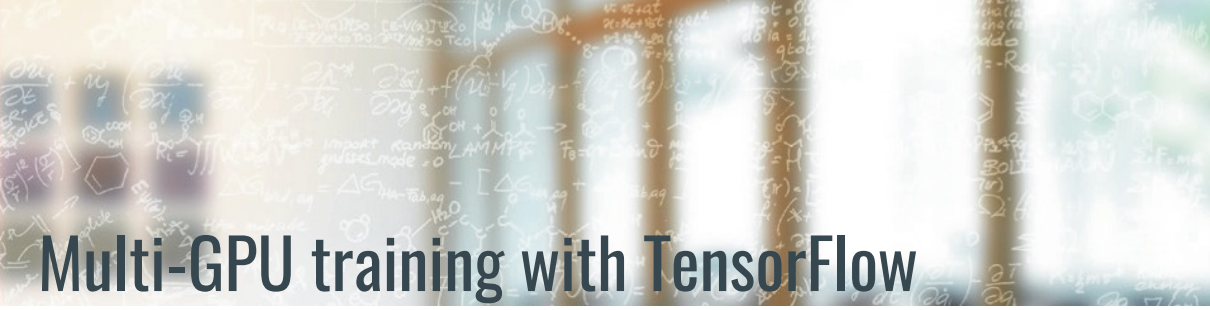


Multi-GPU training with TensorFlow on Piz Daint

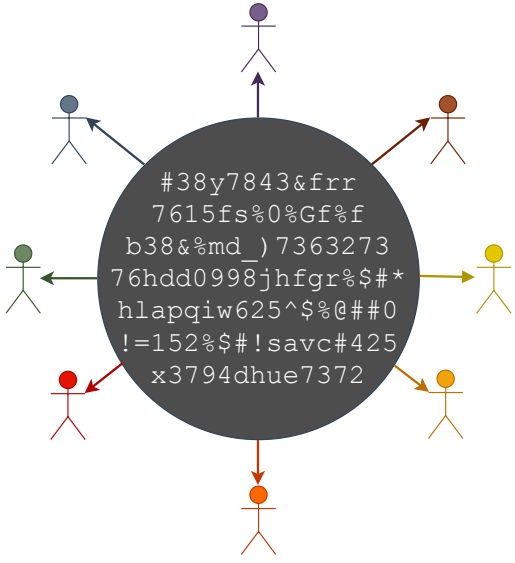
Input pipelines with TensorFlow's `tf.data` API

Rafael Sarmiento and Henrique Mendonça
ETHZürich / CSCS
7th-8th September 2020

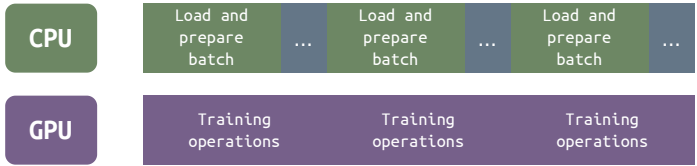
ETH zürich
CSCS
Centro Svizzero di Calcolo Scientifico
Swiss National Supercomputing Centre

Outline

- Input pipelines
- [lab] Building input pipelines
- Read and write TFRecord files
- [lab] Reading and writing TFRecord files


#38y7843&frr
7615fs%0%Gf%f
b38&%md_)7363273
76hdd0998jhfgr%$#*
hlapqiw625^$%@##0
!=152%$#!savc#425
x3794dhue7372

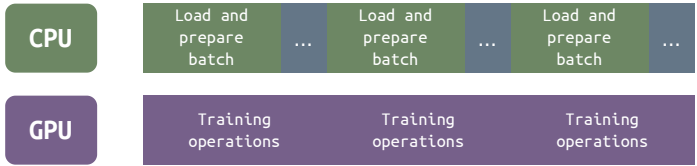
Pipelining


CPU
Load and prepare batch
...
Load and prepare batch
...
Load and prepare batch
...
GPU
Training operations
Training operations
Training operations

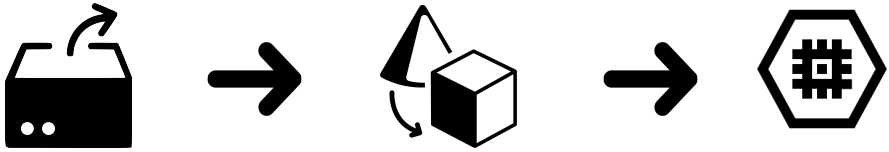
Pipelining


CPU
Load and prepare batch
Idle while training
Load and prepare batch
Idle while training
GPU
Waiting for data
Training operations
Waiting for data
Training operations

Pipelining


CPU
Load and prepare batch
...
Load and prepare batch
...
Load and prepare batch
...
GPU
Training operations
Training operations
Training operations

Input pipelines



Extract
Read data from persistent storage (HDD, SSD, GCS, HDFS, ...)

Transform
Use **CPU cores** to parse and perform preprocessing operations on the data, shuffling and batching

Load
Load the transformed data onto the accelerator devices that execute the model

TensorFlow's `tf.data` API

```
# Extract
dataset = tf.data.TFRecordDataset("./train.tfrecords")
```

TensorFlow's `tf.data` API

```
# Extract
dataset = tf.data.TFRecordDataset("./train.tfrecords")

# Transform
dataset = dataset.shuffle(1000)
dataset = dataset.batch(64)
dataset = dataset.map(resize_images)
dataset = dataset.repeat(100)
```

TensorFlow's `tf.data` API

```
# Extract
dataset = tf.data.TFRecordDataset("./train.tfrecords")

# Transform
dataset = dataset.shuffle(1000)
dataset = dataset.batch(64)
dataset = dataset.map(resize_images)
dataset = dataset.repeat(100)

# 'Load' happens automatically!
```

[lab] Building input pipelines

Let's open an empty notebook and create some simple input pipelines. A good starting point can be to generate random numpy data and create the pipeline including maps, filters, batching, shuffling and repeating operations. `tf.data.Dataset.from_tensor_slices` can be used for the *extract* phase.

Let's start then with the notebook `1_getting_started_with_tensorflows_dataset_api.ipynb`, that's on the folder `input_pipelines/`.

Then we continue to the rest of the notebooks `*_getting_started_with_tensorflows_dataset_api.ipynb` to try other examples.

TFRecord files

- TFRecord is the format of data storage recommended for TensorFlow
- TFRecord is a simple record-oriented binary format
- Data is serialized with protocol buffers and stored as collections of meaningful units (records) in contrast to a byte-oriented filesystem, where the data is treated as an unformatted stream of bytes
- On the deep learning context each record would be an item of the dataset

Best practices for storing data

- Avoid storing data as multiple small files.
- Divide large datasets ($\gg$1 Gb) into TFRecord files of about 150 Mb.
- For smaller data sets (200MB-1GB), a single TFRecord is enough.
- Randomly divide the data into multiple files and then shuffle the filenames uniformly. This helps to decrease the size of the shuffle buffer.
- Read the data from `$SCRATCH`.

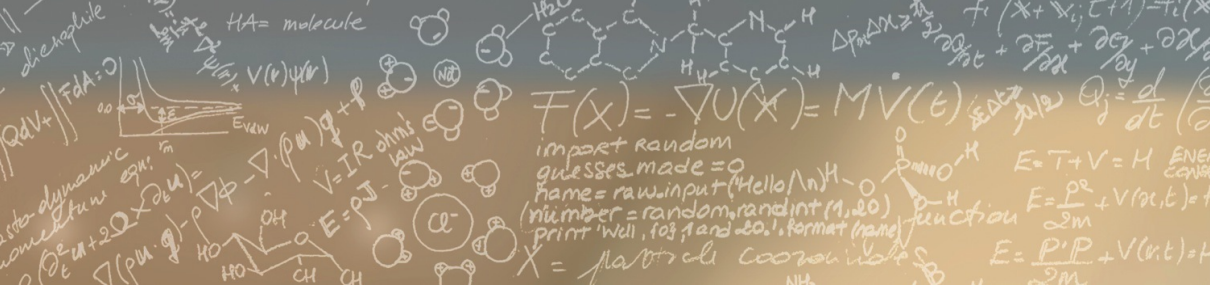
[lab] Reading and writing TFRecord files

Let's run the notebook `read_and_write_TFRecord_files.ipynb`. We are going to write two cat images to a TFRecord file and then we are going to read them to check that the they can be recovered correctly.

Following the same steps, write the MNIST dataset to a TFRecord file. We will use it later on a lab.

You can get the MNIST data as numpy arrays with

```
from tensorflow.keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```



Thank you for your attention!