Python overhead: Reducing the burden

Michael Bareford, EPCC, The University of Edinburgh

m.bareford@epcc.ed.ac.uk

epcc

Reusing this material

Cirrus - SGI ICE XA Supercomputer

- 280 compute nodes (10,080 cores)
  - Dual socket Intel Xeon E5-2695, 36c, 2.1 GHz
  - 256 GiB memory per node
- 36 GPU compute nodes (144 GPUs, 1,440 cores)
  - Dual socket Intel Xeon Gold 6248, 40c, 2.5 GHz
  - Four NVIDIA Tesla V100-SXM2-16GB (Volta) GPUs
    - PCIe connections
  - 384 GiB memory
- Infiniband Fabric
  - Single Infiniband interface, 54.5 Gbps
- File storage
  - /home, 1.5 PB, Ceph
  - /work, 400 TB, Lustre
  - /scratch, 256 TB, RPOOL (solid state)

https://www.cirrus.ac.uk/

cirrus epcc

Python on Cirrus: TCL modules

TCL (Tool Command Language) module files

```
python/3.7.16
python/3.9.13
```

CPU nodes

```
python/3.8.16-gpu
python/3.9.13-gpu
python/3.10.8-gpu
```

GPU nodes

Python on Cirrus: Miniconda installs

TCL (Tool Command Language) module files

```
python/3.7.16
python/3.9.13
```

CPU nodes

```
python/3.8.16-gpu
python/3.9.13-gpu
python/3.10.8-gpu
```

GPU nodes

Running "`module help ...`" will show that each module is a Miniconda installation.

Miniconda is a lightweight Python distribution that allows you put together the minimal set of Python packages needed for your requirements.

Miniconda contains a Python package manager called `pip` (as well as `conda`).

Python on Cirrus: Miniconda installs

TCL (Tool Command Language) module files

CPU nodes

```
python/3.7.16
python/3.9.13
```

GPU nodes

```
python/3.8.16-gpu
python/3.9.13-gpu
python/3.10.8-gpu
```

Running "`module help ...`" will show that each module is a Miniconda installation.

CPU nodes

```
23.1.0-1-py37
4.12.0-py39
```

Miniconda versions

GPU nodes

```
23.3.1-0-py38
4.12.0-py39
22.11.1-1-py310
```

Miniconda versions

Python on Cirrus: Python packages

python/3.9.13

dask	2023.5.0
ipyparallel	8.6.1
jupyterlab-server	2.10.3
mpi4py	3.1.3
numpy	1.24.3
pandas	2.0.1
scipy	1.10.1
	...

python/3.9.13-gpu

cupy-cuda116	10.3.1
dask	...
ipyparallel	...
jupyterlab-server	...
mpi4py	3.1.3
numpy	...
pandas	...
pycuda	2022.1
scipy	1.9.0
	...

Python on Cirrus: Python packages

python/3.9.13

dask	2023.5.0
ipyparallel	8.6.1
jupyterlab-server	2.10.3
mpi4py	3.1.3

openmpi/4.1.4
ucx/1.9.0
...

python/3.9.13-gpu

cupy-cuda116	10.3.1
dask	...
ipyparallel	...
jupyterlab-server	...
mpi4py	3.1.3

openmpi/4.1.4-cuda-11.6
ucx/1.9.0-cuda-11.6

nvidia/nvhpc-nompi/22.2

Python on Cirrus: Virtual environments

```
[auser@cirrus-login1 auser]$ module load python/3.9.13-gpu
```

Python on Cirrus: Virtual environments

```
[auser@cirrus-login1 auser]$ module load python/3.9.13-gpu

  MY_VENV_ROOT=${HOME/home/work}/pyenvs/myvenv

  python -m venv --system-site-packages ${MY_VENV_ROOT}

  extend-venv-activate ${MY_VENV_ROOT}

  source ${MY_VENV_ROOT}/bin/activate
```

Python on Cirrus: Virtual environments

```
[auser@cirrus-login1 auser]$ module load python/3.9.13-gpu

  MY_VENV_ROOT=${HOME/home/work}/pyenvs/myvenv

  python -m venv --system-site-packages ${MY_VENV_ROOT}

  extend-venv-activate ${MY_VENV_ROOT}

  source ${MY_VENV_ROOT}/bin/activate
```

Python on Cirrus: Virtual environments

```
[auser@cirrus-login1 auser]$ module load python/3.9.13-gpu

  MY_VENV_ROOT=${HOME/home/work}/pyenvs/myvenv

  python -m venv --system-site-packages ${MY_VENV_ROOT}

  extend-venv-activate ${MY_VENV_ROOT}

  source ${MY_VENV_ROOT}/bin/activate
```

Python on Cirrus: Virtual environments

```
[auser@cirrus-login1 auser]$ module load python/3.9.13-gpu

  MY_VENV_ROOT=${HOME/home/work}/pyenvs/myvenv

  python -m venv --system-site-packages ${MY_VENV_ROOT}

  extend-venv-activate ${MY_VENV_ROOT}

  source ${MY_VENV_ROOT}/bin/activate
```

Python on Cirrus: Virtual environments

```
[auser@cirrus-login1 auser]$ module load python/3.9.13-gpu

  MY_VENV_ROOT=${HOME/home/work}/pyenvs/myvenv

  python -m venv --system-site-packages ${MY_VENV_ROOT}

  extend-venv-activate ${MY_VENV_ROOT}

  source ${MY_VENV_ROOT}/bin/activate

(myvenv) [auser@cirrus-login1 auser]$ python -m pip install <package name>
                                      python -m pip install <package name>==<version>

(myvenv) [auser@cirrus-login1 auser]$ deactivate
[auser@cirrus-login1 auser]$
```

Python on Cirrus: Local and base packages

${MYVENV_ROOT}/lib/python3.9/site-packages

```
...
metis-0.2a5.dist-info
metis.py
pyfr
pyfr-1.15.0.dist-info
```

/mnt/lustre/indy2lfs/sw/miniconda3/4.12.0-py39-gpu/lib/python3.9/site-packages

```
anyio
anyio-3.6.1.dist-info
appdirs-1.4.4-py3.9.egg
...
```

python/3.9.13-gpu

Python on Cirrus: Further customisation

${MY_VENV_ROOT}/bin/activate

```
# This file must be used with "source bin/activate" *from bash*
# you cannot run it directly

# *** ADD EXTRA ACTIVATION COMMANDS HERE ***

...

deactivate () {
    ...
    unset VIRTUAL_ENV
    if [ ! "${1:-}" = "nondestructive" ] ; then
    # Self destruct!
        unset -f deactivate
        # *** ADD EXTRA DEACTIVATION COMMANDS HERE ***
    fi
}

...
```

Python on Cirrus: Running jobs

submit-myvenv.slurm

```
#!/bin/bash

#SBATCH --job-name=myvenv
#SBATCH --account=[budget code]
#SBATCH --partition=gpu
#SBATCH --qos=gpu
#SBATCH --nodes=2
#SBATCH --gres=gpu:4
#SBATCH --time=24:00:00
#SBATCH --exclusive

source ${HOME/home/work}/pyenvs/myvenv/bin/activate

srun --ntasks=8 --tasks-per-node=4 --cpus-per-task=10 \
    python myvenv-script.py
```

Python on Cirrus: Running jobs

submit-myvenv.slurm

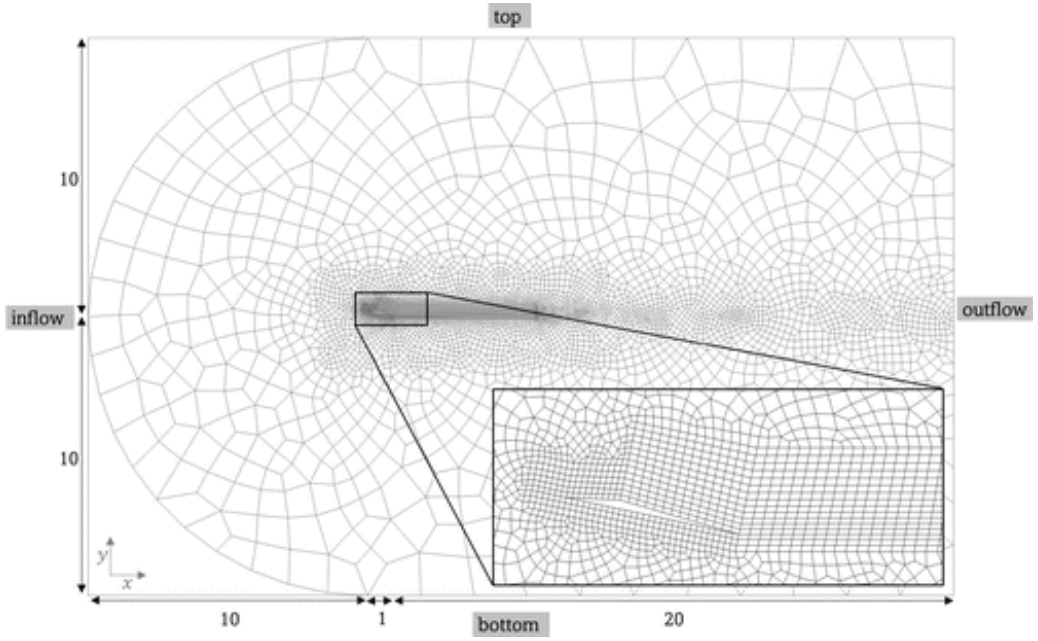
```
#!/bin/bash

#SBATCH --job-name=myvenv
#SBATCH --account=[budget code]
#SBATCH --partition=gpu
#SBATCH --qos=gpu
#SBATCH --nodes=2
#SBATCH --gres=gpu:4
#SBATCH --time=24:00:00
#SBATCH --exclusive

source ${HOME/home/work}/pyenvs/myvenv/bin/activate

srun --ntasks=8 --tasks-per-node=4 --cpus-per-task=10 \
    python myvenv-script.py
```

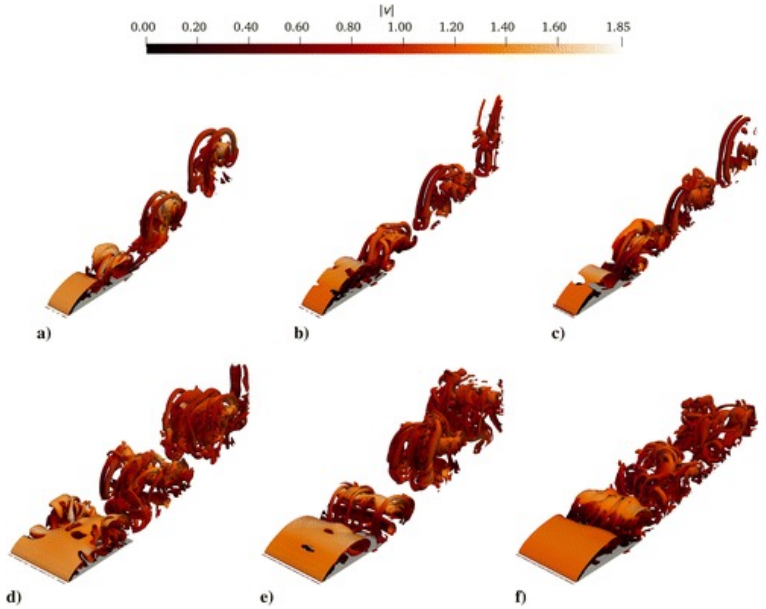
https://cirrus.readthedocs.io/en/main/user-guide/python.html#installing-your-own-python-packages-with-pip

Martian Aerodynamics with PyFR

GPU-accelerated Direct Numerical Simulations (DNS) of flow over a triangular aerofoil under Martian atmospheric conditions using PyFR.

Lidia Caros Roca et al 2022
https://doi.org/10.2514/1.J061454





PyFR is an open-source Python based framework for solving advection-diffusion type problems on streaming architectures using the Flux Reconstruction approach of Huynh.

https://www.pyfr.org/

Martian Aerodynamics with PyFR: Computational performance

- Parallel efficiency falls below 50% for multi-node runs when using **HPE MPT (MPI)**.

GPUs	Nodes	Runtime per checkpoint [min]		Parallel efficiency [%]	
		HPE MPT 2.22		HPE MPT 2.22	
2	1	130		n/a	
4	1	82		79	
16	4	37		44	

Martian Aerodynamics with PyFR: Computational performance

- Parallel efficiency at 80% for multi-node runs when using **OpenMPI**.

GPUs	Nodes	Runtime per checkpoint [min]		Parallel efficiency [%]	
		HPE MPT 2.22	OpenMPI 4	HPE MPT 2.22	OpenMPI 4
2	1	130	109	n/a	n/a
4	1	82	65	79	84
16	4	37	17	44	80

- Built **OpenMPI 4.1.4** specifically for NVIDIA V100 GPU nodes
  - `--with-ucx=/mnt/lustre/indy2lfs/sw/ucx/1.9.0-cuda-11.6`
  - `--with-pmi=/mnt/lustre/indy2lfs/sw/pmi2`
  - `--with-cuda=${NVHPC_ROOT}/cuda/11.6`
- Linked mpi4py with **OpenMPI** libraries
  - Parallel efficiency now at 80% for multi-node runs.

Martian Aerodynamics with PyFR: Computational performance

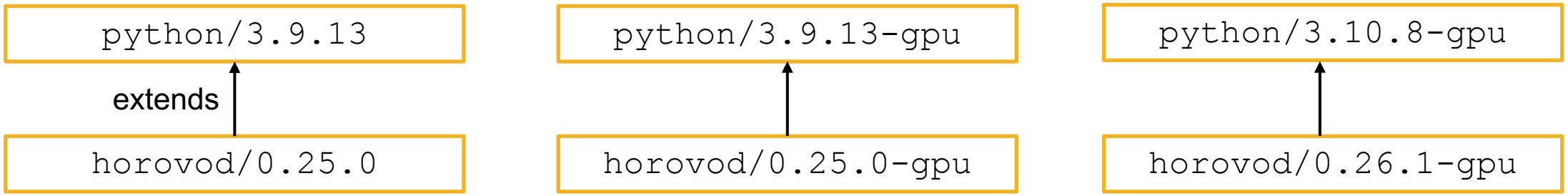
- Parallel efficiency at 80% for multi-node runs when using **OpenMPI**.

GPUs	Nodes	Runtime per checkpoint [min]		Parallel efficiency [%]	
		HPE MPT 2.22	OpenMPI 4	HPE MPT 2.22	OpenMPI 4
2	1	130	109	n/a	n/a
4	1	82	65	79	84
16	4	37	17	44	80

- Built **OpenMPI 4.1.4** specifically for NVIDIA V100 GPU nodes
  - `--with-ucx=/mnt/lustre/indy2lfs/sw/ucx/1.9.0-cuda-11.6`
  - `--with-pmi=/mnt/lustre/indy2lfs/sw/pmi2`
  - `--with-cuda=${NVHPC_ROOT}/cuda/11.6`
- Linked mpi4py with **OpenMPI** libraries
  - Parallel efficiency now at 80% for multi-node runs.

- OpenMPI supports direct GPU-to-GPU communication
  - NVLink intra-node GPU comms
  - Direct to Infiniband for inter-node GPU comms

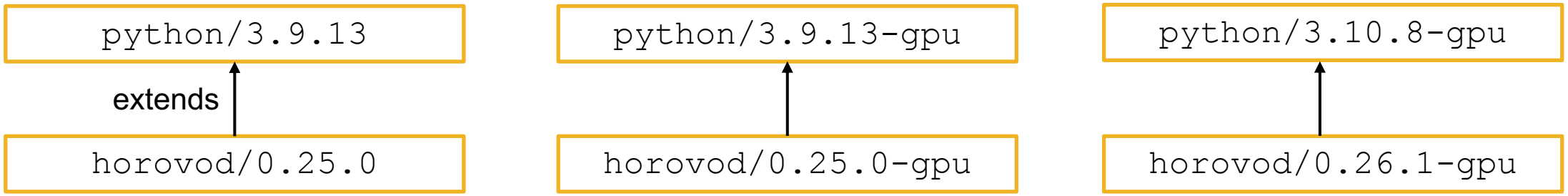
Machine Learning on Cirrus: Modules



- Horovod is a tool for running deep learning frameworks across multiple compute nodes
  https://github.com/horovod/horovod

- Horovod can be used with many machine learning (ML) platforms
  - TensorFlow, PyTorch, Keras, MXNet

Machine Learning on Cirrus: Modules



pip list

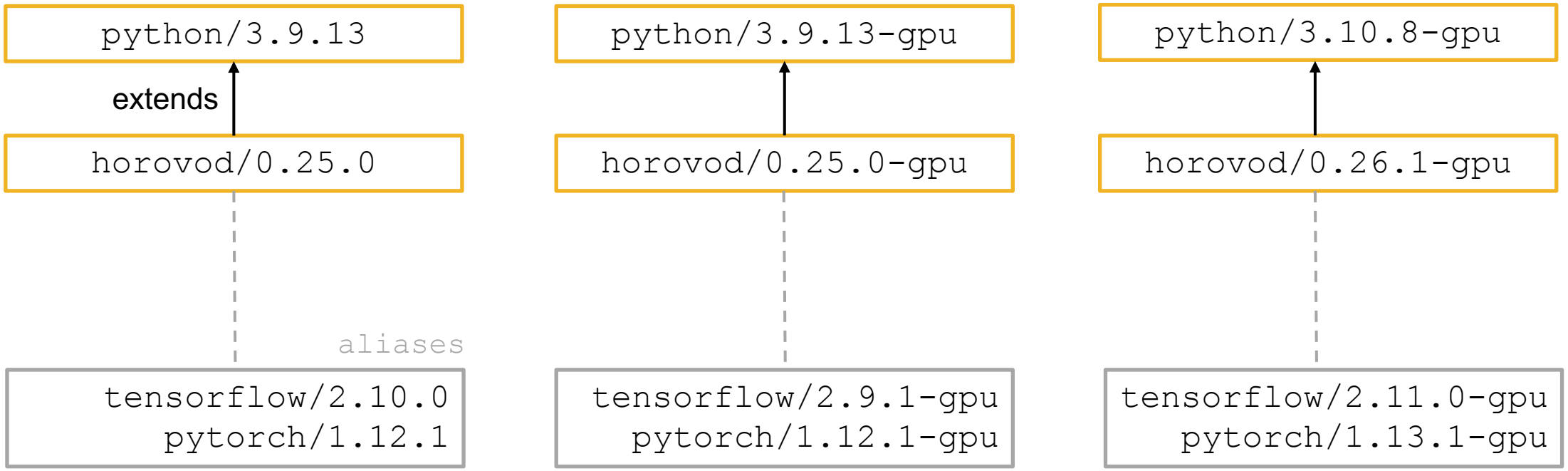
```
                  ...
          mpi4py  3.1.3
                  ...
           mxnet  1.9.1
                  ...
      tensorflow 2.10.0
tensorflow-cpu   2.10.0
                  ...
           torch 1.12.1
                  ...
```

```
                  ...
          mpi4py  3.1.3
                  ...
nvidia-cudnn-cu11 ...
                  ...
      tensorflow  2.9.1
tensorflow-gpu    2.9.1
                  ...
           torch 1.12.1
                  ...
```

```
                  ...
          mpi4py  3.1.4
                  ...
nvidia-cudnn-cu11 ...
                  ...
      tensorflow 2.11.0
tensorflow-gpu   2.11.0
                  ...
           torch 1.13.1
                  ...
```

epcc

Machine Learning on Cirrus: Modules



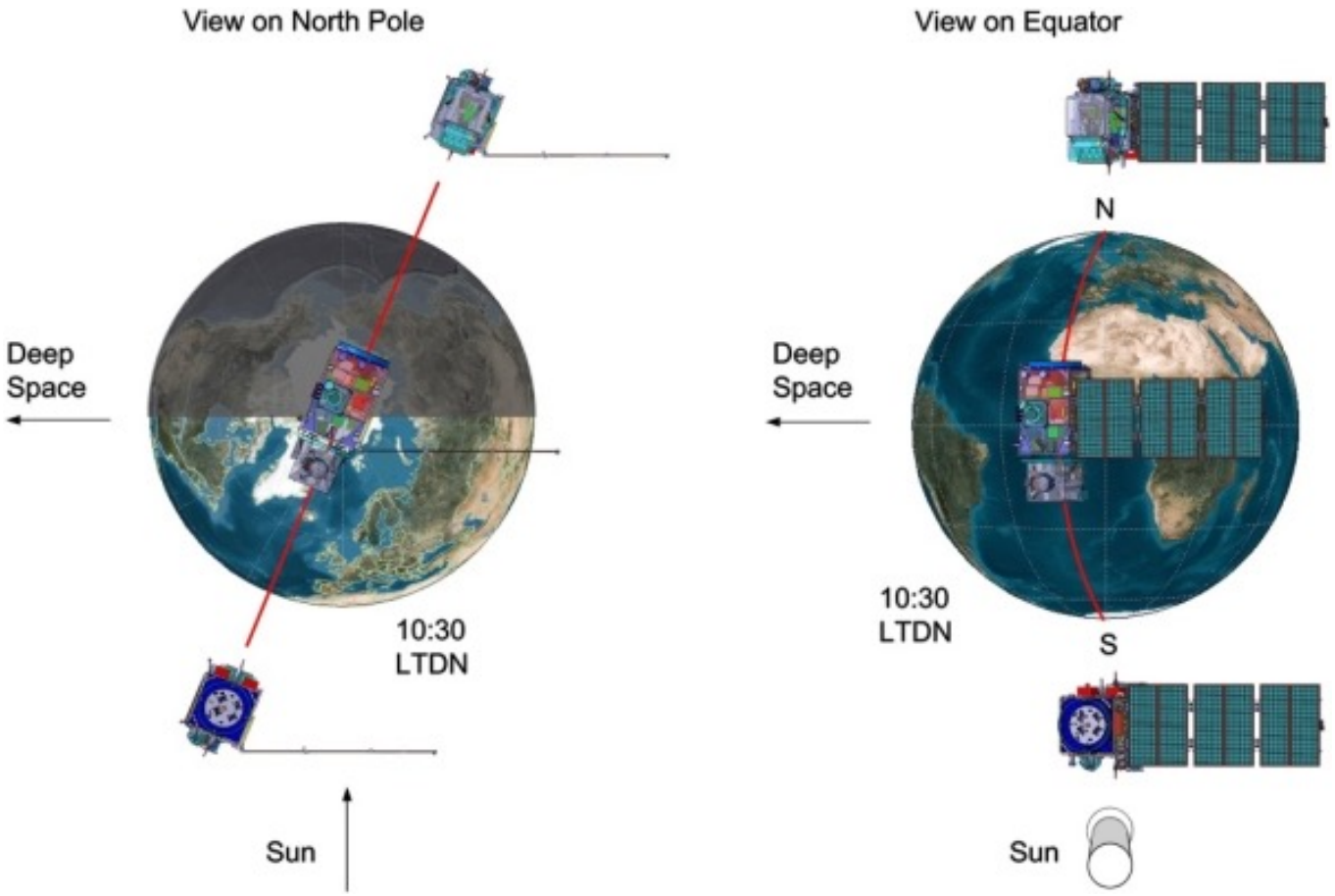
Python on Cirrus: Running TensorFlow

submit-horovod-tensorflow.slurm

```
#!/bin/bash

#SBATCH --job-name=hvtf
#SBATCH --partition=gpu
#SBATCH --qos=gpu
#SBATCH --nodes=4
#SBATCH --gres=gpu:4
...

module load tensorflow/2.9.1-gpu
...

mpirun -n 16 -N 4 -hostfile ./hosts -bind-to none -map-by slot \
    -x HOROVOD_MPI=1 -x HOROVOD_MPI_THREADS_DISABLE=1 \
    -x NCCL_DEBUG=INFO -x LD_LIBRARY_PATH -x PATH \
    python tf_cnn_benchmarks.py \
        --data_format=NCHW --model=resnet50 --variable_update=horovod \
        --num_gpus=1 --data_dir=${DATA_DIR} --print_training_accuracy=True
```

https://github.com/hpc-uk/build-instructions/blob/main/pyenvs/horovod/run_horovod_0.25.0_cirrus_gpu.md

Extracting field boundaries from satellite images



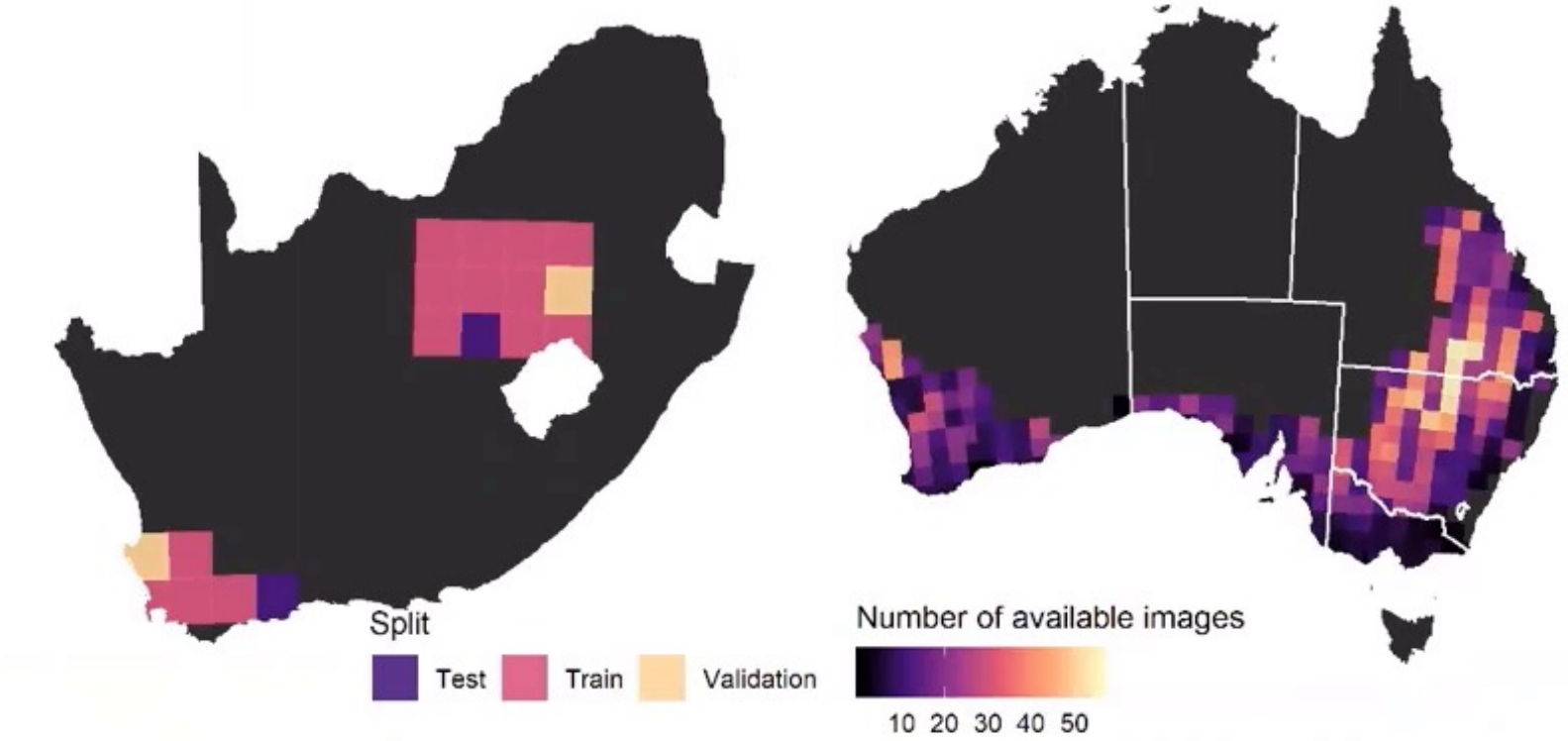
Copernicus SENTINEL-2 Twin Satellites

European **S**pace **A**gency

https://sentinel.esa.int/web/sentinel/missions/sentinel-2/

Extracting field boundaries from satellite images via ML

- ML Training done using a ResUNET model
  - deep layers
  - fewer parameters
- Possible to train using data from one country and then identify field boundaries in another country.



François Waldner, CSIRO Agriculture & Food
https://doi.org/10.1016/j.rse.2020.111741

Field delineation: Software stack

- **Horovod** 0.26.1
  - Python environment based on Python 3.10.8 and featuring TensorFlow 2.11.0.

```
horovod/0.26.1-gpu
```

- **GDAL** 3.6.2
  - The **G**eospatial **D**ata **A**bstraction **L**ibrary is a translator library for raster and vector geospatial data formats.
  - https://gdal.org/

```
gdal/3.6.2-gcc
```

- **Rasterio** 1.2.10
  - Pythonic abstraction of GDAL.
  - https://rasterio.readthedocs.io/en/stable/intro.html

Field delineation: Software stack – Earth Observation framework

- **eo-learn** 0.10.2
  - access and process spatio-temporal image sequences acquired by a satellite fleet
  - https://github.com/sentinel-hub/eo-learn

- **eo-flow** 1.2.0
  - combines Earth Observation data objects with TensorFlow
  - https://github.com/sentinel-hub/eo-flow

- **field-delineation**
  - https://github.com/sentinel-hub/field-delineation
  - custom repo folder provided by user

Field delineation: Full software stack

```
field-delineation
    eo-flow 1.2.0
  eo-learn 0.10.2
  rasterio 1.2.10
```

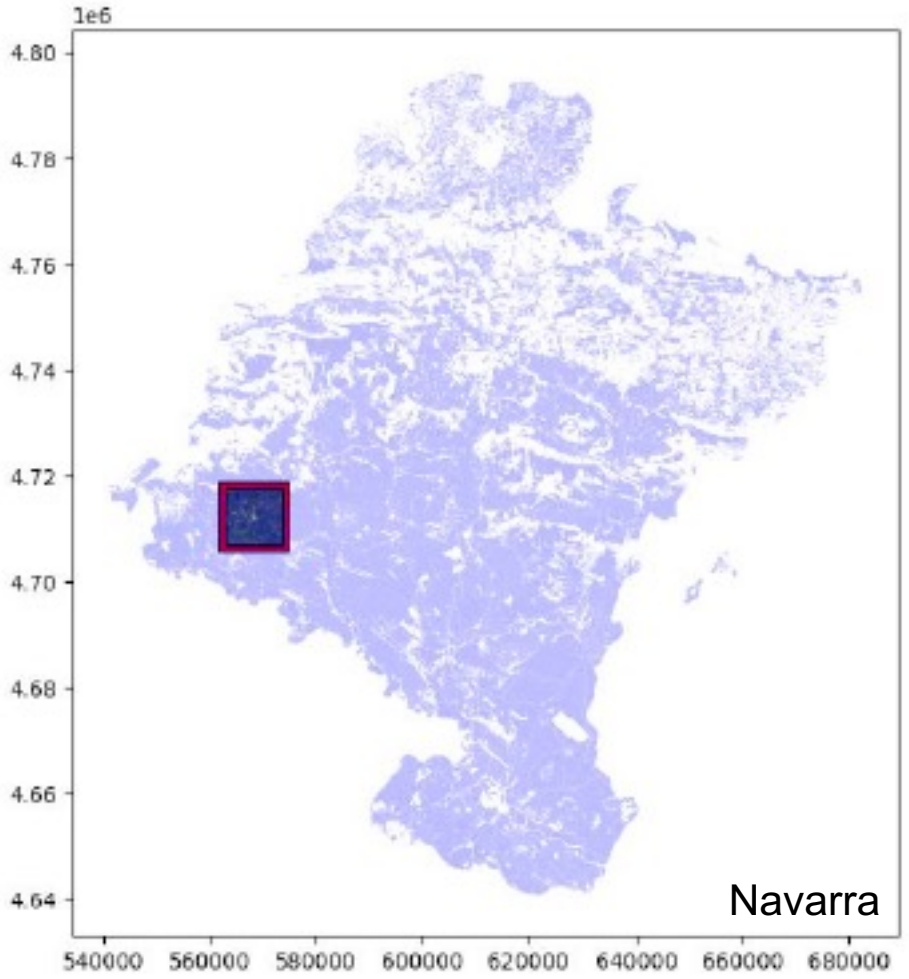
Local virtual environment

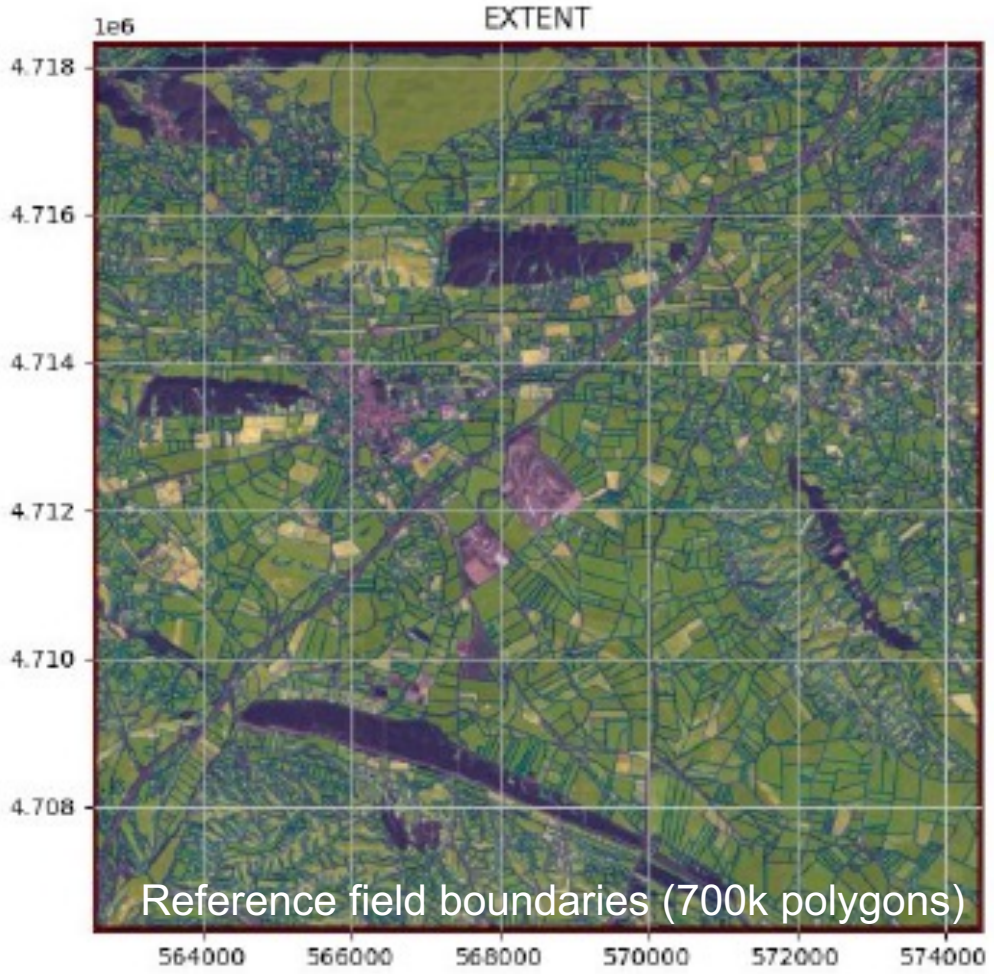
```
gdal/3.6.2

horovod/0.26.1-gpu
python/3.10.8-gpu

openmpi/4.1.4-cuda-11.6

nvidia/nvhpc-no-mpi/22.2
nvidia/cudnn/8.6.0-cuda-11.6
```

Centrally-installed modules

Field delineation: Navarra, Spain

Dr Simon Fraval
Global Academy of Agriculture and Food Security
University of Edinburgh





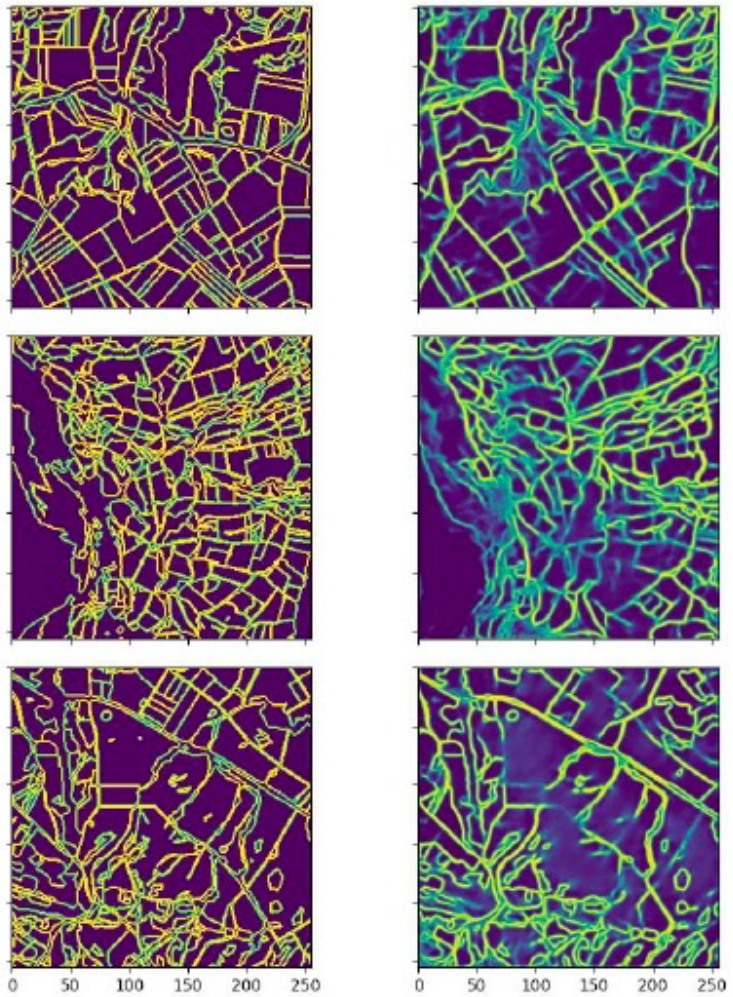
cirrus

epcc

Field delineation: Results

Dr Simon Fraval
Global Academy of Agriculture and Food Security
University of Edinburgh

- Field delineation environment was run on one GPU node using all four GPUs.
- Able to train with higher resolution data (from 10 to 4 m).
- Training took 41 hours (164 GPU hours).
- Achieved validation accuracy of 93% - an improvement of 16% from previous work done on local cluster (NVIDIA Tesla K80).
  - although, model tends to under-segment sub-divisions of larger fields
- Aggregated five million polygons for training, covering regions in Europe, Africa and South East Asia.



Reference Prediction

ARCHER2 - HPE Cray EX Supercomputer

- 5,860 compute nodes (750,080 cores)
  - Dual socket AMD EPYC 7742, 64c, 2.0 GHz
    - 128 cores per node
    - CPU turbo boost available, ≥ 2.25 GHz
  - 256 GB memory per node (2 GB per core)
    - 584 high memory compute nodes (512 GB)

- HPE Cray Slingshot interconnect
  - Two 100 Gbps Slingshot interfaces per node
  - Dragonfly topology

- 14.4 PB ClusterStor L300 Lustre file systems
- 1 PB ClusterStor E1000F solid state storage

https://www.archer2.ac.uk/

archer2

epcc

ARCHER2 - HPE Cray EX Supercomputer

- 5,860 compute nodes (750,080 cores)
  - Dual socket AMD EPYC 7742, 64c, 2.0 GHz
    - 128 cores per node
    - CPU turbo boost available, ≥ 2.25 GHz
  - 256 GB memory per node (2 GB per core)
    - 584 high memory compute nodes (512 GB)
- HPE Cray Slingshot interconnect
  - Two 100 Gbps Slingshot interfaces per node
  - Dragonfly topology
- 14.4 PB ClusterSto
- 1 PB ClusterStor E1

Codes from materials science domain are well used on ARCHER2, e.g., VASP, CP2K, GROMACS, CASTEP, LAMMPS.

https://www.archer2.ac.uk/

archer2

epcc

Python on ARCHER2: Lmod modules

Lmod module files

```
PrgEnv-cray, cce/15.0.0
 PrgEnv-gnu, gcc/11.2.0
PrgEnv-aocc, aocc/3.2.0
```

Cray Programming Environment (CPE) 22.12

```
cray-python/3.9.13.1
```

Built using GCC 11.2.0

Python on ARCHER2: Lmod modules

Lmod module files

```
PrgEnv-cray, cce/15.0.0
 PrgEnv-gnu, gcc/11.2.0
PrgEnv-aocc, aocc/3.2.0
```

CPE 22.12

```
cray-python/3.9.13.1
```

Built using GCC 11.2.0

cray-python/3.9.13.1

dask	2022.2.1
mpi4py	3.1.3
numpy	1.21.5
pandas	1.4.2
scipy	1.6.2
	...

Python on ARCHER2: Lmod modules

Lmod module files

```
PrgEnv-cray, cce/15.0.0
 PrgEnv-gnu, gcc/11.2.0
PrgEnv-aocc, aocc/3.2.0
```

CPE 22.12

```
cray-python/3.9.13.1
```

Built using GCC 11.2.0

cray-python/3.9.13.1

dask	2022.2.1
mpi4py	3.1.3
numpy	1.21.5
pandas	1.4.2
scipy	1.6.2

```
cray-mpich/8.1.23
cray-libsci/22.12.1.1
```

Python on ARCHER2: Virtual environments

```
auser@ln1:~> module load cray-python/3.9.13.1

   MY_VENV_ROOT=${HOME/home/work}/pyenvs/myvenv

   python -m venv --system-site-packages ${MY_VENV_ROOT}

   source ${MY_VENV_ROOT}/bin/activate



(myvenv) auser@ln1:~> python -m pip install <package name>
                      python -m pip install <package name>==<version>

(myvenv) auser@ln1:~> deactivate
auser@ln:~>
```

Python on ARCHER2: Virtual environments for ML

```
tensorflow/2.12.0
pytorch/2.0.0
```

ML Modules

```
auser@ln1:~> module load tensorflow/2.12.0

   MY_VENV_ROOT=${HOME/home/work}/pyenvs/myvenv

   python -m venv --system-site-packages ${MY_VENV_ROOT}

   extend-venv-activate ${MY_VENV_ROOT}

   source ${MY_VENV_ROOT}/bin/activate


(myvenv) auser@ln1:~> python -m pip install <package name>
                      python -m pip install <package name>==<version>


(myvenv) auser@ln1:~> deactivate
auser@ln:~>
```

Python on Cirrus: Local and base packages

${MYVENV_ROOT}/lib/python3.9/site-packages

```
...
dgl
dgl-1.1.1.dist-info
```

/work/y07/shared/python/core/pytorch/2.0.0/python/3.9.13.1/lib/python3.9/site-packages

```
torch
torch-2.0.0+cpu.dist-info
...
```

pytorch/2.0.0

/opt/cray/pe/python/3.9.13.1/lib/python3.9/site-packages

```
...
mpi4py
mpi4py-3.1.3-py3.9.egg-info
...
```

cray-python/3.9.13.1

Python on ARCHER2: Further customisation

${MY_VENV_ROOT}/bin/activate

```
# This file must be used with "source bin/activate" *from bash*
# you cannot run it directly

# *** ADD EXTRA ACTIVATION COMMANDS HERE ***

...

deactivate () {
    ...
    unset VIRTUAL_ENV
    if [ ! "${1:-}" = "nondestructive" ] ; then
    # Self destruct!
        unset -f deactivate
        # *** ADD EXTRA DEACTIVATION COMMANDS HERE ***
    fi
}

...
```

Python on ARCHER2: Running jobs

submit-myvenv.slurm

```
#!/bin/bash

#SBATCH --job-name=myvenv
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=64
#SBATCH --cpus-per-task=2
#SBATCH --time=00:10:00
#SBATCH --account=[budget code]
#SBATCH --partition=standard
#SBATCH --qos=standard

source ${HOME/home/work}/pyenvs/myvenv/bin/activate

export SRUN_CPUS_PER_TASK=${SLURM_CPUS_PER_TASK}

srun --distribution=block:block --hint=nomultithread \
    python myvenv-script.py
```

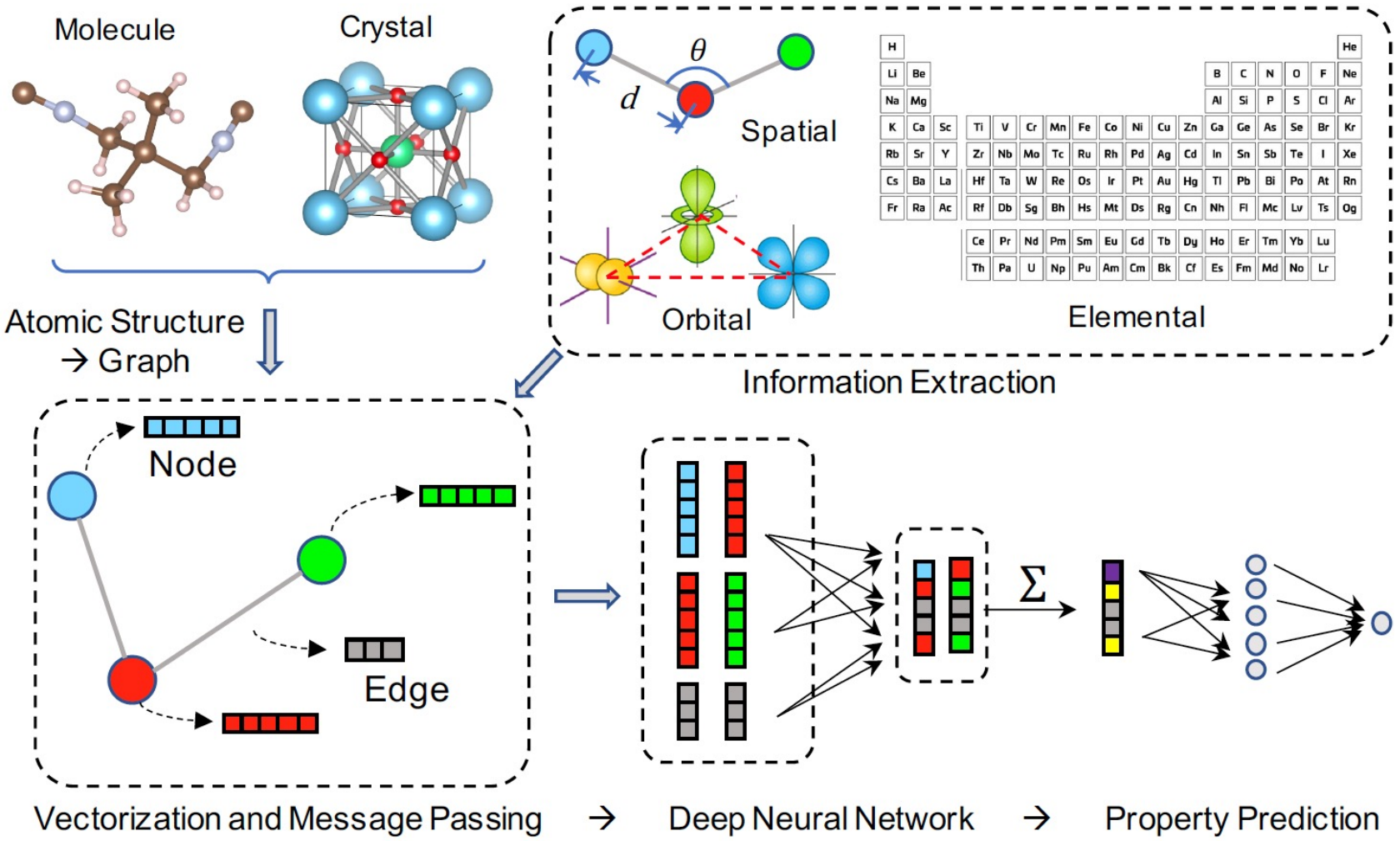
Python on ARCHER2: Running jobs

submit-myvenv.slurm

```
#!/bin/bash

#SBATCH --job-name=myvenv
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=64
#SBATCH --cpus-per-task=2
#SBATCH --time=00:10:00
#SBATCH --account=[budget code]
#SBATCH --partition=standard
#SBATCH --qos=standard

source ${HOME/home/work}/pyenvs/myvenv/bin/activate

export SRUN_CPUS_PER_TASK=${SLURM_CPUS_PER_TASK}

srun --distribution=block:block --hint=nomultithread \
    python myvenv-script.py
```

https://docs.archer2.ac.uk/user-guide/python/#installing-your-own-python-packages-with-pip

Applying Deep Graph Learning to Molecular Graphs



Graph-based deep learning frameworks for molecules and solid-state materials

Weiyi Gong & Qimin Yan, 2021

https://doi.org/10.1016/j.commatsci.2021.110332

DEEP GRAPH LIBRARY

https://www.dgl.ai/

archer2

epcc

Python on ARCHER2: Installing Deep Graph Library (DGL)

https://www.dgl.ai/

```
auser@ln1:~> module load pytorch/2.0.0

   GRAPHER_PYENV_ROOT=${HOME/home/work}/pyenvs/grapher

   python -m venv --system-site-packages ${GRAPHER_PYENV_ROOT}

   extend-venv-activate ${GRAPHER_PYENV_ROOT}

   source ${GRAPHER_PYENV_ROOT}/bin/activate


   python -m pip install dgl -f https://data.dgl.ai/wheels/repo.html

   python -m pip install dglgo -f https://data.dgl.ai/wheels-test/repo.html

   python -m pip install pymatgen torch-geometric
```

Python on ARCHER2: Running a DGL job

submit-grapher.slurm

```
#!/bin/bash

#SBATCH --job-name=grapher
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=128
#SBATCH --cpus-per-task=1
...

source ${HOME/home/work}/pyenvs/grapher/bin/activate

export SRUN_CPUS_PER_TASK=${SLURM_CPUS_PER_TASK}
export MPICH_DPM_DIR=${SLURM_SUBMIT_DIR}/dpmdir

export DGLBACKEND=pytorch
export MPI4PY_FUTURES_MAX_WORKERS=$((SLURM_NTASKS-1))

srun --ntasks=${SLURM_NTASKS} \
     python -m mpi4py.futures ${SLURM_SUBMIT_DIR}/grapher.py
```

Python on ARCHER2: Converting molecular graphs to neural networks

grapher.py

```
#!/usr/bin/env python

...
import dgl
from mpi4py.futures import MPIPoolExecutor
...

def generate(cif_id, args):
  ...

...

if __name__ == '__main__':

  executor = MPIPoolExecutor()
  executor.map(generate, inputs_cif, inputs_args)
  executor.shutdown()
```

Python on ARCHER2: Converting molecular graphs to neural networks

grapher.py

```
#!/usr/bin/env python

...
import dgl
from mpi4py.futures import MPIPoolExecutor
...

def generate(cif_id, args):
  ...

...

if __name__ == '__main__':

  executor = MPIPoolExecutor()
  executor.map(generate, inputs_cif, inputs_args)
  executor.shutdown()
```

- Each molecular graph is stored within a Cystallographic Information Format (CIF) file.
- The **generate** subroutine converts the CIF file to a PyTorch model (PT) file.
  - Uses **pymatgen**, **dgl** and **torch** packages
- PyTorch models could be used to identify molecules or as input to machine learning.

Python on ARCHER2: Converting molecular graphs to neural networks

grapher.py

```
#!/usr/bin/env python

...
import dgl
from mpi4py.futures import MPIPoolExecutor
...

def generate(cif_id, args):
  ...

...

if __name__ == '__main__':

  executor = MPIPoolExecutor()
  executor.map(generate, inputs_c
  executor.shutdown()
```

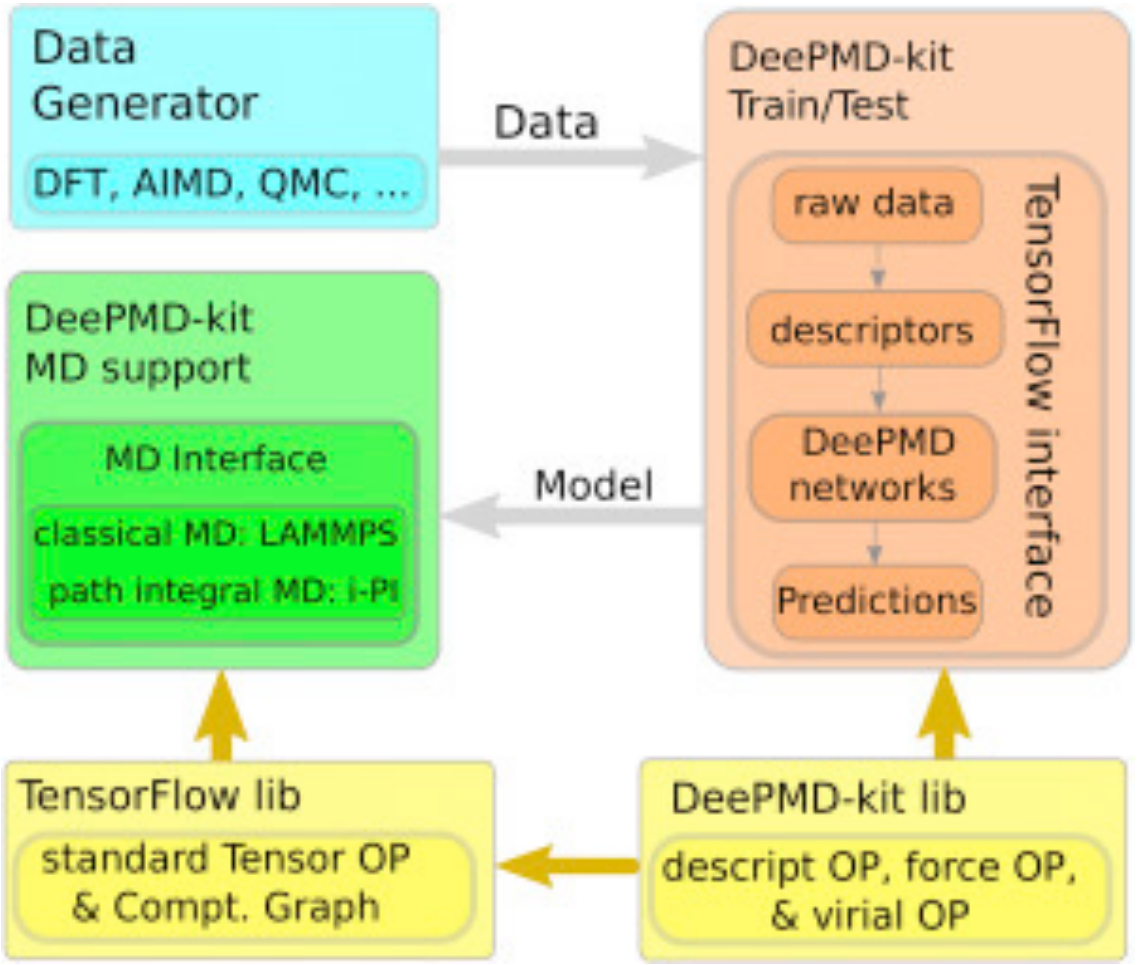
- **`MPIPoolExecutor`** runs a task farm where the work of converting ***n*** CIF files is divided amongst ***m*** workers.
- Use of **`MPIPoolExecutor`** is limited to one ARCHER2 compute node.
- However, can run multiple single-node jobs within a larger job.
  - Using 20 nodes, 50,000 CIF files can be converted to PT files approx. 40 mins.

Python on ARCHER2: Converting molecular graphs to neural networks

grapher.py

```
#!/usr/bin/env python

...
import dgl
from mpi4py import MPI
from mpi4py.futures import MPICommExecutor
...

def generate(cif_id, args):
  ...

...

if __name__ == '__main__':

  with MPICommExecutor() as executor:
    executor.map(generate, inputs_cif, inputs_args)
```

Applying Deep Learning to Molecular Dynamics



DeePMD-kit: A deep learning package for many-body potential energy representation and molecular dynamics

Han Wang, Linfeng Zhang, Jiequn Han, Weinan E., 2018

https://doi.org/10.1016/j.cpc.2018.03.016

DeePMD-kit

https://github.com/deepmodeling/deepmd-kit

https://docs.deepmodeling.com/projects/deepmd/en/master/#

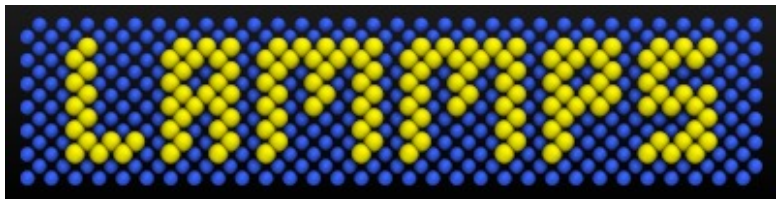
Python on ARCHER2: Installing DeePMD and LAMMPS

```
auser@ln1:~> module load PrgEnv-gnu tensorflow/2.12.0
```

```
${HOME/home/work}/pyenvs/deepmd-lammps

        deepmd

        deepmd-kit

        lammps-stable_23Jun2022_update3
```

Local virtual environment directory structure

https://github.com/deepmodeling/deepmd-kit



https://www.lammps.org/#gsc.tab=0

Python on ARCHER2: Installing DeePMD-kit

Python Interface

```
cd ${DEEPMD_LAMMPS_ROOT}/deepmd-kit

deepmd_source_dir=`pwd`

python -m pip install .
```

C++ Interface

```
cd ${DEEPMD_LAMMPS_ROOT}/deepmd-kit/source/build

deepmd_root=${DEEPMD_LAMMPS_ROOT}/deepmd

cmake .. -D CMAKE_INSTALL_PREFIX=${deepmd_root} \
         -D USE_TF_PYTHON_LIBS=TRUE \
         -D LAMMPS_SOURCE_ROOT=${deepmd_lammps_root} \
         -D MPIEXEC_EXECUTABLE=/usr/bin/srun

make -j 4 install
```

Python on ARCHER2: Installing LAMMPS

```
cd ${deepmd_lammps_root}/build

cmake ../cmake \
      -D CMAKE_CXX_COMPILER=CC \
      -D CMAKE_INSTALL_PREFIX=${deepmd_root} \
      -D CMAKE_INSTALL_LIBDIR=lib \
      -D CMAKE_INSTALL_FULL_LIBDIR=${deepmd_root}/lib \
      -D LAMMPS_INSTALL_RPATH=ON \
      -D MPIEXEC_EXECUTABLE=/usr/bin/srun \
      -D BUILD_MPI=on \
      -D BUILD_SHARED_LIBS=yes \
      -D FFT=FFTW3 \
      -D FFTW3_INCLUDE_DIR=${FFTW_INC} \
      -D FFTW3_LIBRARY=${FFTW_DIR}/libfftw3_mpi.so \
      -D PKG_PLUGIN=ON -D PKG_KSPACE=ON -D PKG_MOLECULE=ON

make -j 4 install
```

Python on ARCHER2: Running a DeePMD-LAMMPS job

submit-deepmd.slurm

```
#!/bin/bash

#SBATCH --job-name=deepmd
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --cpus-per-task=4
...

source ${HOME/home/work}/pyenvs/deepmd-lammps/deepmd/bin/activate

export OMP_NUM_THREADS=1
export OMP_PLACES=cores

export TF_INTRA_OP_PARALLELISM_THREADS=${OMP_NUM_THREADS}

srun --ntasks=${SLURM_NTASKS} lmp -in plugin.in
```

Python on ARCHER2: Running a DeePMD-LAMMPS job

submit-deepmd.slurm

```
#!/bin/bash

#SBATCH --job-name=deepmd
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --cpus-per-task=4
...

srun --ntasks=${SLURM_NTASKS} lmp -in plugin.in
```

plugin.in

```
log log_full.lammps append

plugin load /work/z19/z19/mrb23cab/pyenvs/deepmd-lammps/deepmd/lib/libdeepmd_lmp.so

units metal
atom_style full

...
```

Python on ARCHER2: Running a DeePMD-LAMMPS job

submit-deepmd.slurm

```
#!/bin/bash

#SBATCH --job-name=deepmd
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=32
#SBATCH --cpus-per-task=4
...

source ${HOME/h

export OMP_NUM_
export OMP_PLACES=cores

export TF_INTRA_OP_PARALLELISM_THREADS=${OMP_NUM_THREADS}

srun --ntasks=${SLURM_NTASKS} lmp -in plugin.in
```

For a 300 atom (metal) system, performance is approx. 90k steps per hour
One step is 0.001 ps of simulation time

Python on Cirrus/ARCHER2: Further examples

PyCylon
A Python wrapper for Cylon, a data engineering toolkit designed to work with AI/ML systems and integrate with data processing systems.
https://cylondata.org/

USPEX
Crystal structure prediction
https://uspex-team.org/en

GPAW
A density-functional theory code.
https://wiki.fysik.dtu.dk/gpaw/

EasyVVUQ
A tool for **V**erification, **V**alidation and **U**ncertainty **Q**uantification for a wide variety of simulations.
https://easyvvuq.readthedocs.io/en/dev/

Further Work

- Support users who wish to do multi-node ML runs
  - Have run an ImageNet (ResNet50) benchmark on multiple Cirrus GPU nodes.
    - Achieved 72% parallel efficiency running TensorFlow on 64 GPUs.

- A side effect of installing newer versions of TensorFlow (2.12.0) and PyTorch (2.0.0) on ARCHER2 is that the `numpy` and `scipy` packages are updated.
  - this overrides the `scipy 1.6.2` and `numpy 1.21.5` provided by `cray-python/3.9.13.1` that were built with `cray-libsci/22.12.1.1`.

https://cirrus.readthedocs.io/en/main/user-guide/python.html

https://docs.archer2.ac.uk/user-guide/python