Distributed Deep Learning Using Hopsworks

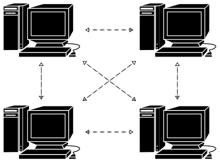
SF Machine Learning
Mesosphere

Kim Hammar
kim@logicalclocks.com

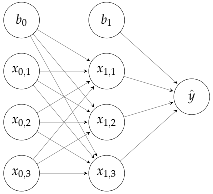
LOGICAL CLOCKS

DISTRIBUTED COMPUTING + DEEP LEARNING = ?

Distributed Computing



Deep Learning



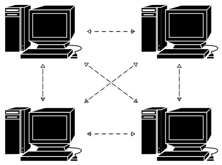
Why Combine the two?

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.

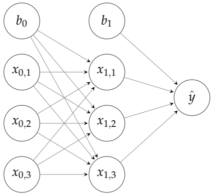
2em1[2] Jeffrey Dean et al. "Large Scale Distributed Deep Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1223–1231.

DISTRIBUTED COMPUTING + DEEP LEARNING = ?

Distributed Computing



Deep Learning



Why Combine the two?

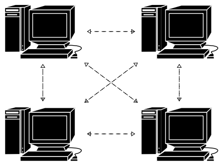
- We like challenging problems ☺

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.

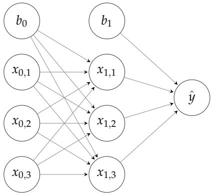
2em1[2] Jeffrey Dean et al. “Large Scale Distributed Deep Networks”. In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1223–1231.

DISTRIBUTED COMPUTING + DEEP LEARNING = ?

Distributed Computing



Deep Learning



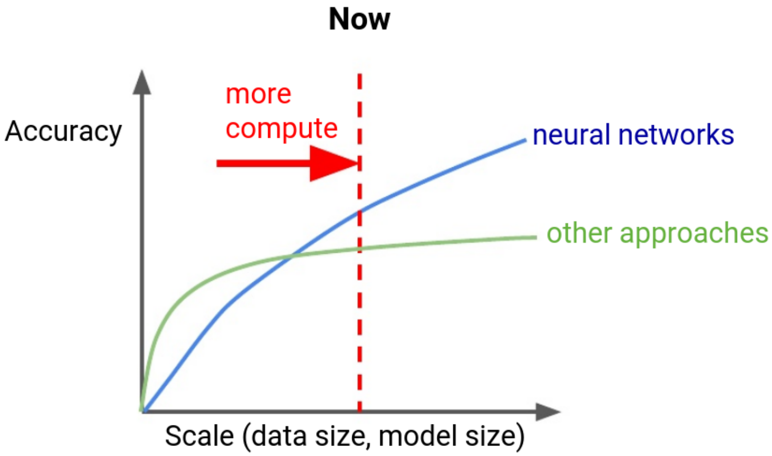
Why Combine the two?

- 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.

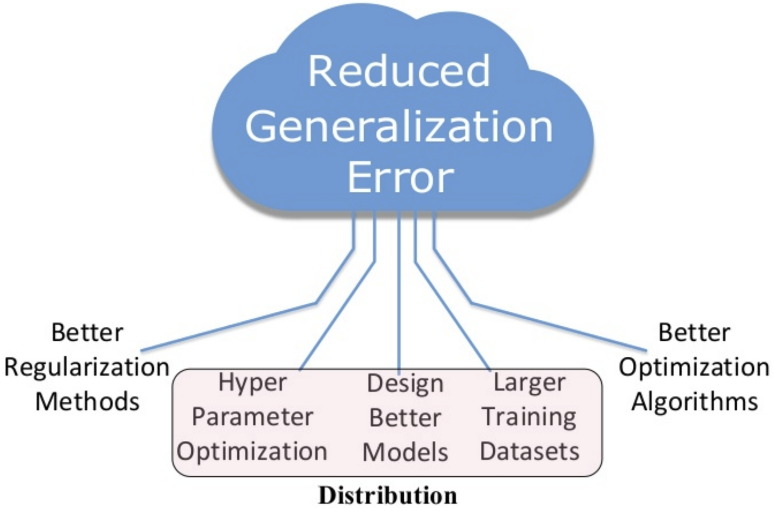
DISTRIBUTED DEEP LEARNING (DDL): PREDICTABLE SCALING



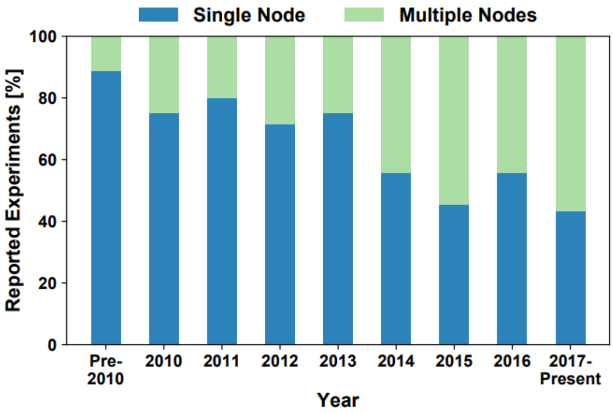
3

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


Reduced Generalization Error
Better Regularization Methods
Hyper Parameter Optimization
Design Better Models
Larger Training Datasets
Better Optimization Algorithms
Distribution

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

Amazon SageMaker

HOROVOD

Microsoft CNTK

BigDL

TensorflowOnSpark

Distributed TF

CaffeOnSpark

PYTORCH

APACHE Spark

mxnet

DL4J

Chainer MN

TonY

Companies using DDL

facebook

Google

UBER

zalando

YAHOO!

NETFLIX

amazon

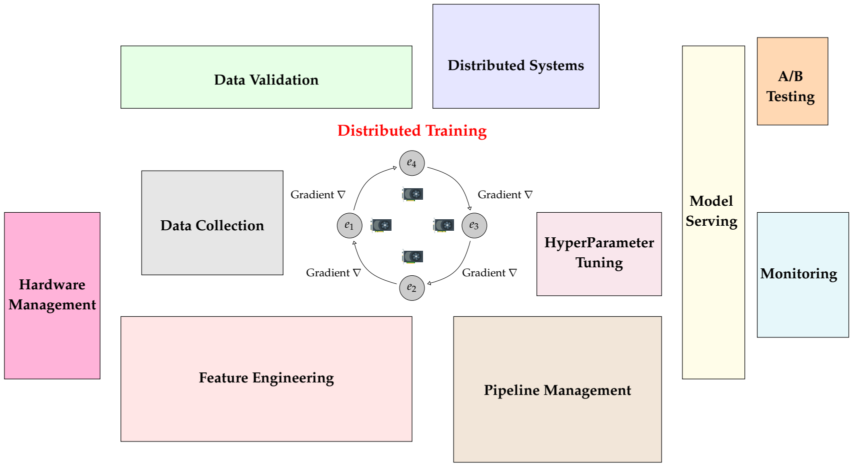
LinkedIn

Bai du 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** using Hopsworks, Metadata Store, PySpark, and Maggy[5]

2em1[5] Moritz Meister and Sina Sheikholeslami. *Maggy*. https://github.com/logicalclocks/maggy. 2019.

OUTLINE

1. **Hopsworks**: Background of the platform
2. **Managed Distributed Deep Learning** using HopsYARN, HopsML, PySpark, and Tensorflow
3. **Black-Box Optimization** using Hopsworks, Metadata

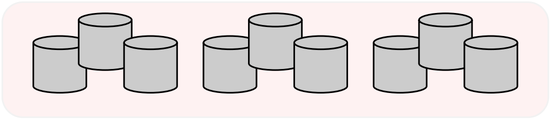
Store, PySpark, and Maggy[6]

2em1[6] Moritz Meister and Sina Sheikholeslami. *Maggy*. https://github.com/logicalclocks/maggy. 2019.

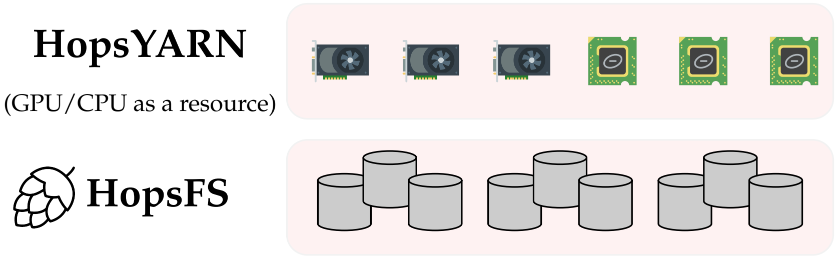
HOPSWORKS

HOPSWORKS

HopsFS



HOPSWORKS



HOPSWORKS

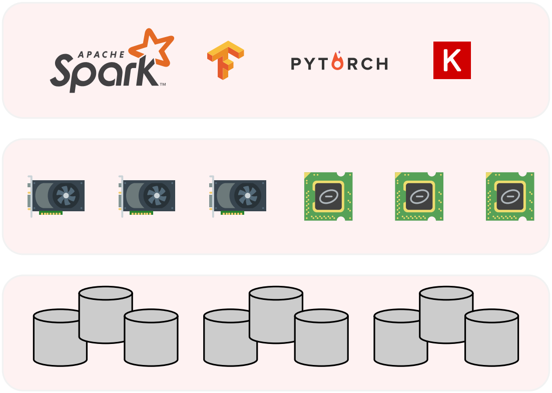
Frameworks

(ML/Data)

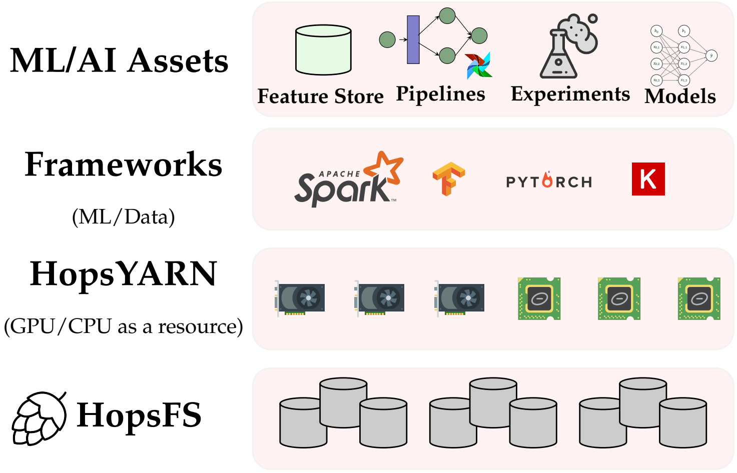
HopsYARN

(GPU/CPU as a resource)

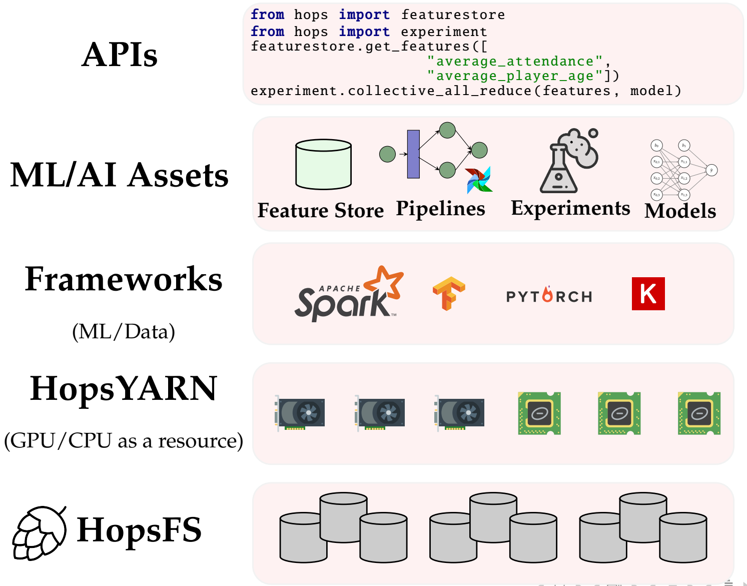
HopsFS



Hopsworks


ML/AI Assets
Feature Store
Pipelines
Experiments
Models
Frameworks
(ML/Data)
APACHE Spark
PYTORCH
K
HopsYARN
(GPU/CPU as a resource)
HopsFS

HOPSWORKS



APIs

```
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

Feature Store **Pipelines** **Experiments** **Models**

Frameworks

(ML/Data)

HopsYARN

(GPU/CPU as a resource)

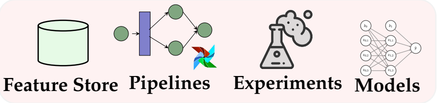
HopsFS

HOPSWORKS

APIs

```
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

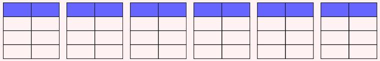
(ML/Data)

HopsYARN

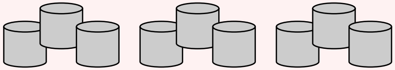
(GPU/CPU as a resource)

Distributed Metadata

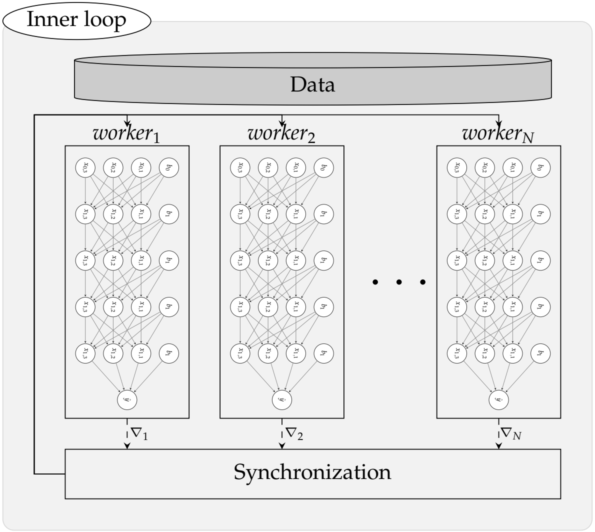
(Available from REST API)



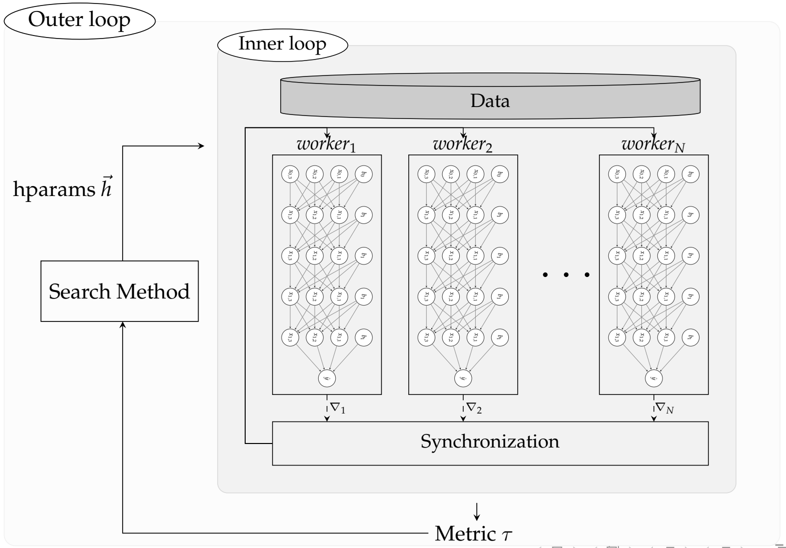
HopsFS



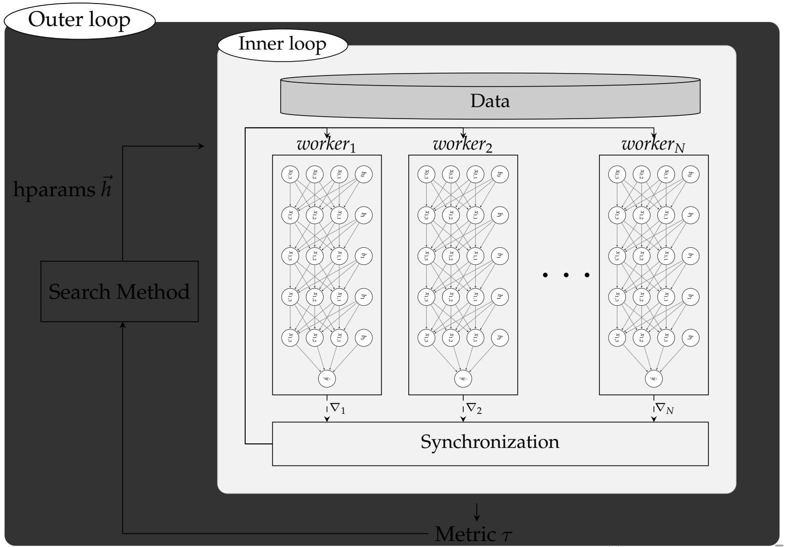
INNER AND OUTER LOOP OF LARGE SCALE DEEP LEARNING



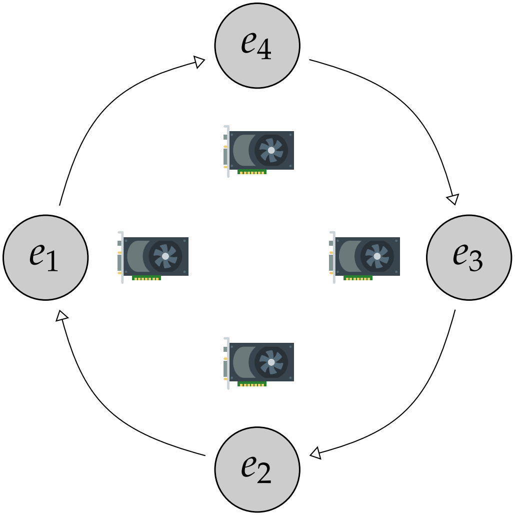
INNER AND OUTER LOOP OF LARGE SCALE DEEP LEARNING



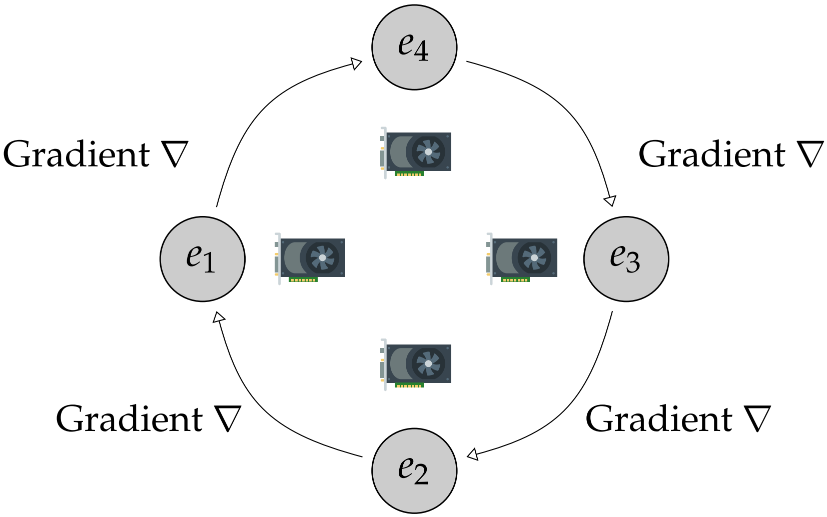
Inner and Outer Loop of Large Scale Deep Learning



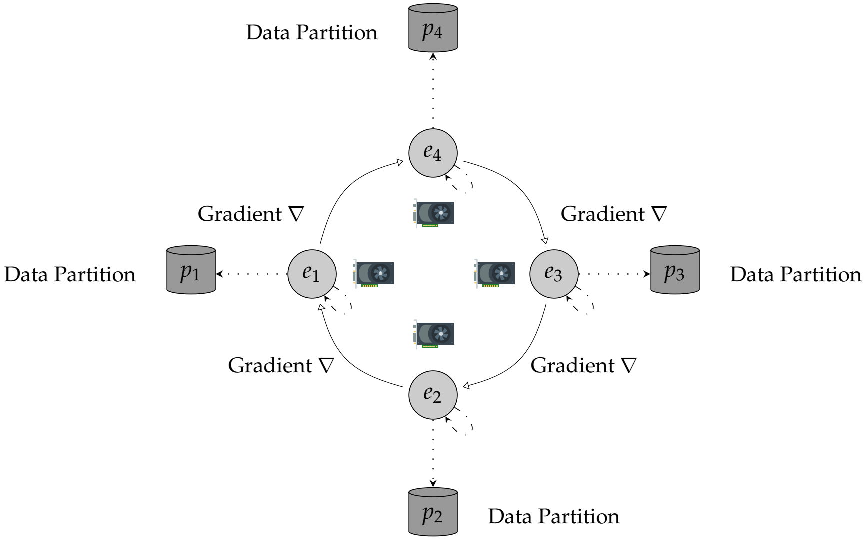
INNER LOOP: DISTRIBUTED DEEP LEARNING



Inner Loop: Distributed Deep Learning



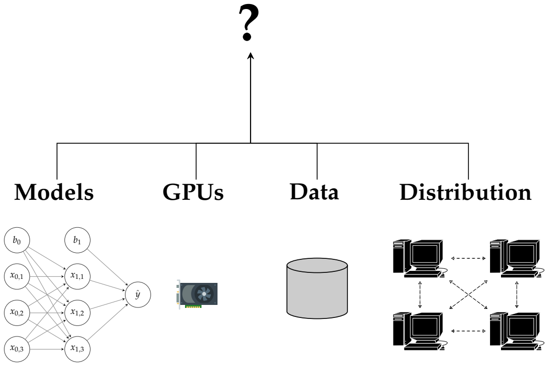
INNER LOOP: DISTRIBUTED DEEP LEARNING



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

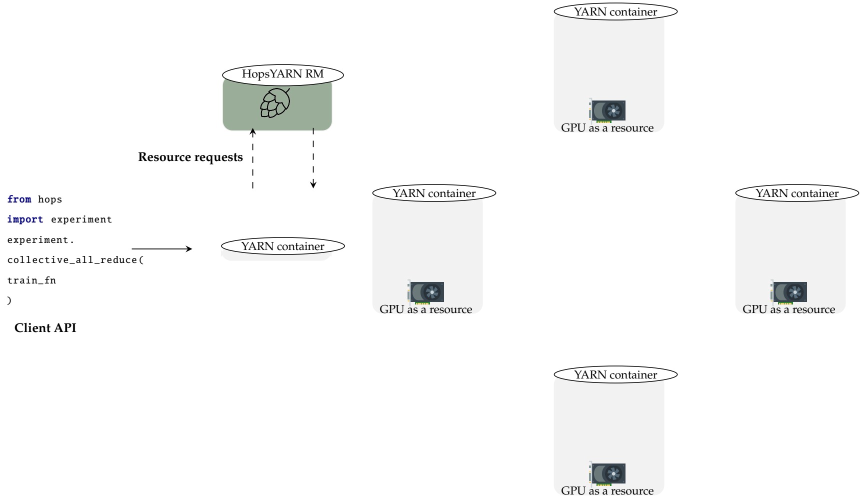


HOPSWORKS DDL SOLUTION

HOPSWORKS DDL SOLUTION

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

HOPSWORKS DDL SOLUTION

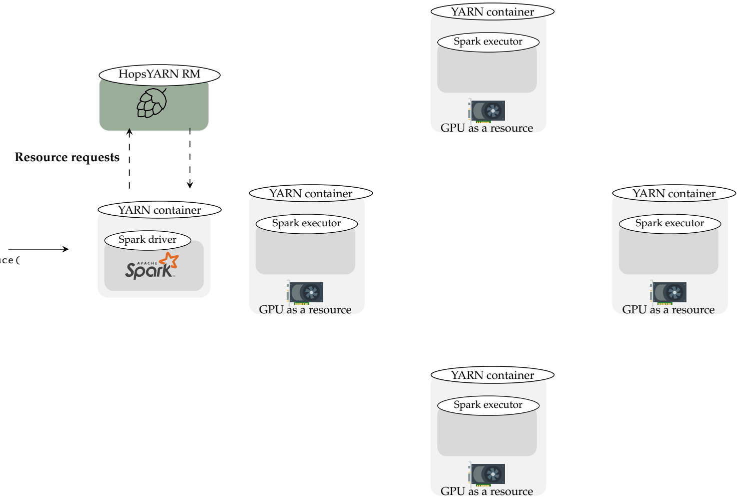


```
from hops
import experiment
experiment.
collective_all_reduce(
train_fn
)
```

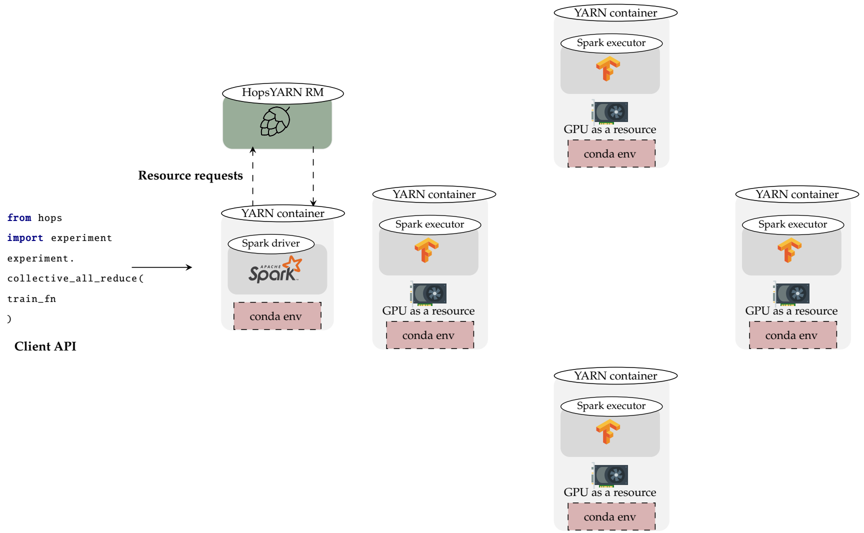
HOPSWORKS DDL SOLUTION

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

Client API

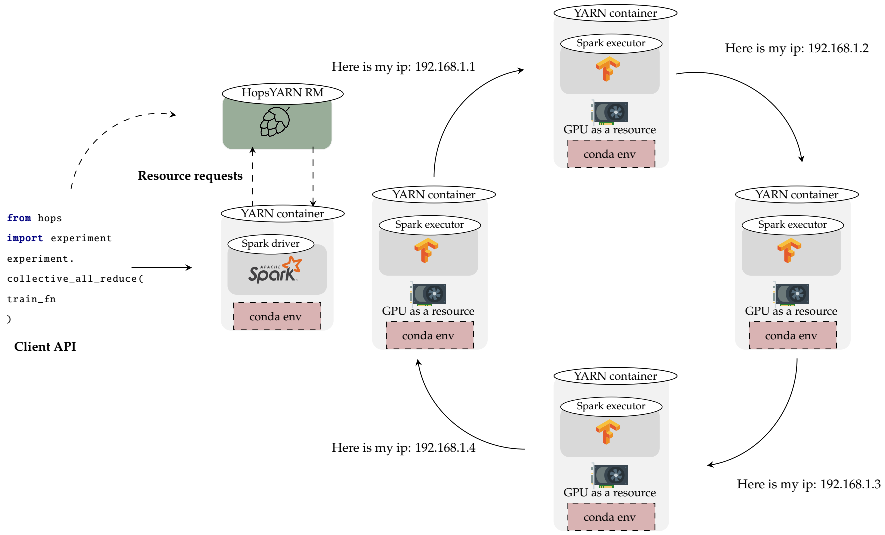


HOPSWORKS DDL SOLUTION



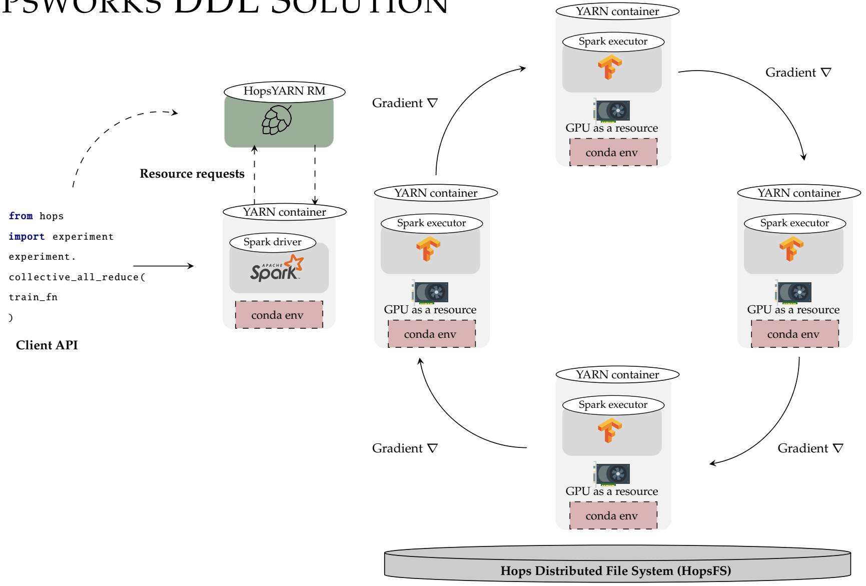
```
from hops
import experiment
experiment.
collective_all_reduce(
train_fn
)
```

HOPSWORKS DDL SOLUTION



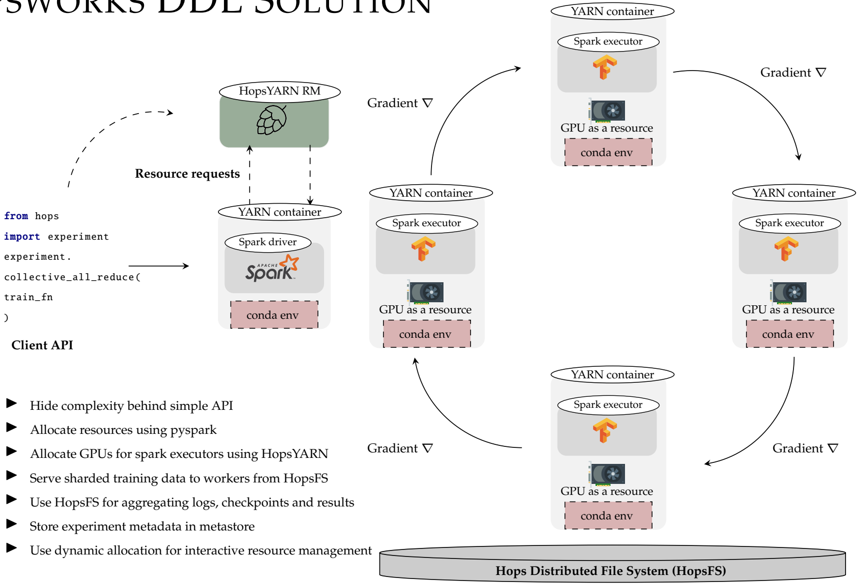
```
from hops
import experiment
experiment.
collective_all_reduce(
train_fn
)
```

HOPSWORKS DDL SOLUTION



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

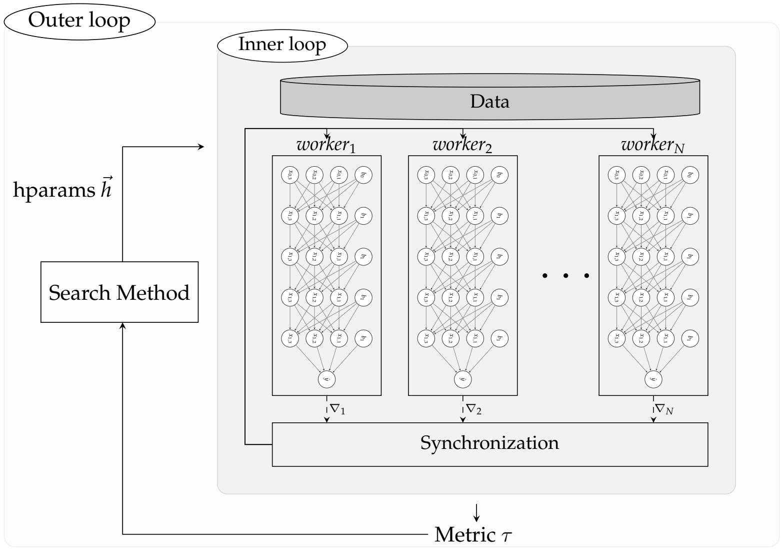
HOPSWORKS DDL SOLUTION



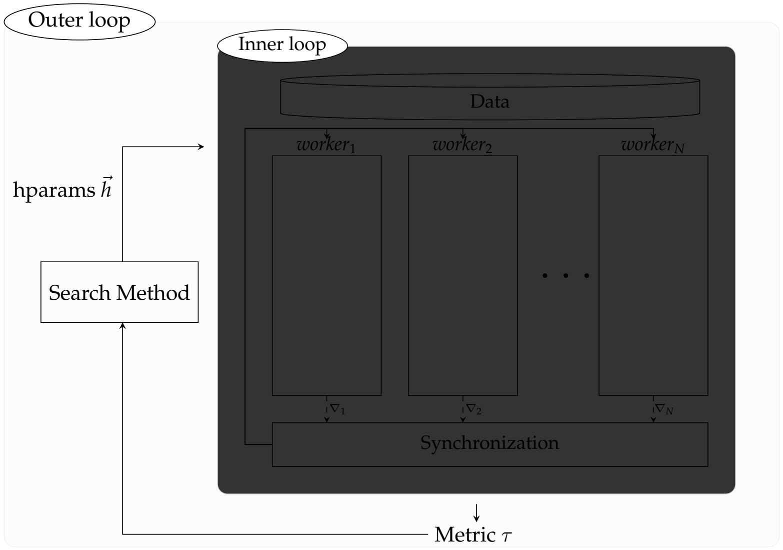
```
from hops
import experiment
experiment.
collective_all_reduce(
train_fn
)
```

- 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

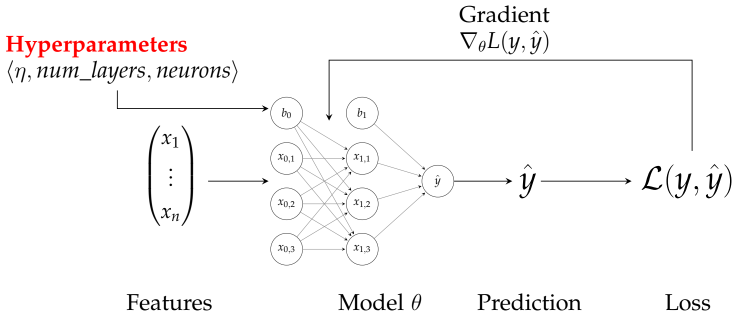
OUTER LOOP: BLACK BOX OPTIMIZATION



OUTER LOOP: BLACK BOX OPTIMIZATION



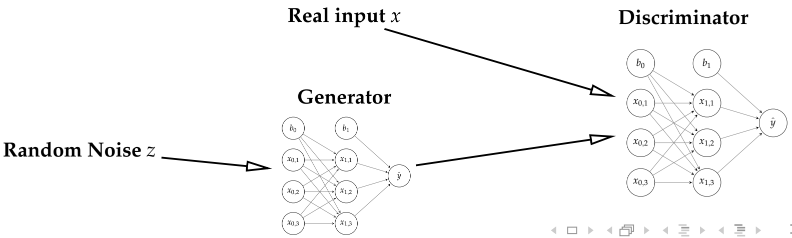
OUTER LOOP: BLACK BOX OPTIMIZATION



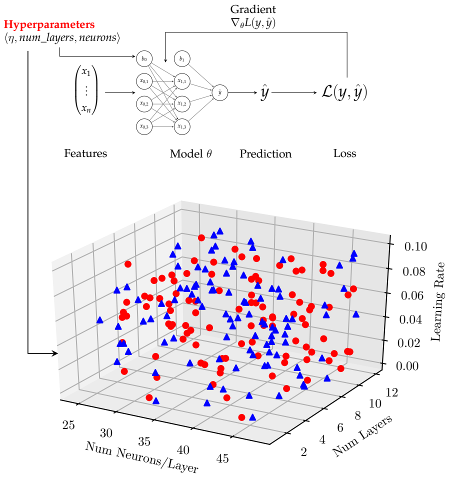
OUTER LOOP: BLACK BOX OPTIMIZATION

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.

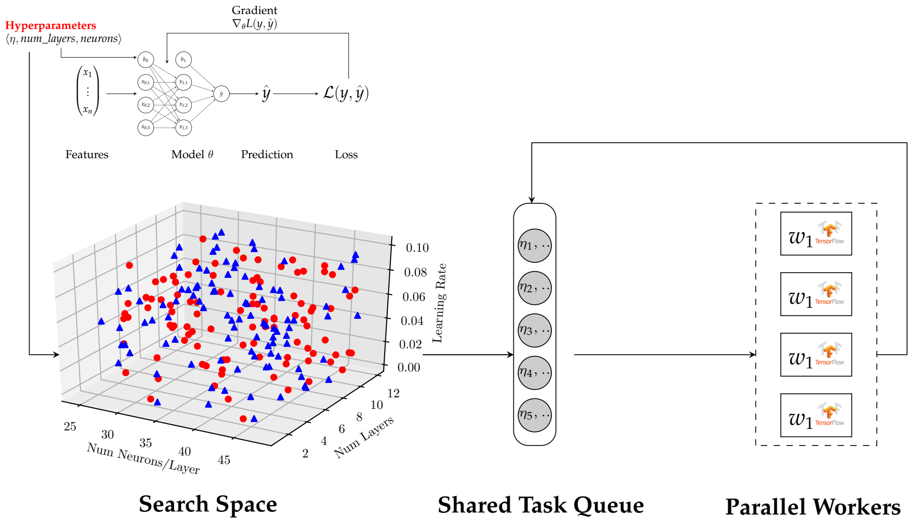


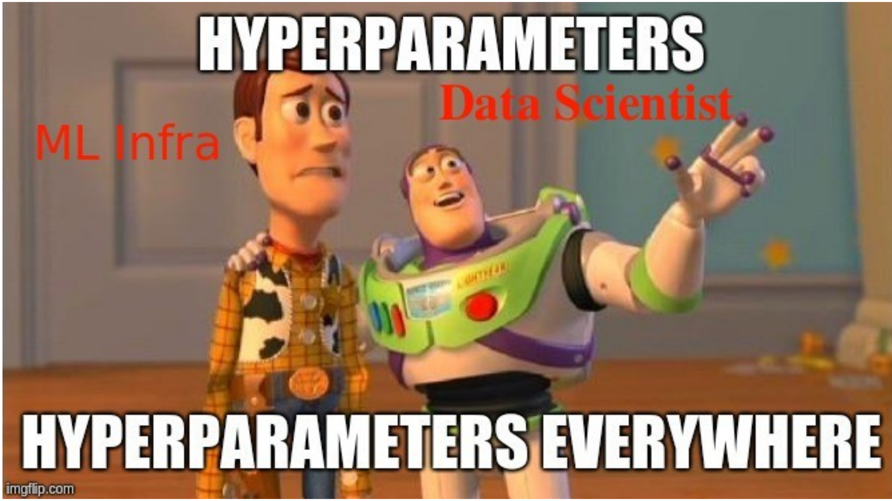
OUTER LOOP: BLACK BOX OPTIMIZATION



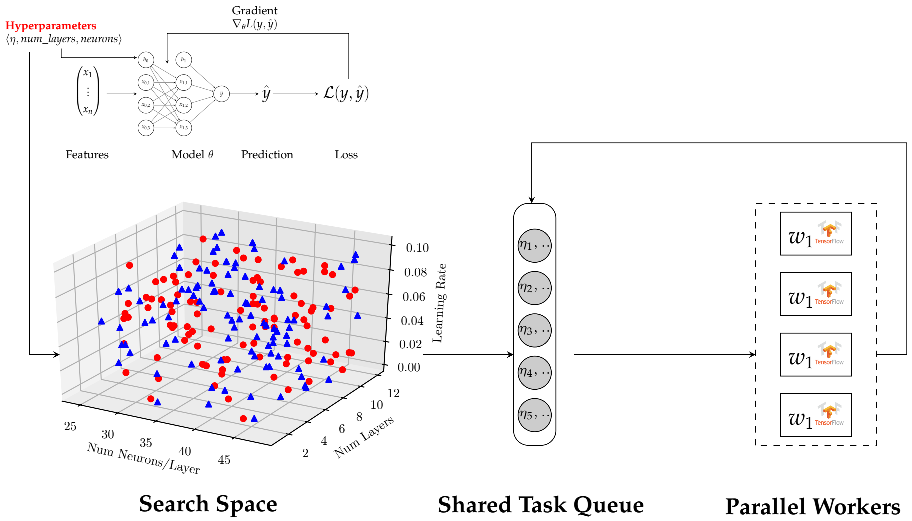
Search Space

OUTER LOOP: BLACK BOX OPTIMIZATION

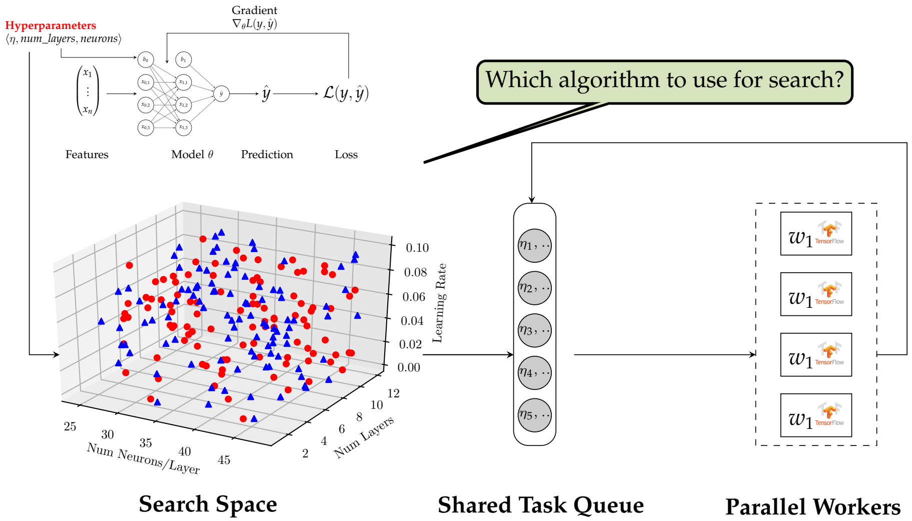



HYPERPARAMETERS
ML Infra
Data Scientist
HYPERPARAMETERS EVERYWHERE
imgflip.com

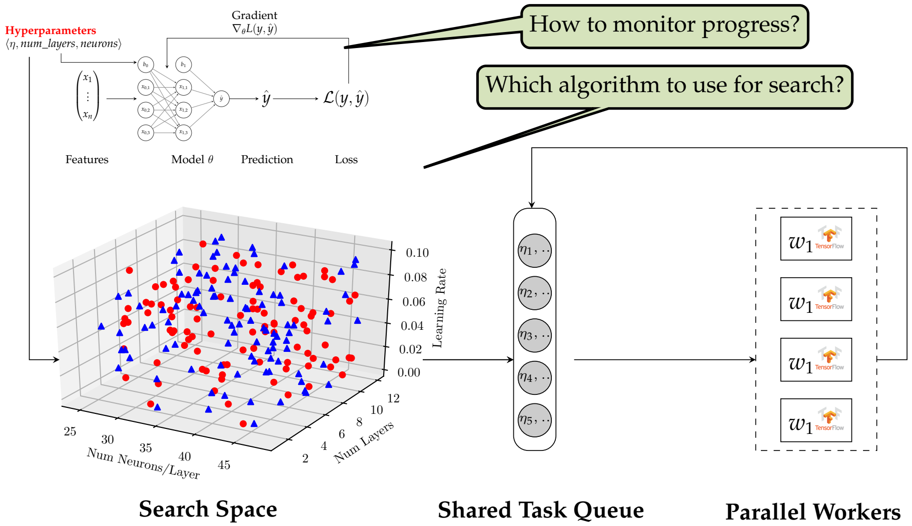
Outer Loop: Black Box Optimization



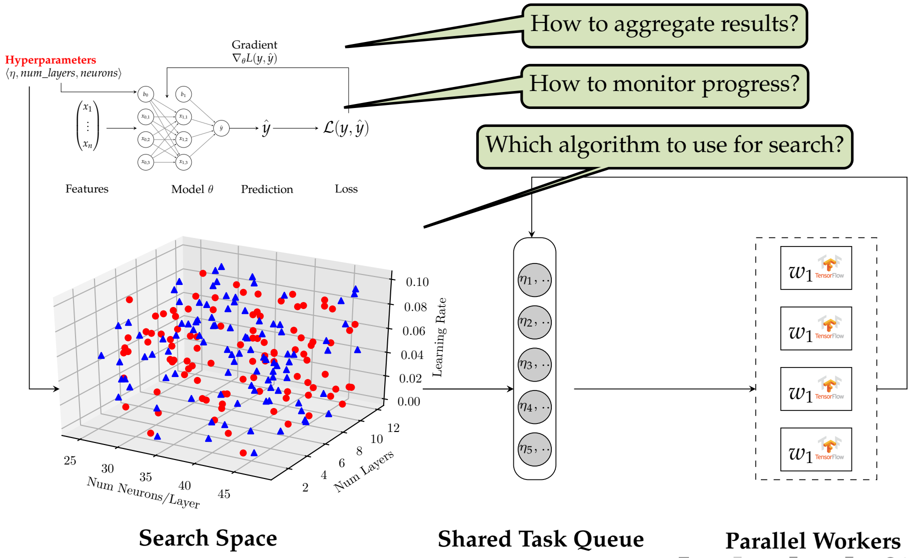
Outer Loop: Black Box Optimization



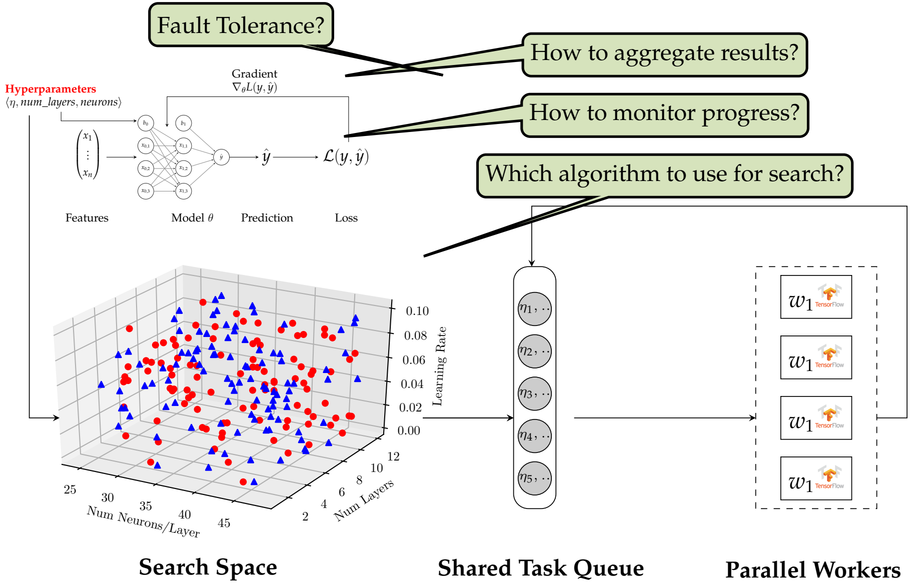
OUTER LOOP: BLACK BOX OPTIMIZATION



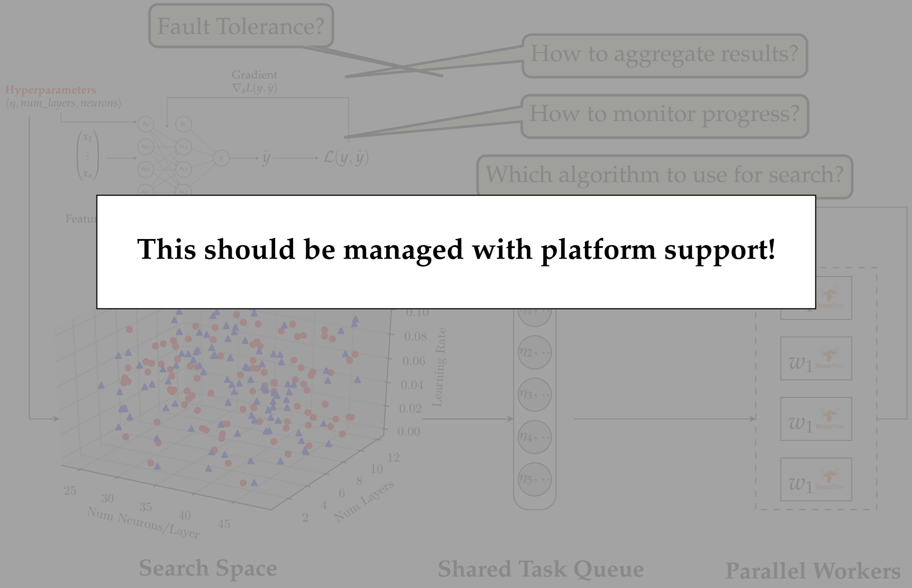
OUTER LOOP: BLACK BOX OPTIMIZATION



OUTER LOOP: BLACK BOX OPTIMIZATION



Outer Loop: Black Box Optimization



This should be managed with platform support!

MAGGY: A FRAMEWORK FOR SYNCHRONOUS/ASYNCHRONOUS HYPERPARAMETER TUNING ON HOPSWORKS[7]

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

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

2em1[7] Authors of Maggy: Moritz Meister and Sina Sheikholeslami. Author of the base framework that Maggy builds on: Robin Andersson

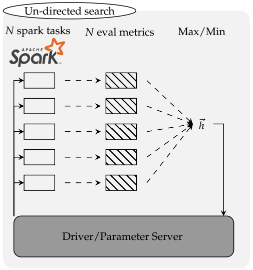
MAGGY: A FRAMEWORK FOR SYNCHRONOUS/ASYNCHRONOUS HYPERPARAMETER TUNING ON HOPSWORKS[7]

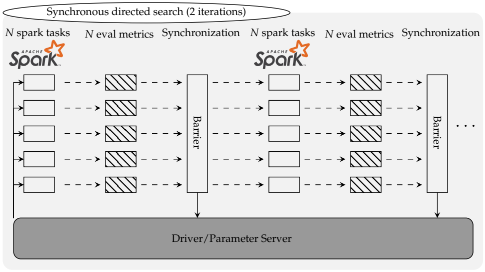
A flexible framework for running different black-box optimization algorithms on Hopsworks

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

2em1[7] 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

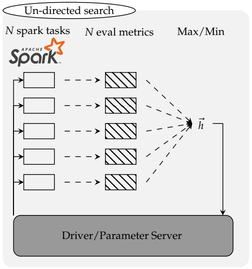


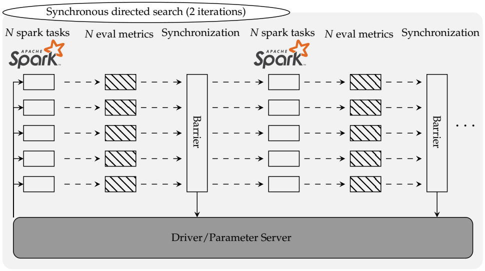


- Parallel undirected/synchronous search is trivial using Spark and a distributed file system
- Example of un-directed search algorithms: **random and grid search**
- Example of synchronous search algorithms: **differential evolution**

FRAMEWORK SUPPORT FOR SYNCHRONOUS SEARCH ALGORITHMS

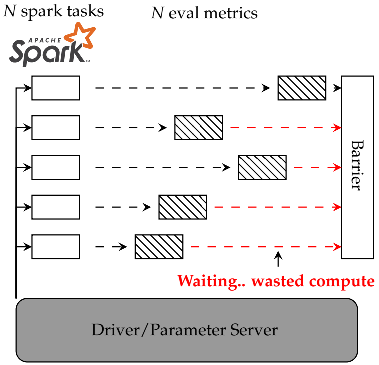
Fits very well with Spark BSP model





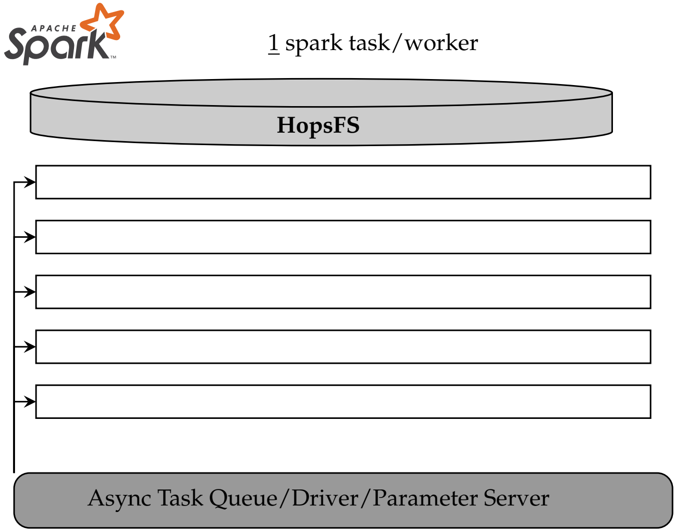
- Parallel un-directed/synchronous search is trivial using Spark and a distributed file system
- Example of un-directed search algorithms: **random and grid search**
- Example of synchronous search algorithms: **differential evolution**

PROBLEM WITH THE BULK-SYNCHRONOUS PROCESSING MODEL FOR PARALLEL SEARCH

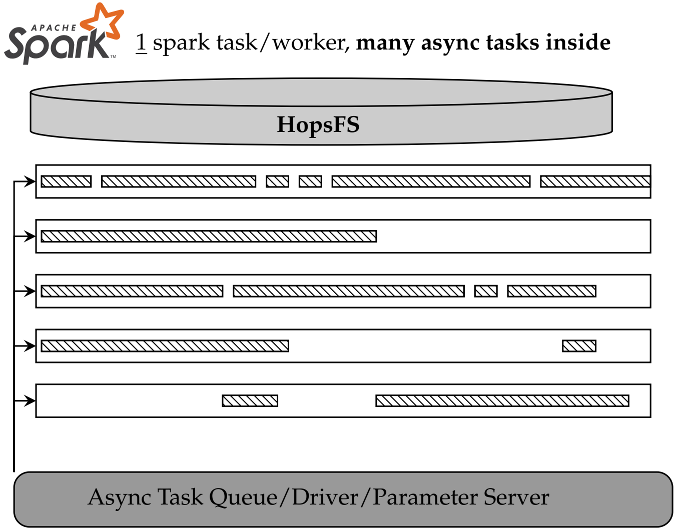


- Synchronous search is sensitive to stragglers and not suitable for early stopping
- ... For large scale search problems we need asynchronous search
- **Problem:** Asynchronous search is much harder to implement with big data processing tools such as Spark

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

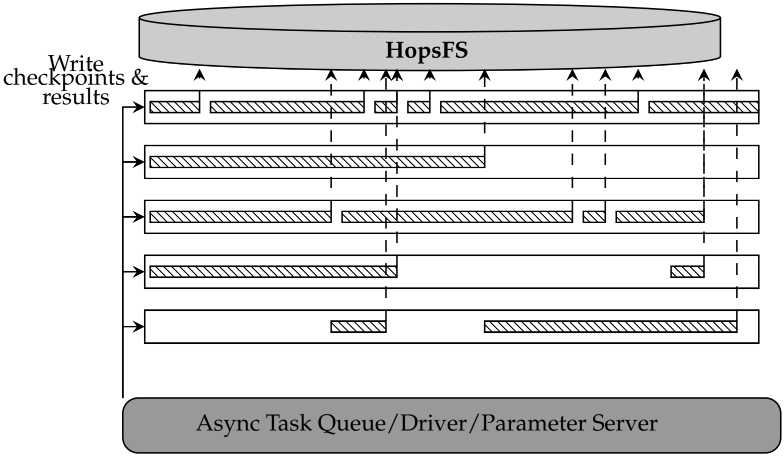


ENTER MAGGY: A FRAMEWORK FOR RUNNING ASYNCHRONOUS SEARCH ALGORITHMS ON HOPS

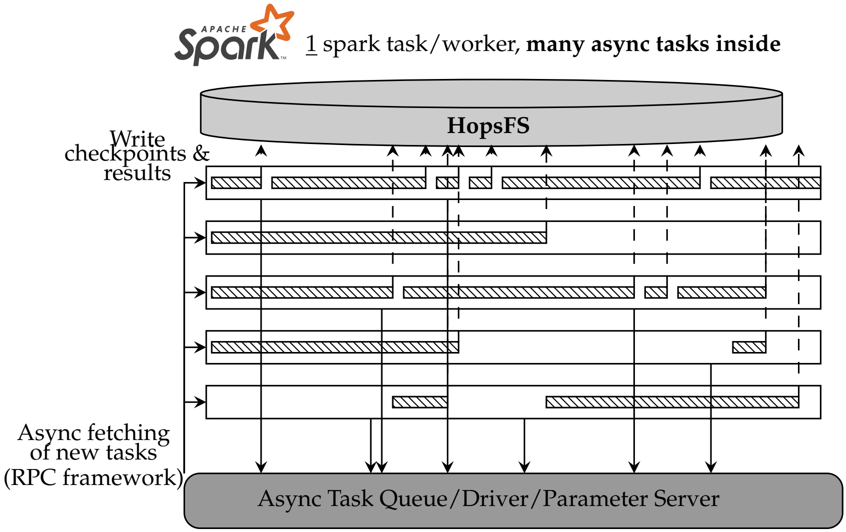

1 spark task/worker, **many async tasks inside**
HopsFS
Async Task Queue/Driver/Parameter Server

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

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

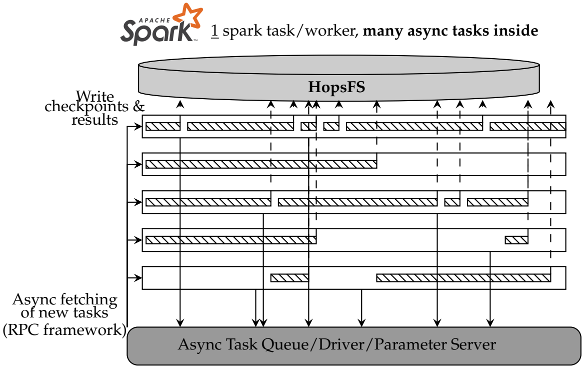

HopsFS
Write checkpoints & results
Async Task Queue/Driver/Parameter Server

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

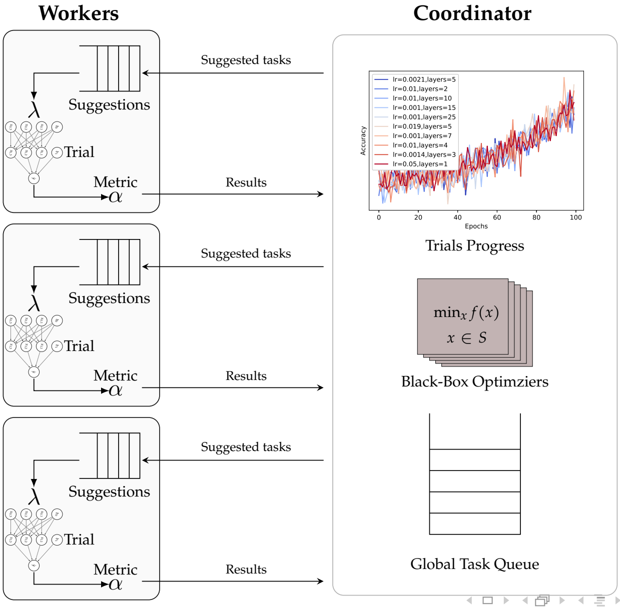

APACHE Spark
1 spark task/worker, **many async tasks inside**
HopsFS
Write checkpoints & results
Async fetching of new tasks (RPC framework)
Async Task Queue/Driver/Parameter Server

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

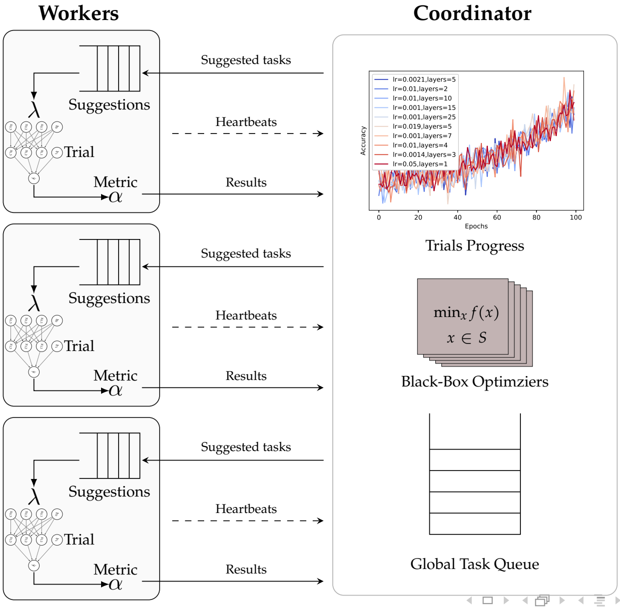
- Robust against stragglers
- Supports early stopping
- Fault tolerance with checkpointing
- Monitoring with Tensorboard
- Log aggregation with HopsFS
- Simple API and extendable


1 spark task/worker, **many async tasks inside**
HopsFS
Write checkpoints & results
Async fetching of new tasks (RPC framework)
Async Task Queue/Driver/Parameter Server

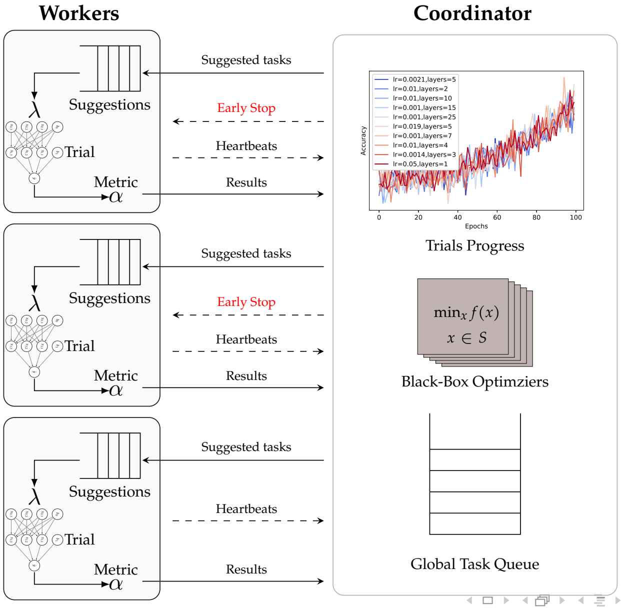
MAGGY: ASYNCHRONOUS SEARCH WORKFLOW



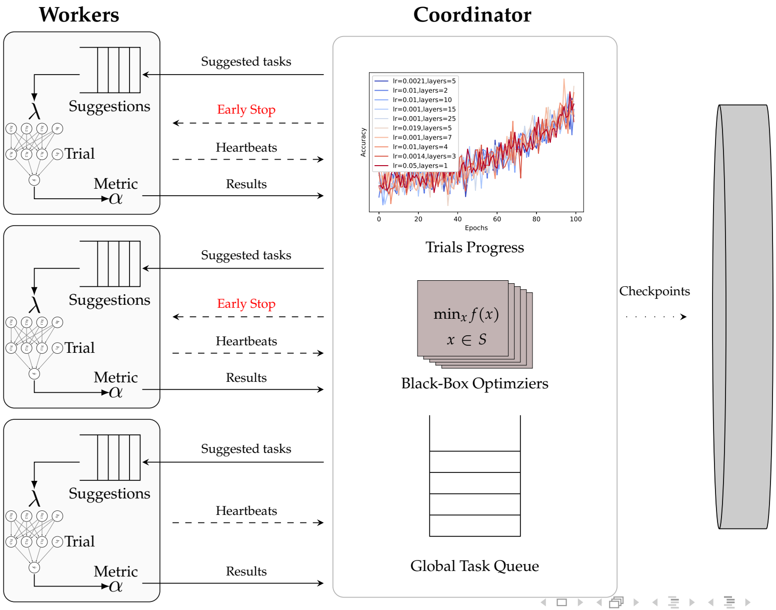
MAGGY: ASYNCHRONOUS SEARCH WORKFLOW



MAGGY: ASYNCHRONOUS SEARCH WORKFLOW



MAGGY: ASYNCHRONOUS SEARCH WORKFLOW



MAGGY: API

```
class RandomSearch(AbstractOptimizer):

  def initialize(self):
       # ..

  def get_suggestion(self, trial=None):
      # ..

  def finalize_experiment(self, trials):
     # ..

  def early_check(self, to_check, trials, direction):
     # ..
```

MAGGY: API

Users have to extend the AbstractOptimizer base class to implement their own algorithms.

```
class RandomSearch(AbstractOptimizer):

    def initialize(self):
        # ..

    def get_suggestion(self, trial=None):
        # ..

    def finalize_experiment(self, trials):
        # ..

    def early_check(self, to_check,
                    trials, direction):
        # ..
```

MAGGY: API

Users have to extend the AbstractOptimizer base class to implement their own algorithms.

Initializing search space

```
class RandomSearch(AbstractOptimizer):

  def initialize(self):
        # ..

  def get_suggestion(self, trial=None):
       # ..

  def finalize_experiment(self, trials):
      # ..

 def early_check(self, to_check,
                        trials, direction):
      # ..
```

MAGGY: API

Users have to extend the AbstractOptimizer base class to implement their own algorithms.

Initializing search space

Suggestions to be evaluated by workers

```
class RandomSearch(AbstractOptimizer):

  def initialize(self):
      # ..

  def get_suggestion(self, trial=None):
      # ..

  def finalize_experiment(self, trials):
      # ..

  def early_check(self, to_check,
                  trials, direction):
      # ..
```

MAGGY: API

Users have to extend the AbstractOptimizer base class to implement their own algorithms.

Initializing search space

Suggestions to be evaluated by workers

Aggregate results

```
class RandomSearch(AbstractOptimizer):

  def initialize(self):
      # ..

  def get_suggestion(self, trial=None):
      # ..

  def finalize_experiment(self, trials):
      # ..

  def early_check(self, to_check,
                  trials, direction):
      # ..
```

MAGGY: API

Users have to extend the AbstractOptimizer base class to implement their own algorithms.

Initializing search space

Suggestions to be evaluated by workers

Aggregate results

Configure early-stop policy

```
class RandomSearch(AbstractOptimizer):

    def initialize(self):
        # ..

    def get_suggestion(self, trial=None):
        # ..

    def finalize_experiment(self, trials):
        # ..

    def early_check(self, to_check,
                    trials, direction):
        # ..
```

MAGGY: API

```
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")
```

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

LOGICAL CLOCKS

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, and Rasmus Toivonen. And our interns: Moritz Meister and Sina Sheikholeslami.

REFERENCES

- Example notebooks https://github.com/logicalclocks/hops-examples
- HopsML[8]
- Hopsworks[9]
- Hopsworks' feature store[10]
- Maggy https://github.com/logicalclocks/maggy

2em1[8] Logical Clocks AB. *HopsML: Python-First ML Pipelines*. https://hops.readthedocs.io/en/latest/hopsml/hopsML.html. 2018.

2em1[9] Jim Dowling. *Introducing Hopsworks*. https://www.logicalclocks.com/introducing-hopsworks/. 2018.

2em1[10] Kim Hammar and Jim Dowling. *Feature Store: the missing data layer in ML pipelines?* https://www.logicalclocks.com/feature-store/. 2018.