HECC Data Science Platform

6/3/2020

NASA Advanced Supercomputing Division

Outline

- **About the Data Science Team**
- **HECC Data Science**
  - Data Science Conda Environments
  - Deploying Environments on Pleiades
  - Distributed TensorFlow using GPUs
  - Dask (Distributed Python using CPUs)
- **Pilot Projects**
  - Solar Cell Materials Prediction
  - Carbon Nanotube Neural Network Gas Sensor
  - System Log Aggregation

Question? Use the Webex chat facility to ask the Host

About the Data Science Team

- Shubha Ranjan – Data Science Group Lead
- Data Science Team Members
  - Thaddeus Norman
  - Jeff Becker
  - Samson Cheung
  - Jonathan Gee
  - Mark Jacquet
  - Thai Truong

HECC Data Science

- Data Science Conda Environments
  - TensorFlow: tf1_14, tf2, tf2_1
  - PyTorch: pyt1_3
- Deploying Environments on Pleiades
  - Accessing Environments
    - module use -a /swbuild/analytix/tools/modulefiles
    - module load miniconda3/v4
    - source activate <*environment_name*>

Common Python Packages

- Pandas – Data analytic platform
- Dask – Distributed data analytic platform
- Scikit-learn – Machine learning platform
- Matplotlib – Plotting and graphics platform
- Pillow – A fork of Python Image Library (PIL)
- Jupyter Lab – A collaborative and integrated development environment
- To request or suggest a package:
  - Contact: support@nas.nasa.gov

Creating a Custom Conda Environment

- Accessing NAS conda environments:
  - module use -a /swbuild/analytix/tools/modulefiles
  - module load miniconda3/v4
  - source activate *environment_name*
- Adding packages for local use:
  - pip install --user *package_name*
- Creating a new conda environment:
  - export CONDA_ENVS_PATH=/nobackup/$USER/.conda/envs
  - conda create –-name *my_env* --clone base

Jupyter Lab & Web UIs

- Web UIs and security
  - Web servers allow access to your file system!
  - When using a web server you must be on a compute node!
  - Compute nodes have restricted access
  - Compute nodes will not allow you to download files from internet
- Interactive Mode
  - qsub –I –l select *node …*
- Reserved Front End
  - pbs_rfe --duration *10+* --model *bro*

Question? Use the Webex chat facility to ask the Host

Setting Up Jupyter Lab

1. Log into rfe or PBS interactive mode:
   - ssh *node_name*
   - qsub –I –l select=1:ncpus=28:model=bro
2. Load data analytics modules:
   - module use -a /swbuild/analytix/tools/modulefiles
   - module load miniconda3/v4
   - source activate tf2
3. Generate Jupyter config file:
   - jupyter notebook --generate-config

Securing & Verifying Jupyter Setup

1. Setup a Jupyter password & secure config:
   - jupyter notebook password
   - chmod 700 $HOME/.jupyter
   - openssl req -x509 -config /swbuild/analytix/tools/jupyter_setup/jupyter_ssl.conf \ -newkey rsa -out $HOME/.jupyter/jupyter.pem -days 1000
   - cp /swbuild/analytix/tools/jupyter_setup/jupyter_notebook_config.py $HOME/.jupyter/
2. Verifying setup:
   - $HOME/.jupyter
   - (jupyterlab)% ls -l $HOME/.jupyter/
   - -rw------- 1 103 Jun 17 13:47 jupyter_notebook_config.json
   - -rw------- 1 1704 Jun 17 14:35 jupyter.key
   - -rw------- 1 1155 Jun 17 14:35 jupyter.pem
   - -rw------- 1 29875 Jun 17 14:35 jupyter_notebook_config.py

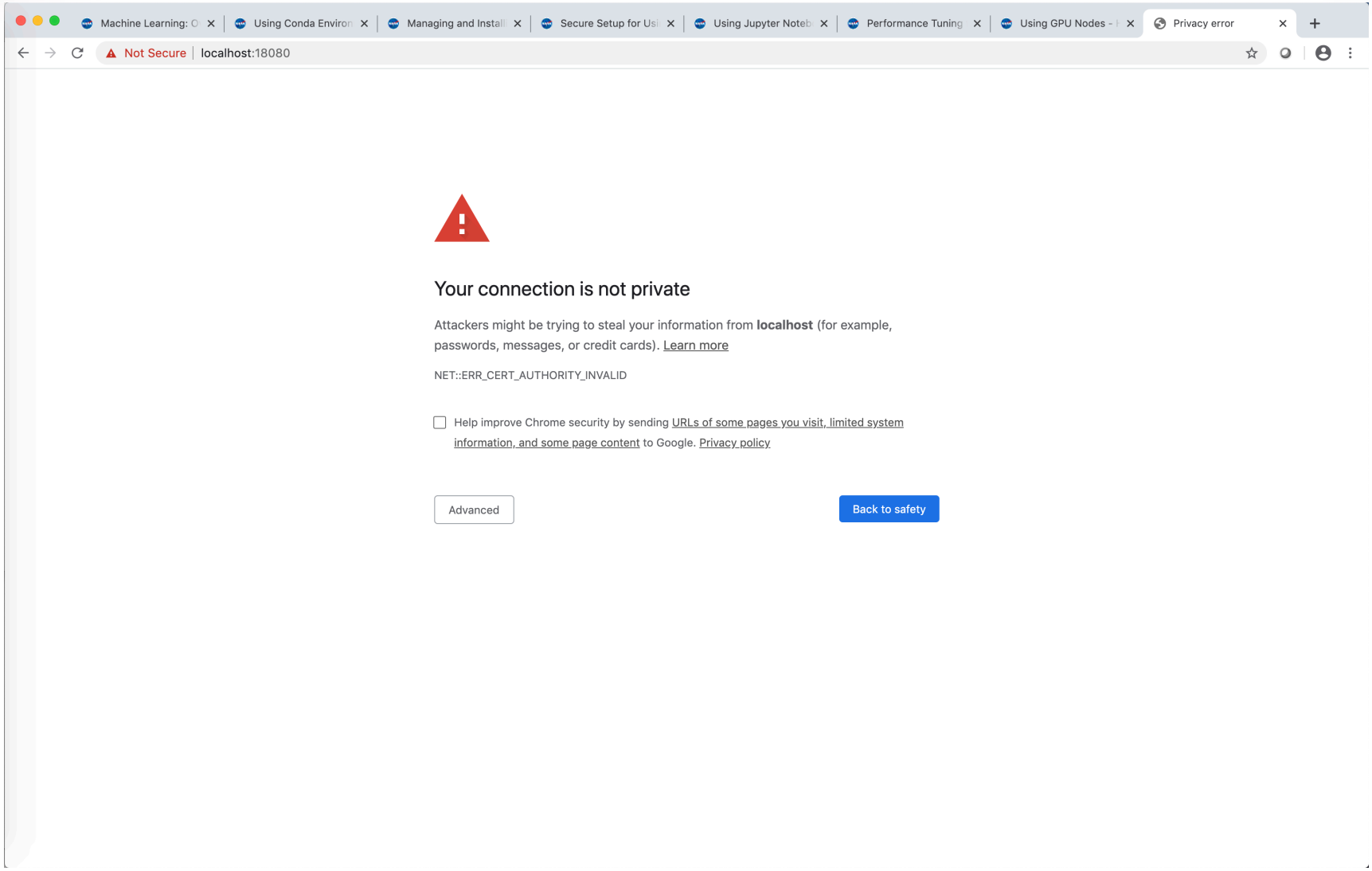
Jupyter Config Settings

- In jupyter_notebook_config.py confirm:
  - #c.NotebookApp.certfile = os.path.join(our_dir, 'jupyter.pem')
  - #c.NotebookApp.keyfile = os.path.join(our_dir, 'jupyter.key')
  - c.NotebookApp.certfile = '/home6/*username*/.jupyter/jupyter.pem'
  - c.NotebookApp.keyfile = '/home6/*username*/.jupyter/jupyter.key’

Running Jupyter Lab

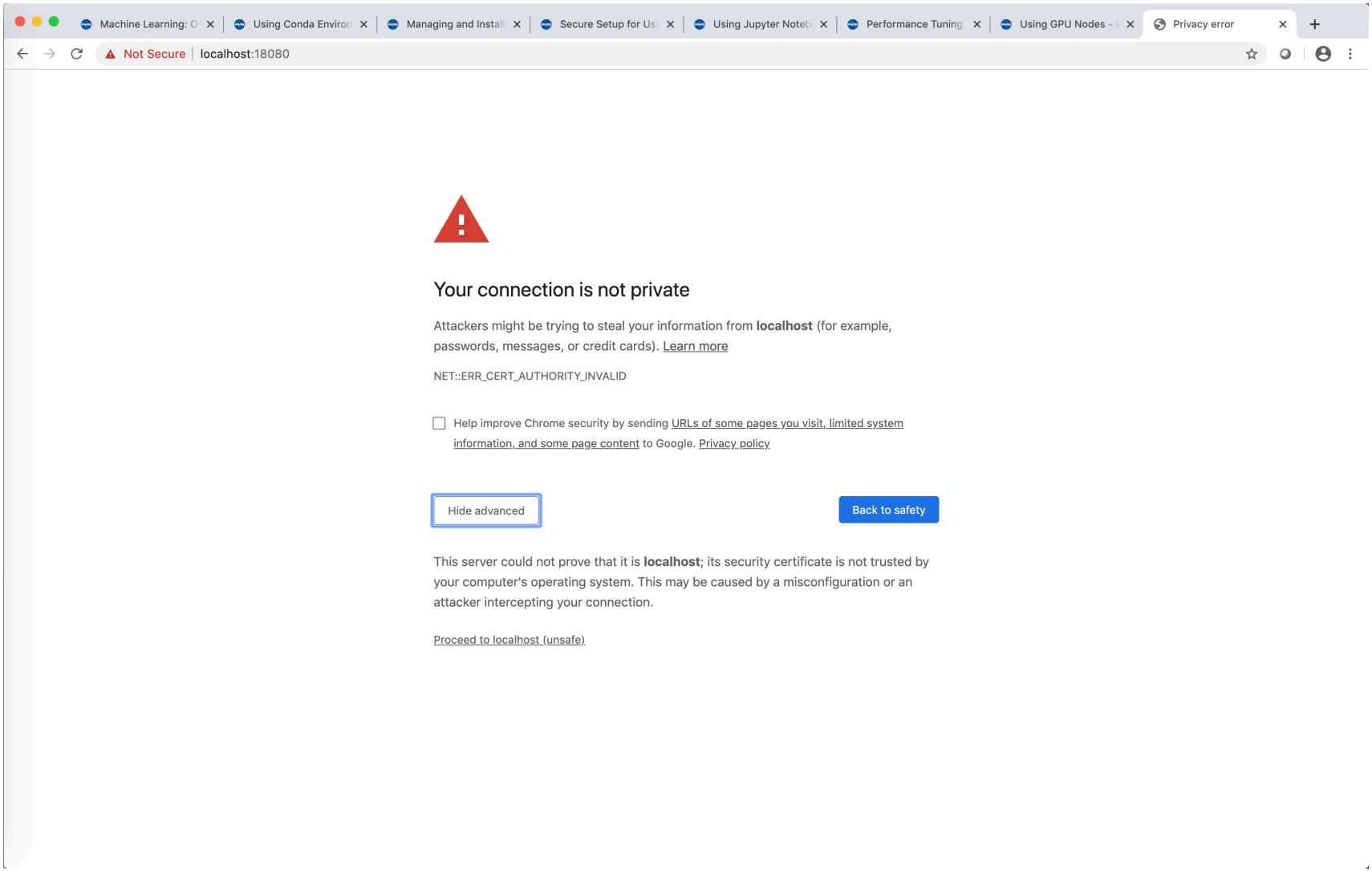
- Setup SSH passthrough:
  - https://www.nas.nasa.gov/hecc/support/kb/one-step-connection-using-public-key-and-passthrough_62.html
- Start PBS interactive session:
  - qsub -I -q k40 –l select=1:ncpus=16:model=san_gpu
- Or log onto rfe:
  - ssh *node_name*
- Start conda environments and Jupyter:
  - module use -a /swbuild/analytix/tools/modulefiles
  - module load miniconda3/v4
  - source activate tf2
  - jupyter lab –no-browser
- In a new shell on your local machine:
  - ssh -o "StrictHostKeyChecking ask" -L 8080:localhost:8888 –o ProxyJump=sfe,pfe21 node_name

First Login to Jupyter-Initial Screen



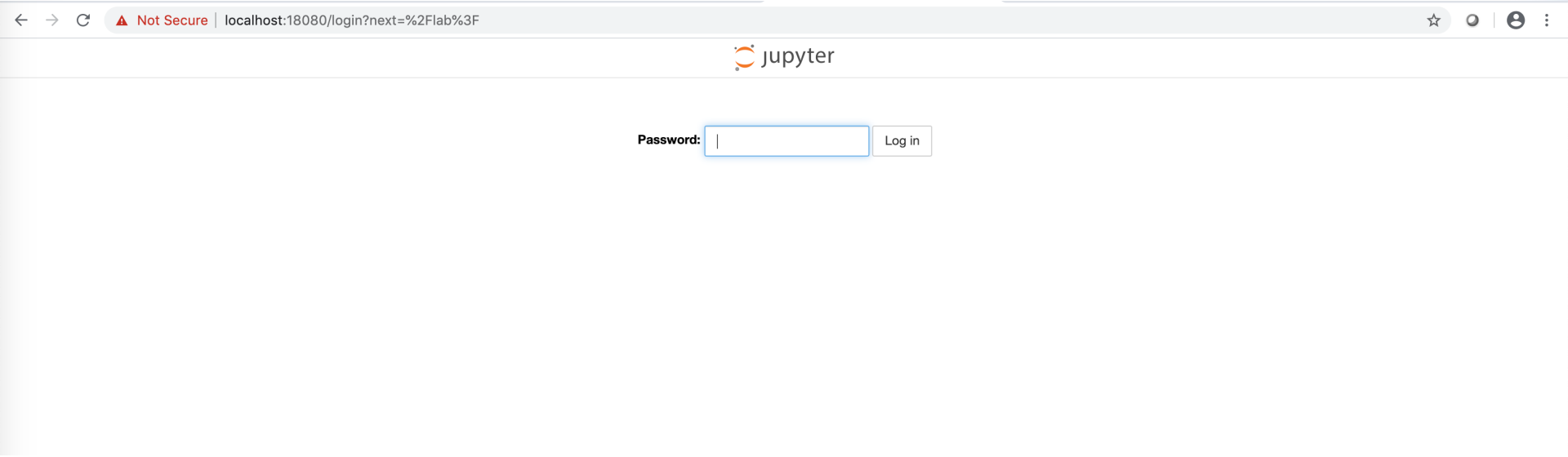
Question? Use the Webex chat facility to ask the Host

First Login to Jupyter-Second Screen



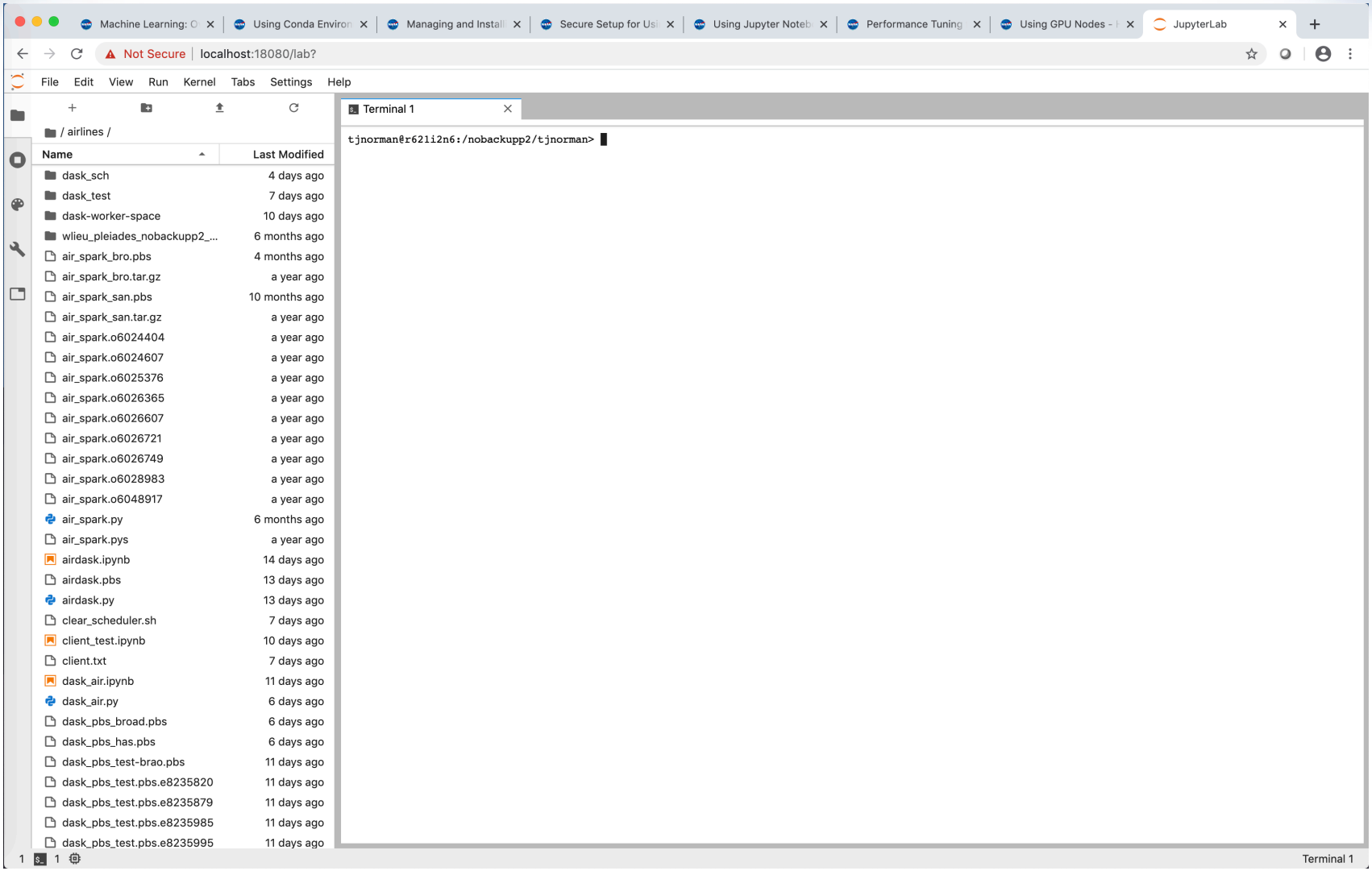
Question? Use the Webex chat facility to ask the Host

First Login to Jupyter-Enter Your Password



Question? Use the Webex chat facility to ask the Host

First Login-Jupyter Interface



Deep Learning Tutorials

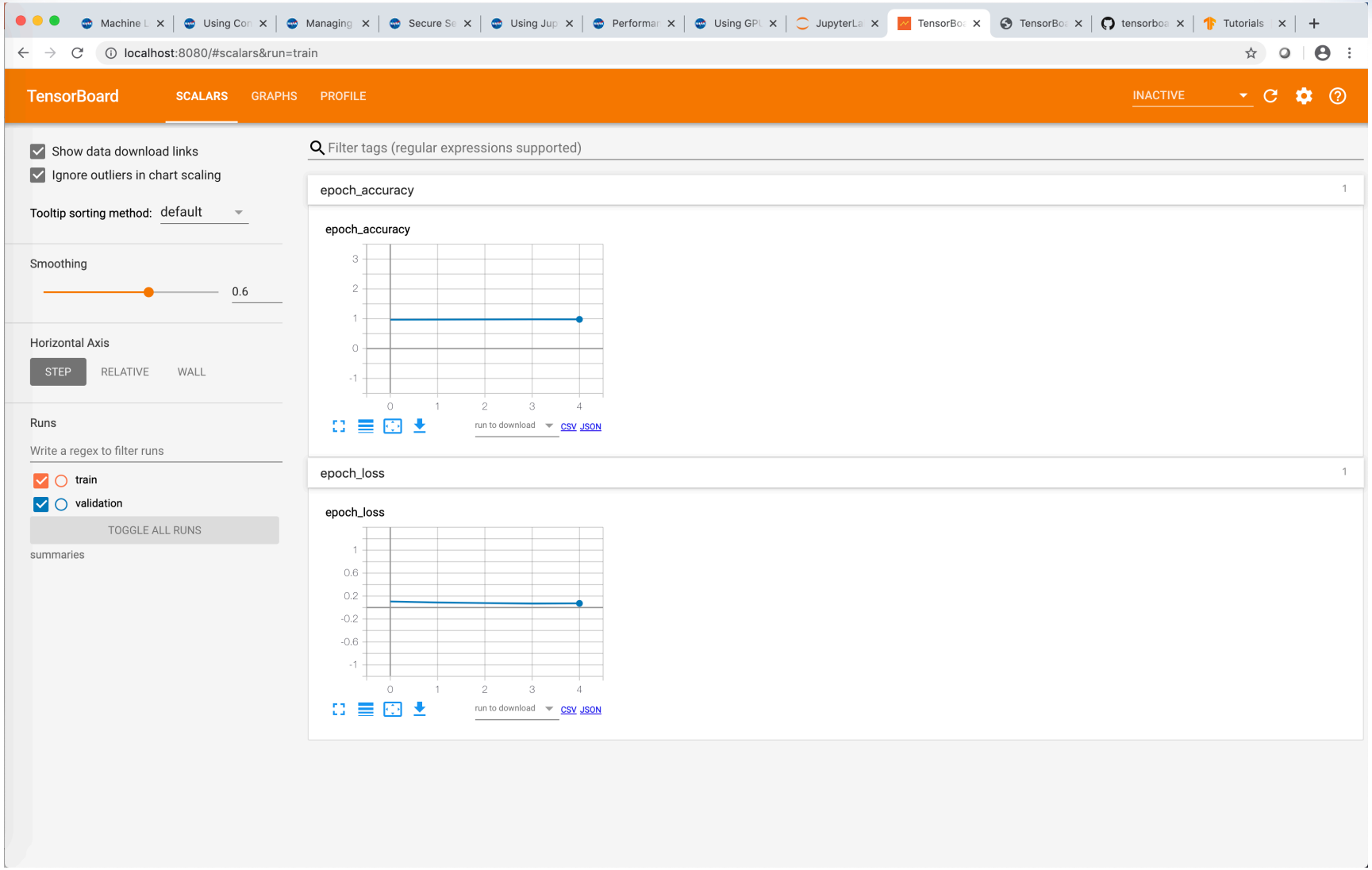
- TensorFlow:
  - https://www.tensorflow.org/tutorials
  - https://developers.google.com/machine-learning/crash-course
- PyTorch
  - https://pytorch.org/tutorials/

Question? Use the Webex chat facility to ask the Host

TensorFlow & Tensorboard

- TensorFlow environments:
  - TensorFlow 1.14: source activate tf1_14
  - TensorFlow 2.0: source activate tf2
  - TensorFlow 2.1: source activate tf2_1
- To use Tensorboard:
  - Log onto RFE or start PBS interactive mode
  - Activate tf2 environment
  - At command prompt: tensorboard --logdir= *target_directory*
  - On your local machine:
    - ssh -o "StrictHostKeyChecking ask" -L 8085:localhost:6006 -o ProxyJump=sfe,pfe21 node_name

Tensorboard Example: Scaler



Tensorboard Example: Hyperparameters

- Import functions:
  - import tensorflow as tf
  - from tensorboard.plugins.hparams import api as hp
- Setup hyperparameters:
  - HP_NUM_UNITS = hp.HParam('num_units', hp.Discrete([16, 32]))
  - HP_DROPOUT = hp.HParam('dropout', hp.RealInterval(0.1, 0.2))
  - HP_OPTIMIZER = hp.HParam('optimizer', hp.Discrete(['adam', 'sgd']))
- Setup dictionary of hyperparameters:
  - hparams = {
  - HP_NUM_UNITS: num_units,
  - HP_DROPOUT: dropout_rate,
  - HP_OPTIMIZER: optimizer,
  - }
- Insert into keras:
  - tf.keras.layers.Dense(hparams[HP_NUM_UNITS], activation=tf.nn.relu),
  - tf.keras.layers.Dropout(hparams[HP_DROPOUT]),
  - model.compile(optimizer=hparams[HP_OPTIMIZER], …..)

Tensorboard Example: Hyperparameters

JupyterLab | TensorBoard | Hyperparameter Tuning with th

localhost:8080/#hparams

TensorBoard SCALARS HPARAMS INACTIVE

Hyperparameters

- [x] num_units
  - [x] 16.000
  - [x] 32.000
- [x] dropout

Min: -infinity

Max: +infinity

- [x] optimizer
  - [x] adam
  - [x] sgd

Metrics

- [x] Accuracy

Min: -infinity Max: +infinity

Status

[x] Unknown [x] Success [x] Failure [x] Running

Sorting

Sort by Direction

Paging

Number of matching session groups: 8

Page # 1 / 1 Max # of session groups per page: 100

TABLE VIEW PARALLEL COORDINATES VIEW SCATTER PLOT MATRIX VIEW

Trial ID	Show Metrics	num_units	dropout	optimizer	Accuracy
3df0d7cf35bec5a...	☐	32.000	0.20000	sgd	0.89330
3ec2aed9e07589f...	☐	32.000	0.20000	adam	0.93400
53bf5bece9190fa...	☐	16.000	0.20000	adam	0.91980
5b97f3c2967245b...	☐	16.000	0.10000	adam	0.91970
6826c7fa3322d82...	☐	32.000	0.10000	adam	0.93880
7684dcc13358fd0...	☐	16.000	0.20000	sgd	0.88380
7b29a731e3daca7...	☐	32.000	0.10000	sgd	0.89560
ae235909ec4e4d9...	☐	16.000	0.10000	sgd	0.89040

Using GPU Nodes

- GPU nodes currently available
  - san_gpu (Sandy Bridge GPU)
  - #PBS -l select=xx:ncpus=yy:model=san_gpu –q k40

  - sky_gpu (Skylake GPU)
  - #PBS -l select=xx:ncpus=yy:model=sky_gpu –q v100
  - #PBS -l select=xx:ncpus=yy:model=sky_gpu:naccelerators=8 –q devel

  - xx == number of nodes
  - yy == number of cpus

Distributing TensorFlow Over GPUs

- TensorFlow Documentation:
  - https://www.tensorflow.org/guide/distributed_training
  - https://www.tensorflow.org/tutorials/distribute/multi_worker_with_keras
- TensorFlow mirroredstrategy:
  - mirrored_strategy = tf.distribute.MirroredStrategy()
  - mirrored_strategy = tf.distribute.MirroredStrategy(devices=["/gpu:0", "/gpu:1"])
- Example Code:

```
mirrored_strategy = tf.distribute.MirroredStrategy()
with mirrored_strategy.scope():
    model = Model()
    model.complile(loss = …, optimizer = …, …)
model.fit(x_train, y_train, batch_size, epochs, …)
```

Distributing TensorFlow Over Multi GPU Nodes

- Four Step Process:
  1. Get nodes from PBS
  2. Run shell script on each node
  3. Assign nodes to TensorFlow
  4. Setup model and data in MultiWorkerMirroredStrategy() scope
- Step 1: Get nodes from PBS

```
#in PBS script get node information from nodefile
NODES=($( cat $PBS_NODEFILE | sort | uniq ))
NUM_OF_NODES=${#NODES[@]}
```

Distributing TensorFlow Over Multi GPU Nodes

- Step 2: Run shell script on each node

```
#worker number for other nodes
C=1

# for each node that's not the current node
for node in ${NODES[@]}
do
  if [[ $node != $(eval hostname) ]]
    then
    # ssh into each node and run the .sh script with node info
    ssh $node "$BASE/script.sh $C ${NODES[*]}" &
    C=$(($C+1))
    sleep 2
  fi
Done
#run the main worker node
$BASE/script.sh 0 ${NODES[*]}
```

Still in PBS

1. run script.sh in each worker node

2. run script.sh in master node

Distributing TensorFlow Over Multi GPU Nodes

- The Shell Script – script.sh
  - #cd to code location cd /nobackup/$USER
  - BASE=/nobackup/$USER/directory
  - module purge
  - module -a use /swbuild/analytix/tools/modulefiles
  - module load miniconda3/v4
  - source activate tf2
  - #run python script with inputs from this .sh script python $BASE/*python_code.py* $@

Distributing TensorFlow Over Multi GPU Nodes

- Step 3: Assign nodes to TensorFlow

In python code

```
base_dir = os.path.dirname(os.path.abspath(__file__))
# -------------- Set up TF_CONFIG -------------- #
# load the index and node info from the command line
index = int(sys.argv[1])
verbose = index < 1
nodes=[]
# node names for each node, append a port # to the names
for i in range(2,len(sys.argv)):
    nodes.append(sys.argv[i] + ':2001')
# set TF_CONFIG variable that MultiWorkerMirroredStrategy needs
os.environ['TF_CONFIG'] = json.dumps({
    'cluster': {
        'worker': nodes
    },
    'task': {'type': 'worker', 'index': index}
})
```

1. Collect node information
2. Assign node to TF_CONFIG

Distributing TensorFlow Over Multi GPU Nodes

- Step 4: Setup model in data in MWMS scope

```
#use the distributed strategy and create model and data in strategy scope
strategy =
tf.distribute.experimental.MultiWorkerMirroredStrategy(tf.distribute.
experimental.CollectiveCommunication.AUTO)

with strategy.scope():
    datasets = make_datasets()
     model = Model()
     model.compile(loss = …, optimizer = …, …)
model.fit(x_train, y_train, batch_size, epochs, …)
```

Using Dask

- Distributed Python Environment
  - https://tutorial.dask.org/
  - https://github.com/dask/dask-tutorial
  - https://docs.dask.org/en/latest/setup/hpc.html
- Single Node Interactive Mode:
  - from dask.distributed import Client, LocalCluster
  - cluster = LocalCluster()
  - client = Client(cluster)
- Multiple Node Interactive Mode:
  - mpirun -np <number of cpus> dask-mpi --no-nanny --interface ib0 --local-directory /nobackup/<username>/dask_sch/ --scheduler-file /nobackup/<username>/dask_sch/ sched.json
  - client = Client(scheduler_file='/../dask_sch/sched.json')

Submitting a Dask Job

In PBS script:

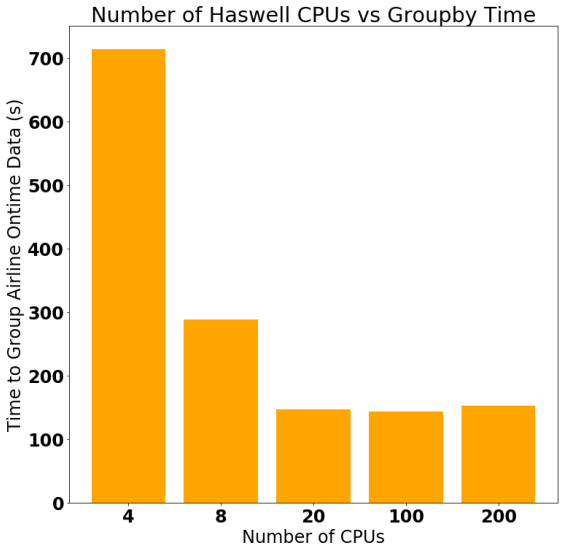
```
#clear and create directory for dask scheduler
if [ -d dask_sch ]
then
    echo "removing old scheduler directory"
    rm -r dask_sch
    echo "creating new scheduler directory"
    mkdir dask_sch
else
    echo "creating scheduler directory"
    mkdir dask_sch
fi

#run program
#MPI is included in the environment through MPICH.
mpirun -np <number of cpus> dask-mpi --no-nanny --interface ib0 --local-directory
/nobackup/username/dask_sch/ --scheduler-file /nobackup/username /dask_sch/sched.json & python
<filename.py>
```

In Python Code:

```
from dask.distributed import Client
client = Client(scheduler_file='/…/dask_sch/sched.json')
```

Example: Airline Ontime Data Set

- Dask Code:
  - from dask.distributed import Client
  - client = Client(scheduler_file='/nobackupp2/tjnorman/airlines/dask_sch/sched.json’)
  - df_airlines = dd.read_csv(data + '/On_Time_Reporting_Carrier_On_Time_Performance_*.csv',
  - assume_missing=True, dtype={'Year':'int64’, ….. }, encoding='ISO-8859–1’)
  - df_air = df_airlines[[…*selected columns…*]]
  - df_air_year = df_air.groupby('Year').mean()
  - df_air_year = df_air_year.reset_index()
  - df_air_year = df_air_year.compute()
  - df_air_year.to_csv('ontime_data_279_dis.csv', index=False)



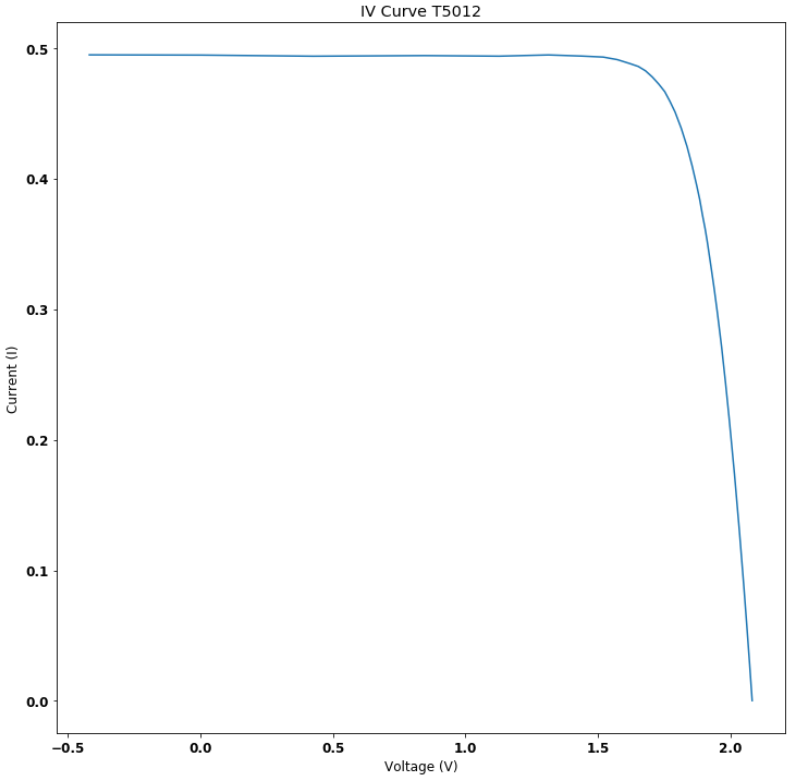
Pilot Projects

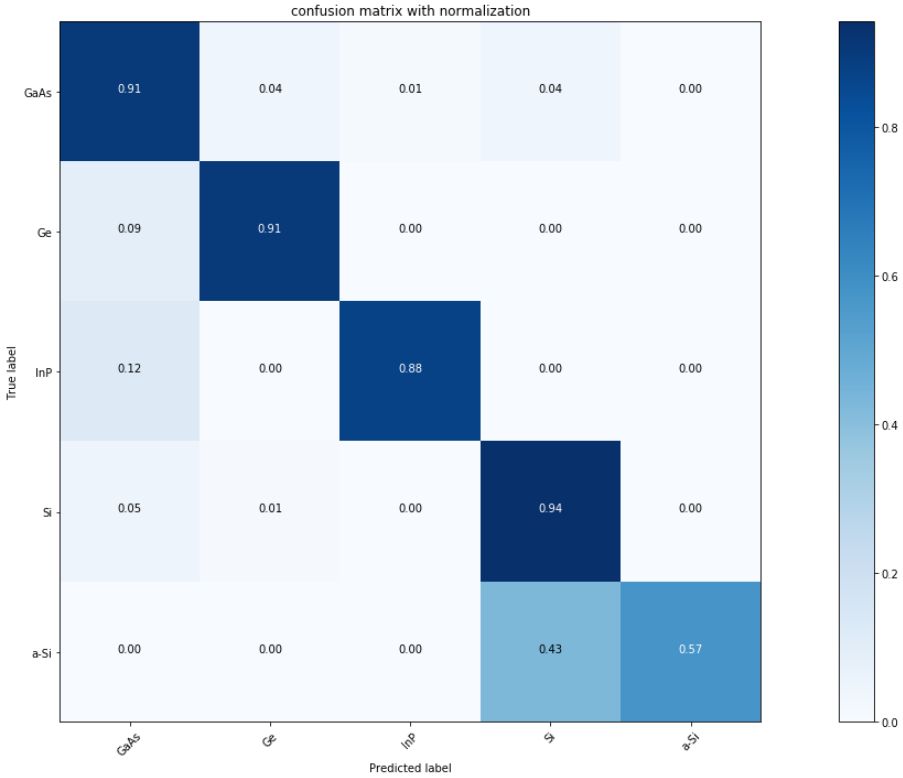
- What Are Pilot Projects?
  - Highlight Existing Capabilities
  - Explore New Technology
- How Do You Initiate One?
  - Contact Shubha Ranjan: shubha.ranjan@nasa.gov
- Here Are Some Examples:
  - Solar Cell Materials Prediction
  - Carbon Nanotube Neural Network Gas Sensor
  - System Log Aggregation

Question? Use the Webex chat facility to ask the Host

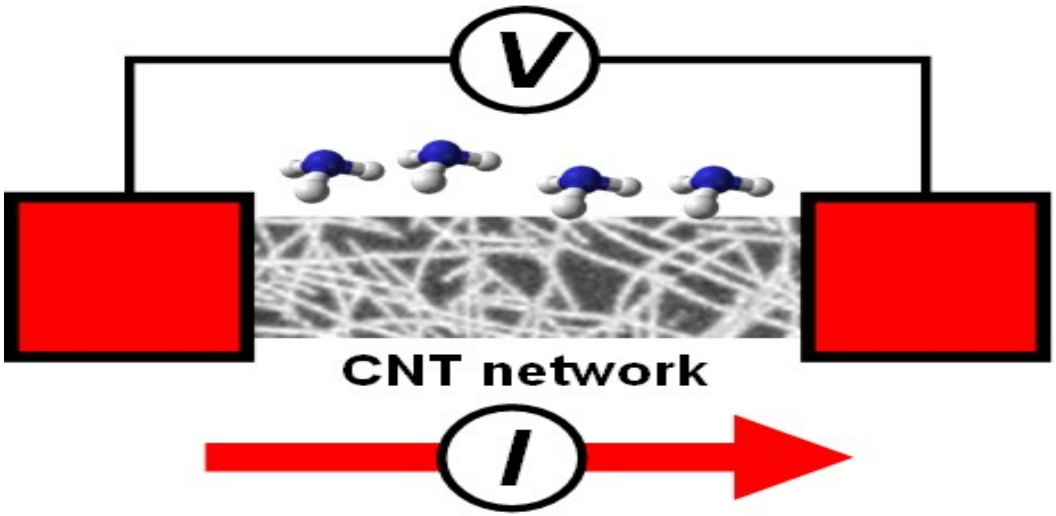
Predicting PV Cell Material Form Electrical Data

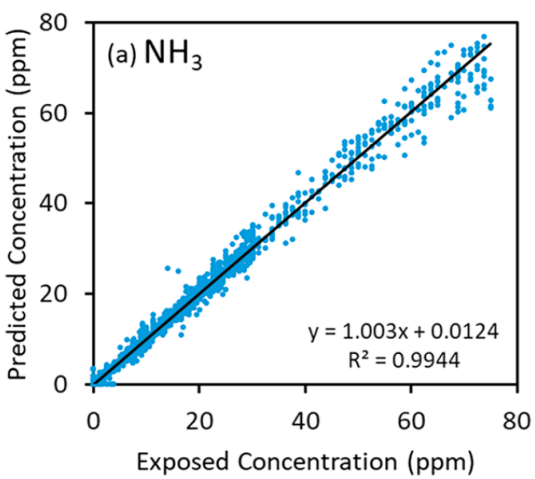
- Compounds targeted in model:
  - GaAs (1055), Ge (323), InP (27), Si (1156), a-Si (22)
  - SCC, RCS, RSH, RS, DIF, VOC

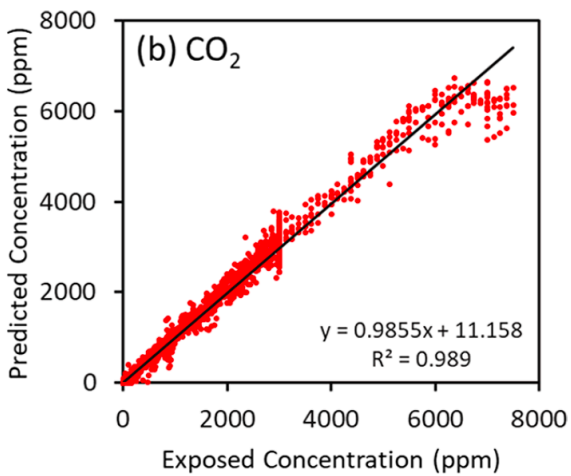




Carbon Nanotube NN Gas Sensor







CNT Pros
- Inexpensive
- Easy to fabricate
- Resistance Sensitive to gas concentration

CNT Cons
- Non-linear response
- Difficult to model

Neural Network Solution
- Trained model to predict gas concentration
- Recommended path for future work

System Log Aggregation

Problem:

- System log data stored in various different files
- Interest in understanding log statement to predict node failure

Solution:

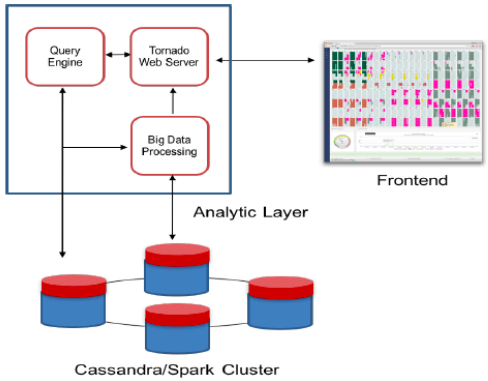


Fig. 3. Overall architecture of the Log Analytics Framework consisting of the Cassandra distributed NoSQL database and the Apache Spark in-memory data processing engine

- Park et al: Titan supercomputer at the Oak Ridge Leadership Computing Facility (OLCF)
- 2017 IEEE International Conference on Cluster Computing (DOI 10.1109/CLUSTER.2017.113

Software Stack Description

- Web Interface – Jupyter Lab:
  - ➢ Gives access to Python, Dask and related data science tools
- Memory Management – Arrow:
  - ➢ Facilitates data transfer between Spark and Python data formats
- Query Engine – Spark:
  - ➢ Will allow rapid retrieval and manipulation of the data
- Database – Cassandra:
  - ➢ Setup in single node mode for prototyping purposes

Summary

- Conda environments based on popular Deep Learning Frameworks
- Additional Software added on user request
- Secure Jupyter Lab by using RFE or PBS interactive
- Help get you started with Machine Learning
- Pilot Projects
- We’re Here To Help!

Question? Use the Webex chat facility to ask the Host