binary classification if 2 sentences go together



Devlin, etal, 2019

2 Fine tune on variety of tasks

Keras NLP package has several BERT versions

So let's start with 'bert-small' (28M parameters)

For reference:
$BERT_{BASE}$ (L=12, H=768, Attn=12, Total Parameters=110M)
$BERT_{LARGE}$ (L=24, H=1024, Attn=16, Total Parameters=340M).

models/

bert_tiny_en_uncased	BERT	4M	2-layer BERT model where all input is lowercased. Trained on English Wikipedia + BooksCorpus.
bert_small_en_uncased	BERT	28M	4-layer BERT model where all input is lowercased. Trained on English Wikipedia + BooksCorpus.
bert_medium_en_uncased	BERT	41M	8-layer BERT model where all input is lowercased. Trained on English Wikipedia + BooksCorpus.
bert_base_en_uncased	BERT	109M	12-layer BERT model where all input is lowercased. Trained on English Wikipedia + BooksCorpus.
bert_base_en	BERT	108M	12-layer BERT model where case is maintained. Trained on English Wikipedia + BooksCorpus.
bert_base_zh	BERT	102M	12-layer BERT model. Trained on Chinese Wikipedia.
bert_base_multi	BERT	177M	12-layer BERT model where case is maintained. Trained on trained on Wikipedias of 104 languages

Notebook exercise using KerasNLP

- In a terminal window start the notebook session for keras-nlp

  …]$ cd KerasTest

  …]$ bash jupyter-keras-nlp-cpu2res.sh

  (need to set up conda environment)

- Open the URL and look for:

  BERT_FineTune_v3.ipynb notebook

- Select Edit->clear all cells; Run -> Run all cells

(or select Kernel -> restart Kernal and run all cells)

python - add_weig | Getting Started wi | python - keras loa | install tensorflow 2 | SI2023_BERT_... - J

napped-dedicator-diagnosis.expanse-user-content.sdsc.edu/lab/tree/WKSHOPS/NRP24/Intro-Transfor...

Finish update

Course | Prompt En...

All Bookmarks

File Edit View Run Kernel Tabs Settings Help

Filter files by name

/ ... / NRP24 / Intro-Transformer /

Name	Last Modified
keras-nlp...	2 days ago
keras-nlp...	yesterday
keras-nlp...	yesterday
mystart-j...	2 days ago
mystart-j...	yesterday
mystart-j...	yesterday
SI2023_B...	40 seconds ago
SI2023_D...	yesterday
SI2023_t...	2 days ago
SI2023_Tr	2 days ago

SI2023_BERT_FineTune_v3.ipy

Code

Python 3 (ipykernel)

[12]:
```
myES = tf.keras.callbacks.EarlyStopping(monitor='val_sparse_categorical_accuracy', mode='m

fit_history=mymodel.fit(textin_trn_ds,
    validation_data=textin_val_ds,
    epochs=4,shuffle=True,   #just run a few epochs
    callbacks=[myES])

print('done training')
```

```
Epoch 1/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 28s 2s/step - loss: 0.6665 - sparse_categorical_accuracy: 0.5587
- val_loss: 0.8364 - val_sparse_categorical_accuracy: 0.6500
Epoch 2/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 11s 2s/step - loss: 0.6064 -
- val_loss: 0.7352 - val_sparse_categorical_accuracy:
Epoch 3/10
5/5 ━━━━━━━━━━━━━━━━━━━━ 11s 2s/step - loss: 0.5802 -
- val_loss: 0.6733 - val_sparse_categorical_accuracy:
Epoch 4/10
```

Would you like to receive official Jupyter news? Please read the privacy policy.

Open privacy policy Yes No

Simple 0 1 Python 3 (ipykernel) | Idle Mode: Edit Ln 5, Col 13 SI2023_BERT_FineTune_v3.ipynb 1

52°F 11:06 PM 3/17/2024

END