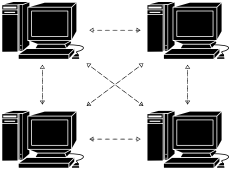
Distributed Deep Learning Using Hopsworks

EIT Big Data Summer School

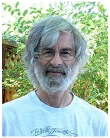
Kim Hammar
kim@logicalclocks.com

DISTRIBUTED COMPUTING + DEEP LEARNING = ?

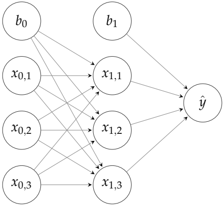
Distributed Computing



"A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable."
- Leslie Lamport



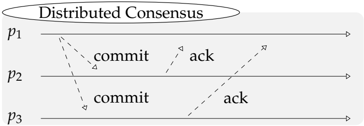
Deep Learning

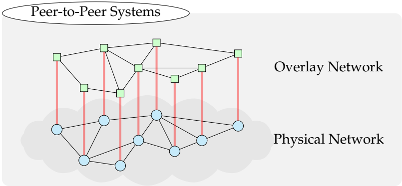


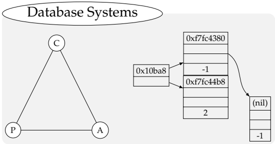
"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P," improves with experience E.
- Tom Mitchell

Why Combine the two?

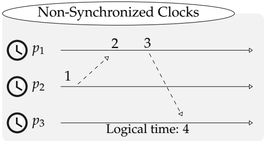
DISTRIBUTED COMPUTING IN 2 MINUTES

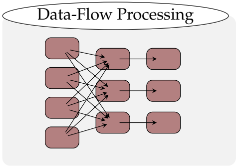




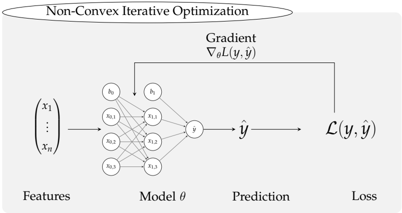


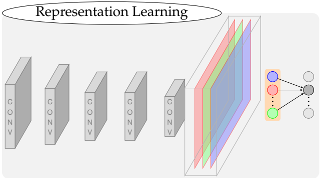
Big Data

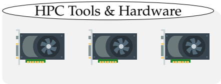




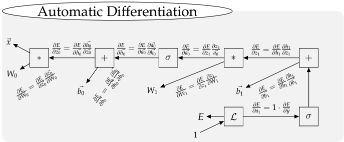
Deep Learning in 2 minutes

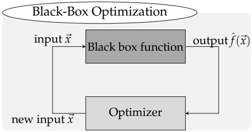






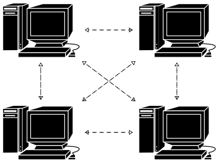
Big Data



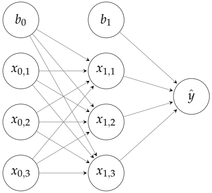


DISTRIBUTED COMPUTING + DEEP LEARNING = ?

Distributed Computing



Deep Learning



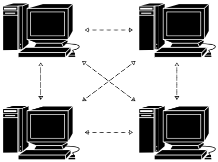
Why Combine the two?

2em1[1] Chen Sun et al. “Revisiting Unreasonable Effectiveness of Data in Deep Learning Era”. In: *CoRR* abs/1707.02968 (2017). arXiv: 1707.02968. URL: http://arxiv.org/abs/1707.02968.

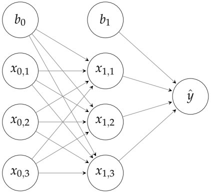
2em1[2] Jeffrey Dean et al. “Large Scale Distributed Deep Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1223–1231.

DISTRIBUTED COMPUTING + DEEP LEARNING = ?

Distributed Computing



Deep Learning



Why Combine the two?

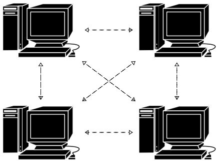
- We like challenging problems ☺

2em1[1] Chen Sun et al. "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In: *CoRR* abs/1707.02968 (2017). arXiv: 1707.02968. URL: http://arxiv.org/abs/1707.02968.

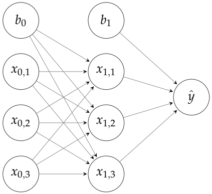
2em1[2] Jeffrey Dean et al. "Large Scale Distributed Deep Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1223–1231.

DISTRIBUTED COMPUTING + DEEP LEARNING = ?

Distributed Computing



Deep Learning



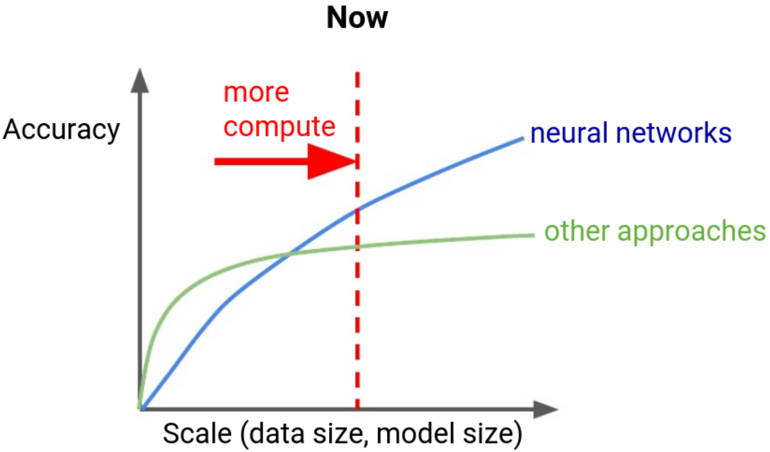
Why Combine the two?

- We like challenging problems ☺
- More productive data science
- Unreasonable effectiveness of data[1]
- To achieve state-of-the-art results[2]

2em1[1] Chen Sun et al. "Revisiting Unreasonable Effectiveness of Data in Deep Learning Era". In: *CoRR* abs/1707.02968 (2017). arXiv: 1707.02968. URL: http://arxiv.org/abs/1707.02968.

2em1[2] Jeffrey Dean et al. "Large Scale Distributed Deep Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1223–1231.

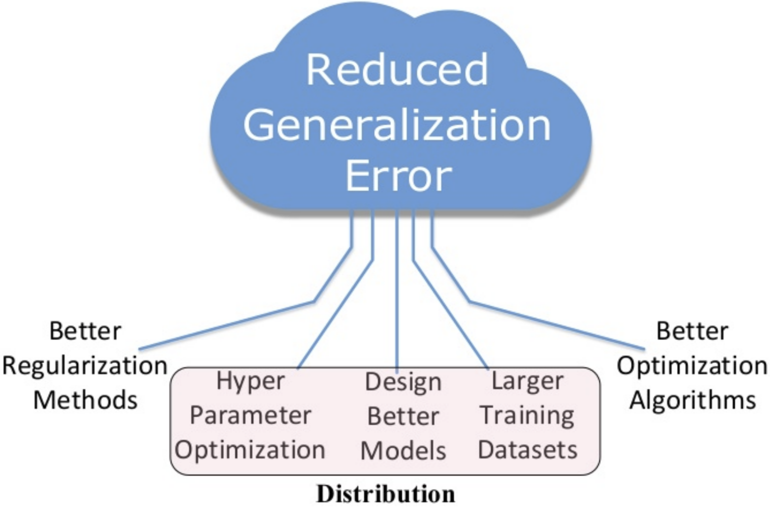
DISTRIBUTED DEEP LEARNING (DDL): PREDICTABLE SCALING



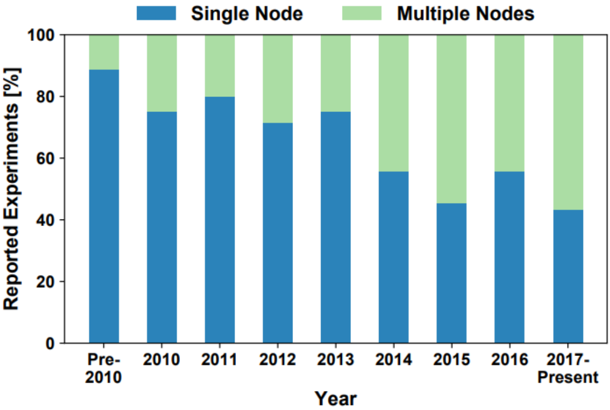
3

2em1[3] Jeff Dean. *Building Intelligent Systems withLarge Scale Deep Learning*. https://www.scribd.com/document/355752799/Jeff-Dean-s-Lecture-for-YC-AI. 2018.

DISTRIBUTED DEEP LEARNING (DDL): PREDICTABLE SCALING



DDL IS NOT A SECRET ANYMORE



(b) Training with Single vs. Multiple Nodes [4]

2em1[4] Tal Ben-Nun and Torsten Hoefler. "Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis". In: *CoRR* abs/1802.09941 (2018). arXiv: 1802.09941. URL: http://arxiv.org/abs/1802.09941.

DDL IS NOT A SECRET ANYMORE

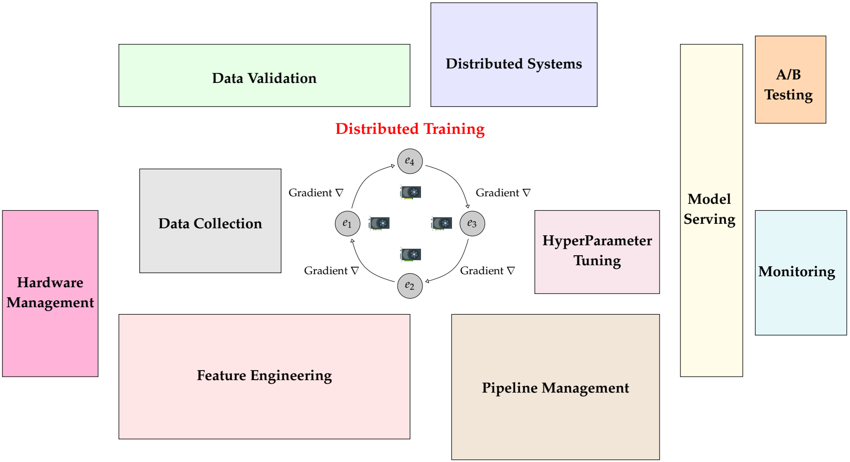
Frameworks for DDL

- Kubeflow
- Microsoft CNTK
- Horovod
- Amazon SageMaker
- BigDL
- TensorflowOnSpark
- Distributed TF
- CaffeOnSpark
- PyTorch
- Apache Spark
- mxnet
- DL4J
- ChainerMN
- TonY

Companies using DDL

- facebook
- Google
- UBER
- zalando
- YAHOO!
- Twitter
- NETFLIX
- amazon
- LinkedIn
- Baidu 百度
- IBM
- Microsoft

DDL REQUIRES AN ENTIRE SOFTWARE/INFRASTRUCTURE STACK



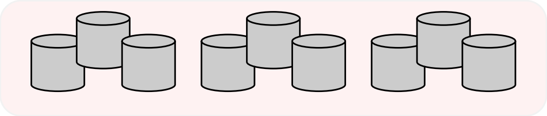
OUTLINE

1. **Hopsworks**: Background of the platform
2. **Managed Distributed Deep Learning** using HopsYARN, HopsML, PySpark, and Tensorflow
3. **Black-Box Optimization (Hyperparameter Tuning)** using Hopsworks, Metadata Store and PySpark
4. **Short Break**
5. **Demo**, end-to-end ML pipeline
6. **Hands-on Workshop**, try out Hopsworks on SICS ICE cluster

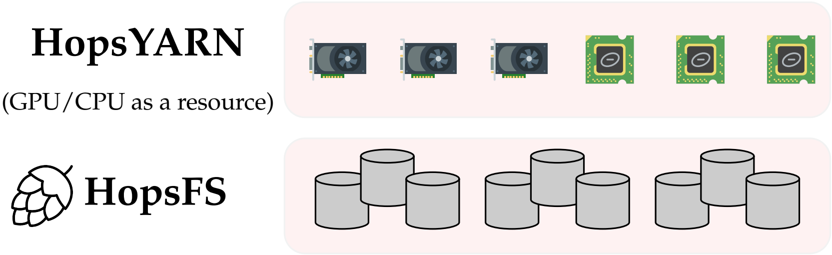
HOPSWORKS

Hopsworks

HopsFS



HOPSWORKS



HOPSWORKS

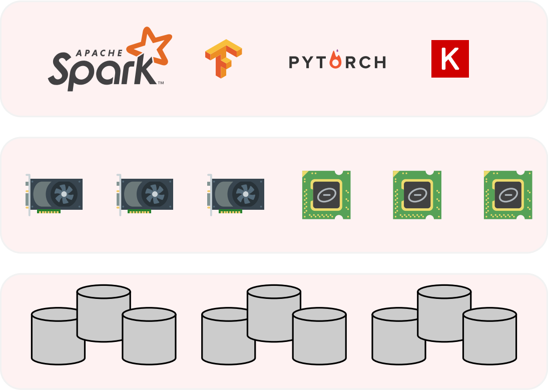
Frameworks

(ML/Data)

HopsYARN

(GPU/CPU as a resource)

HopsFS



Hopsworks

ML/AI Assets

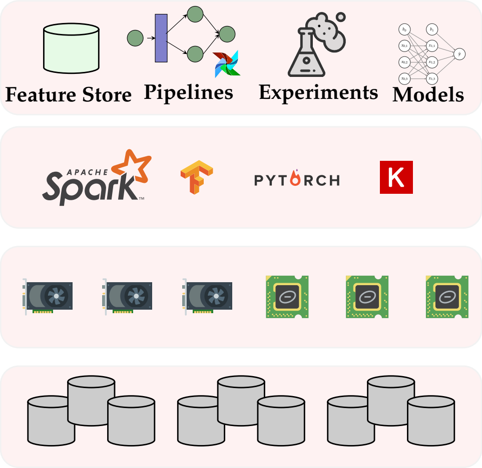
Frameworks

(ML/Data)

HopsYARN

(GPU/CPU as a resource)

HopsFS

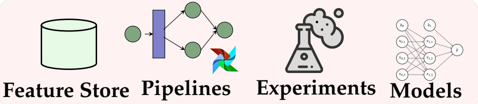


Hopsworks

APIs

```python
from hops import featurestore
from hops import experiment
featurestore.get_features([
                    "average_attendance",
                    "average_player_age"])
experiment.collective_all_reduce(features, model)
```

ML/AI Assets



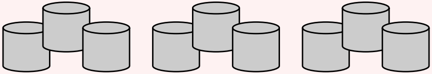
Frameworks

(ML/Data)

HopsYARN

(GPU/CPU as a resource)

HopsFS

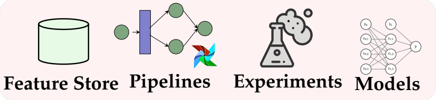


HOPSWORKS

APIs

```python
from hops import featurestore
from hops import experiment
featurestore.get_features([
                    "average_attendance",
                    "average_player_age"])
experiment.collective_all_reduce(features, model)
```

ML/AI Assets



Frameworks

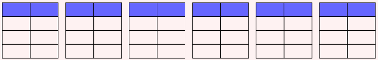
(ML/Data)

HopsYARN

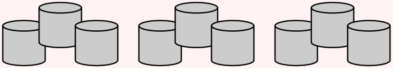
(GPU/CPU as a resource)

Distributed Metadata

(Available from REST API)

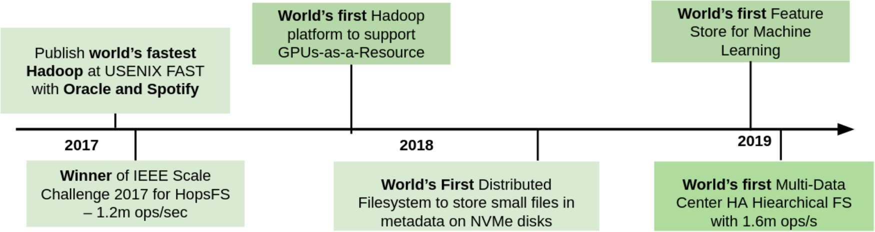


HopsFS



THE TEAM

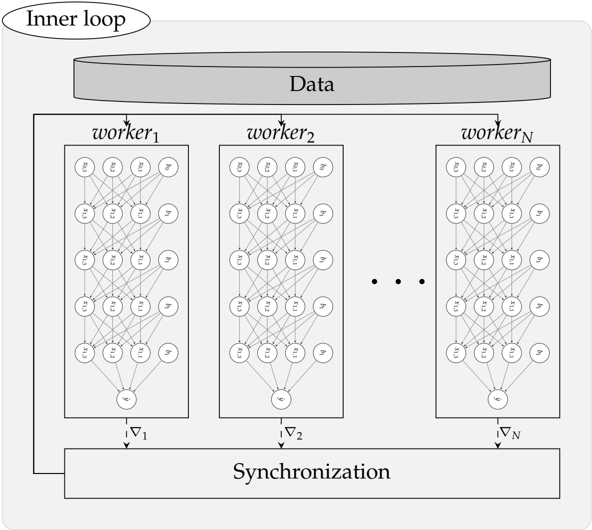
Hops & Hopsworks History



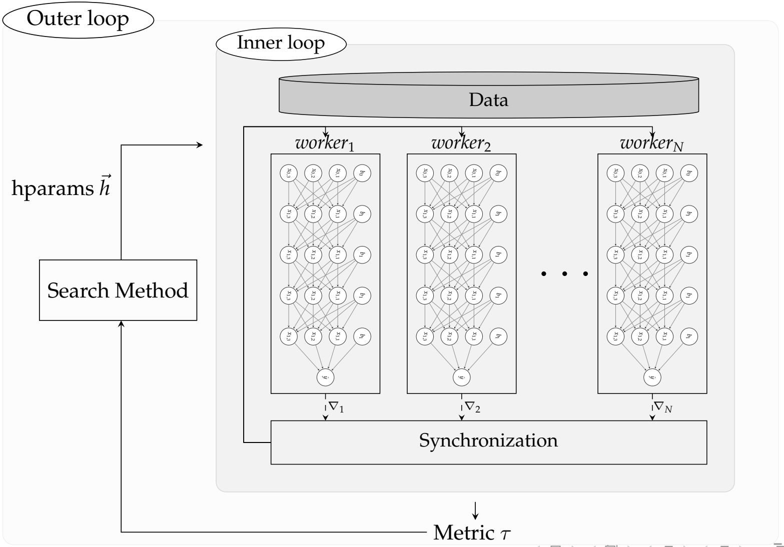
*"If you're working with big data and Hadoop, **this one paper could repay your investment** in the Morning Paper many times over.... **HopFS is a huge win.**"*

Adrian Colyer, The Morning Paper

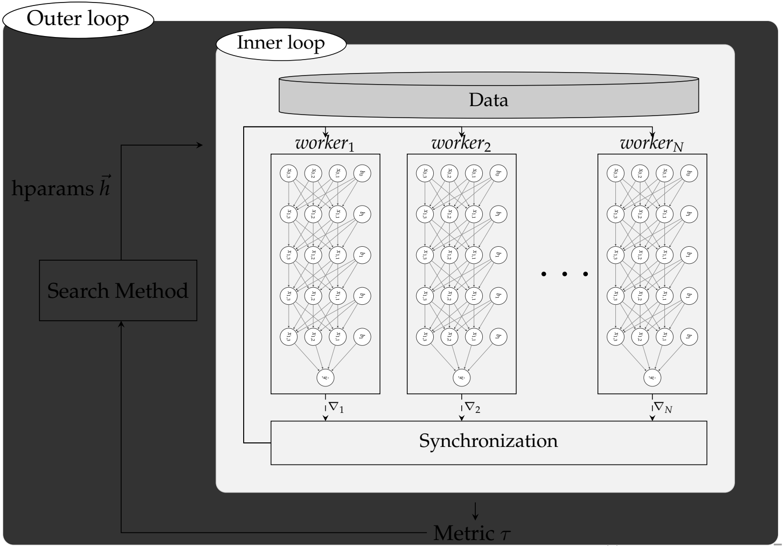
INNER AND OUTER LOOP OF LARGE SCALE DEEP LEARNING



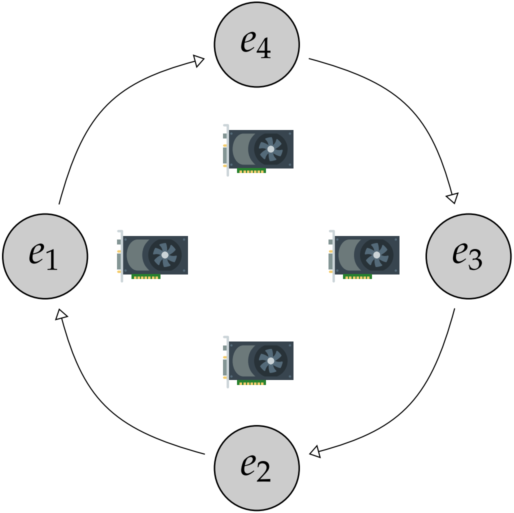
INNER AND OUTER LOOP OF LARGE SCALE DEEP LEARNING



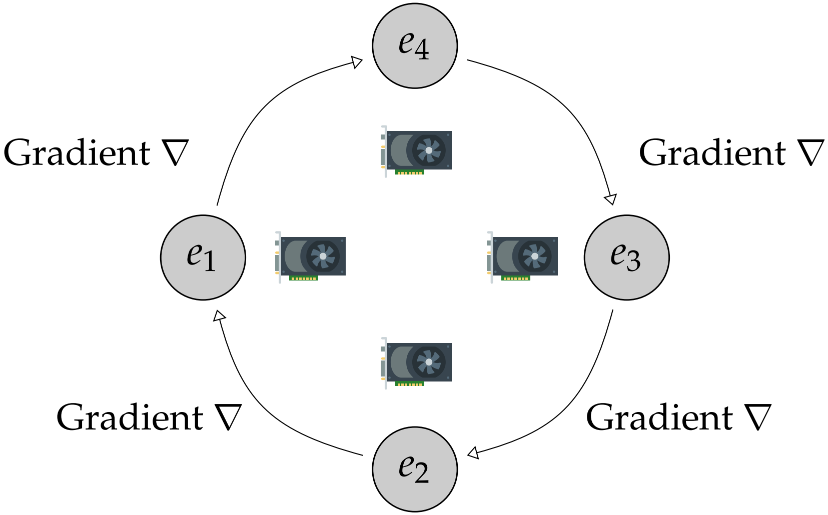
INNER AND OUTER LOOP OF LARGE SCALE DEEP LEARNING



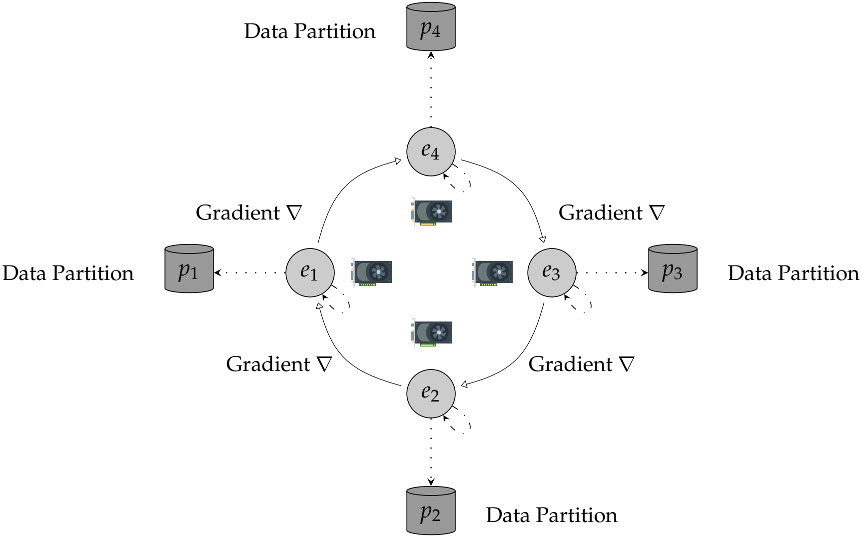
Inner Loop: Distributed Deep Learning



INNER LOOP: DISTRIBUTED DEEP LEARNING



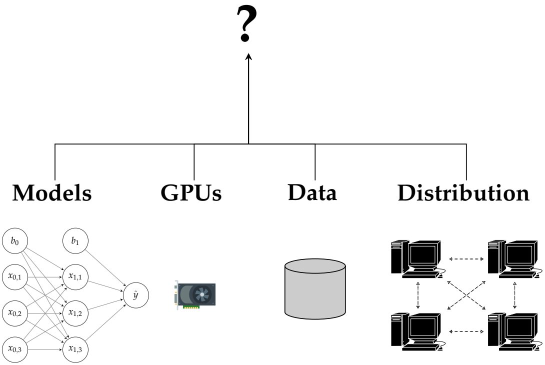
INNER LOOP: DISTRIBUTED DEEP LEARNING



DISTRIBUTED DEEP LEARNING IN PRACTICE

- Implementation of distributed algorithms is becoming a **commodity** (TF, PyTorch etc)

- **The hardest part of DDL is now:**
  - Cluster management
  - Allocating GPUs
  - Data management
  - Operations & performance



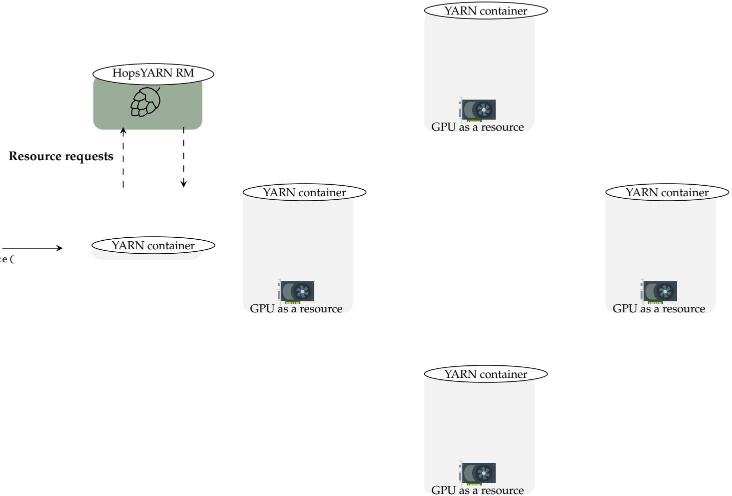
HOPSWORKS DDL SOLUTION

Hopsworks DDL Solution

```python
from hops import experiment

experiment.collective_all_reduce(train_fn)
```

HOPSWORKS DDL SOLUTION



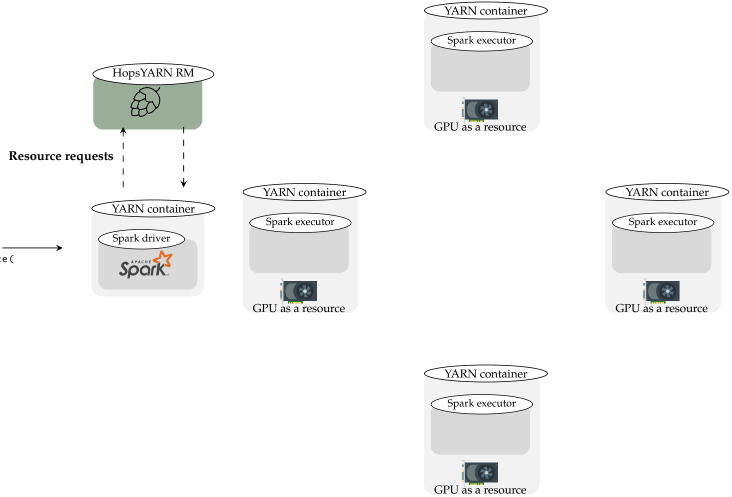

```
from hops
import experiment
experiment.
collective_all_reduce(
train_fn
)
```

Client API

HOPSWORKS DDL SOLUTION

```python
from hops
import experiment
experiment.
collective_all_reduce(
train_fn
)
```

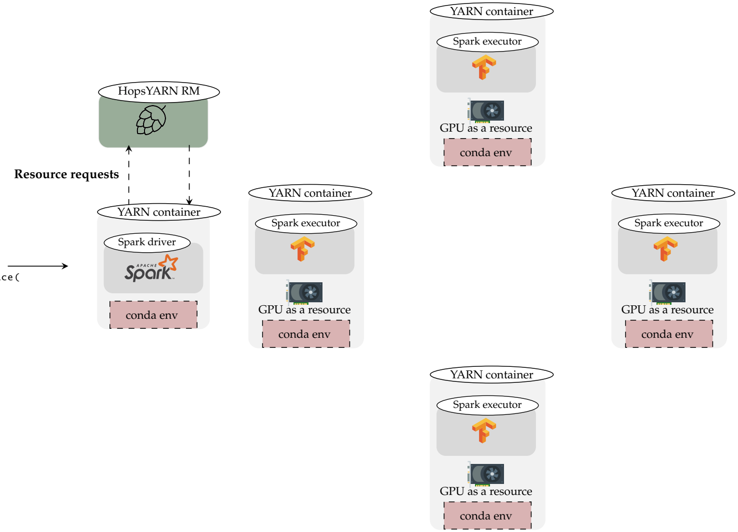
Client API



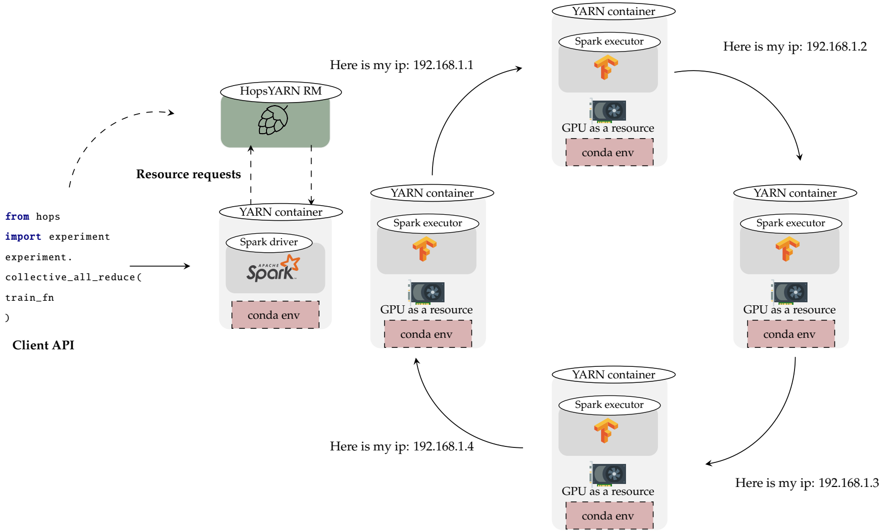
HOPSWORKS DDL SOLUTION

```
from hops
import experiment
experiment.
collective_all_reduce(
train_fn
)
```

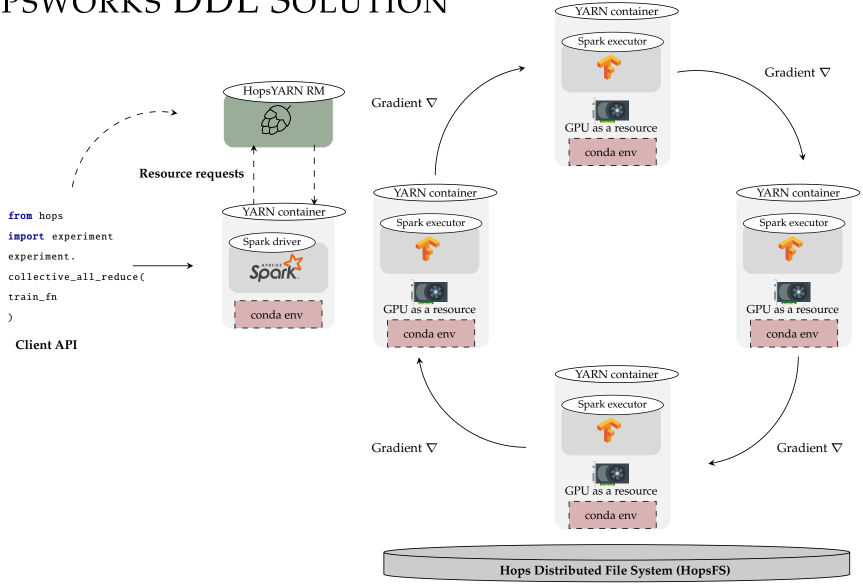
Client API



HOPSWORKS DDL SOLUTION



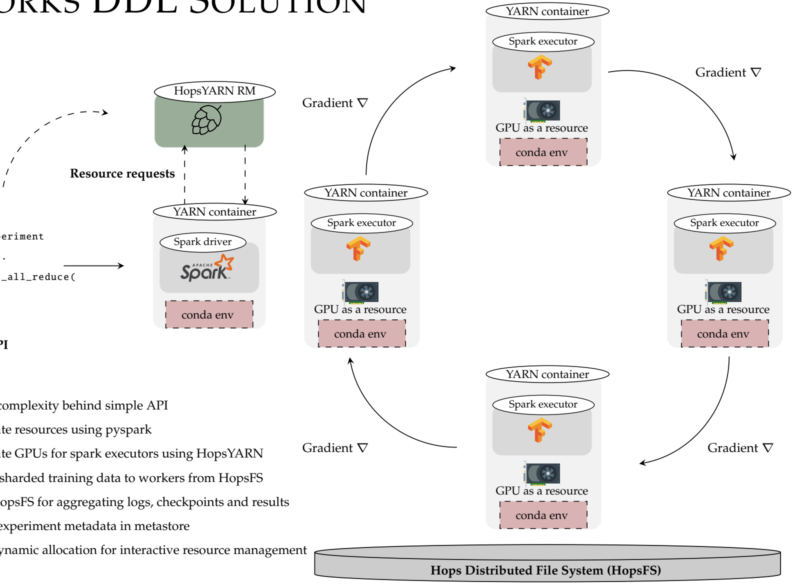
HOPSWORKS DDL SOLUTION



HOPSWORKS DDL SOLUTION

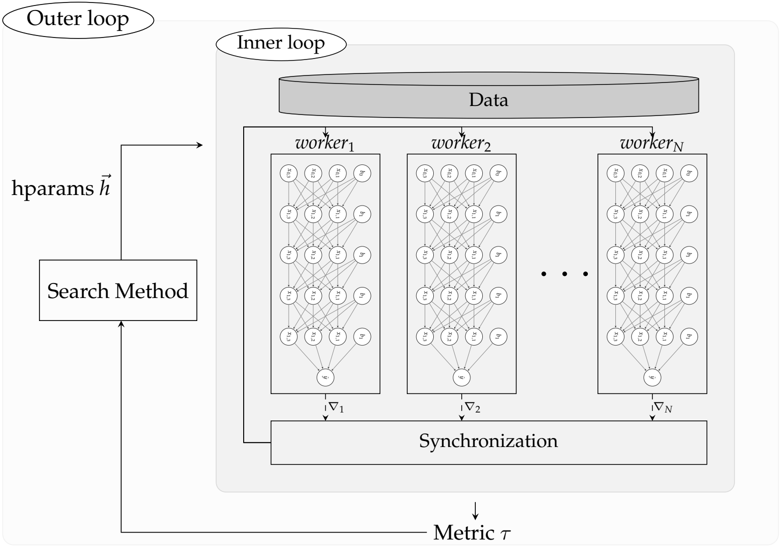
```python
from hops
import experiment
experiment.
collective_all_reduce(
train_fn
)
```

Client API

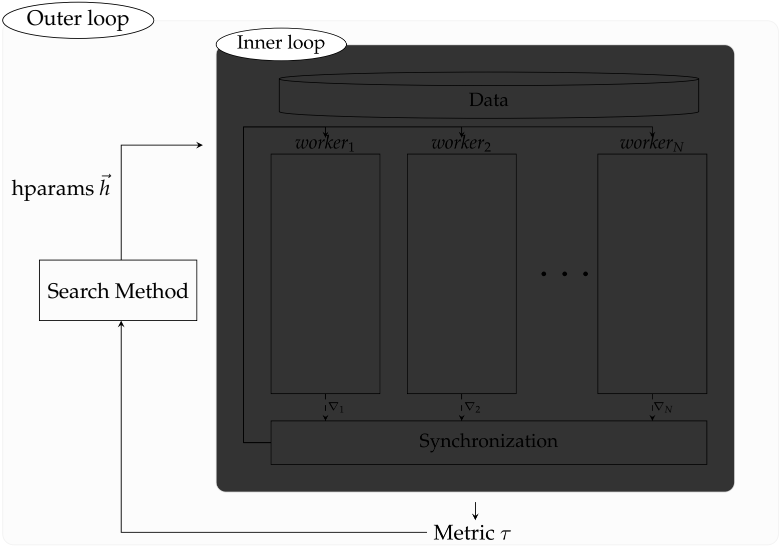


- Hide complexity behind simple API
- Allocate resources using pyspark
- Allocate GPUs for spark executors using HopsYARN
- Serve sharded training data to workers from HopsFS
- Use HopsFS for aggregating logs, checkpoints and results
- Store experiment metadata in metastore
- Use dynamic allocation for interactive resource management

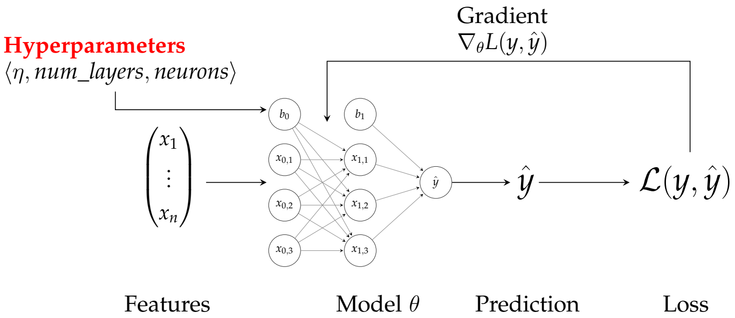
OUTER LOOP: BLACK BOX OPTIMIZATION



OUTER LOOP: BLACK BOX OPTIMIZATION



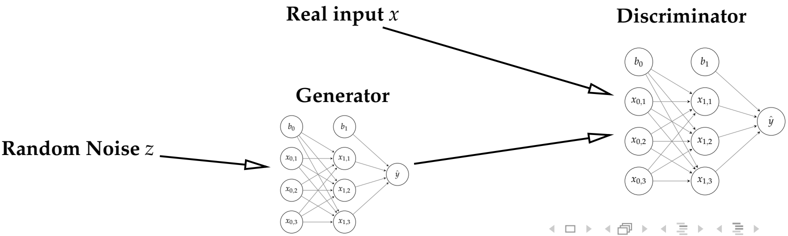
Outer Loop: Black Box Optimization



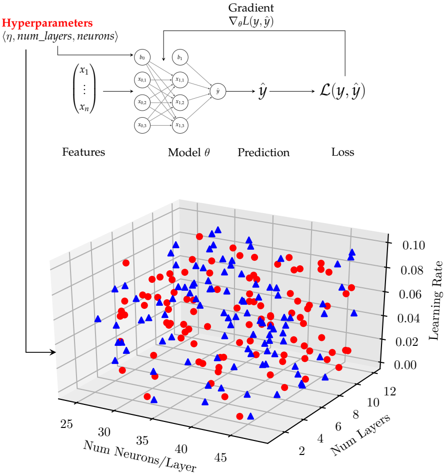
OUTER LOOP: BLACK BOX OPTIMIZATION

Example Use-Case from one of our clients:

- Goal: Train a One-Class GAN model for fraud detection
- Problem: GANs are extremely sensitive to hyperparameters and there exists a very large space of possible hyperparameters.
- Example hyperparameters to tune: learning rates η, optimizers, layers.. etc.

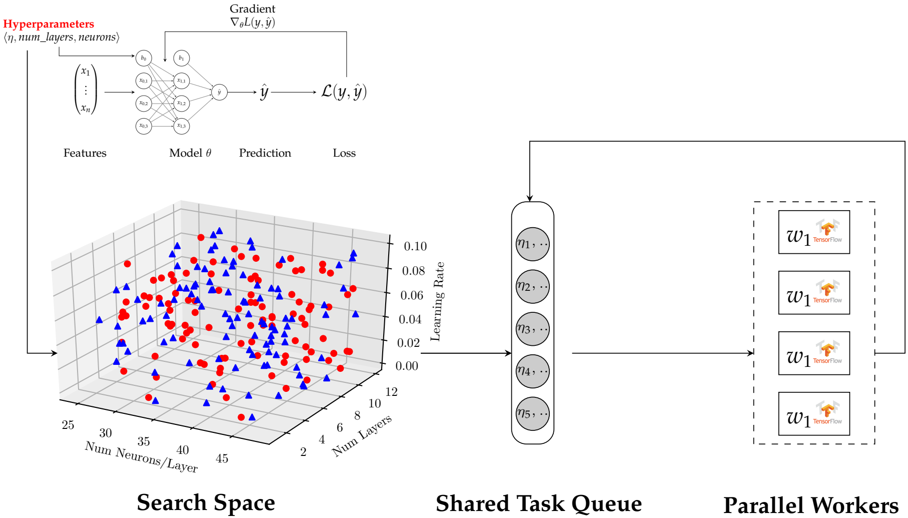


OUTER LOOP: BLACK BOX OPTIMIZATION

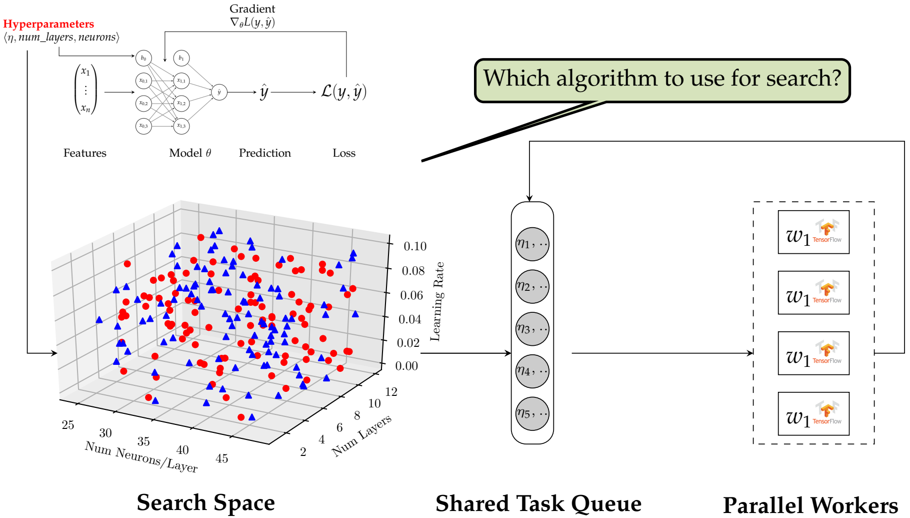


Search Space

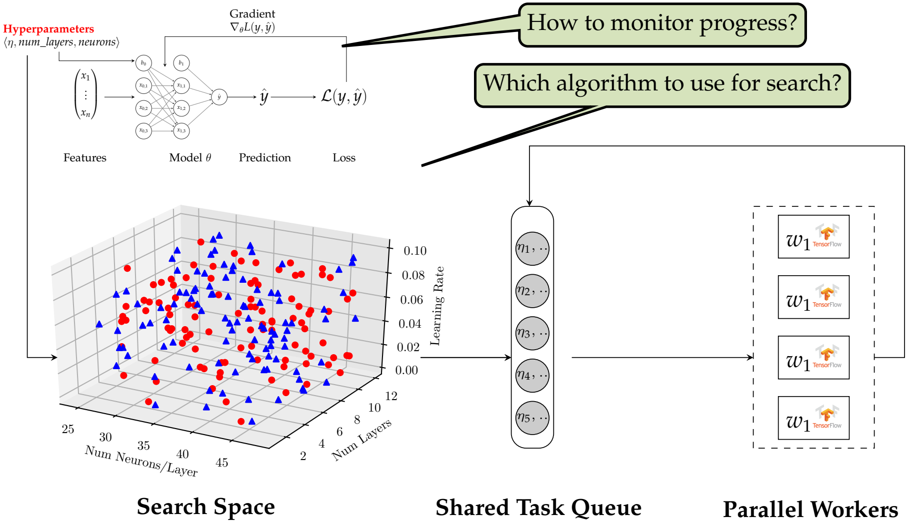
OUTER LOOP: BLACK BOX OPTIMIZATION



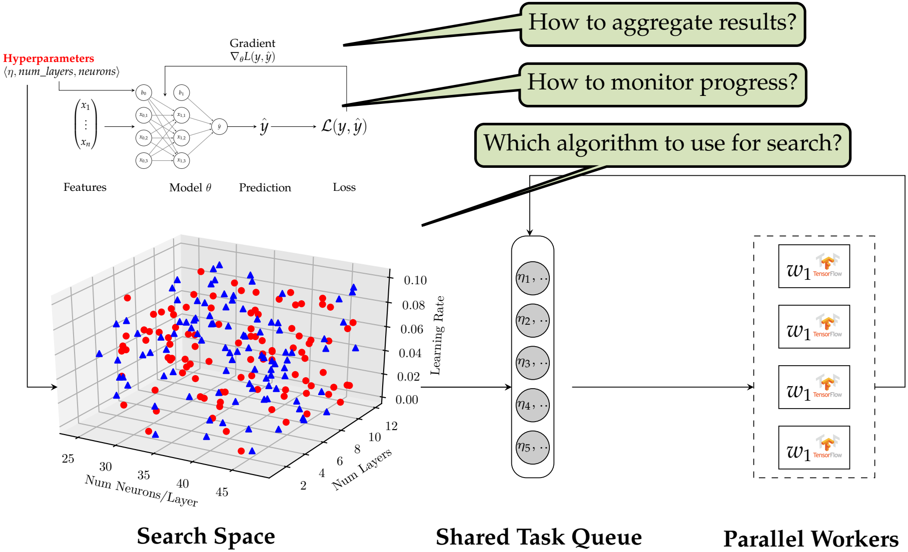
OUTER LOOP: BLACK BOX OPTIMIZATION



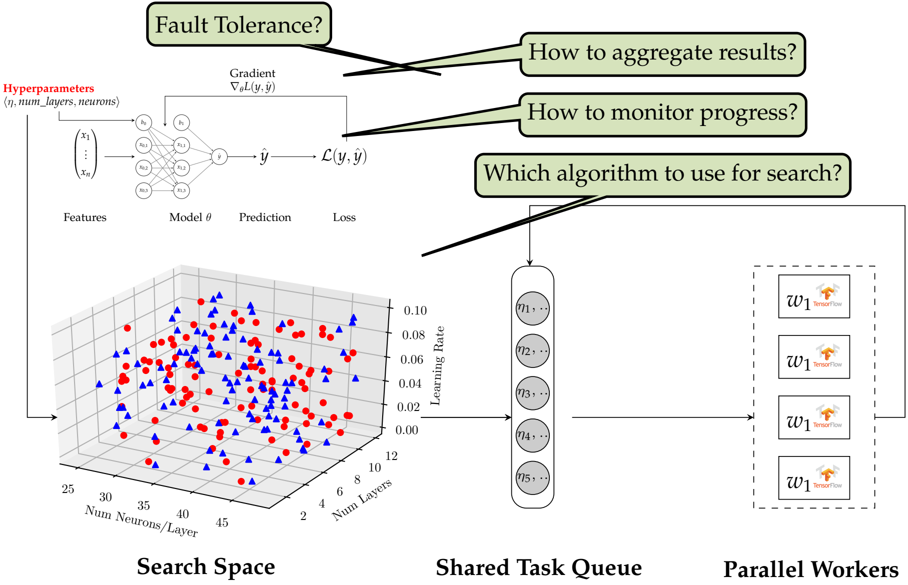
Outer Loop: Black Box Optimization



OUTER LOOP: BLACK BOX OPTIMIZATION



OUTER LOOP: BLACK BOX OPTIMIZATION



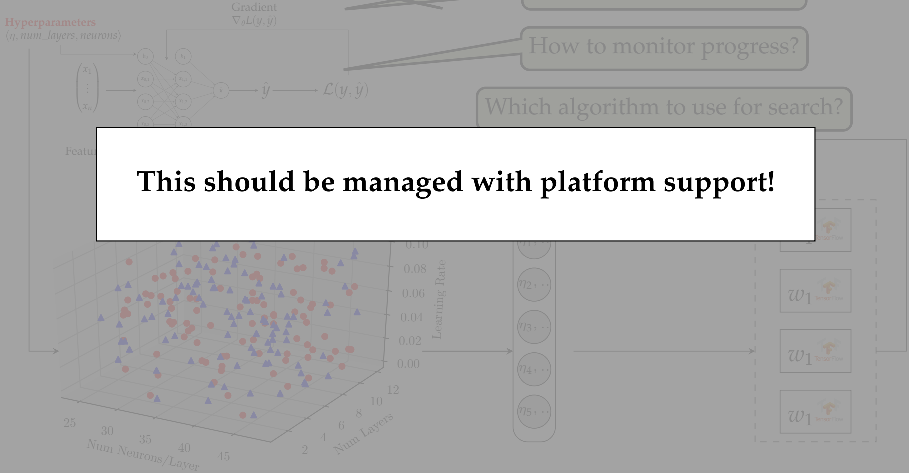
OUTER LOOP: BLACK BOX OPTIMIZATION

Fault Tolerance?

How to aggregate results?

How to monitor progress?

Which algorithm to use for search?



This should be managed with platform support!

Search Space **Shared Task Queue** **Parallel Workers**

Maggy: A Framework for Synchronous/Asynchronous Hyperparameter Tuning on Hopsworks[5]

A flexible framework for running different black-box optimization algorithms on Hopsworks

- ASHA, Hyperband, Differential Evolution, Random search, Grid search, etc.

2em1[5] Authors of Maggy: Moritz Meister and Sina Sheikholeslami. Author of the base framework that Maggy builds on: Robin Andersson

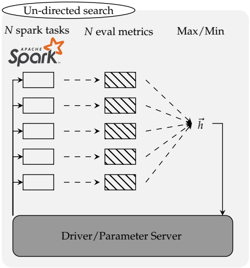
Maggy: A Framework for Synchronous/Asynchronous Hyperparameter Tuning on Hopsworks[5]

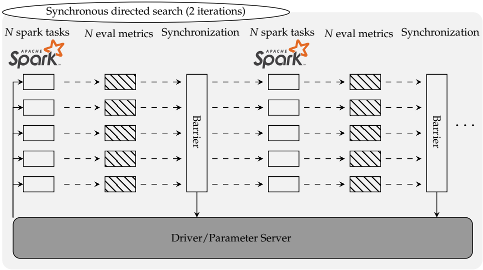
A flexible framework for running different black-box optimization algorithms on Hopsworks

- ASHA, Hyperband, Differential Evolution, Random search, Grid search, etc.

2em1[5] Authors of Maggy: Moritz Meister and Sina Sheikholeslami. Author of the base framework that Maggy builds on: Robin Andersson

FRAMEWORK SUPPORT FOR SYNCHRONOUS SEARCH ALGORITHMS

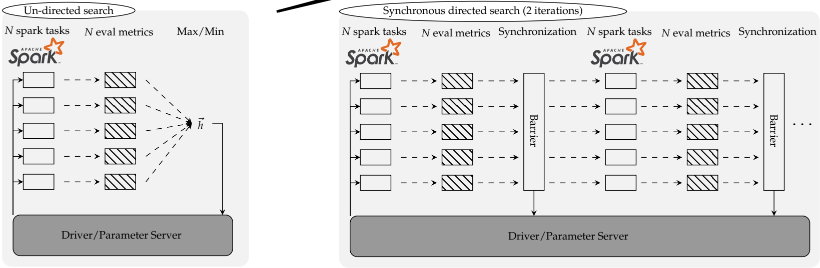




- Parallel undirected/synchronous search is trivial using Spark and a distributed file system
- Example of un-directed search algorithms: **random and grid search**
- Example of synchronous search algorithms: **differential evolution**

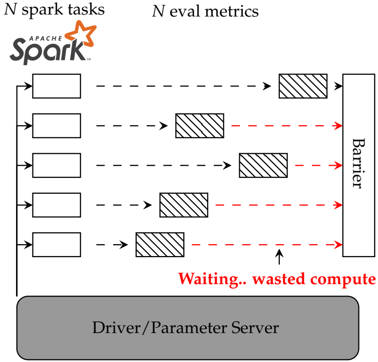
FRAMEWORK SUPPORT FOR SYNCHRONOUS SEARCH ALGORITHMS

Fits very well with Spark BSP model



- Parallel un-directed/synchronous search is trivial using Spark and a distributed file system
- Example of un-directed search algorithms: **random and grid search**
- Example of synchronous search algorithms: **differential evolution**

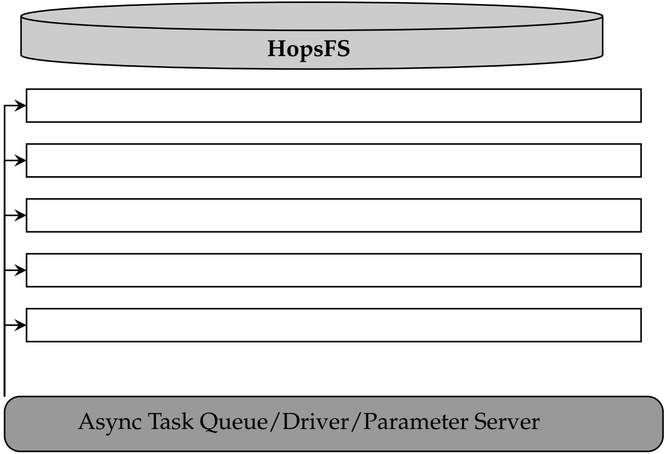
PROBLEM WITH THE BULK-SYNCHRONOUS PROCESSING MODEL FOR PARALLEL SEARCH



- Synchronous search is sensitive to stragglers and not suitable for early stopping
- ... For large scale search problems we need asynchronous search
- **Problem:** Asynchronous search is much harder to implement with big data processing tools such as Spark

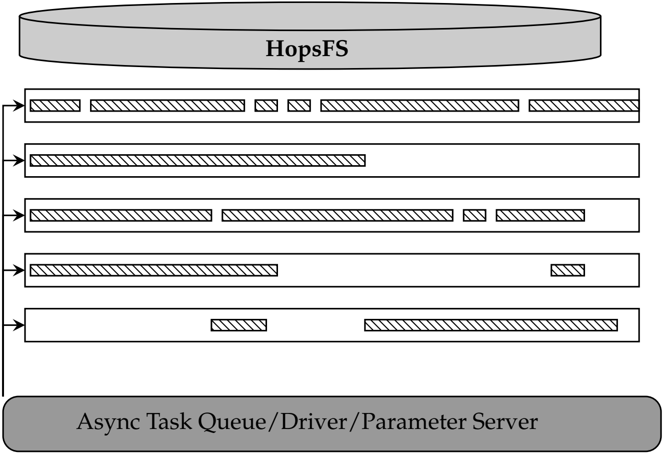
ENTER MAGGY: A FRAMEWORK FOR RUNNING ASYNCHRONOUS SEARCH ALGORITHMS ON HOPS

1 spark task/worker



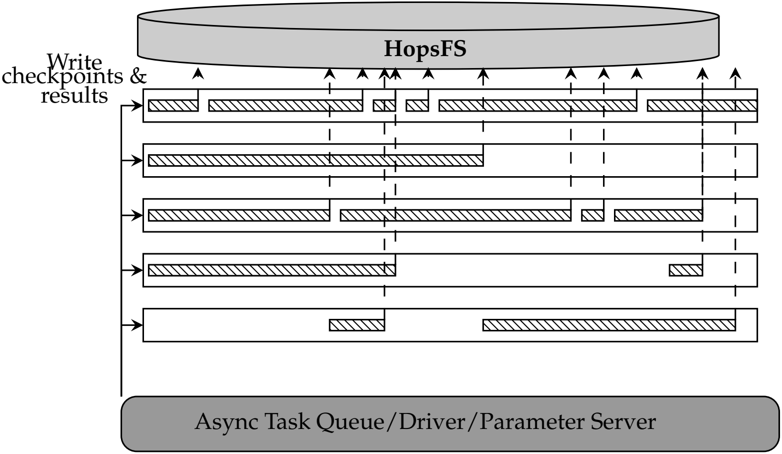
Enter Maggy: A Framework for running Asynchronous Search Algorithms on Hops

1 spark task/worker, **many async tasks inside**



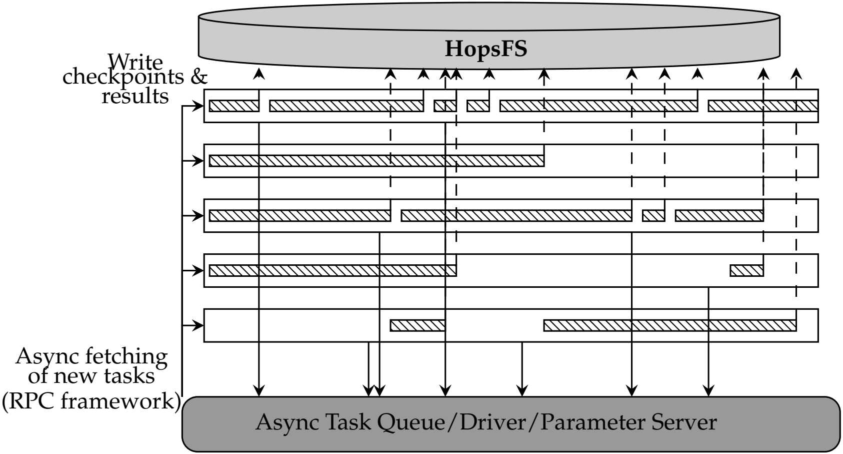
ENTER MAGGY: A FRAMEWORK FOR RUNNING ASYNCHRONOUS SEARCH ALGORITHMS ON HOPS

1 spark task/worker, **many async tasks inside**



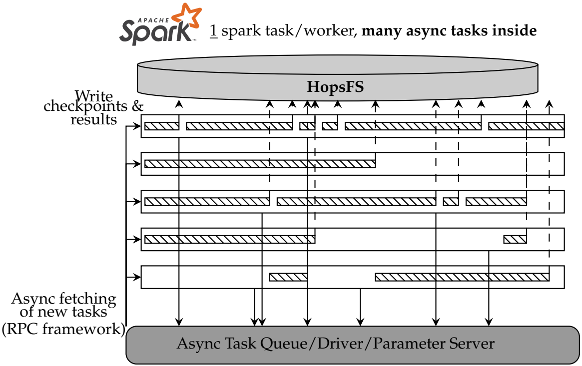
ENTER MAGGY: A FRAMEWORK FOR RUNNING ASYNCHRONOUS SEARCH ALGORITHMS ON HOPS

1 spark task/worker, **many async tasks inside**



ENTER MAGGY: A FRAMEWORK FOR RUNNING ASYNCHRONOUS SEARCH ALGORITHMS ON HOPS

- Robust against stragglers
- Supports early stopping
- Fault tolerance with checkpointing
- Monitoring with Tensorboard
- Log aggregation with HopsFS
- Simple API and extendable

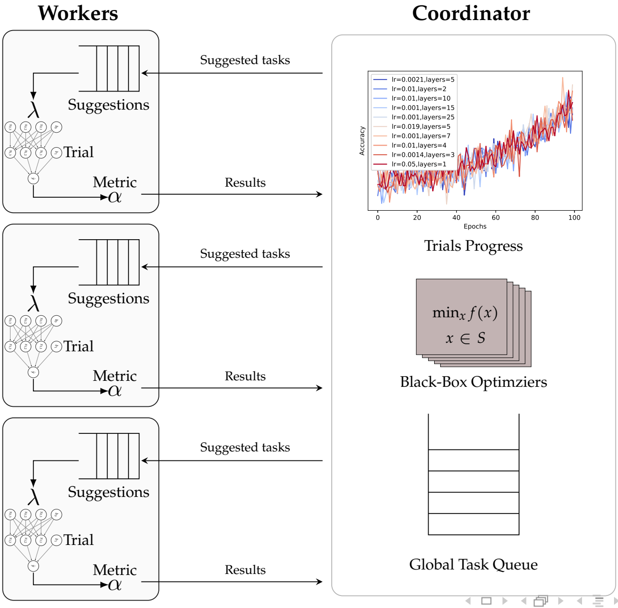


Parallel Experiments

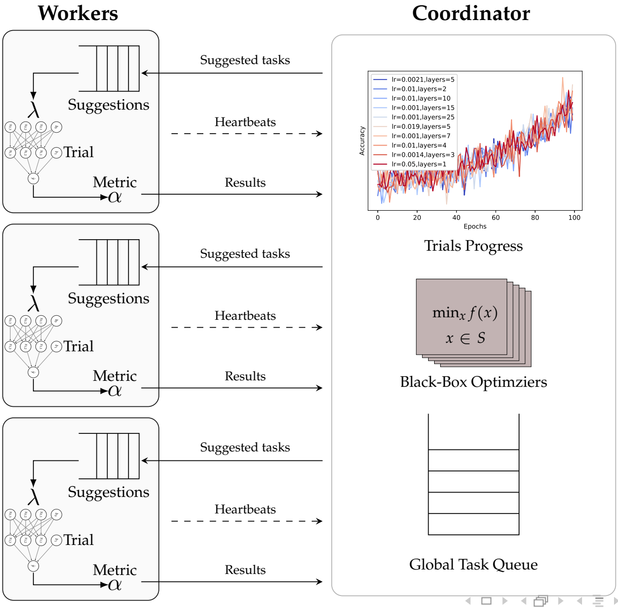
```
from maggy import experiment
from maggy.searchspace import Searchspace
from maggy.randomsearch import RandomSearch

sp = Searchspace(argument_param=('DOUBLE', [1, 5]))
rs = RandomSearch(5, sp)
result = experiment.launch(train_fn,
                           sp, optimizer=rs,
                           num_trials=5, name='test',
                           direction="max")
```

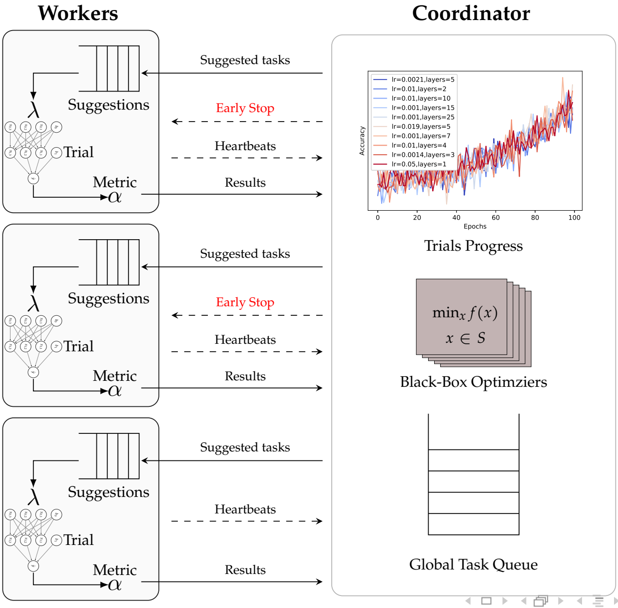
ASYNCHRONOUS SEARCH WORKFLOW



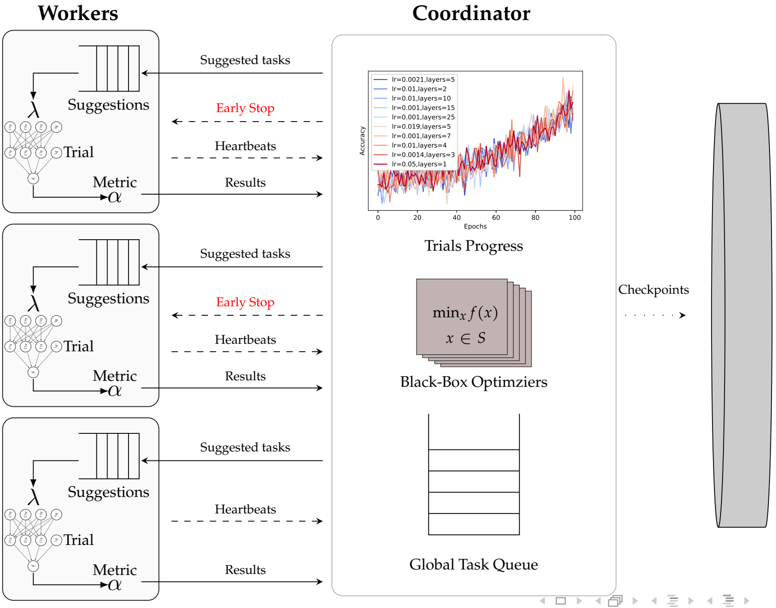
Asynchronous Search Workflow



ASYNCHRONOUS SEARCH WORKFLOW



ASYNCHRONOUS SEARCH WORKFLOW



SUMMARY

- Deep Learning is going distributed
- Algorithms for DDL are available in several frameworks
- Applying DDL in practice brings a lot of operational complexity
- Hopsworks is a platform for scale out deep learning and big data processing
- Hopsworks makes DDL simpler by providing simple abstractions for distributed training, parallel experiments and much more..

@hopshadoop www.hops.io

@logicalclocks www.logicalclocks.com

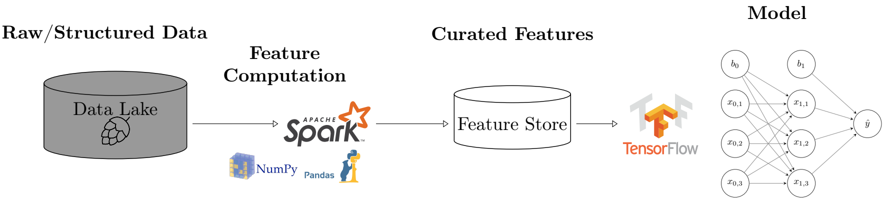
We are open source:
https://github.com/logicalclocks/hopsworks
https://github.com/hopshadoop/hops

Thanks to Logical Clocks Team: Jim Dowling, Seif Haridi, Theo Kakantousis, Fabio Buso, Gautier Berthou, Ermias Gebremeskel, Mahmoud Ismail, Salman Niazi, Antonios Kouzoupis, Robin Andersson, Alex Ormenisan, Rasmus Toivonen and Steffen Grohsschmiedt.

During the break..

1. Register for an account at: www.hops.site
2. Follow the instructions at: http://bit.ly/2OI4Ggt
3. Cheatsheet (for copy-paste): http://snurran.sics.se/hops/kim/workshop_cheat.txt

Demo-Setting



Hands-on Workshop

1. If you haven't registered, do it now on hops.site
2. Cheatsheet: http://snurran.sics.se/hops/kim/workshop_cheat.txt
3. Python API Docs: http://hops-py.logicalclocks.com/

EXERCISE 1 (HELLO HOPSWORKS)

1. Create a Deep Learning Tour Project on Hopsworks
2. Start a Jupyter Notebook with the config:
   - "Experiment" Mode
   - 1 GPU
   - 4000 (MB) memory for the driver (appmaster)
   - 8000 (MB) memory for the executor
   - Rest can be default
3. Create a new "PySpark" notebook
4. In the first cell, write:

```
print("Hello Hopsworks")
```

5. Execute the cell (Ctrl + <Enter>)

EXERCISE 2 (DISTRIBUTED HELLO HOPSWORKS WITH GPU)

1. Add a new cell with the contents:

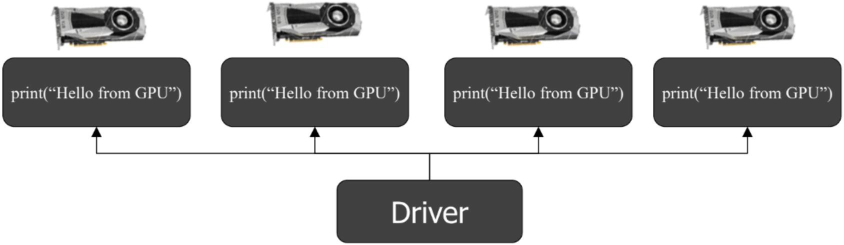
```python
def executor():
    print("Hello from GPU")
```

2. Add a new cell with the contents:

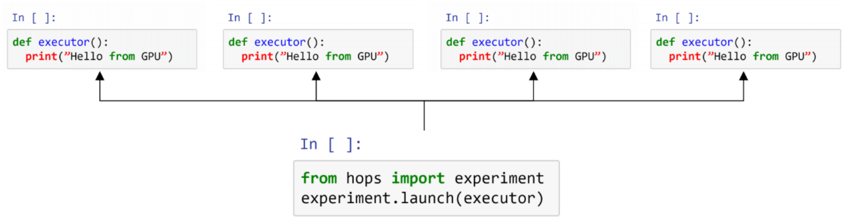
```python
from hops import experiment
experiment.launch(executor)
```

3. Execute the two cells in order (Ctrl + <Enter>)
4. Go to the Application UI

EXERCISE 2 (DISTRIBUTED HELLO HOPSWORKS WITH GPU)



Exercise 2 (Distributed Hello Hopsworks with GPU)



EXERCISE 3 (SAVE DATA IN THE FEATURE STORE)

1. Enable the Feature Store service in your project
2. Add a new cell with the contents:

```
from hops import featurestore
import numpy as np
featurestore.create_featuregroup(np.random.rand(20,10), "eit_school")
```

3. Add a new cell with the contents:

```
featurestore.get_featuregroup("eit_school").show(5)
```

4. Add a new cell with the contents:

```
%%local
%matplotlib inline
from hops import featurestore
featurestore.visualize_featuregroup_correlations("eit_school")
```

5. Execute the cells in order and then go to the featurestore registry

EXERCISE 4 (LOAD MNIST FROM HOPSFS)

1. Add a new cell with the contents:

```
from hops import hdfs
import tensorflow as tf
def create_tf_dataset():
    train_files = [hdfs.project_path() +
                  "TestJob/data/mnist/train/train.tfrecords"]
    dataset = tf.data.TFRecordDataset(train_files)
    def decode(example):
        example = tf.parse_single_example(example,{
                    'image_raw': tf.FixedLenFeature([], tf.string),
                    'label': tf.FixedLenFeature([], tf.int64)})
        image = tf.reshape(tf.decode_raw(example['image_raw'],
                        tf.uint8), (28,28,1))
        label = tf.one_hot(tf.cast(example['label'], tf.int32), 10)
        return image, label
    return dataset.map(decode).batch(128).repeat()
```

EXERCISE 4 (LOAD MNIST FROM HOPSFS)

2. Add a new cell with the contents:

```
create_tf_dataset()
```

3. Execute the two cells in order (Ctrl + <Enter>)

EXERCISE 5 (DEFINE CNN MODEL)

```python
from tensorflow import keras
def create_model():
    model = keras.Sequential()
    model.add(keras.layers.Conv2D(filters=32, kernel_size=3, padding='same',
                              activation='relu', input_shape=(28,28,1)))
    model.add(keras.layers.BatchNormalization())
    model.add(keras.layers.MaxPooling2D(pool_size=2))
    model.add(keras.layers.Dropout(0.3))
    model.add(keras.layers.Conv2D(filters=64, kernel_size=3,
                              padding='same', activation='relu'))
    model.add(keras.layers.BatchNormalization())
    model.add(keras.layers.MaxPooling2D(pool_size=2))
    model.add(keras.layers.Dropout(0.3))
    model.add(keras.layers.Flatten())
    model.add(keras.layers.Dense(128, activation='relu'))
    model.add(keras.layers.Dropout(0.5))
    model.add(keras.layers.Dense(10, activation='softmax'))
    return model
```

EXERCISE 5 (DEFINE CNN MODEL)

2. Add a new cell with the contents:

```
create_model().summary()
```

3. Execute the two cells in order (Ctrl + <Enter>)

EXERCISE 6 (DEFINE & RUN THE EXPERIMENT)

1. Add a new cell with the contents:

```
from hops import tensorboard
from tensorflow.python.keras.callbacks import TensorBoard
def train_fn():
    dataset = create_tf_dataset()
    model = create_model()
    model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=keras.optimizers.Adam(),metrics=['accuracy'])
    tb_callback = TensorBoard(log_dir=tensorboard.logdir())
    model_ckpt_callback = keras.callbacks.ModelCheckpoint(
                            tensorboard.logdir(), monitor='acc')
    history = model.fit(dataset, epochs=50,
                    steps_per_epoch=80, callbacks=[tb_callback])
    return history.history["acc"][-1]
```

EXERCISE 6 (DEFINE & RUN THE EXPERIMENT)

2. Add a new cell with the contents:

```
experiment.launch(train_fn)
```

3. Execute the two cells in order (Ctrl + <Enter>)
4. Go to the Application UI and monitor the training progress

REFERENCES

- Example notebooks https://github.com/logicalclocks/hops-examples
- HopsML[6]
- Hopsworks[7]
- Hopsworks' feature store[8]
- Maggy https://github.com/logicalclocks/maggy
- HopsFS[9]

2em1[6] Logical Clocks AB. *HopsML: Python-First ML Pipelines*. https://hops.readthedocs.io/en/latest/hopsml/hopsML.html. 2018.

2em1[7] Jim Dowling. *Introducing Hopsworks*. https://www.logicalclocks.com/introducing-hopsworks/. 2018.

2em1[8] Kim Hammar and Jim Dowling. *Feature Store: the missing data layer in ML pipelines?* https://www.logicalclocks.com/feature-store/. 2018.

2em1[9] Salman Niazi et al. "HopsFS: Scaling Hierarchical File System Metadata Using NewSQL Databases". In: *15th USENIX Conference on File and Storage Technologies (FAST 17)*. Santa Clara, CA: USENIX Association, 2017, pp. 89–104. ISBN: 978-1-931971-36-2. URL: https://www.usenix.org/conference/fast17/technical-sessions/presentation/niazi.