OAK RIDGE National Laboratory

Introduction to Open-CE on Summit

Junqi Yin

Analytics/AI Methods at Scale (AAIMS)

Oak Ridge Leadership Computing Facility

ORNL is managed by UT-Battelle, LLC for the US Department of Energy

U.S. DEPARTMENT OF ENERGY

Outline

- What's Open-CE ?
- How's Open-CE deployed on Summit?
- Customize your own environment
- Performance baselines
- Documentations

IBM Watson Machine Learning Community Edition (WML-CE)

module load ibm-wml-ce

Multi-node Multi-GPU ML/DL

Machine learning

RAPIDS

cuDF cuML

Dask, CuPy, XGBoost

Snap ML

- Logistic Regression
- Random Forest
- Decision Tree
- SVM
- Ridge Regression
- Lasso Regression

Deep learning

TensorFlow
TensorFlow Probability
TensorBoard
TensorFlow-Keras

PyTorch

Torchtext
Torchvision

Frameworks

IBM LMS

DALI

Apex

TensorRT

Plugins and Tools

Open Cognitive Environment (Open-CE)

- IBM WML-CE is deprecated after v1.7.0.
- IBM released most conda receipts for WML-CE as an open-source project – Open-CE
- ibm-wml-ce modules are still available on Summit
- open-ce module provides more up-to-date DL frameworks

Open-CE

module load open-ce

Multi-node Multi-GPU ML/DL

Deep learning

Frameworks

Plugins and Tools

Machine learning

py-XGBoost

(Rapids will be deployed as a separated module)

IBM-WML-CE vs Open-CE

Environment	ibm-wml-ce/1.6.1	ibm-wml-ce/1.7.0	open-ce/0.1
Package	IBM DDL 1.5.0	IBM DDL 1.5.1	
	Tensorflow 1.15	Tensorflow 2.1	Tensorflow 2.3
	Pytorch 1.2.0	Pytorch 1.3.1	Pytorch 1.6.0
	Caffe(IBM-enhanced) 1.0.0	Caffe (IBM-enhanced) 1.0.0	
	Horovod v0.18.2 (IBM-DDL Backend)	Horovod v0.19 (NCCL Backend)	Horovod v0.19.5 (NCCL Backend)
Complete List	1.6.2 Software Packages	1.7.0 Software Packages	Open-CE Software Packages

Open-CE deployment on Summit

- Build opence env

```
./opence/opence build env envs/opence-env.yaml
                          --build_types cuda
                          --output_folder condabuild
```

- Build single package

```
./open-ce build feedstock
--working_directory horovod-feedstock
```

- Summit channel

```
OPENCE_CHANNEL=/sw/sources/open-ce/conda-channel
```

envs/opence-env.yaml:

```
imported_envs:
  - pytorch-env.yaml
  - tensorflow-env.yaml
  - xgboost-env.yaml
  - dali-env.yaml  #[build_type == 'cuda']
  - spacy-env.yaml
  - transformers-env.yaml
  - horovod-env.yaml #[build_type == 'cuda']
```

horovod-feedstock:

```
diff --git a/recipe/meta.yaml b/recipe/meta.yaml
index ffc5140..9b675cf 100644
--- a/recipe/meta.yaml
+++ b/recipe/meta.yaml
@@ -28,7 +28,6 @@ requirements:
-    - openmpi        {{ openmpi     }}
     - nccl           {{ nccl        }}
     - tensorflow     {{ tensorflow  }}
     - pytorch        {{ pytorch }}
```

Customize your environment: examples – install Apex

- Install from clone env:

```
module load open-ce/0.1-0
module load gcc/7.4.0

# create cloned conda environment
conda create --name my-opence  --clone open-ce-0.1-0
conda activate my-opence

# install Apex
git clone https://github.com/NVIDIA/apex
cd apex
pip install -v --no-cache-dir --global-option="--cpp_ext" --global-option="--
cuda_ext“  ./
```

Customize your environment: examples – install DALI

- Install from scratch:

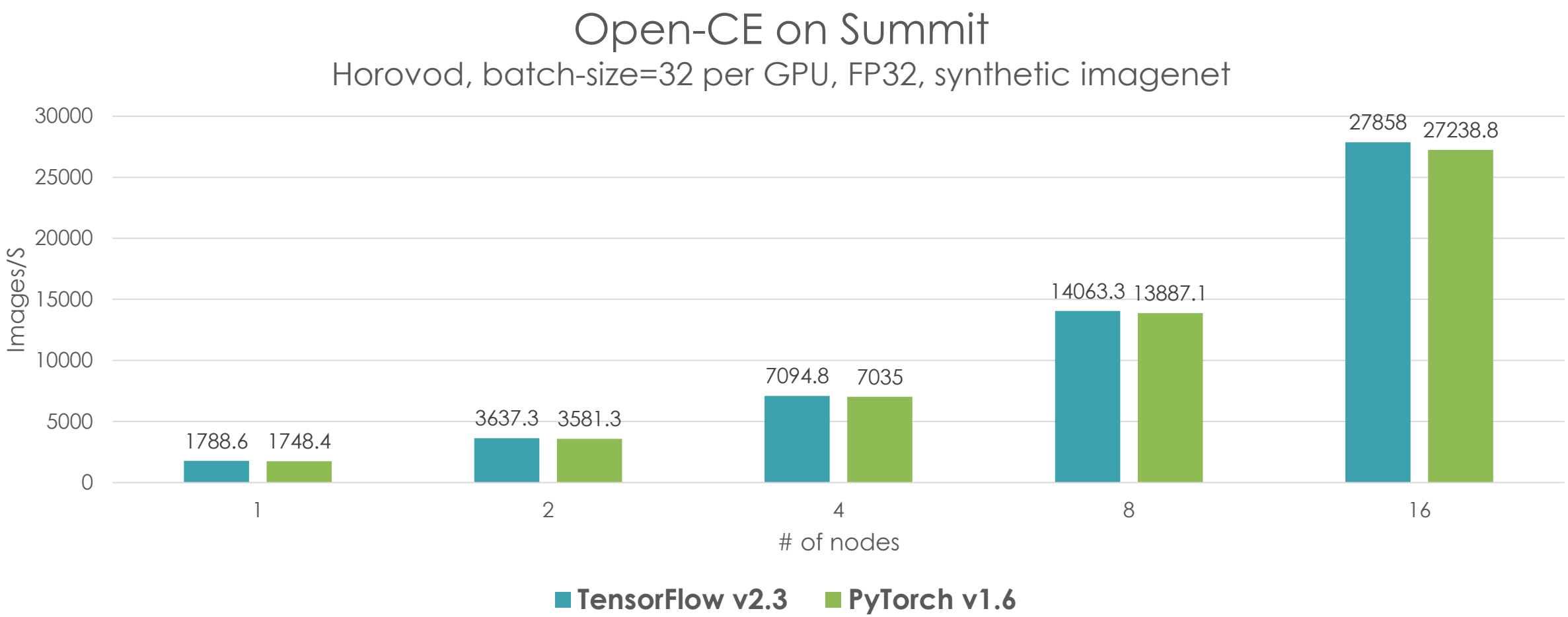
```
module load gcc/7.4.0
export OPENCE_CHANNEL=/sw/sources/open-ce/conda-channel

# install miniconda
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-ppc64le.sh
bash Miniconda3-latest-Linux-ppc64le.sh  -b -p ~/miniconda
export PATH=~/miniconda/bin:$PATH

# create env for dali
conda create –name dali-env python=3.6
conda activate dali-env

# install dali
conda install -c $OPENCE_CHANNEL dali
```

Performance baselines

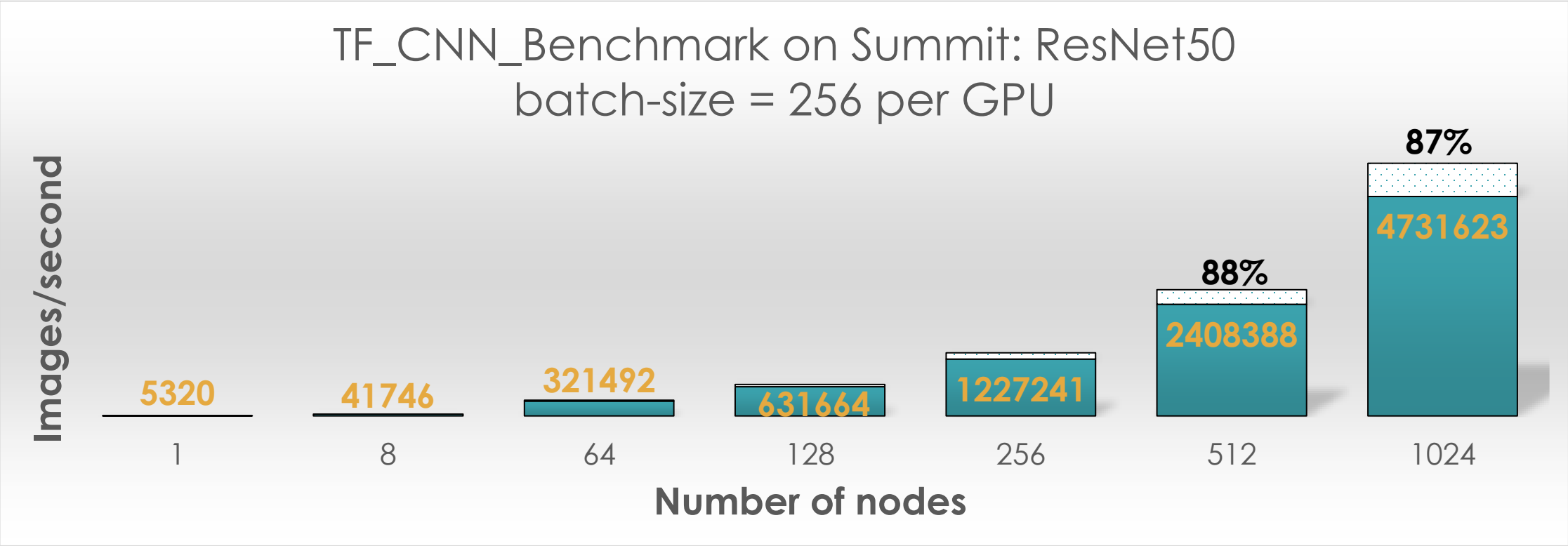
- open-ce module includes horovod examples
- To run distributed training on synthetic data, simply submit the job script

```
#BSUB -P STF011
#BSUB -W 0:10
#BSUB -nnodes 8
#BSUB -q batch
#BSUB -J mldl_test_job
#BSUB -o logs/pyt%J.out
#BSUB -e logs/pyt%J.err

module load open-ce

#TensorFlow
jsrun -bpacked:7 -g6 -a6 -c42 -r1 python
$CONDA_PREFIX/horovod/examples/tensorflow2_synthetic_benchmark.py

#PyTorch
jsrun -bpacked:7 -g6 -a6 -c42 -r1 python
$CONDA_PREFIX/horovod/examples/pytorch_synthetic_benchmark.py
```

Performance baselines



Performance baselines: ResNet50 on ImageNet

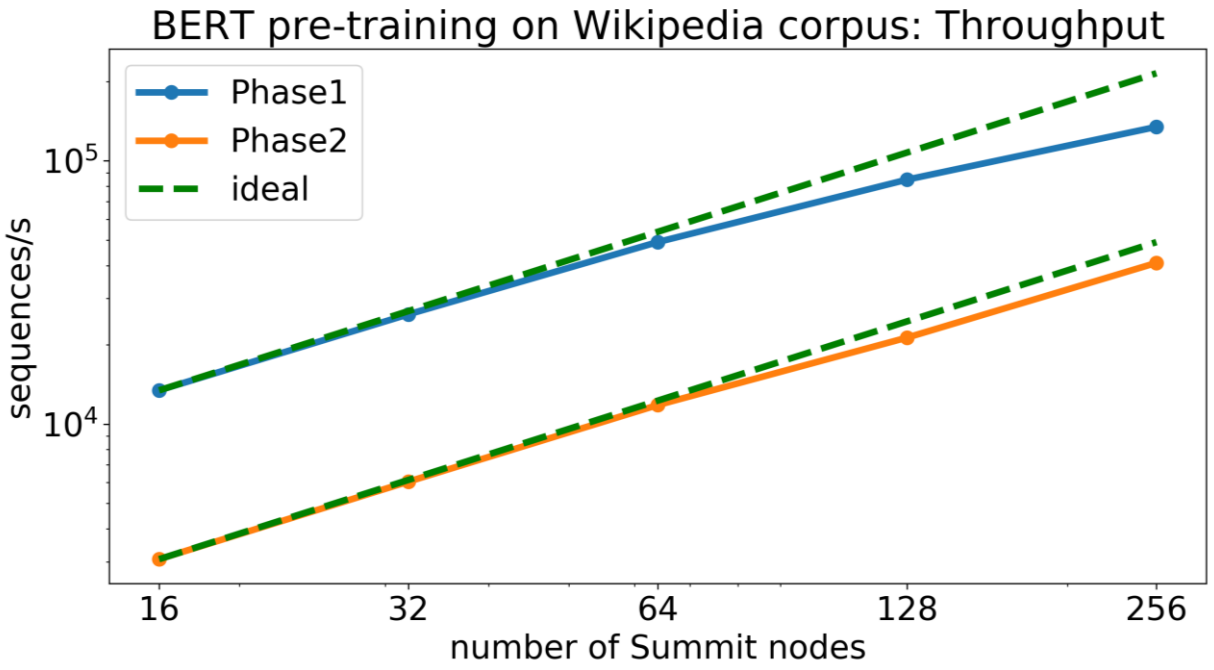


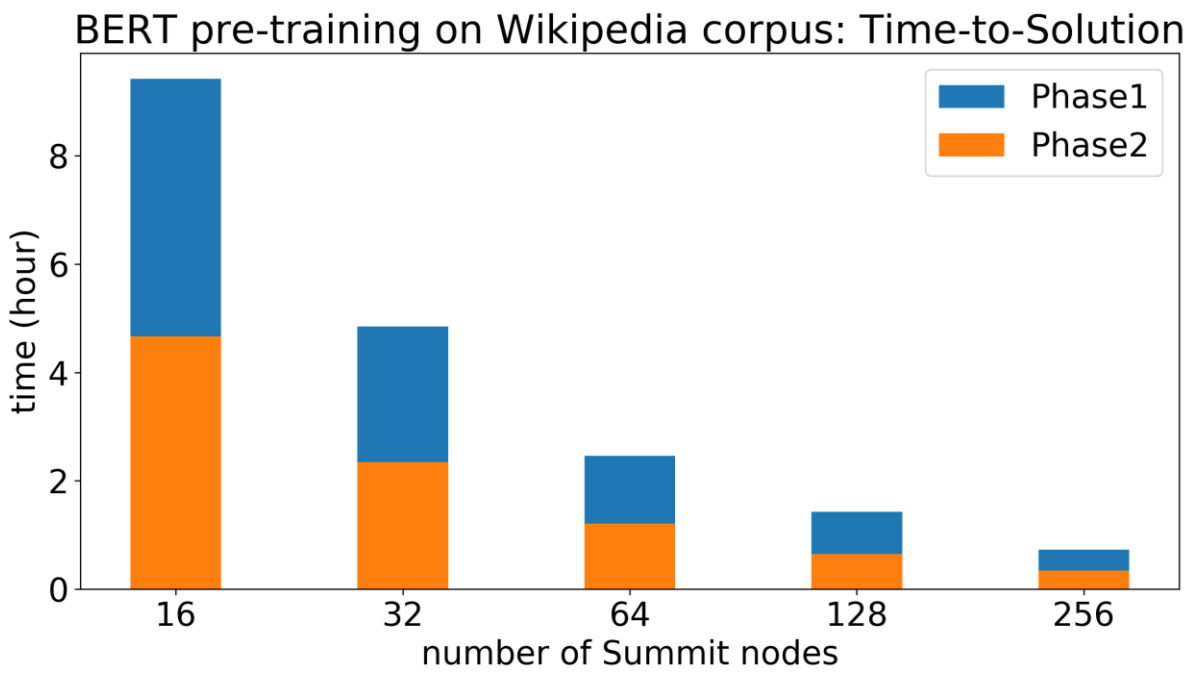
Nodes	Mini-batch size	Top-1 Val accuracy	Training time (min)
16	12288	0.750	27
32	12288	0.766	17
64	15360	0.763	12

TF.distribute & Horovod + LARS

code: TensorFlow distributed example

Performance baselines: BERT on Wikipedia





- Throughput: 63% (phase1) 83% (phase2)
- Time-to-Solution: ~64min on 1536 V100 vs ~76min on 1024 TPUv3 (arXiv:1904.00962)

code: PyTorch BERT example Apex DDP + LAMB

Documentations

- Open-CE repo: https://github.com/open-ce/open-ce
- Summit module docs: https://docs.olcf.ornl.gov/software/analytics/ibm-wml-ce.html
- Distributed DL tutorial: https://code.ornl.gov/olcf-analytics/summit/distributed-deep-learning-examples
- Previous trainings:
  - WML-CE on Summit (slides | recording)
  - Scaling up deep learning application on Summit (slides | recording)
  - ML/DL on Summit (slides | recording)