Distribution Oblivious Training Functions for ML

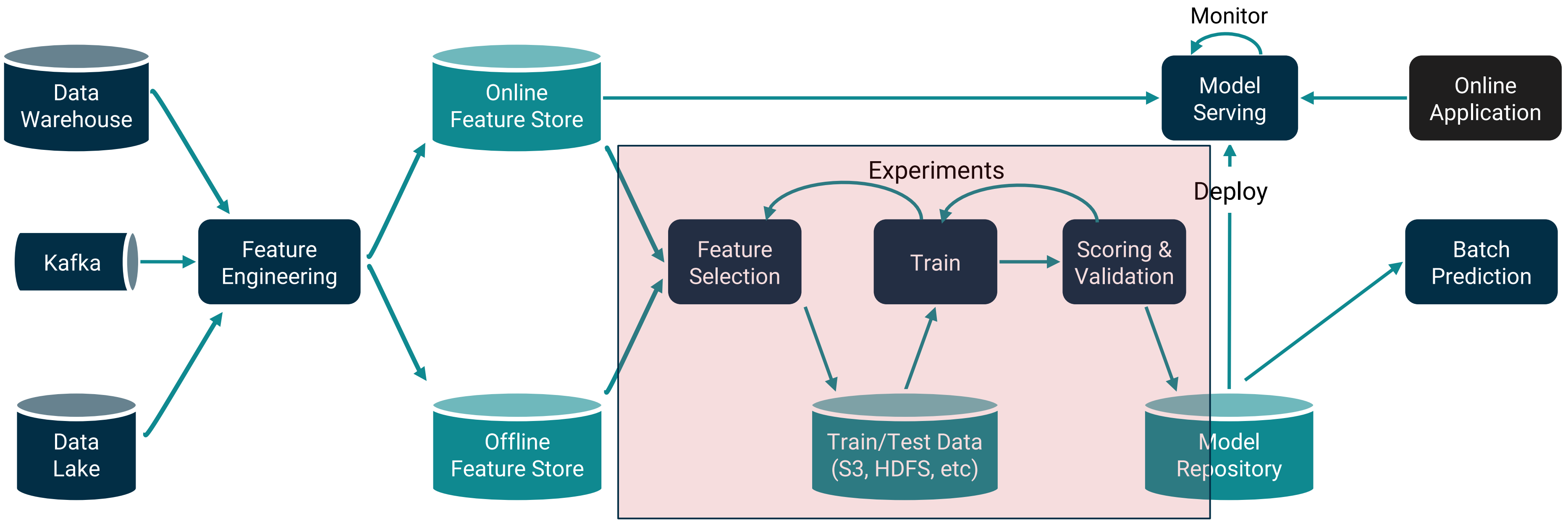
•••

Jim Dowling
Assoc Prof @ KTH
CEO @ Logical Clocks

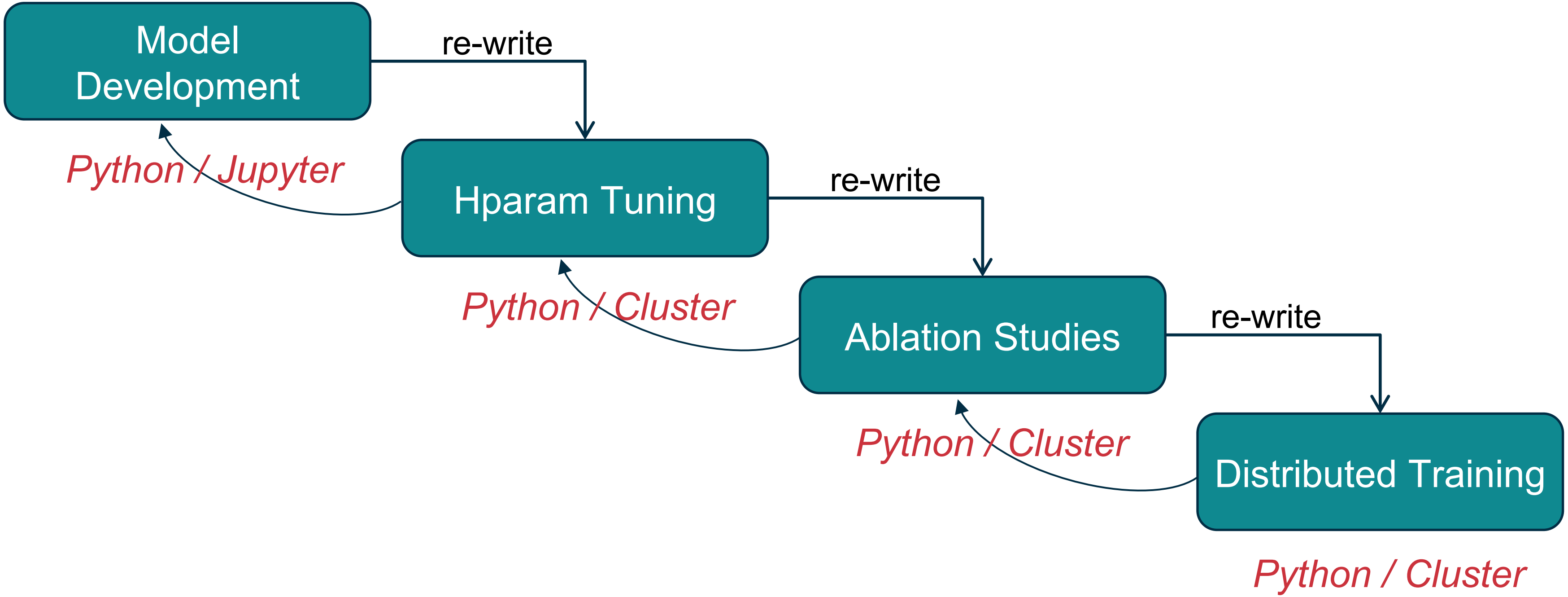
LOGICAL CLOCKS

The Machine Learning Pipeline

1.Feature Engineering → 2. Feature Selection → 3. Training → 4. Serving → 5. Predictions



Model Development in Practice



Challenges

- How do I maintain up to 4 different code bases for training models? DRY training code, please!
- What is Python / Cluster?
  - Dask, PySpark, Distributed TensorFlow, etc?
- Can I have a single execution framework to run all these 4 phases?
  - Kubernetes, python, spark-submit, Jupyter notebook

Programming Problem

- Model development is iterative and moving between distribution contexts requires code updates

Our Solution

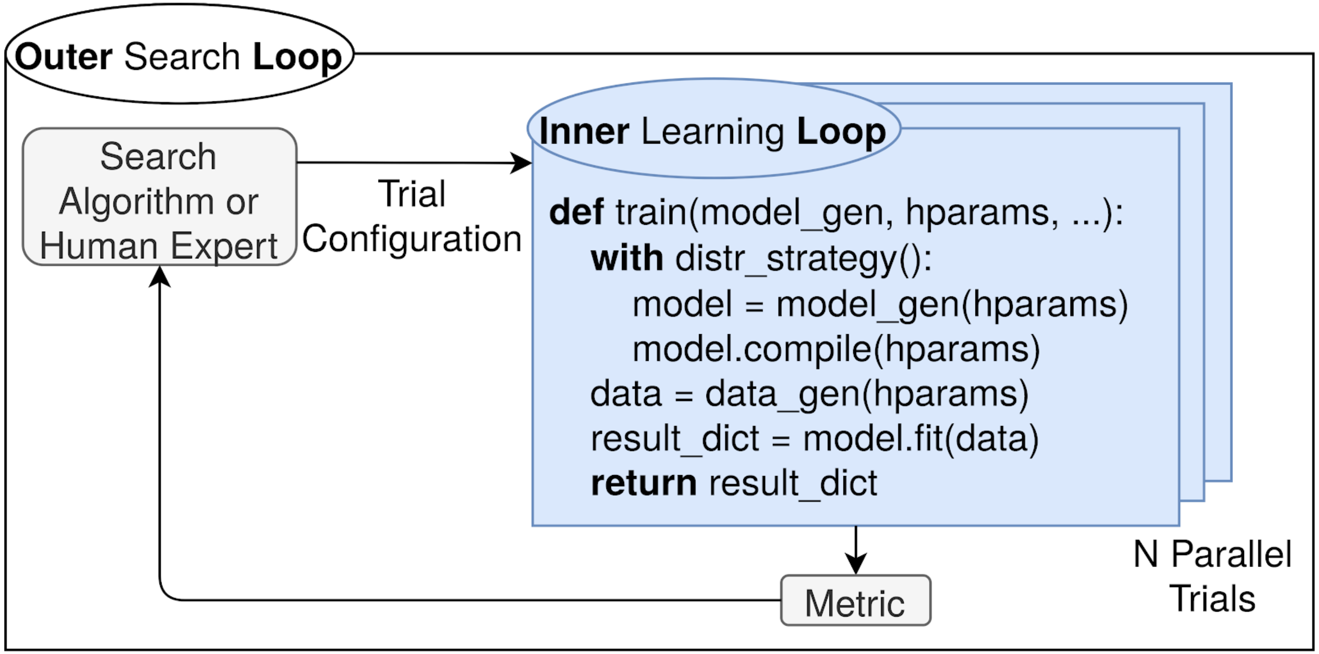
- Make the training loop oblivious to the given distribution context
- **Maggy**: a framework to support the distribution contexts based on PySpark
  https://github.com/logicalclocks/maggy

```python
train_images = mnist.train_images()
train_labels = mnist.train_labels()
test_images = mnist.test_images()
test_labels = mnist.test_labels()

train_images = (train_images / 255) - 0.5
test_images = (test_images / 255) - 0.5

train_images = train_images.reshape((-1, 784))
test_images = test_images.reshape((-1, 784))

model = Sequential([
  Dense(64, activation='relu', input_shape=(784,)),
  Dense(64, activation='relu'),
  Dense(10, activation='softmax'),
])

model.compile(
  optimizer='adam',
  loss='categorical_crossentropy',
  metrics=['accuracy'],
)
```

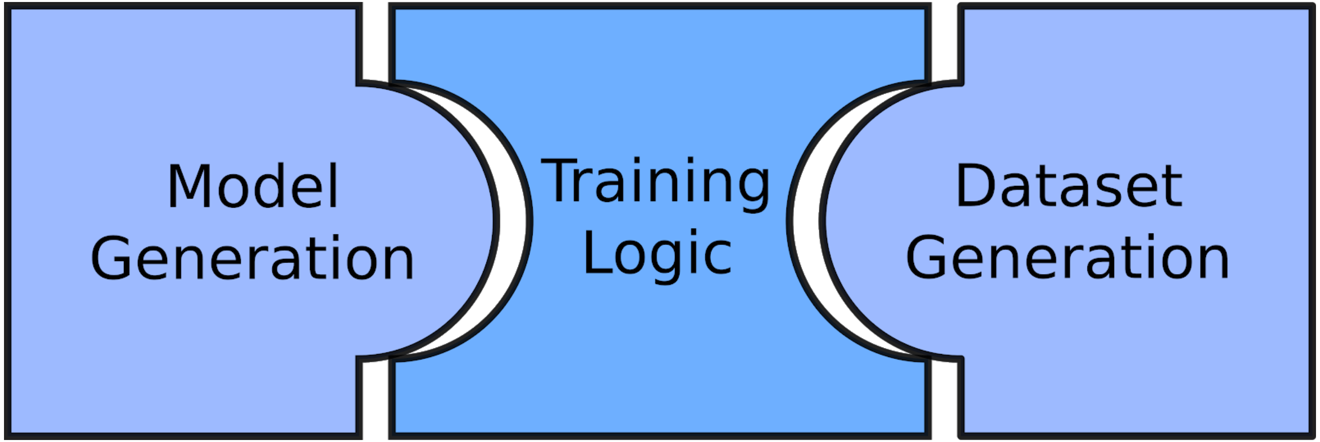
```python
model.fit(
  train_images,
  to_categorical(train_labels),
  epochs=5,
  batch_size=32,
)

model.evaluate(
  test_images,
  to_categorical(test_labels)
)

model.save_weights('model.h5')

predictions =
model.predict(test_images[:5])
```

Hyperparameters

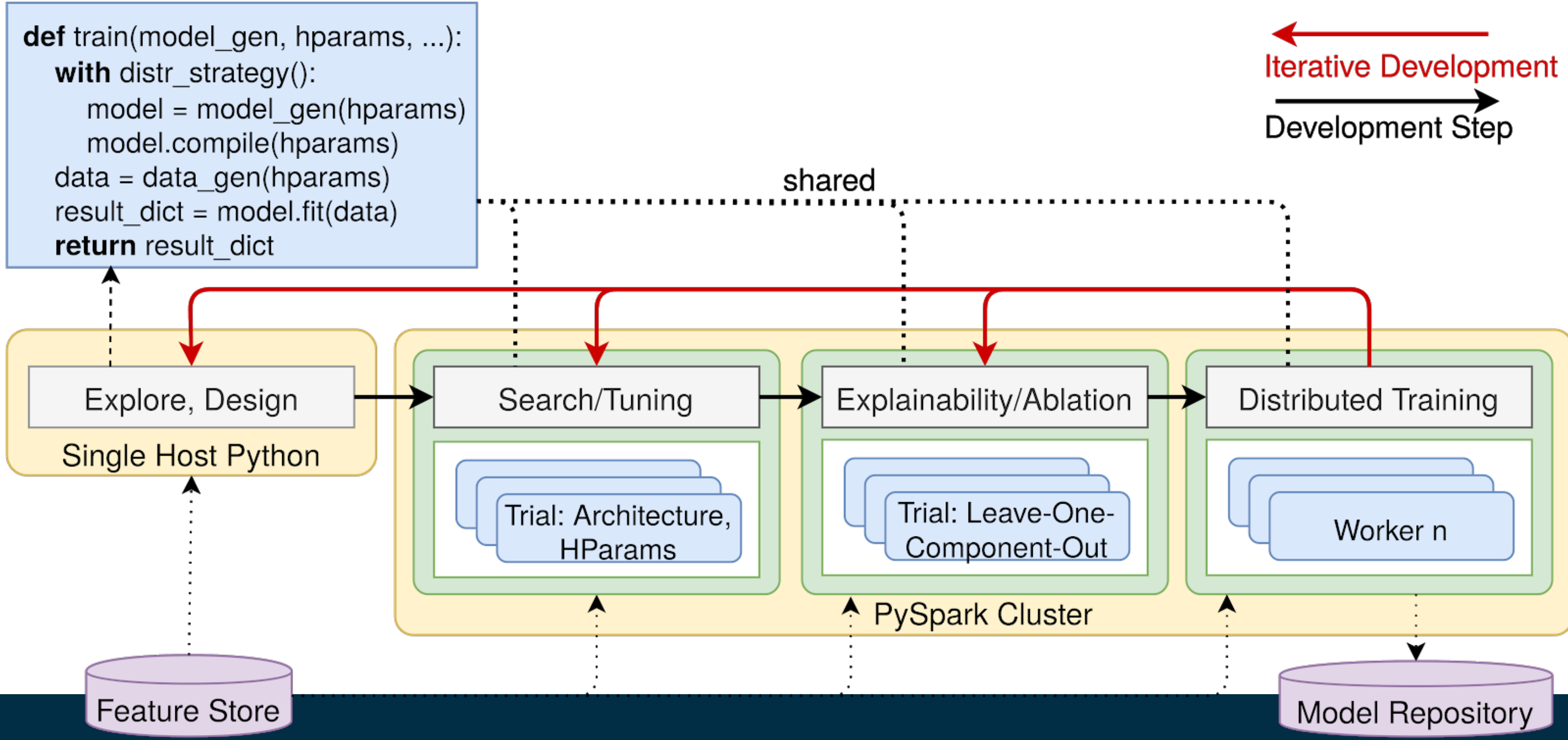
Inner and Outer Loop of Machine Learning



Decouple Training Logic from Model/Dataset Generation



Laptop -> HParam Tuning -> Ablation Study ->Distributed Training



Maggy Framework code

```
def decorator(train_fn, …): # Maggy framework code
    # connect to Maggy server
    # create reporter object to report statistics to the Driver
    train_fn(…)
```

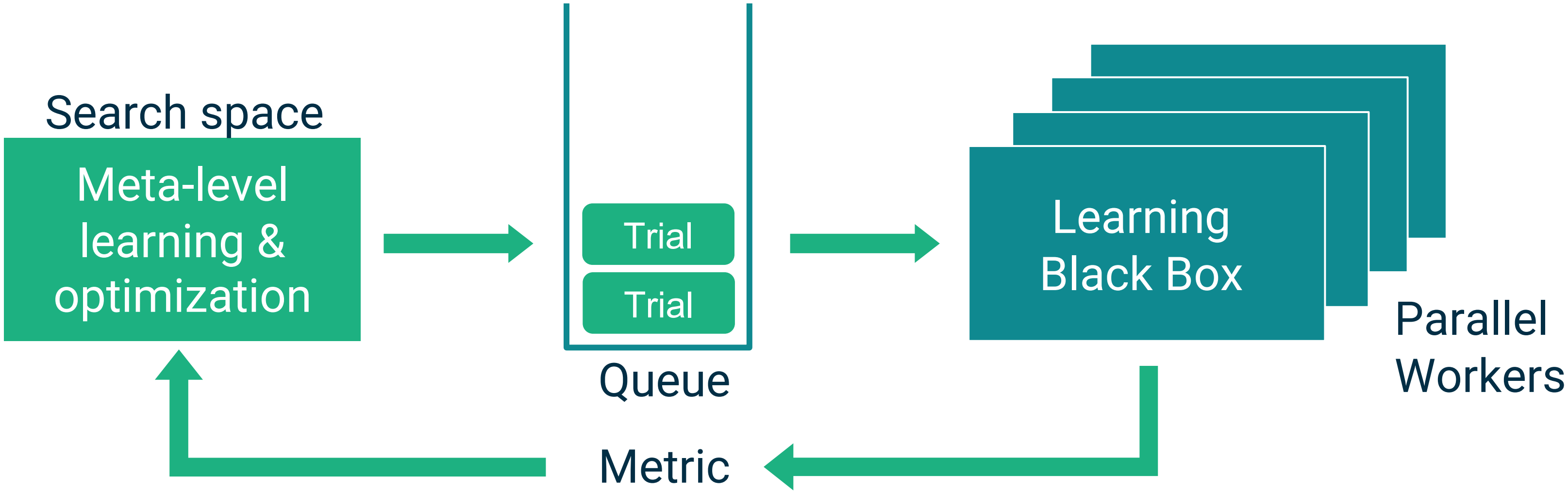
User-defined Training Function executed on all workers

```
def train(model_gen, hparams, dist_strategy, data_gen):
    with dist_strategy():
        model = model_gen.create()
        …..
        model.compile()
        model.fit(data_gen.batch())
```

User-defined Driver sets up dist context runs experiments

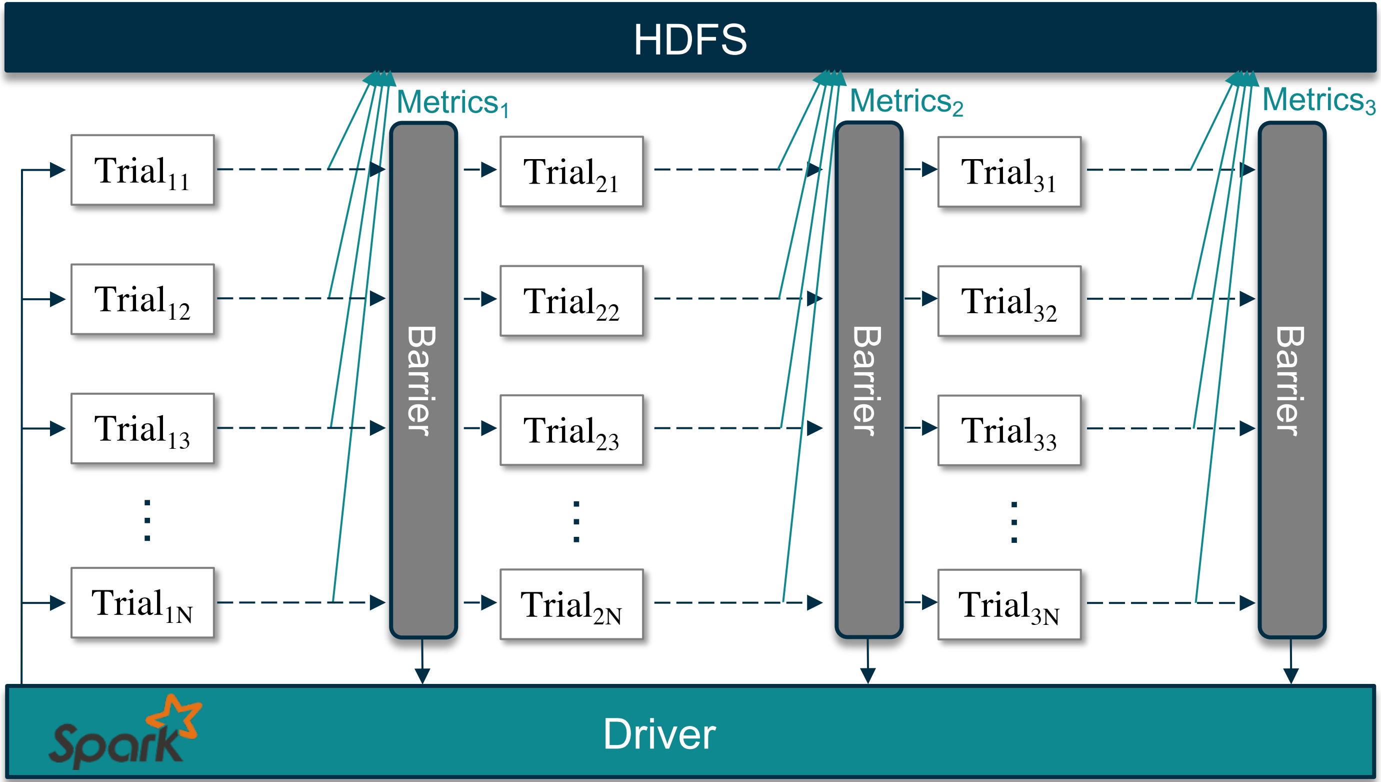
```
from maggy import experiment
experiment.lagom(train, …)
```

Search: Parallel Hyperparameter Tuning with Maggy

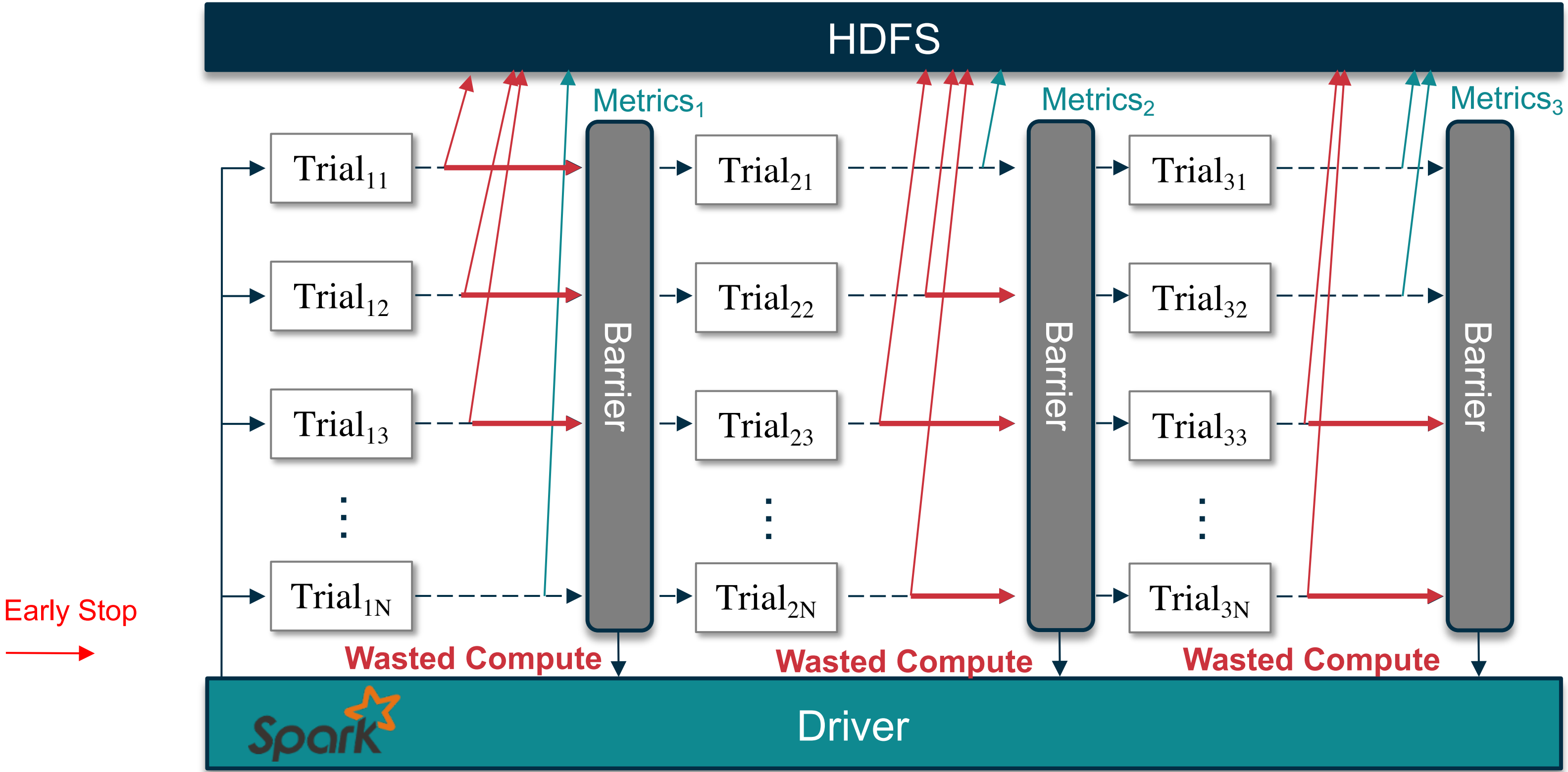


Moritz Meister https://databricks.com/session_eu19/asynchronous-hyperparameter-optimization-with-apache-spark

Synchronous Parallel Trials with PySpark



Synchronous Parallel Trials with Early Stopping



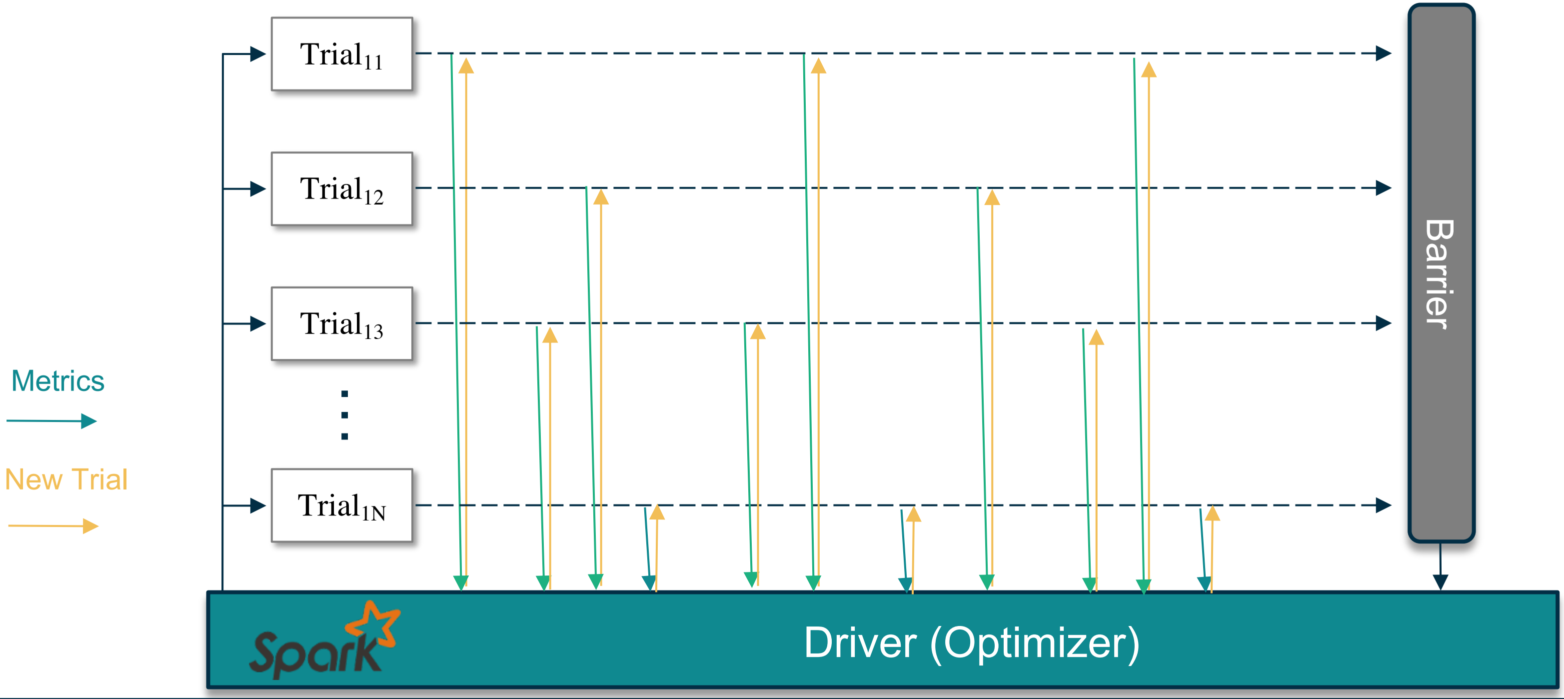
Problem: PySpark is inefficient with Early Stopping

- PySpark's bulk-synchronous execution model prevents efficient use of early-stopping for hyperparameter optimization.

New Framework?

Fix PySpark?

Solution: Long Running Tasks and a RPC framework



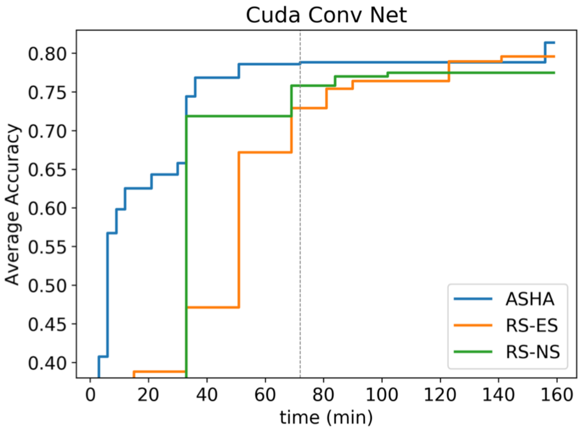
Maggy User API

```python
sp = Searchspace(kernel=('INTEGER', [2, 8]),
                 pool=('INTEGER', [2, 8]))

def train_fn(kernel, pool):
    for i in range(nr_iterations):
        ...


    return accuracy

result = experiment.lagom(train_fn, searchspace=sp,
                          optimizer='randomsearch',
                          num_trials=5, name='demo',
                          direction='max')
```

Develop your own Optimizer

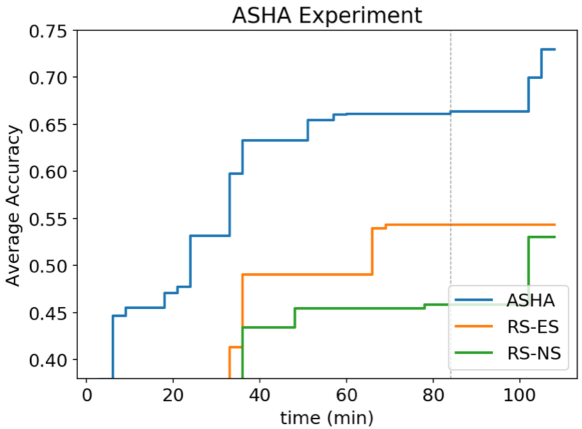
```python
class CustomOptimizer(AbstractOptimizer):
    def initialize(self):
        pass
    def get_suggestion(self, trial=None):
        # Return trial, return None if experiment finished
        pass
    def finalize_experiment(self, trials):
        pass


class CustomEarlyStop(AbstractEarlyStop):
    def earlystop_check(to_check, finalized_trials, direction):
        pass
```

Results



Hyperparameter Optimization Trial

	Best Accuracy (Std)	Trials (Std)	Trials Stopped (Std)
ASHA	0.8136 (0.02)	442 (0.0)	0 (0.0)
RS-ES	0.7958 (0.01)	120 (60.7)	90 (65.3)
RS-NS	0.7747 (0.04)	36 (0.0)	0 (0.0)



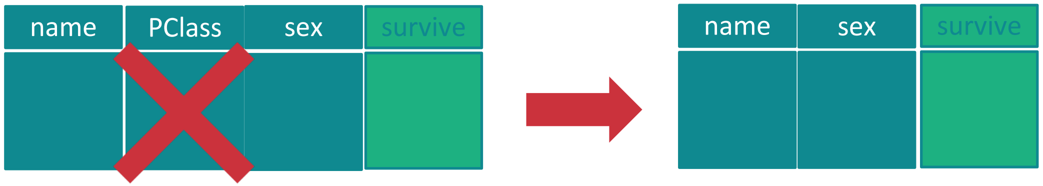
ASHA Validation Trial

	Best Accuracy (Std)	Trials (Std)	Trials Stopped (Std)
ASHA	0.7004 (0.03)	422 (38.69)	0 (0.0)
RS-ES	0.5438 (0.12)	112 (7.53)	63 (6.43)
RS-NS	0.5306 (0.28)	40 (4.51)	0 (0.0)

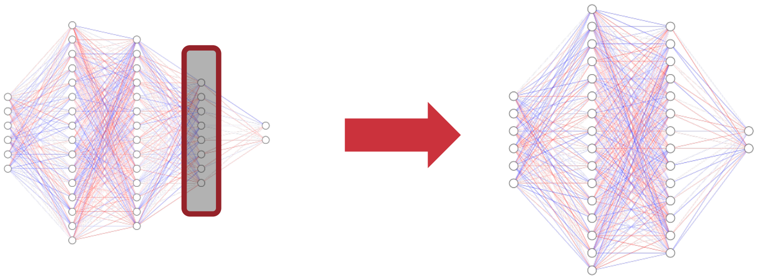
Parallel Ablation Studies

Replacing the Maggy Optimizer with an Ablator:

- Feature Ablation using the Feature Store



- Leave-One-Layer-Out Ablation
- Leave-One-Component-Out (LOCO)



Sina Sheikholeslami https://castor-software-days-2019.github.io/sina

Maggy on Hopsworks

Hopsworks – Award Winning AI Platform

DIGITAL TOP 50

DatSci AWARDS 2019

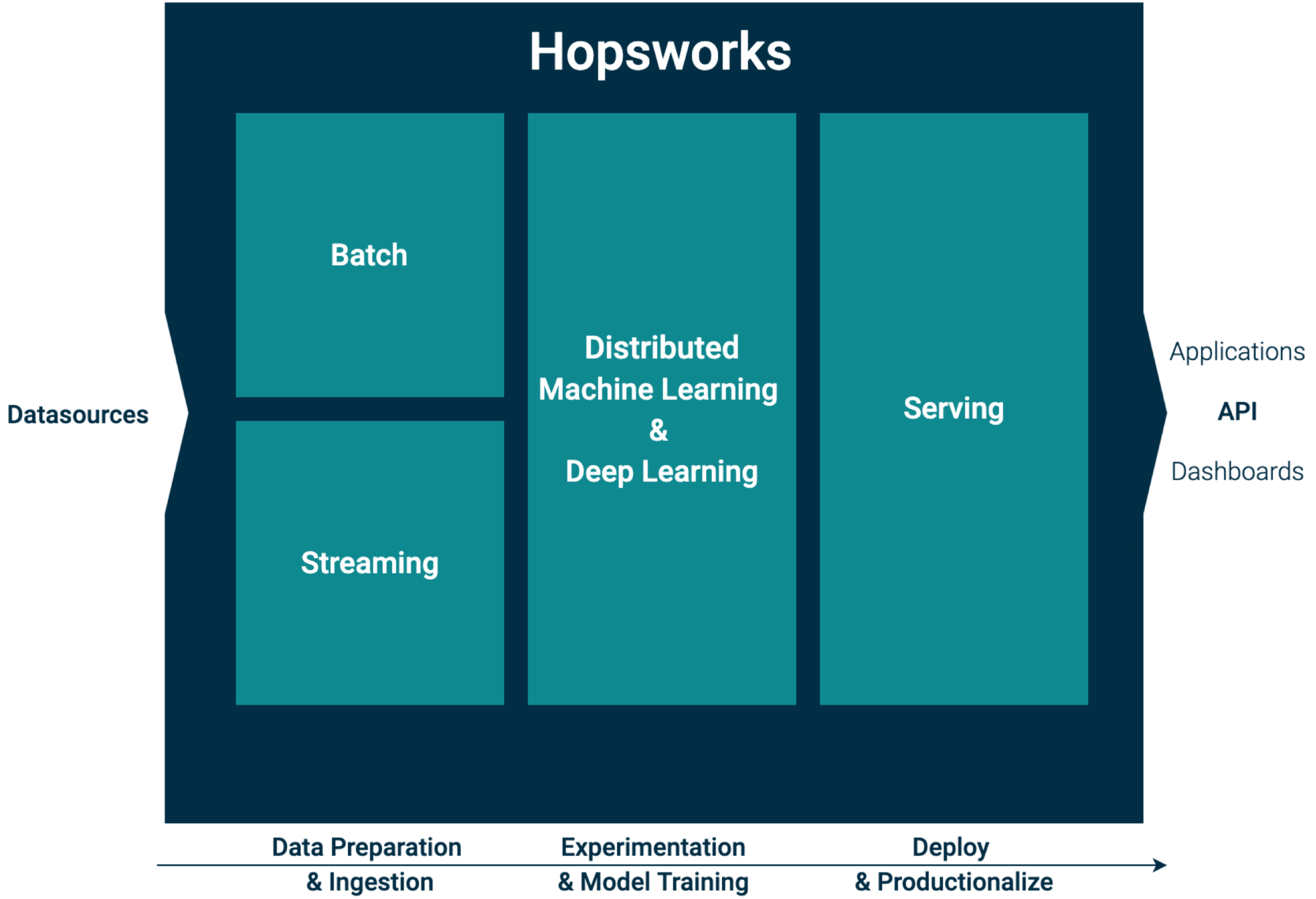
TECHNOLOGY INNOVATION OF THE YEAR

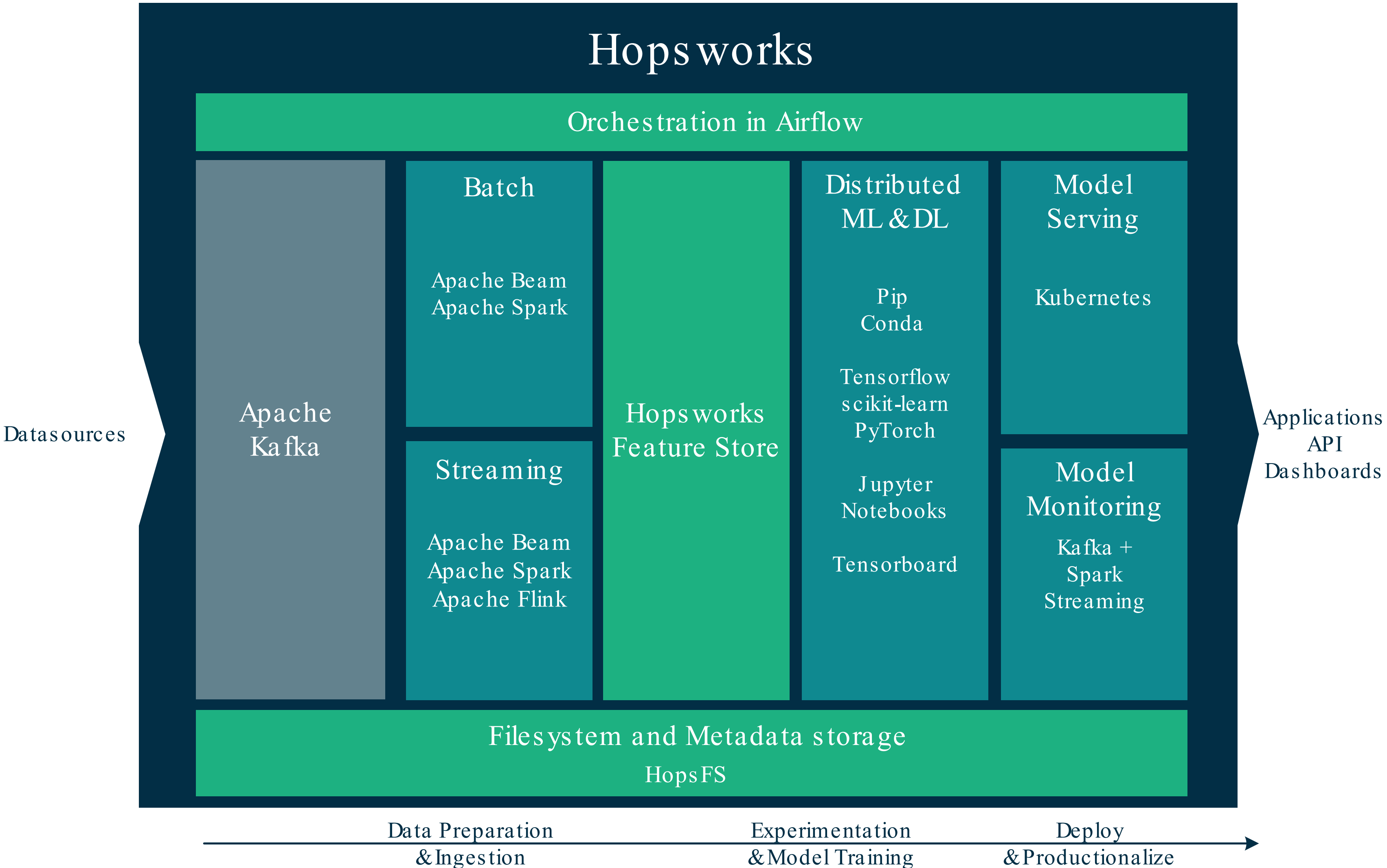
PAPIs.io

AI Startup Battle Winner

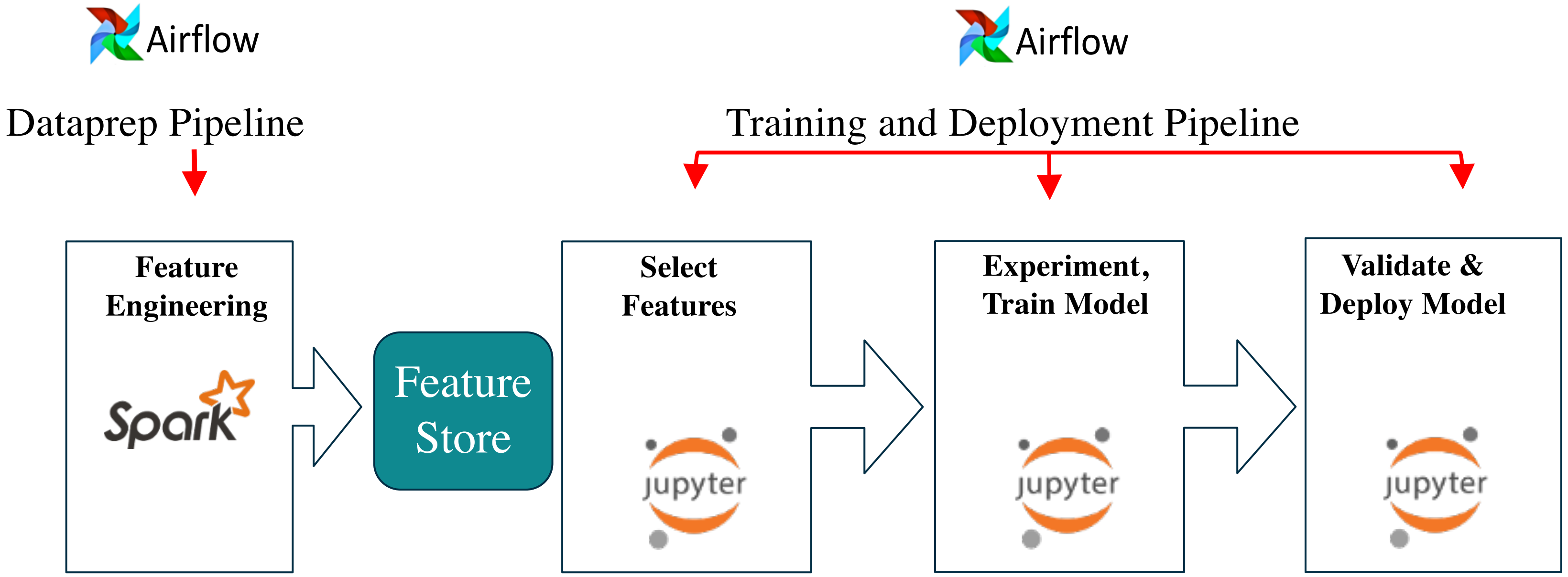
IEEE

CCGRID 2017 SCALE CHALLENGE

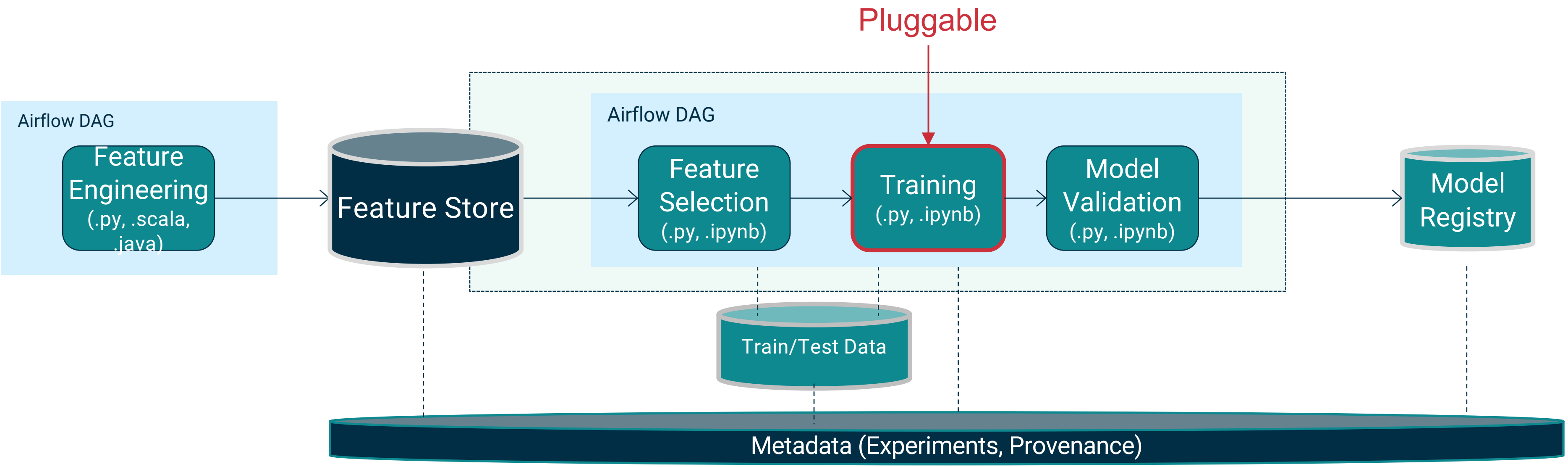

Hopsworks
Batch
Streaming
Distributed
Machine Learning
&
Deep Learning
Serving
Datasources
Applications
API
Dashboards
Data Preparation
& Ingestion
Experimentation
& Model Training
Deploy
& Productionalize


Hopsworks
Orchestration in Airflow
Datasources
Apache Kafka
Batch
Apache Beam
Apache Spark
Streaming
Apache Beam
Apache Spark
Apache Flink
Hopsworks Feature Store
Distributed ML & DL
Pip
Conda
Tensorflow
scikit-learn
PyTorch
Jupyter Notebooks
Tensorboard
Model Serving
Kubernetes
Model Monitoring
Kafka +
Spark
Streaming
Applications
API
Dashboards
Filesystem and Metadata storage
HopsFS
Data Preparation & Ingestion
Experimentation & Model Training
Deploy & Productionalize

Model Training Pipelines (with Notebooks)



Pluggable ML Pipelines



Summary

- Model training and ML pipelines can benefit from framework and DSL support
- Maggy is a framework based on PySpark for transparent distributed ML

- Maggy References:
    - https://databricks.com/session_eu19/asynchronous-hyperparameter-optimization-with-apache-spark
    - https://fosdem.org/2020/schedule/event/maggy/
    - https://www.logicalclocks.com/research/towards-distribution-transparency-for-supervised-ml-with-oblivious-training-functions
    - https://www.logicalclocks.com/blog/scaling-machine-learning-and-deep-learning-with-pyspark-on-Hopsworks
    - https://castor-software-days-2019.github.io/sina (Ablation studies)

Maggy Team

- KTH/LC: Jim Dowling, Amir Payberah, Vlad Vlassov
- PhDs: Moritz Meister, Sina Sheikholeslami
- MScs Students: Kai Jeggle, Alessio Molinari

Thank you!

LOGICAL CLOCKS

Register for a free account at

www.hops.site

Twitter

@logicalclocks

@hopsworks

GitHub

https://github.com/logicalclocks/hopsworks

https://github.com/hopshadoop/hops