Running AI workloads on IBM Power Systems

Maxime Deloche
Deep Learning Engineer

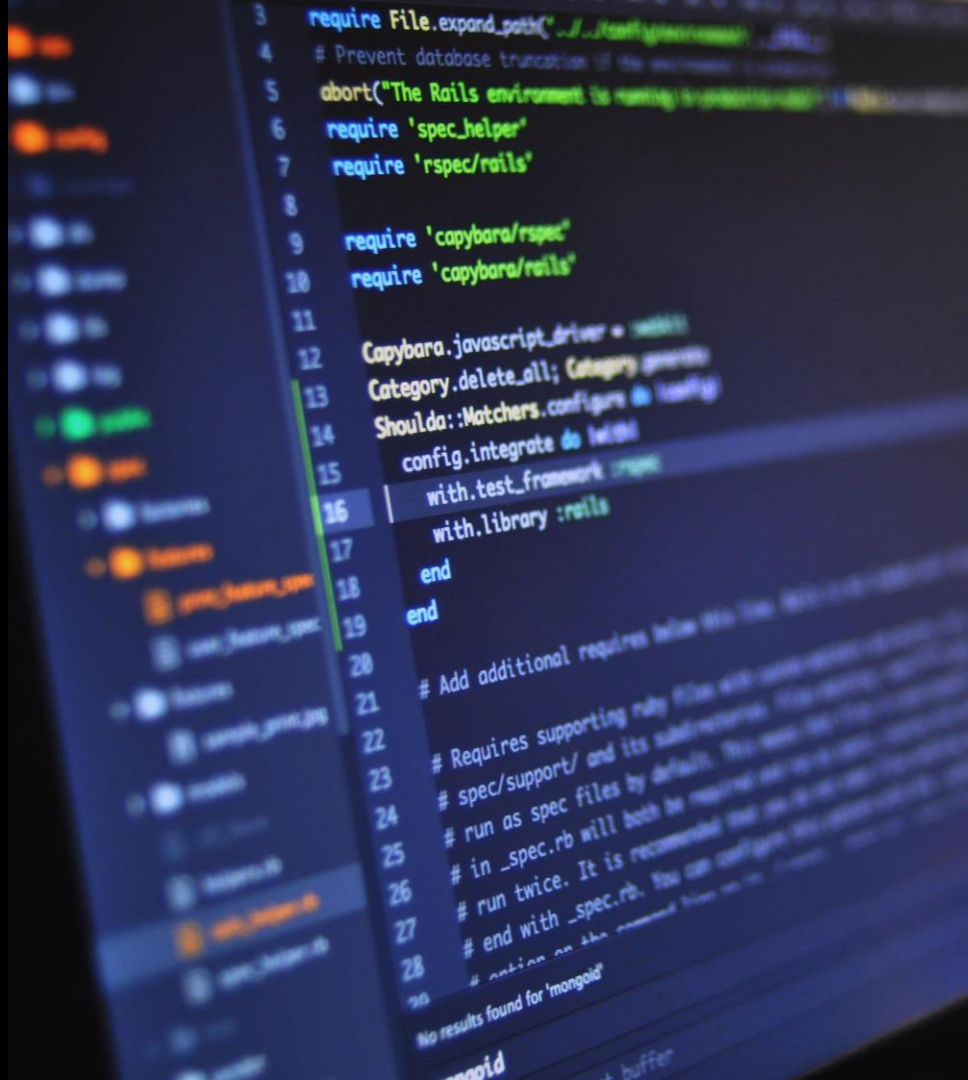
Jean-Armand Broyelle
Cognitive Systems Lab Technical Leader

Cognitive Systems Lab
IBM Garage Montpellier, France

Plan

- **Open Cognitive Environment (OpenCE)**
  - Overview & requirements
  - Release cycle
- **How to use OpenCE**
  - How to build
  - Conda channels
  - Additional repositories
- **Containerization**
  - Advantages
  - Build custom development environments
- **Useful software**
  - Horovod, Dask, scikit-optimize...

Open Cognitive Environment (OpenCE) overview



Open Cognitive Environment (OpenCE)

- Software distribution for AI & Deep Learning applications
- Helps building a complete environment for AI development on Power
- Free and based on open-source SW + optimizations for IBM Power Systems

This toolkit includes…

- Deep learning frameworks
- NVIDIA software: CUDA toolkit, CuDNN, NCCL, DALI, TensorRT…
- Data science libraries: numpy, pandas, scikit-learn, Dask, Horovod, …

OpenCE overview

– https://github.com/open-ce

– Source-to-image project to provide pre-integrated recipes and build scripts

– Packages built to run in a conda environment

– Previously “IBM Watson Machine Learning Community Edition” (WMLCE) and before “IBM PowerAI”

Differences with previous “WMLCE”:

– More flexible “source-to-image” workflow

– Shift from IBM support to community support

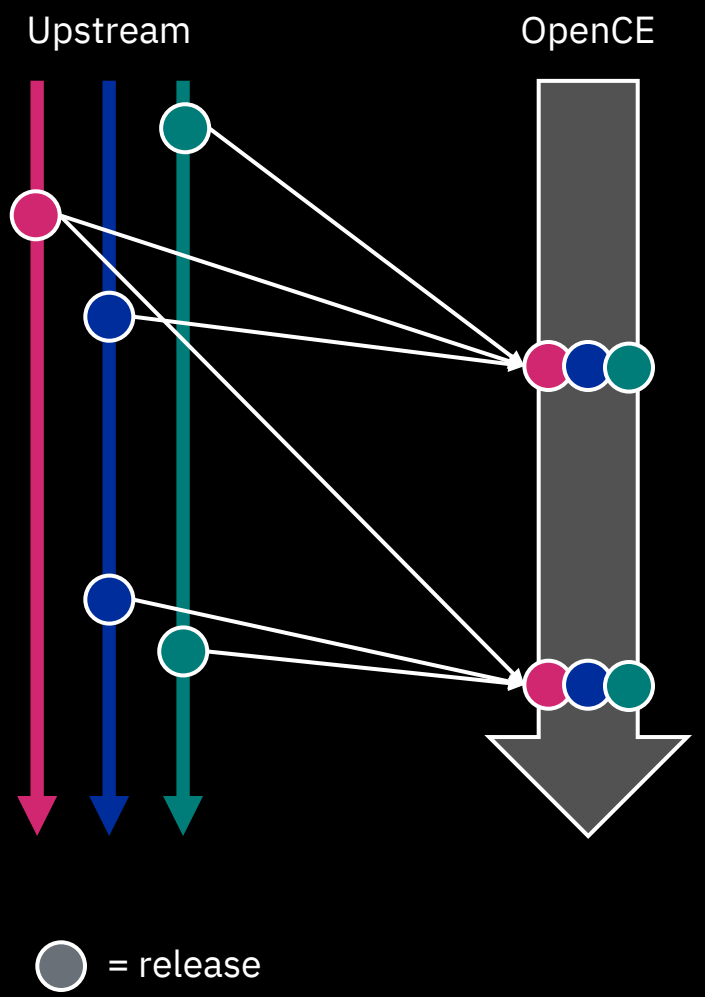
  • Large clusters with active contributors: Oak Ridge National Laboratory “Summit”, MIT “Satori”, Oregon State University…

– Drop support of a few packages and features: Large Model Support (LMS), RAPIDS, SnapML

Release cycle

- Current version is 1.1.3
- Uncoordinated upstream releases
- Rule of thumb: a new release should include a new release of both Tensorflow and Pytorch

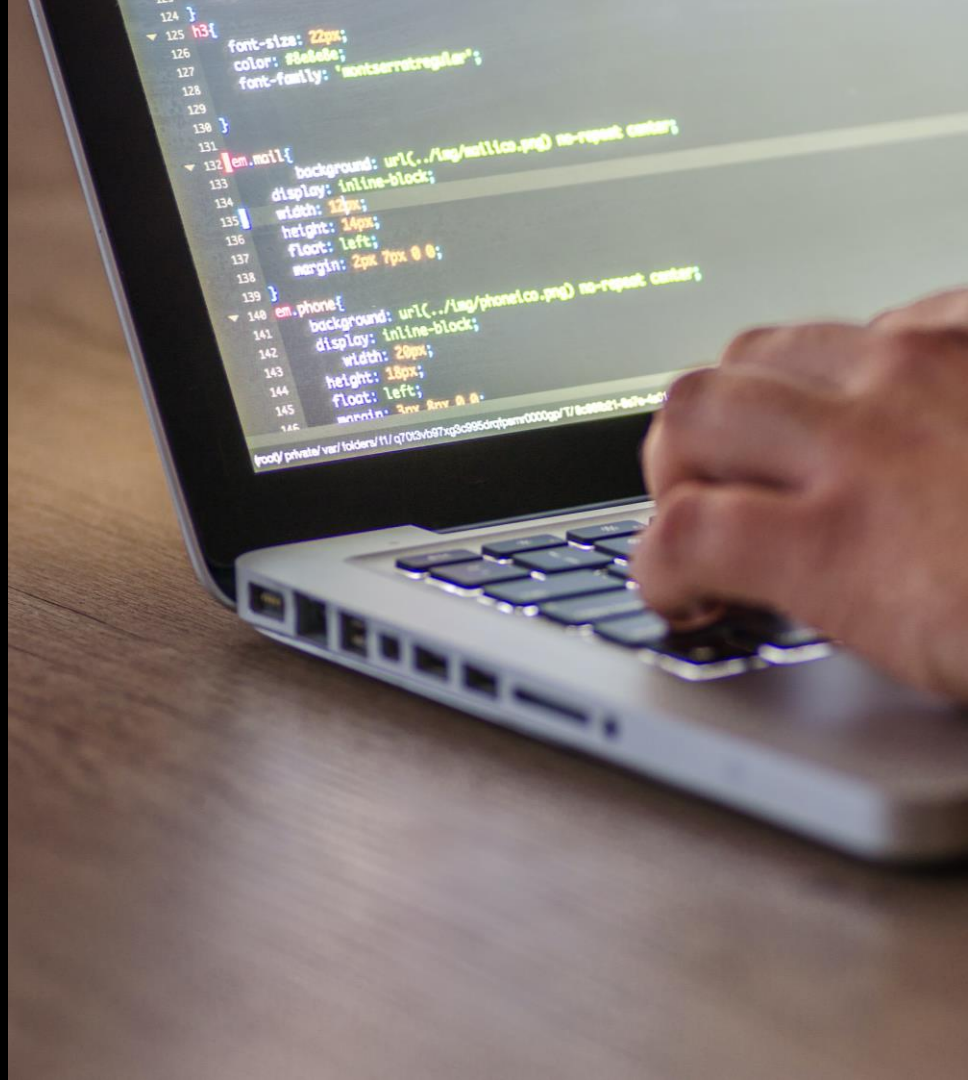
- A release is a set of upstream versions that are guaranteed to work together
- This doesn’t mean that a newer version of an upstream won’t work!



Software requirements

- Operating System
  - RHEL 7.6 or higher
  - Ubuntu 18.04 or higher
- Python >= 3.6
- NVIDIA GPU Drivers v. 440 (when using GPUs)
- Anaconda installer (https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-ppc64le.sh)
  - conda >= 3.8.3 and conda-build >= 3.20.5

How to use OpenCE



OpenCE packaging

Conda is an open-source package manager for multiple languages, not only Python

- Dependencies automatically resolved
- Delivery of packages is continuous
- Conda environments isolate software stacks

Documentation: https://docs.conda.io/projects/conda/en/latest/user-guide/tasks/index.html

Anaconda installer: https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-ppc64le.sh

You need a conda channel. 2 options:

Easy way

- Use an already built, publicly accessible Conda channel of OpenCE: https://opence.mit.edu/
- Built for the IBM Power9 “Satori” cluster at MIT
- Ready to use as conda channel (see Conda section below)
- Useful tips and guides in their documentation: https://mit-satori.github.io/
- Alternative (Oregon State University channel): https://ftp.osuosl.org/pub/open-ce/current/

Alternative

- Build a conda channel of OpenCE
- Cons: you must handle the build and the access to the channel
- Pros: you are autonomous on what packages to build and how often it is updated

https://opence.mit.edu/#/

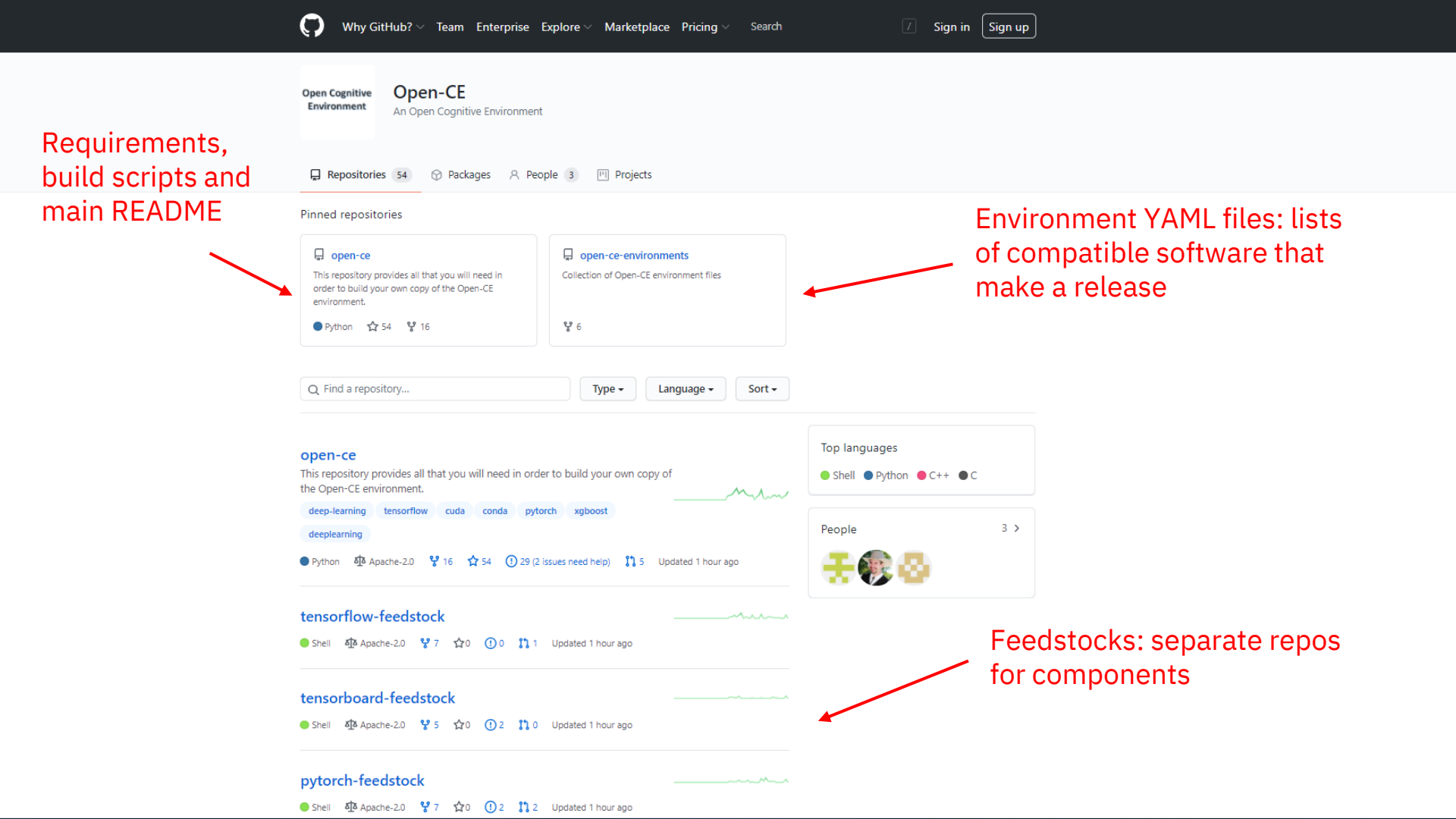
Open Cognitive Environment (OpenCE)

Packages | Documentation

Filters

Search : pytorch

Name	Version	Platform	Summary
_pytorch_select	1.0	linux-ppc64le	Package used to select the specific PyTorch build variant
_pytorch_select	2.0	linux-ppc64le	Package used to select the specific PyTorch build variant
pytorch	1.6.0	linux-ppc64le	Meta-package to install GPU-enabled PyTorch variant
pytorch-base	1.6.0	linux-ppc64le	PyTorch is an optimized tensor library for deep learning using GPUs and CPUs.
pytorch-cpu	1.6.0	linux-ppc64le	Meta-package to install CPU-only PyTorch variant
pytorch	1.7.1	linux-ppc64le	Meta-package to install GPU-enabled PyTorch variant
pytorch-base	1.7.1	linux-ppc64le	PyTorch is an optimized tensor library for deep learning using GPUs and CPUs.
pytorch-cpu	1.7.1	linux-ppc64le	Meta-package to install CPU-only PyTorch variant
pytorch-lightning	0.9.0	noarch	PyTorch Lightning is the lightweight PyTorch wrapper for ML researchers. Scale your models. Write less boilerplate.
pytorch-lightning	1.1.0	noarch	PyTorch Lightning is the lightweight PyTorch wrapper for ML researchers. Scale your models. Write less boilerplate.
pytorch-lightning-bolts	0.2.5	noarch	Pretrained SOTA Deep Learning models, callbacks and more for research and production with PyTorch Lightning and PyTorch.

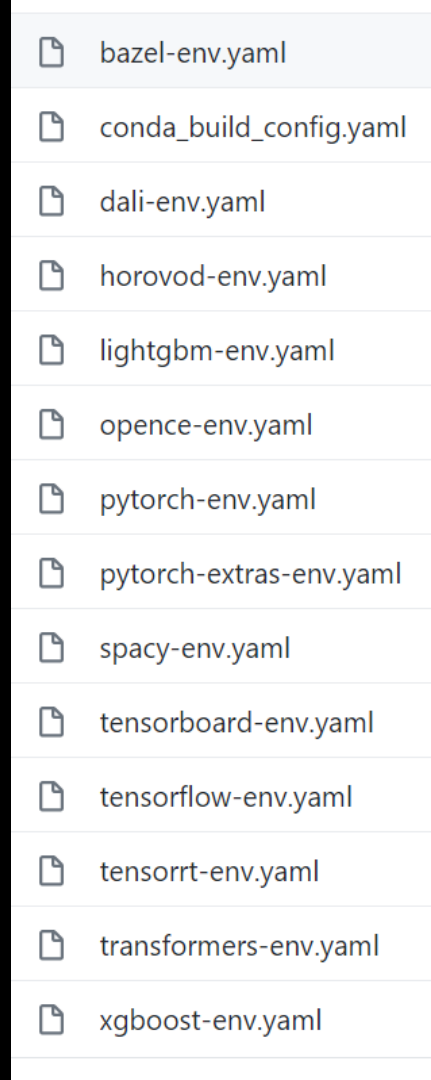

Why GitHub?
Team
Enterprise
Explore
Marketplace
Pricing
Search
Sign in
Sign up
Open Cognitive Environment
Open-CE
An Open Cognitive Environment
Repositories 54
Packages
People 3
Projects
Pinned repositories
open-ce
This repository provides all that you will need in order to build your own copy of the Open-CE environment.
Python 54 16
open-ce-environments
Collection of Open-CE environment files
6
Find a repository...
Type
Language
Sort
open-ce
This repository provides all that you will need in order to build your own copy of the Open-CE environment.
deep-learning tensorflow cuda conda pytorch xgboost deeplearning
Python Apache-2.0 16 54 29 (2 issues need help) 5 Updated 1 hour ago
tensorflow-feedstock
Shell Apache-2.0 7 0 0 1 Updated 1 hour ago
tensorboard-feedstock
Shell Apache-2.0 5 0 2 0 Updated 1 hour ago
pytorch-feedstock
Shell Apache-2.0 7 0 2 2 Updated 1 hour ago
Top languages
Shell Python C++ C
People 3
Requirements, build scripts and main README
Environment YAML files: lists of compatible software that make a release
Feedstocks: separate repos for components

How to build OpenCE (1/2)

- Install requirements (including CUDA if GPUs)
- Git clone the “open-ce” repo
- Pick the environment you want to build from the “open-ce-environments” repo
- Run the “open-ce build env...” command

```
$ git clone https://github.com/open-ce/open-ce.git
$ ./open-ce/open_ce/open-ce build env \
        https://raw.githubusercontent.com/open-ce/\
        open-ce-environments/main/envs/opence-env.yaml
```

- Full “OpenCE” build requires 10 to 15 hours ; and a lot less for smaller YAML envs.



How to build OpenCE (2/2)

- Previous build produces a “./condabuild” dir

```
$ conda install -c ./condabuild \
        numpy tensorflow ...
```

- You can possibly move that directory to a web server and make it available to your users (the same way the MIT OpenCE conda channel works)

- Possibility to build single feedstocks

```
$ open-ce build feedstock ...
```

- Possibility to run tests after a build (defined for each package in their feedstock repo)

```
$ open-ce test env ...
```

- Possibility to build inside a container (clean environment for the build and makes builds more “system independent”)

```
$ open-ce build env \
        --container_build ...
```

Configuration and use

or replace with your own Conda channel

– Add OpenCE to the conda channels (or modify directly “~/.condarc” file):

```
$ conda config --prepend channels http://opence.mit.edu/
```

– List installed packages

```
$ conda list
```

– Search a package

```
$ conda search numpy
```

– Install a package (optionally specifying version)

```
$ conda install numpy=1.17.4
```

– Uninstall a package

```
$ conda remove numpy
```

GPUs and CUDA

- GPU drivers (v. 440 or higher) need to be installed (https://www.nvidia.com/Download/index.aspx)

- CUDA runtime package is available in the Anaconda default channel: “cudatoolkit” (version 10.2 or 11.0)
  - Package link: https://anaconda.org/anaconda/cudatoolkit

- It includes everything needed by GPU-accelerated apps:
  - GPU-accelerated libraries, CUDA runtime, compiler, profiling tools…

Manage Conda environments

Easy to create, install and switch between software stacks!

– List environments

```
$ conda env list
```

– Create an environment

```
$ conda activate my_env
```

– Switch between environments

```
$ conda create -n my_env python=3.7
```

– Conda environments allow to have multiple software stacks:

- Multiple Python versions
- Different frameworks or versions

```
(my_env) mdeloche-14-8sq4l @pwrai ~/documents >
```

Active environment is displayed at the beginning of your prompt.

```
(opence) mdeloche-14-8sq4l @pwrai ~/documents > conda info

     active environment : opence
    active env location : /opt/anaconda/envs/opence
            shell level : 1
       user config file : /home/pwrai/.condarc
 populated config files : /home/pwrai/.condarc
          conda version : 4.7.12
    conda-build version : not installed
         python version : 3.7.7.final.0
       virtual packages : __cuda=10.2
       base environment : /opt/anaconda  (writable)
           channel URLs : https://opence.mit.edu/linux-ppc64le
                          https://opence.mit.edu/noarch
                          https://repo.anaconda.com/pkgs/main/linux-ppc64le
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/r/linux-ppc64le
                          https://repo.anaconda.com/pkgs/r/noarch
          package cache : /opt/anaconda/pkgs
                          /home/pwrai/.conda/pkgs
       envs directories : /opt/anaconda/envs
                          /home/pwrai/.conda/envs
                platform : linux-ppc64le
              user-agent : conda/4.7.12 requests/2.23.0 CPython/3.7.7 Linux/4.14.0-115.18.1.el7a.ppc64le ubuntu/18.04.4 glibc/2.27
                 UID:GID : 2051:2051
              netrc file : None
            offline mode : False

(opence) mdeloche-14-8sq4l @pwrai ~/documents >
```

Additional repositories

- Supplementary channel (useful packages that are not part of OpenCE):
  - https://anaconda.org/powerai

- “old” WMLCE main channel:
  - https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda/

- “old” WMLCE early-access channel:
  - https://public.dhe.ibm.com/ibmdl/export/pub/software/server/ibm-ai/conda-early-access/

If your package is still not available?

- Try pip:
  - Install pip with conda, and search for your package
  - If pip finds the package:
    - If there is a binary built for ppc64le, it will install it and dependencies
    - If not, it will try to install it using its “setup.py” file (see setuptools)

Warning	This conflicts if the same package is installed in Conda: only use it when you can’t find it in the various Conda repositories!

- If pip compilation fails, try to recompile the package manually

Hint	Build a **wheel file** (self-contained pip package) that you can then save and re-use on any ppc64le system (and possibly add it to your built Conda channel).

Air-gapped systems

- Local channel needed, not only with packages from OpenCE but also standard packages from main repos
- Solution: think ahead about your stack and create a “custom” conda channel

1. On an internet-facing system, clean the cache

```
$ conda clean --all
```

2. Build an environment will all packages you need ; and locate the cache

```
$ conda create -n my_env PACKAGES
$ conda info | grep “package cache”
```

3. Copy all “.tar.bz2” and “.conda” files from the cache to “MY_CHANNEL/noarch/” on the air-gapped system
4. Install Anaconda or your air-gapped system and build a local conda channel from it (this creates metadata files needed by conda)

```
$ conda index MY_CHANNEL/
```

5. Install using that directory, or add it to your conda configured channels

```
$ conda install –c ./MY_CHANNEL/ \
        PACKAGES
```

```
$ conda config --prepend channels \
        file://path/to/MY_CHANNEL/
```

Containerization

Container images

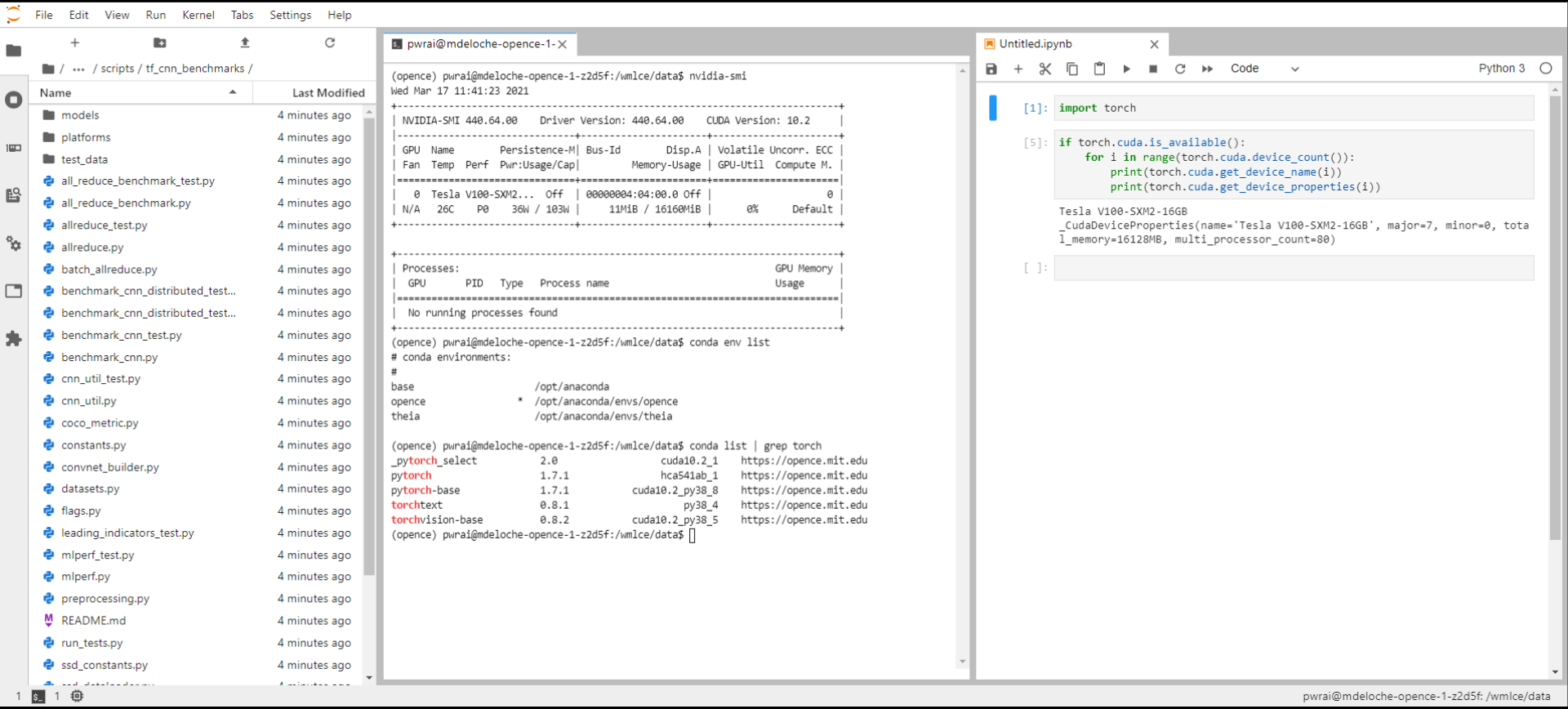
- Advantages of working in containers:
  - Isolation of users on your system
  - Resources limits (GPUs in particular)
  - Custom software stack for each container
- Can run on a container orchestrator (Kubernetes/Red Hat Openshift) or standalone using Docker commands

- No (known) OpenCE available images yet
- Previous “WMLCE” IBM images: https://hub.docker.com/r/ibmcom/powerai
- The image we built for our customers (that you can freely modify and reuse): https://gitlab.com/PSLC/wmlce-mop

Your own minimal image

```
FROM ubuntu:20.04

ENV PATH /opt/conda/bin:$PATH


RUN apt-get update -y && apt-get install -y wget


RUN wget --quiet \
        https://repo.anaconda.com/archive/Anaconda3-2020.11-Linux-ppc64le.sh && \
        sh ./Anaconda3-2020.11-Linux-ppc64le.sh -b -p /opt/conda && \
        conda config --prepend https://opence.mit.edu/ && \
        conda create -n opence python=3.8 tensorflow pytorch

# Add additional libs, conda environments, GUI, SSH service...


CMD ["bash", "-i"]
```

JupyterLab (1/2)



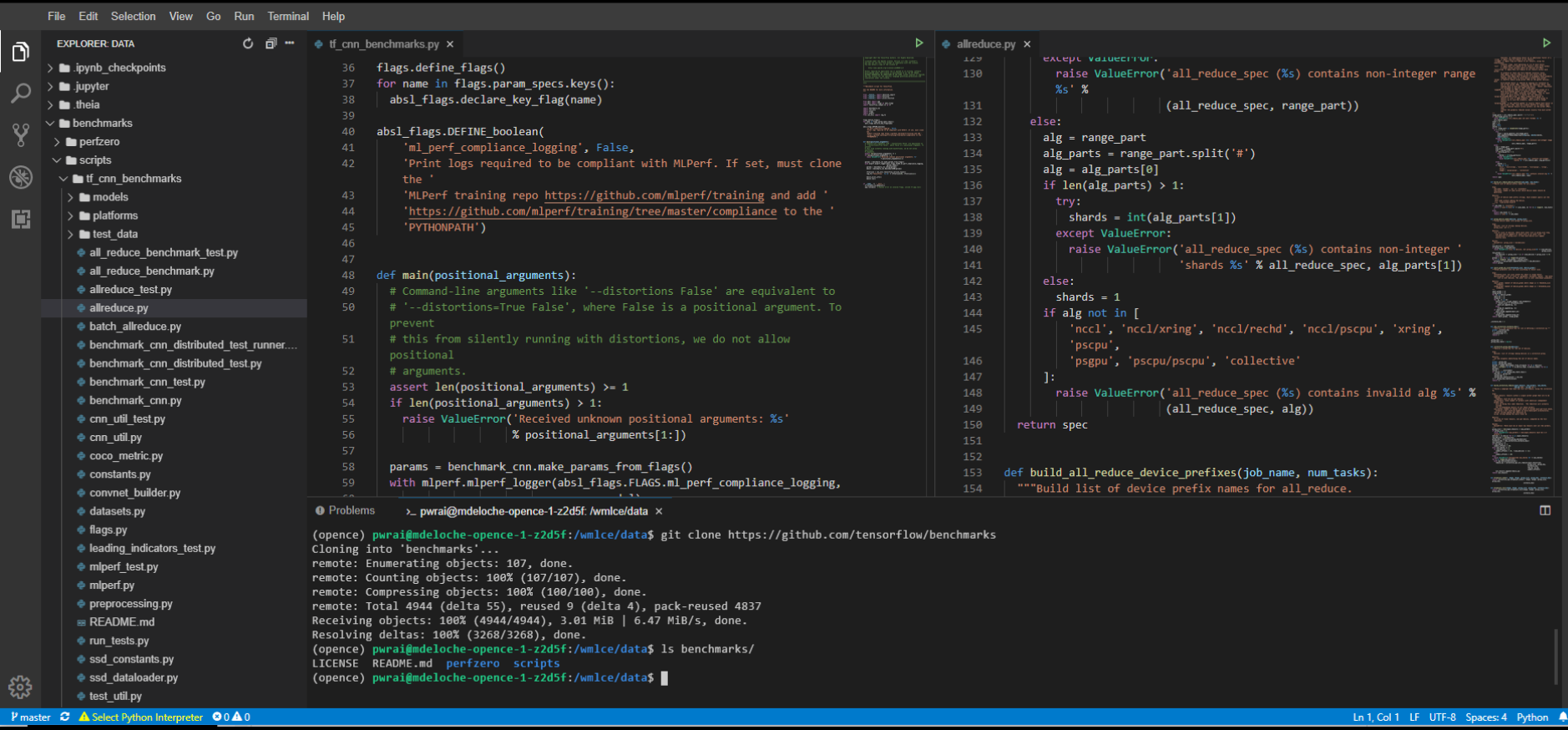
JupyterLab (2/2)

Source: https://github.com/jupyterlab/jupyterlab

Doc: https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html

```
RUN conda run -n opence conda install -y nodejs jupyterlab


CMD [“/opt/anaconda/envs/opence/bin/jupyter”, \
        “lab”, \
        “--ip=$(hostname)”, \
        “--no-browser”, \
        “--notebook-dir=/home/user/”]
```

Theia (1/2)


File Edit Selection View Go Run Terminal Help
EXPLORER: DATA
.ipynb_checkpoints
.jupyter
.theia
benchmarks
perfzero
scripts
tf_cnn_benchmarks
models
platforms
test_data
all_reduce_benchmark_test.py
all_reduce_benchmark.py
allreduce_test.py
allreduce.py
batch_allreduce.py
benchmark_cnn_distributed_test_runner....
benchmark_cnn_distributed_test.py
benchmark_cnn_test.py
benchmark_cnn.py
cnn_util_test.py
cnn_util.py
coco_metric.py
constants.py
convnet_builder.py
datasets.py
flags.py
leading_indicators_test.py
mlperf_test.py
mlperf.py
preprocessing.py
README.md
run_tests.py
ssd_constants.py
ssd_dataloader.py
test_util.py
tf_cnn_benchmarks.py
flags.define_flags()
for name in flags.param_specs.keys():
absl_flags.declare_key_flag(name)
absl_flags.DEFINE_boolean(
'ml_perf_compliance_logging', False,
'Print logs required to be compliant with MLPerf. If set, must clone the '
'MLPerf training repo https://github.com/mlperf/training and add '
'https://github.com/mlperf/training/tree/master/compliance to the '
'PYTHONPATH')
def main(positional_arguments):
# Command-line arguments like '--distortions False' are equivalent to
# '--distortions=True False', where False is a positional argument. To prevent
# this from silently running with distortions, we do not allow positional
# arguments.
assert len(positional_arguments) >= 1
if len(positional_arguments) > 1:
raise ValueError('Received unknown positional arguments: %s'
% positional_arguments[1:])
params = benchmark_cnn.make_params_from_flags()
with mlperf.mlperf_logger(absl_flags.FLAGS.ml_perf_compliance_logging,
allreduce.py
raise ValueError('all_reduce_spec (%s) contains non-integer range %s' %
(all_reduce_spec, range_part))
else:
alg = range_part
alg_parts = range_part.split('#')
alg = alg_parts[0]
if len(alg_parts) > 1:
try:
shards = int(alg_parts[1])
except ValueError:
raise ValueError('all_reduce_spec (%s) contains non-integer '
'shards %s' % all_reduce_spec, alg_parts[1])
else:
shards = 1
if alg not in [
'nccl', 'nccl/xring', 'nccl/rechd', 'nccl/pscpu', 'xring',
'pscpu',
'psgpu', 'pscpu/pscpu', 'collective'
]:
raise ValueError('all_reduce_spec (%s) contains invalid alg %s' %
(all_reduce_spec, alg))
return spec
def build_all_reduce_device_prefixes(job_name, num_tasks):
"""Build list of device prefix names for all_reduce.
Problems
pwrai@mdeloche-opence-1-z2d5f: /wmlce/data
(opence) pwrai@mdeloche-opence-1-z2d5f:/wmlce/data$ git clone https://github.com/tensorflow/benchmarks
Cloning into 'benchmarks'...
remote: Enumerating objects: 107, done.
remote: Counting objects: 100% (107/107), done.
remote: Compressing objects: 100% (100/100), done.
remote: Total 4944 (delta 55), reused 9 (delta 4), pack-reused 4837
Receiving objects: 100% (4944/4944), 3.01 MiB | 6.47 MiB/s, done.
Resolving deltas: 100% (3268/3268), done.
(opence) pwrai@mdeloche-opence-1-z2d5f:/wmlce/data$ ls benchmarks/
LICENSE README.md perfzero scripts
(opence) pwrai@mdeloche-opence-1-z2d5f:/wmlce/data$
master
Select Python Interpreter
Ln 1, Col 1
LF
UTF-8
Spaces: 4
Python

Theia (2/2)

Doc: https://theia-ide.org/docs/

```
COPY theia-package.json /tmp/theia-install/package.json

RUN cd /tmp/theia-install && \
        conda create -y -n theia python=2 nodejs && \
        conda run -n theia npm install -g yarn && \
        conda run -n theia yarn && \
        conda run -n theia yarn theia build

CMD [“conda”, “run”, “-n”, “theia”, \
        “yarn”, “theia”, “start”, “--hostname“, “127.0.0.1”]
```

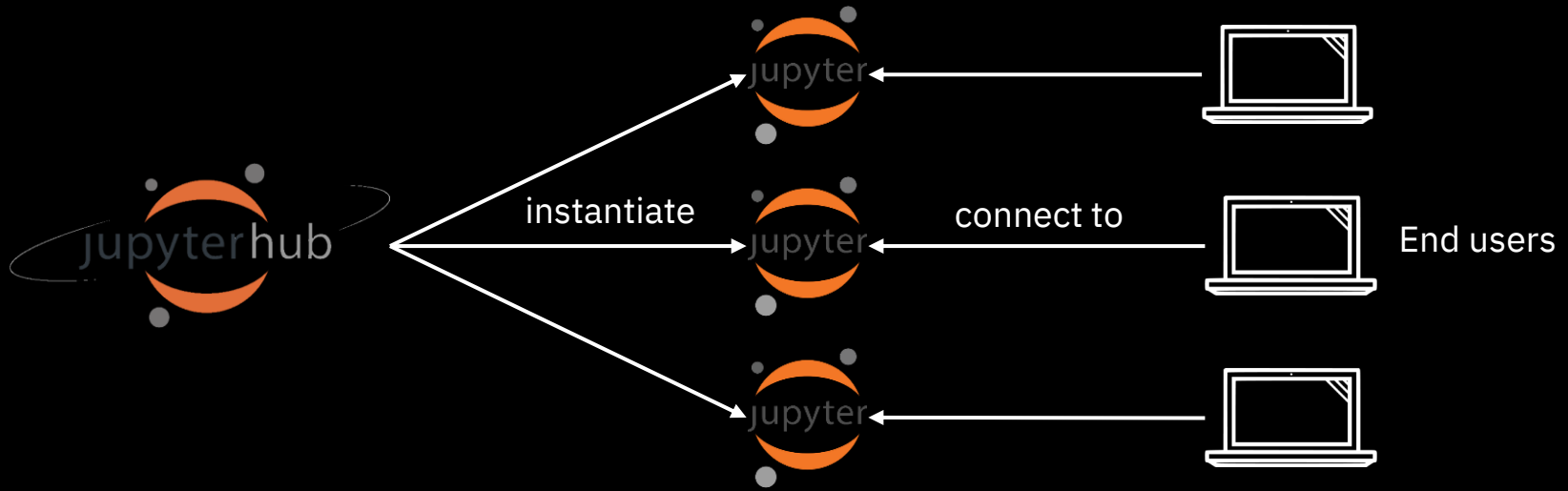
Containerization - Final thoughts (1/2)

- Those are 2 examples among many web-based IDEs. You can take advantage of them even when not running in containers.
- Containers allow to keep the host environment clean and stable, and still let the cluster users customize their work environment and tailor them to their development needs.

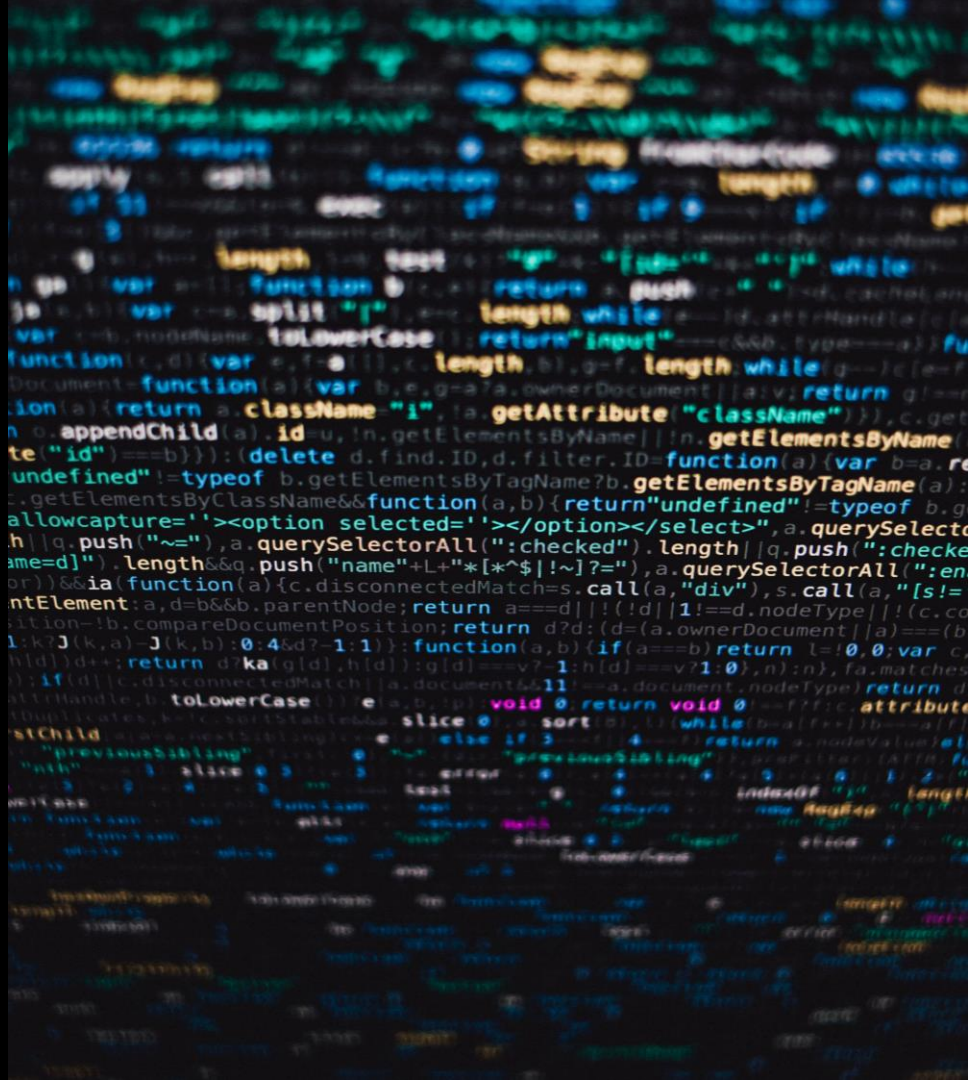
- When building containers, you can use CI/CD pipelines (Github Actions, Gitlab CI/CD, ...) to simplify the development workflow: by automating the build, test and push to a registry steps.

Containerization - Final thoughts (2/2)

- JupyterHub (https://github.com/jupyterhub/jupyterhub) is multi-user hub of Jupyter Notebooks
  - Spawns and manages multiple instances of single-user environments
  - You can bring in your own Docker image!
    (see https://jupyterhub-dockerspawner.readthedocs.io/en/latest/)



Useful software



Horovod (1/3)

- Distributed deep learning training framework
- Interfaces with Tensorflow, Keras, Pytorch and MXNet

- Based for underlying communications on:
  - Use MPI (Message Passing Interface) to communicate on the control plane
  - Use Nvidia NCCL to communicate on the data plane (MPI-compatible library with operations optimized for GPUs, especially “allreduce”)

- Latest version 0.21.0 included in OpenCE, as well as NCCL and OpenMPI

Horovod (2/3)

Keras sample code

```
# initialize Horovod
horovod.init()

# pin GPU to be used to process local rank (one GPU per process)
config.gpu_options.visible_device_list = str(horovod.local_rank())

# define dataset and model
train_dataset = ...
model = Sequential().add(...)

# create a distributed optimizer
opt = horovod.DistributedOptimizer(keras.optimizers.Adam())

# compile the Keras model
model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=opt, metrics=['accuracy’])

# train the model
# Callback broadcasts initial variable states from rank 0 to all other processes.
# This is necessary to ensure consistent initialization of all workers.
model.fit(train_dataset, callbacks=[horovod.callbacks.BroadcastGlobalVariablesCallback(0)])
```

Horovod (3/3)

Horovod defines a “horovodrun” command that is a wrapper to “mpirun” calls.

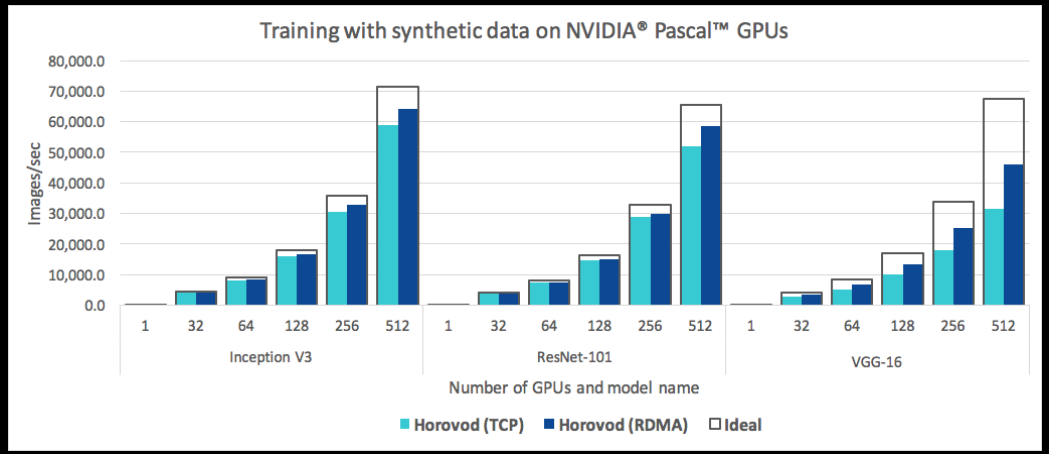
– Run on single host with 4 GPUs:

```
$ horovodrun -np 4 -H localhost:4 python train.py
```

– Run on 3 hosts with 4 GPUs each:

```
$ horovodrun -np 12 -H server1:4,server2:4,server3:4 python train.py
```

– Benchmarks shows a scaling of ~90% on 128 4-GPU servers

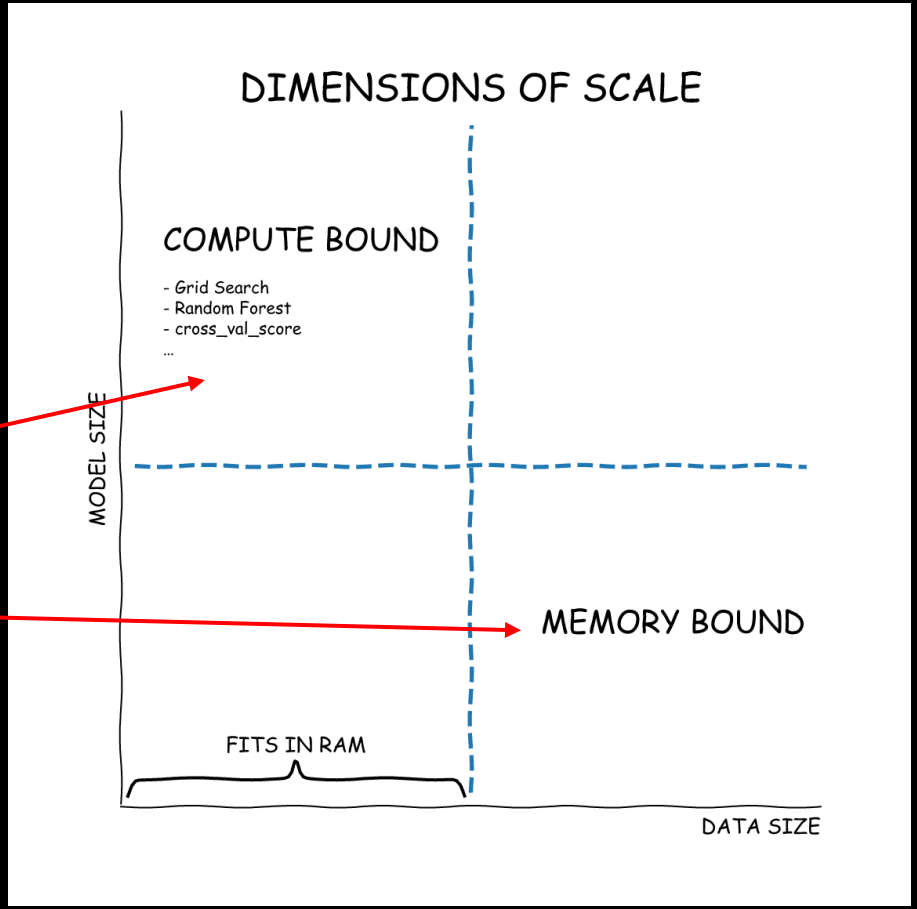


Dask (1/2)

- Parallel computing library for Python – scale from a laptop to a cluster with the same familiar API

- 2 components:
  - Distributed task scheduler
  - Dask Collections optimized for distributed environments and larger-than-memory data

- Great doc: https://docs.dask.org/en/latest/



Dask (2/2)

– Familiar programming interfaces:

```
import dask.dataframe as dd # pandas-like

df = dd.read_csv(“*.csv”)
df.groupby(df.user_id).value.mean().compute()
```

```
import dask.array as da # numpy-like
f = h5py.File(“myfile.hdf5”)
x = da.from_array(f['/big-data’],
          chunks=(1000, 1000))
x - x.mean(axis=1).compute()
```

– Lazy evaluation: build a graph of operations that is triggered by calling “compute()”

– Dask-ML extends scikit-learn with the same API

  • Can also be used with Pytorch, Keras, and Tensorflow using packages that bring a Scikit-learn like API to these frameworks (Skorch, SciKeras…)

– See https://ml.dask.org/index.html

Scikit-optimize (1/3)

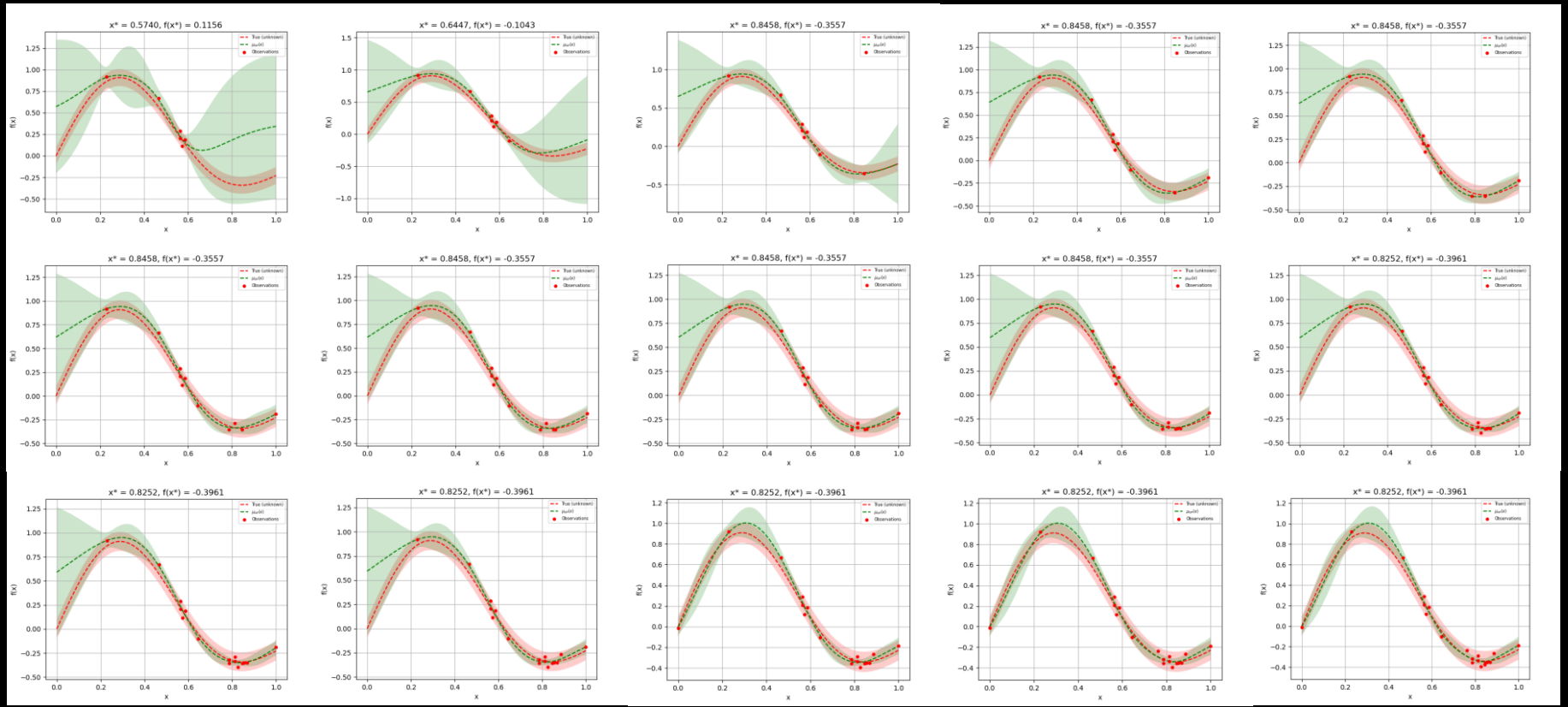
- Hyper-parameter optimization library
- Based on scikit-learn but not limited to it, it can optimize any function with parameters

- Embeds Bayesian Optimization algorithm
  - Black box function (no gradients)
  - Expensive to evaluate
  - Observations are noisy
- Typically a neural network training

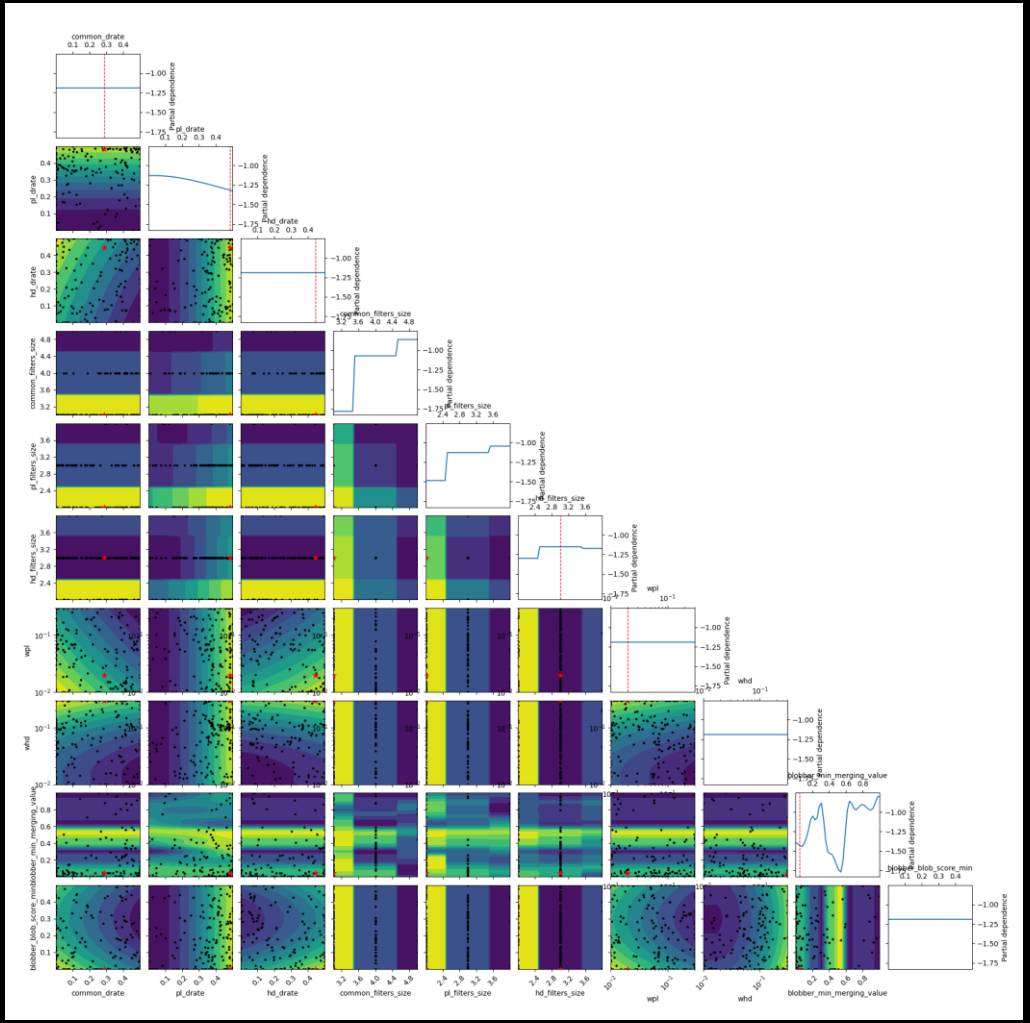
- To be installed with ‘pip’

```
import skopt
import matplotlib.pyplot as plt
import numpy as np

DIMENSIONS = [
    skopt.space.Real(0.0, 1.0, name="x")
]

def f(x, noise_level=0.1):
    return np.sin(x[0]*5) * \
        (1-np.tanh(x[0]**2)) \
        + np.random.randn() * noise_level


opt = skopt.Optimizer(DIMENSIONS)

for i in range(20):
    x = opt.ask()
    y = f(x)
    result = opt.tell(x, y)
print(result)
```

Scikit-optimize (2/3)



Scikit-optimize (3/3)



Thank you!
Questions?