Maggy

–

Open-Source Asynchronous Distributed Hyperparameter Optimization Based on Apache Spark

Moritz Meister
moritz@logicalclocks.com
@morimeister

FOSDEM'20
02.02.2020

LOGICAL CLOCKS

The Bitter Lesson (of AI)*

"Methods that scale with computation are the future of AI"**

"The two (general purpose) methods that seem to scale …

… are **search** and **learning**."*

Rich Sutton
(Father of Reinforcement Learning)

* http://www.incompleteideas.net/IncIdeas/BitterLesson.html

** https://www.youtube.com/watch?v=EeMCEQa85tw

The Answer

Spark scales with available compute!

Spark is the answer!

1 + 1 = 3

Distribution and Deep Learning

Generalization Error

Better Regularization Methods

Better Optimization Algorithms

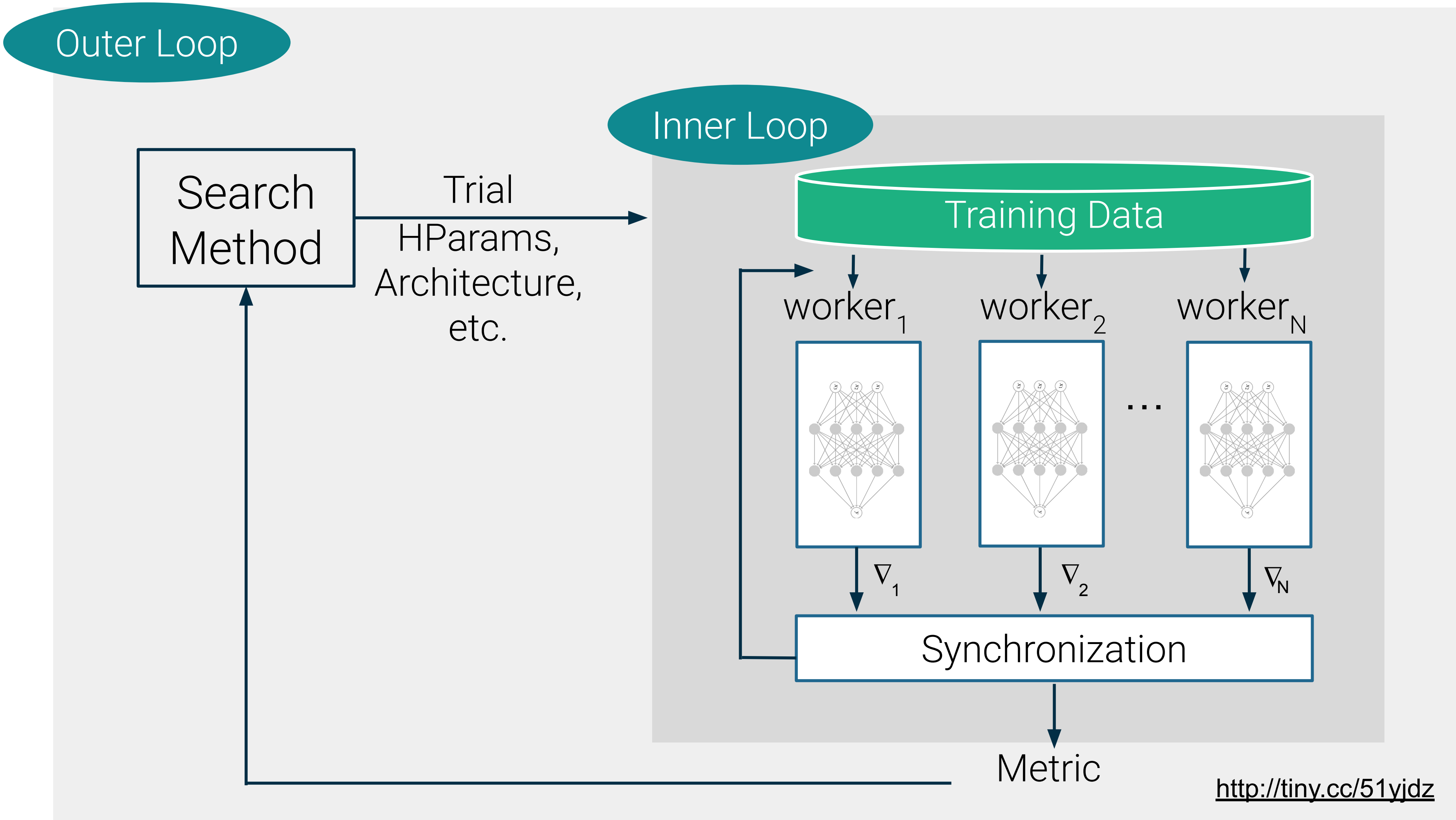
Larger Training Datasets

Hyperparameter Optimization

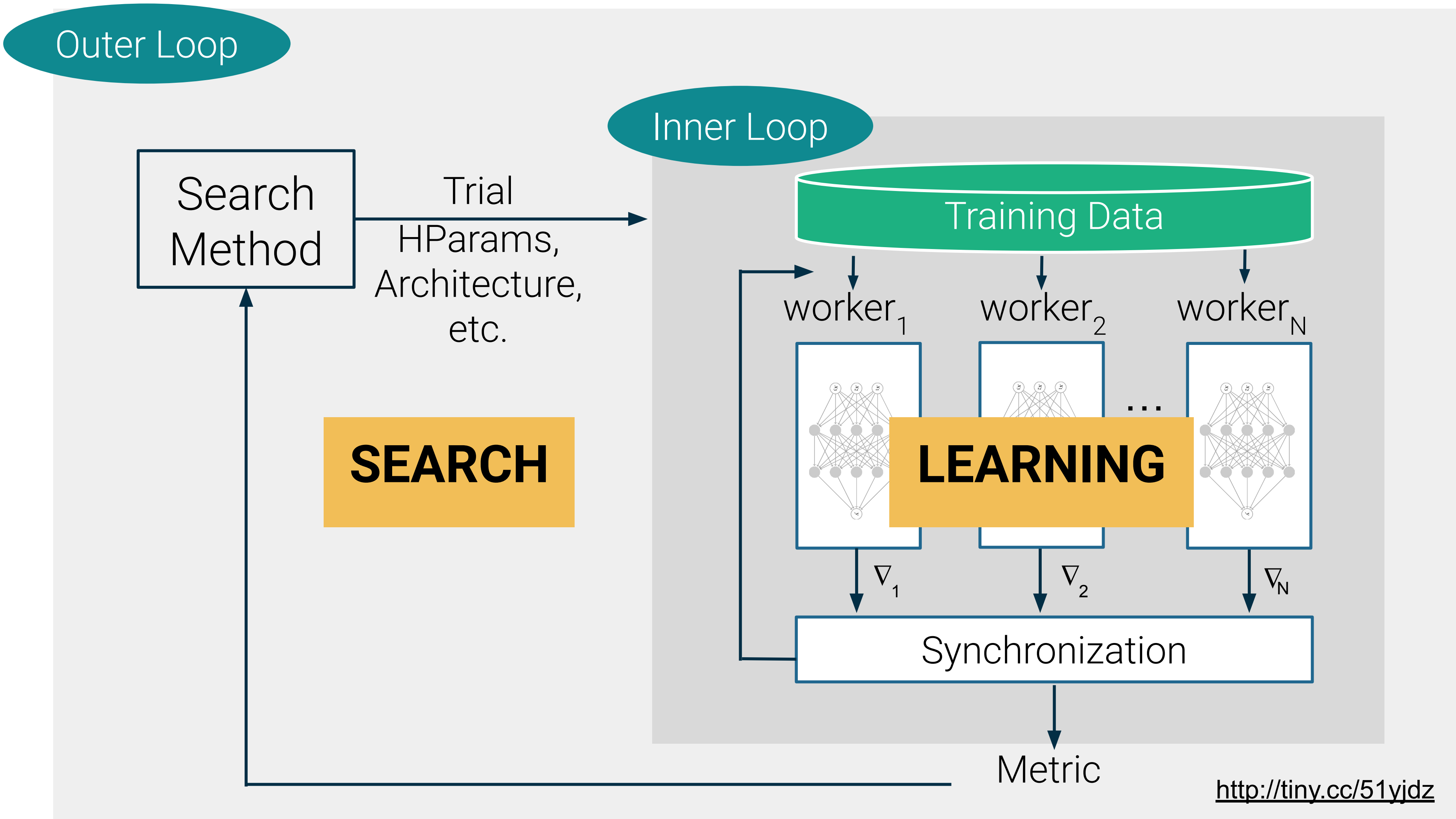
Design Better Models

Distribution

Inner and Outer Loop of Deep Learning

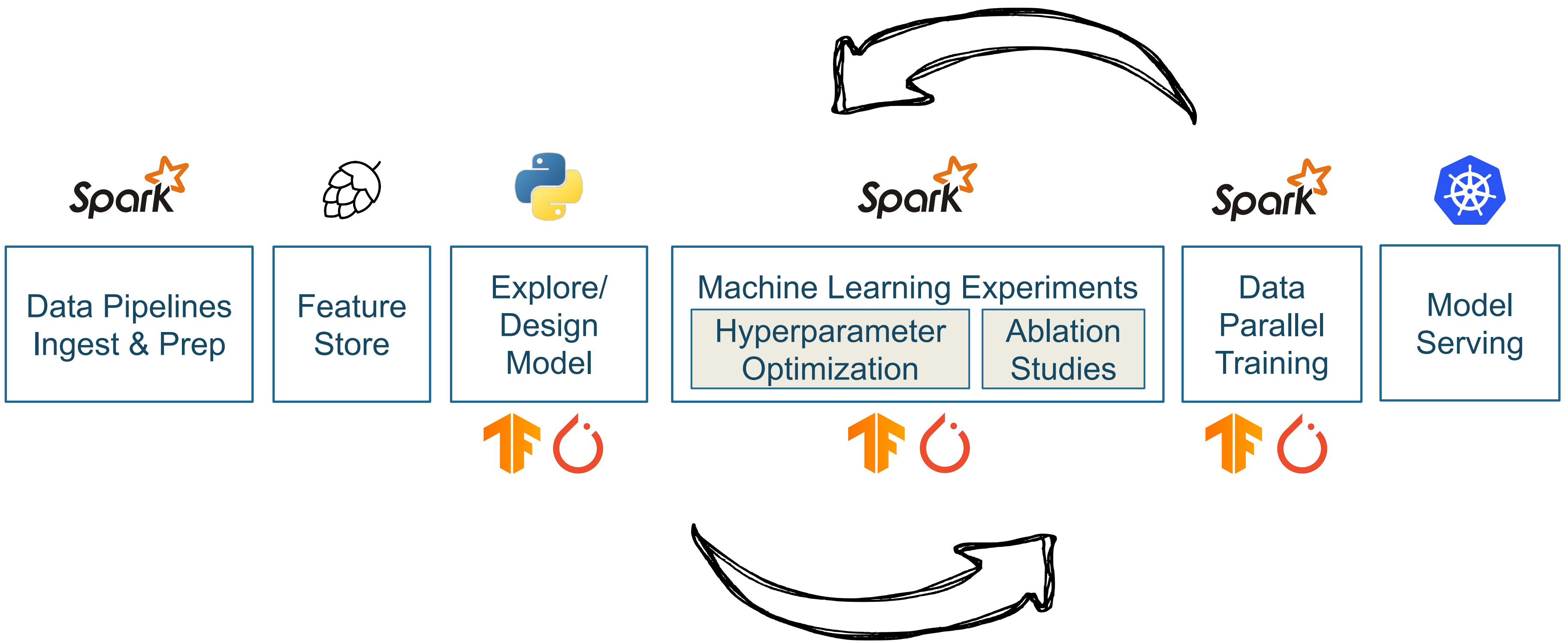


Inner and Outer Loop of Deep Learning



http://tiny.cc/51yjdz

In Reality This Means Rewriting Training Code



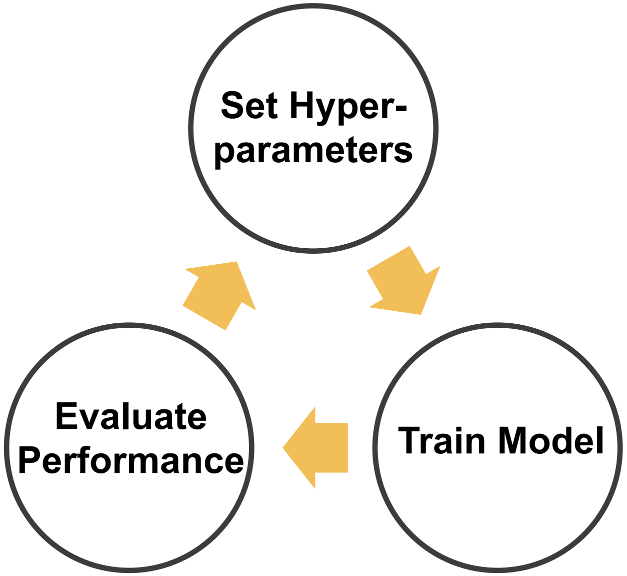
Towards Distribution Transparency

Inner Loop

The distribution oblivious training function (pseudo-code):

```
def train(model_gen, hparams, ...):
    with distr_strategy():
      model = model_gen(hparams)
      model.compile(hparams)
      data = data_gen(hparams)
      result_dict = model.fit(data)
    return result_dict
```

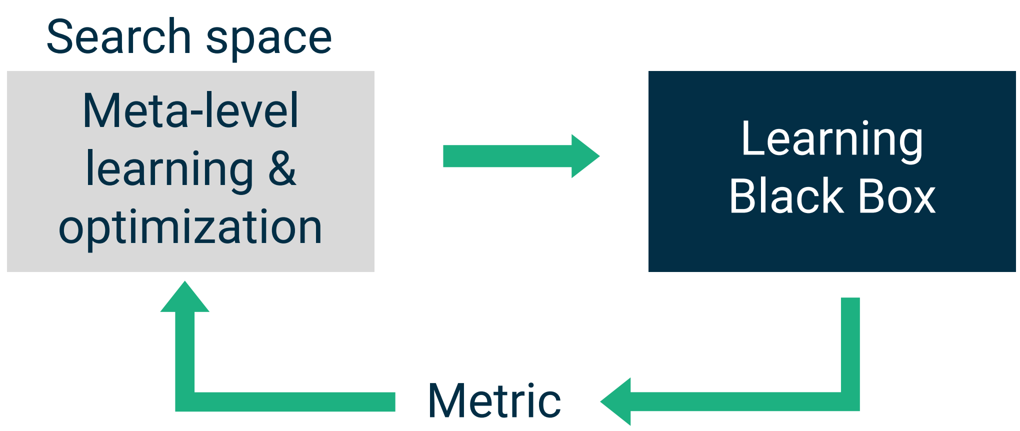
Towards Distribution Transparency



- Trial and Error is slow
- Iterative approach is greedy
- Search spaces are usually large
- Sensitivity and interaction of hyperparameters

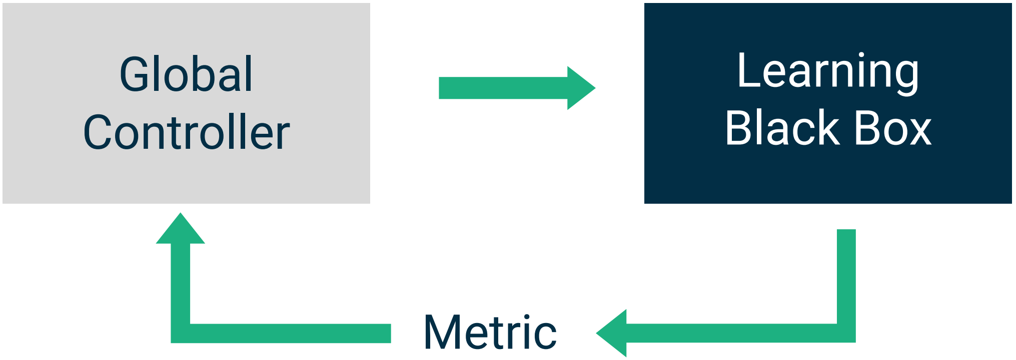
Sequential Black Box Optimization

Outer Loop



Sequential Search

Outer Loop

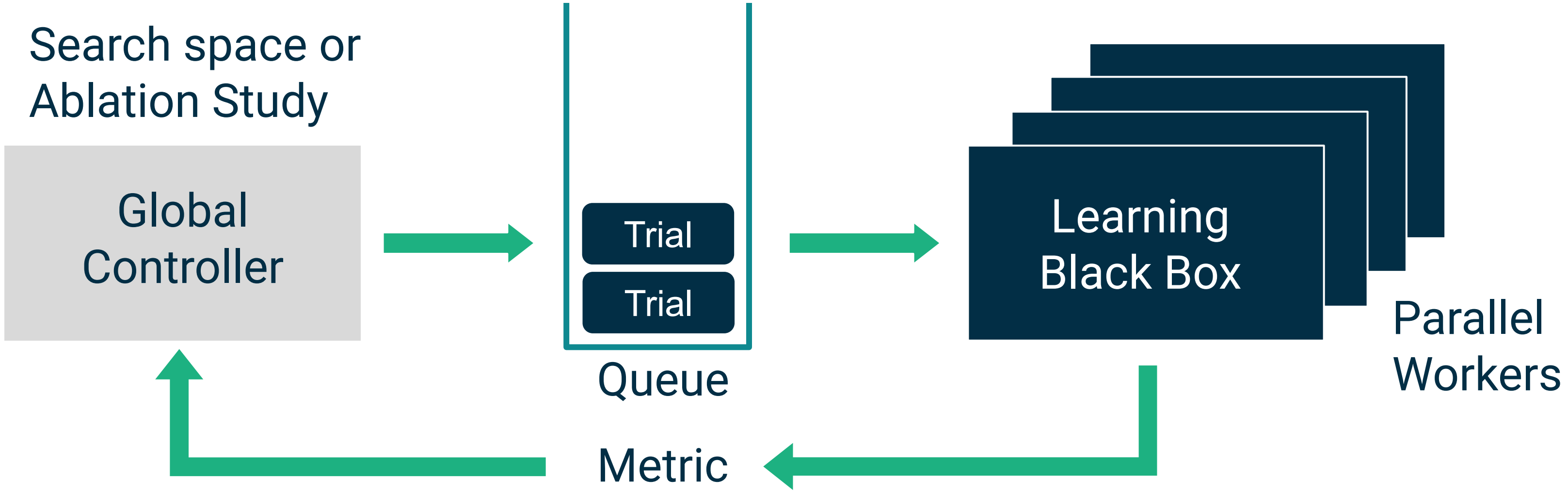


Parallel Search

Outer Loop

Which algorithm to use for search?

How to monitor progress?



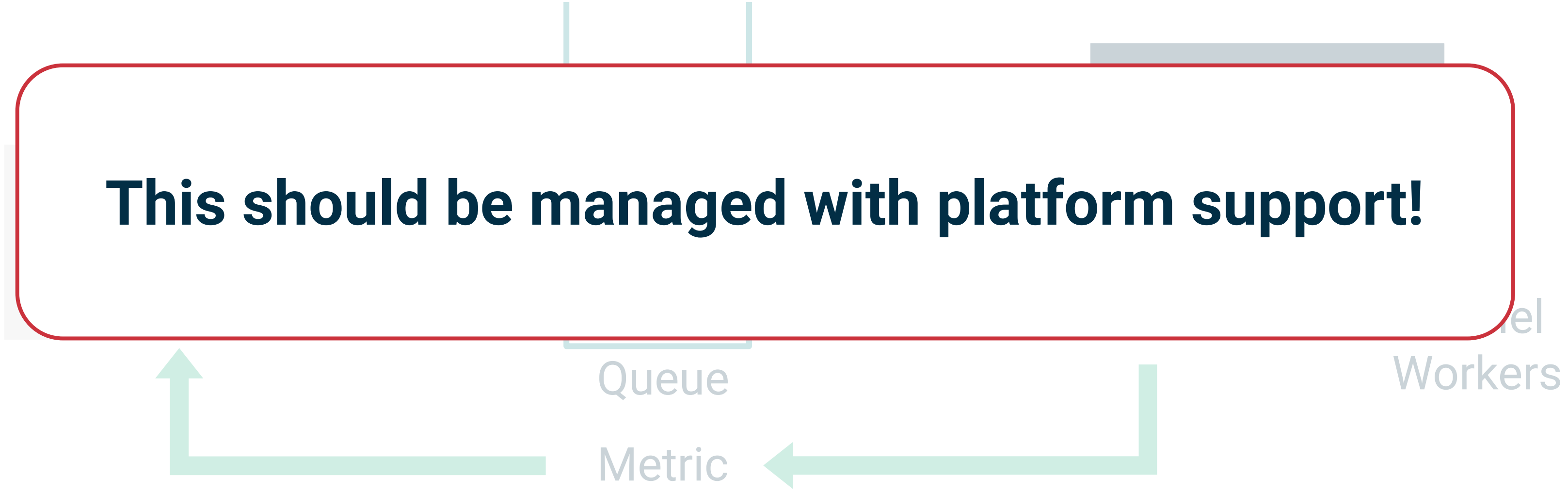
How to aggregate results?

Fault Tolerance?

Parallel Search

Which algorithm to use for search?

How to monitor progress?



This should be managed with platform support!

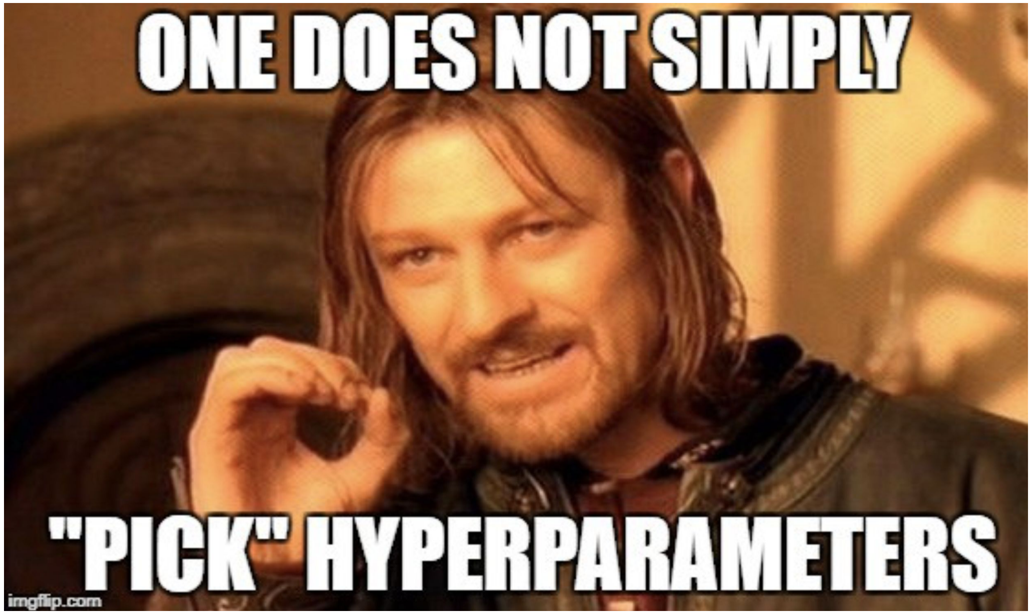
How to aggregate results?

Fault Tolerance?

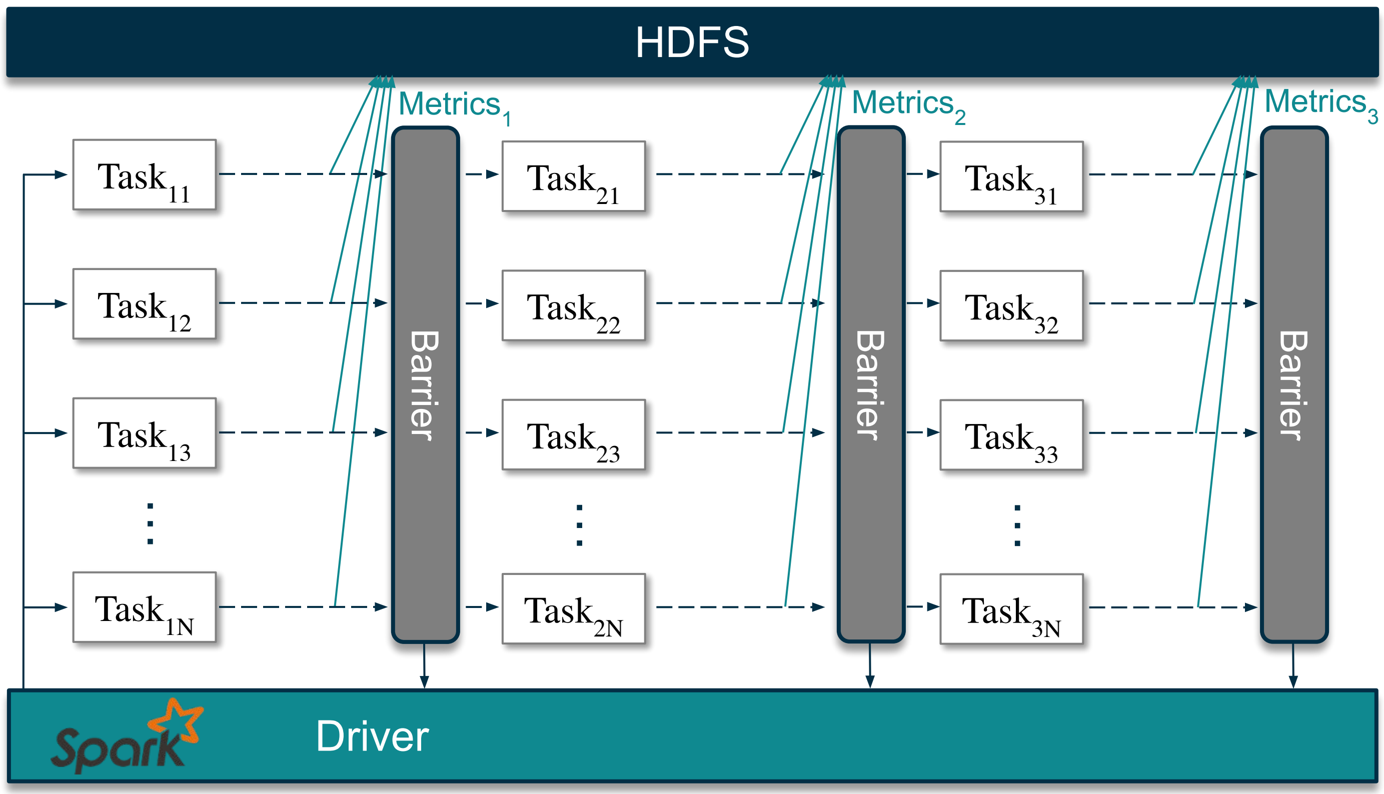
Maggy

A flexible framework for asynchronous parallel execution of trials for ML experiments on Hopsworks:

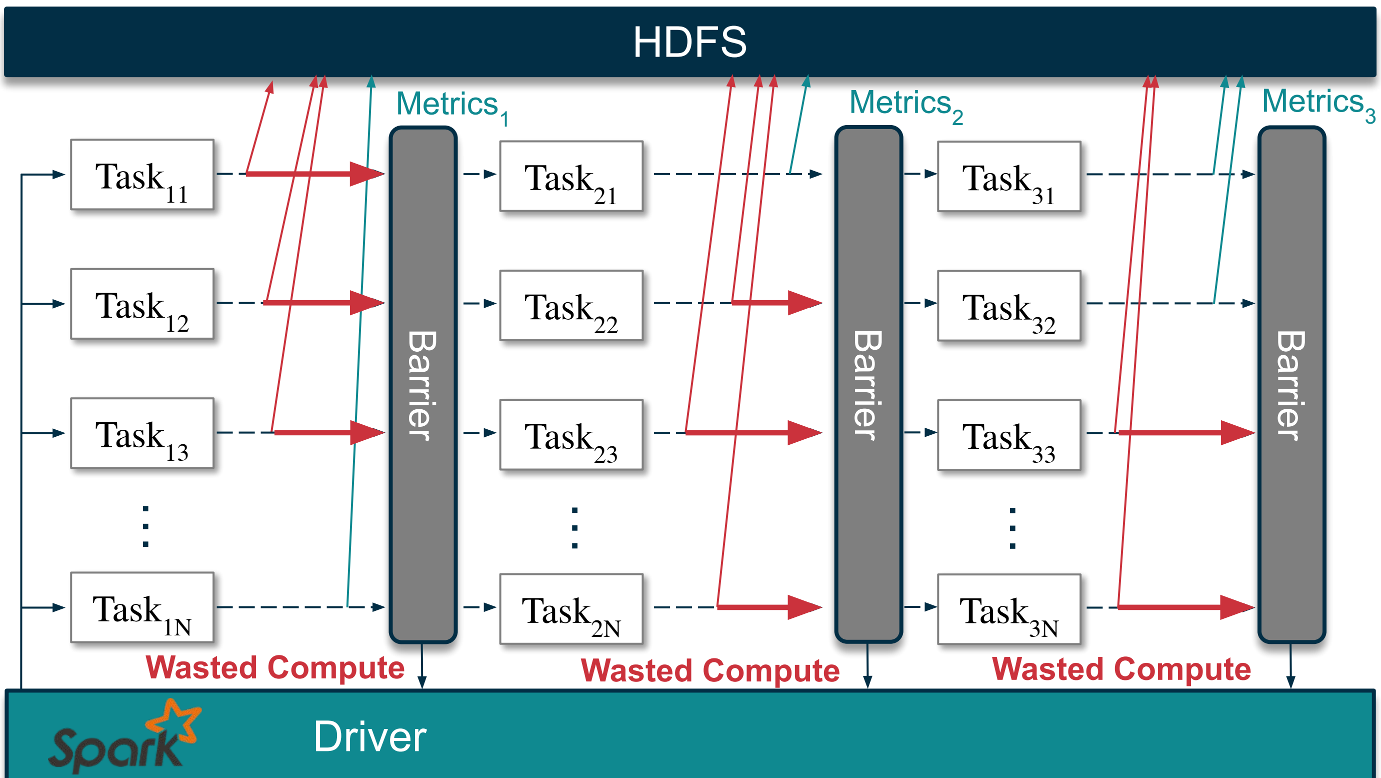
ASHA, Random Search,
Grid Search,
LOCO-Ablation,
Bayesian Optimization
and more to come...



Synchronous Search



Add Early Stopping and Asynchronous Algorithms



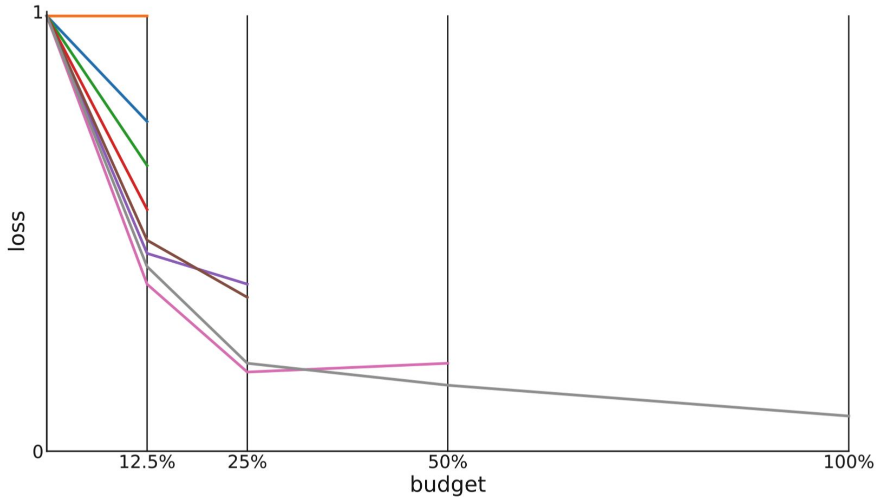
Performance Enhancement

Early Stopping:

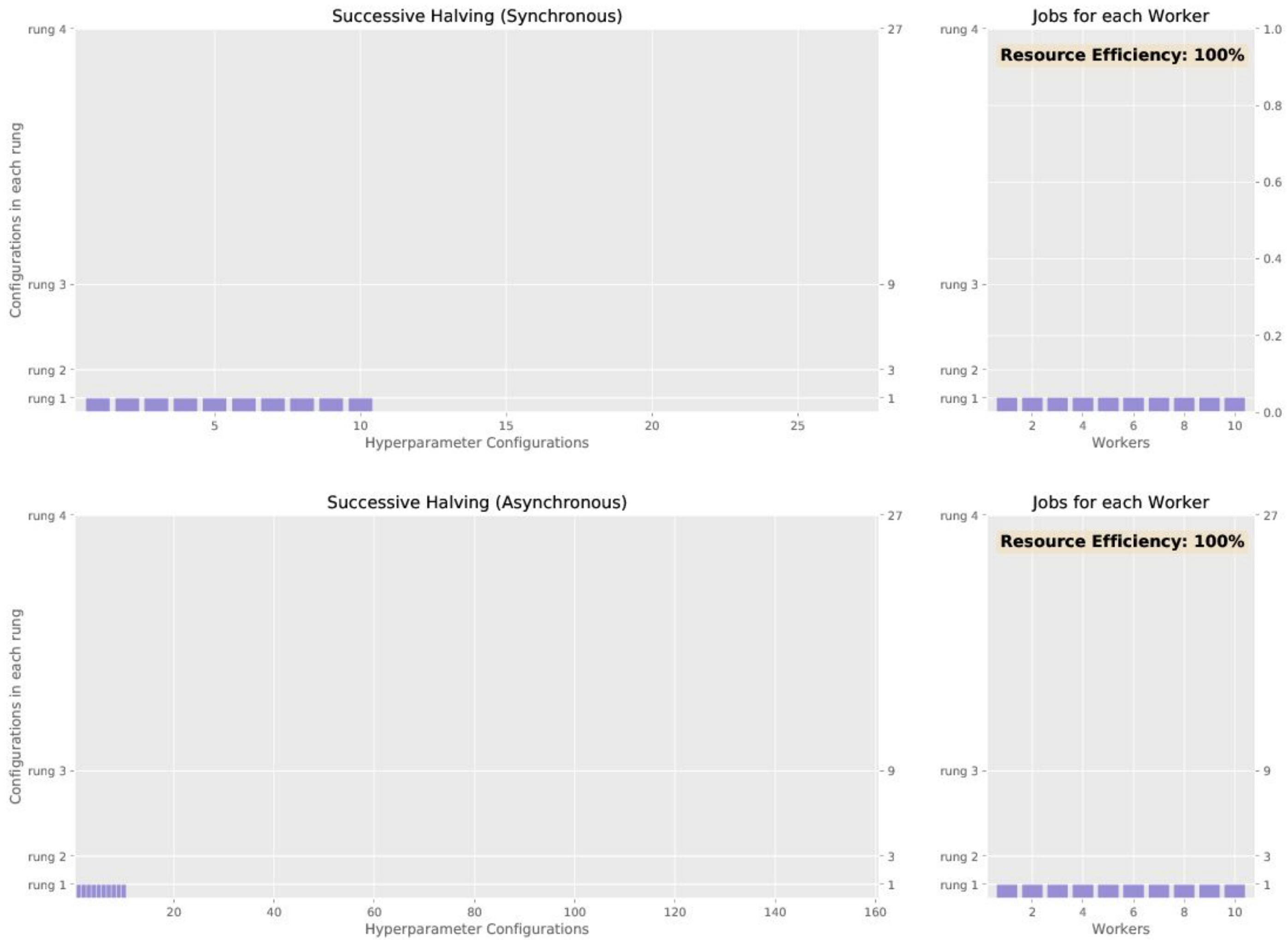
- Median Stopping Rule
- Performance curve prediction

Multi-fidelity Methods:

- Successive Halving Algorithm
- Hyperband



Asynchronous Successive Halving Algorithm

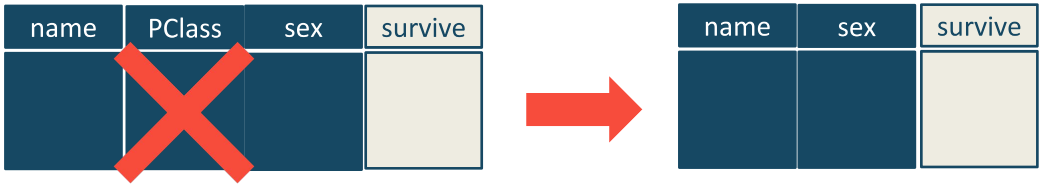


Animation: https://blog.ml.cmu.edu/2018/12/12/massively-parallel-hyperparameter-optimization/
Liam Li et al. “Massively Parallel Hyperparameter Tuning”. In: CoRR abs/1810.05934 (2018).

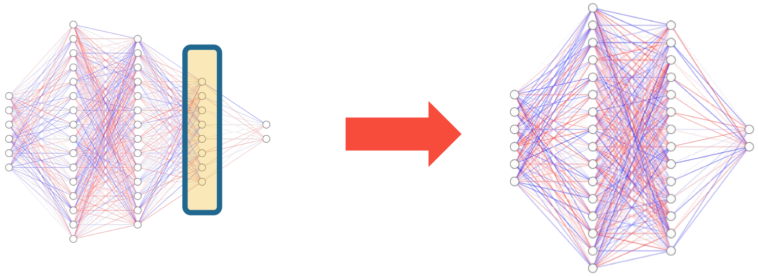
Ablation Studies

Replacing the Maggy Optimizer with an Ablator:

- Feature Ablation using the Feature Store



- Leave-One-Layer-Out Ablation

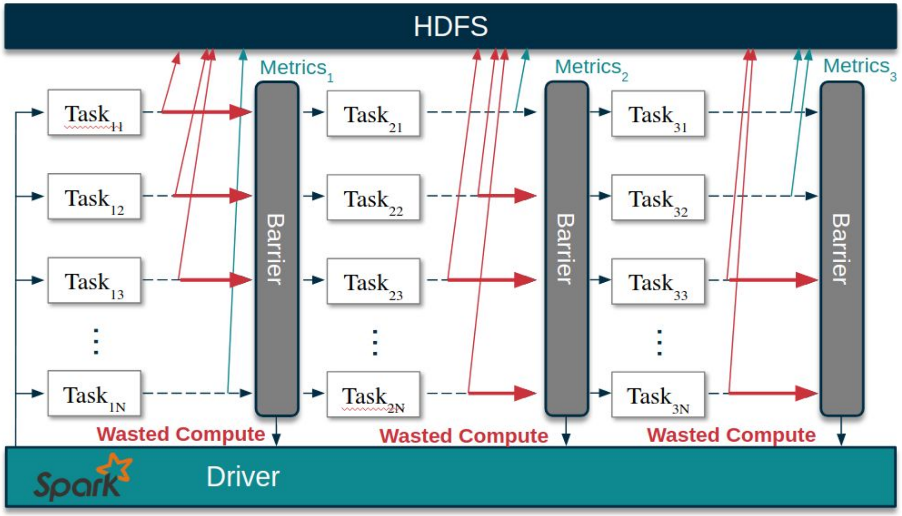


- Leave-One-Component-Out (LOCO)

Challenge

How can we fit this into the bulk synchronous execution model of Spark?

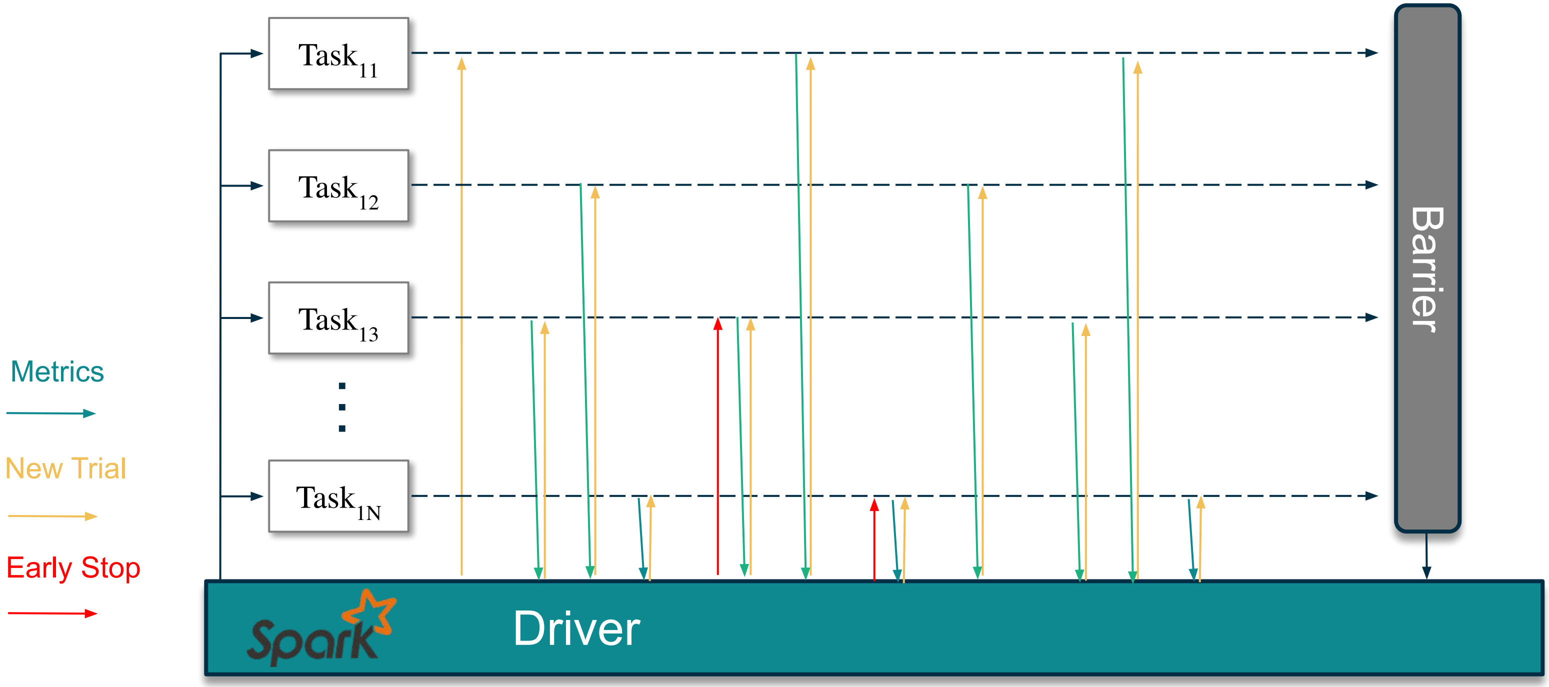
Mismatch: Spark Tasks and Stages vs. Trials



Databricks' approach: Project Hydrogen (barrier execution mode) & SparkTrials in Hyperopt

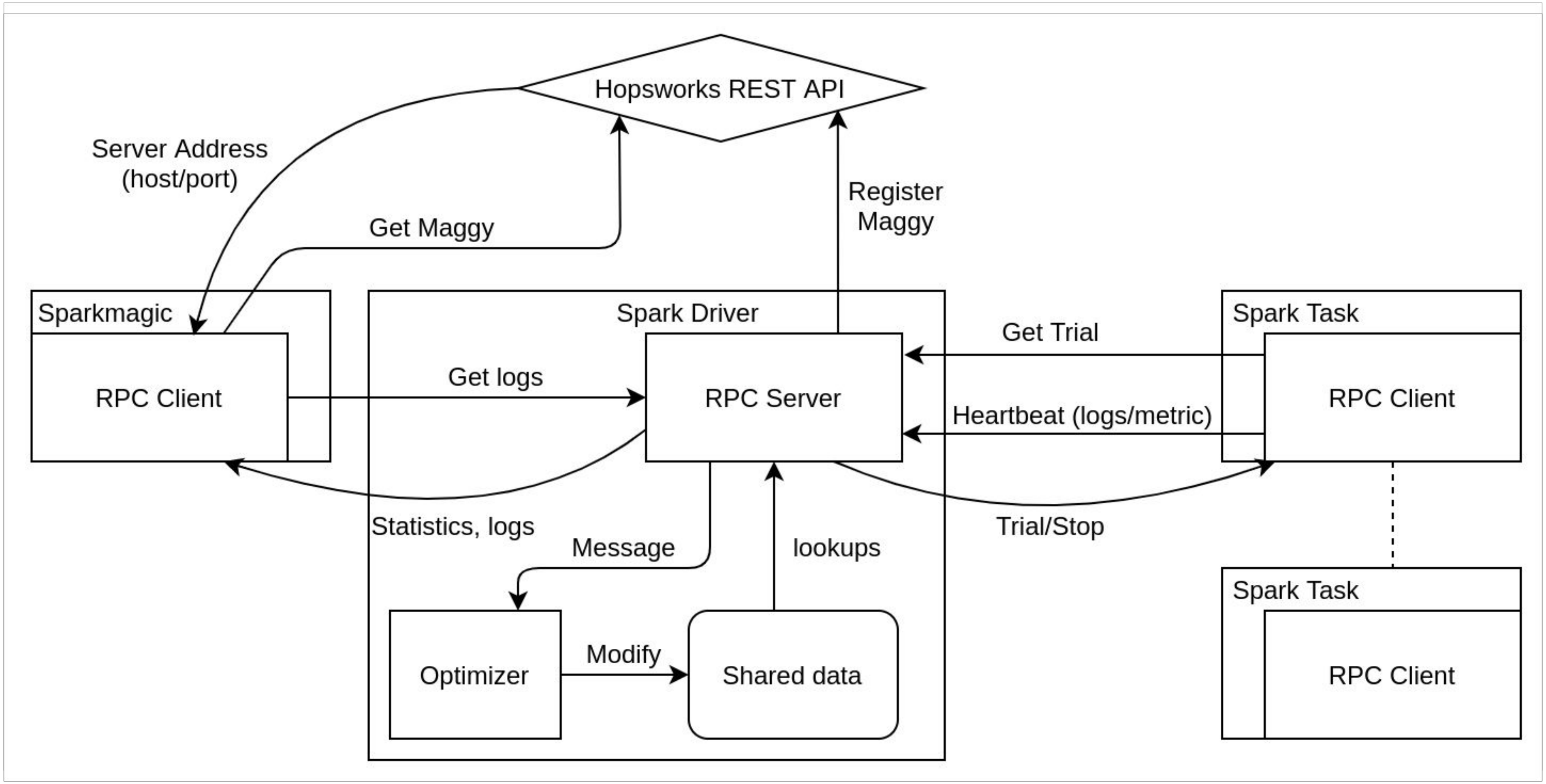
The Solution

Long running tasks and communication:



HyperOpt: One Job/Trial, requiring many Threads on Driver

Enter Maggy



Hopsworks REST API
Server Address
(host/port)
Get Maggy
Register
Maggy
Sparkmagic
RPC Client
Spark Driver
RPC Server
Get logs
Get Trial
Heartbeat (logs/metric)
Spark Task
RPC Client
Statistics, logs
Message
lookups
Trial/Stop
Optimizer
Modify
Shared data
Spark Task
RPC Client

User API

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

def train_fn(kernel, pool):
      from maggy import KerasBatchEnd
      ...
      model.compile()
      model.fit(...,callbacks=[KerasBatchEnd(metric='acc'), tb_callback]
      ...
      return accuracy

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

Developer API

```
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
```

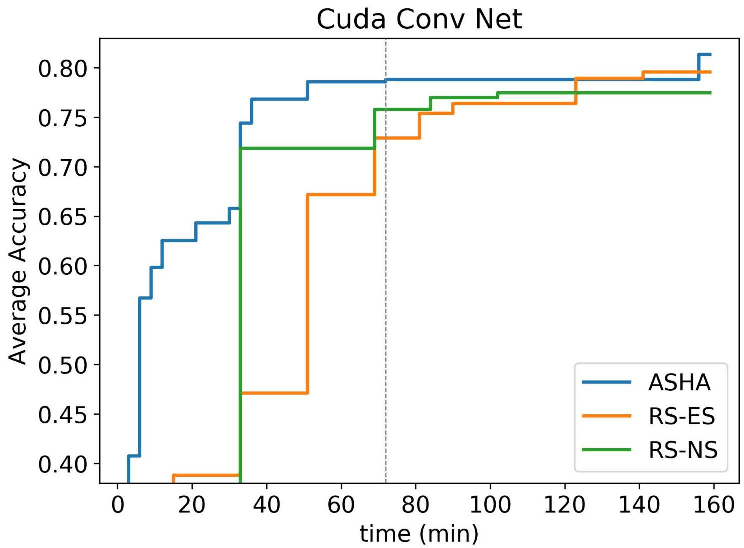
Ablation API

```
ablation_study = AblationStudy('titanic_train_dataset',
                               label_name='survived')

ablation_study.features.include('pclass', 'fare')

ablation_study.model.layers.include('my_dense_two',
                                    'my_dense_three')

ablation_study.model.layers.include_groups(['my_dense_two',
                                            'my_dense_four'])

ablation_study.model.layers.include_groups(prefix='my_dense')
```

Ablation API

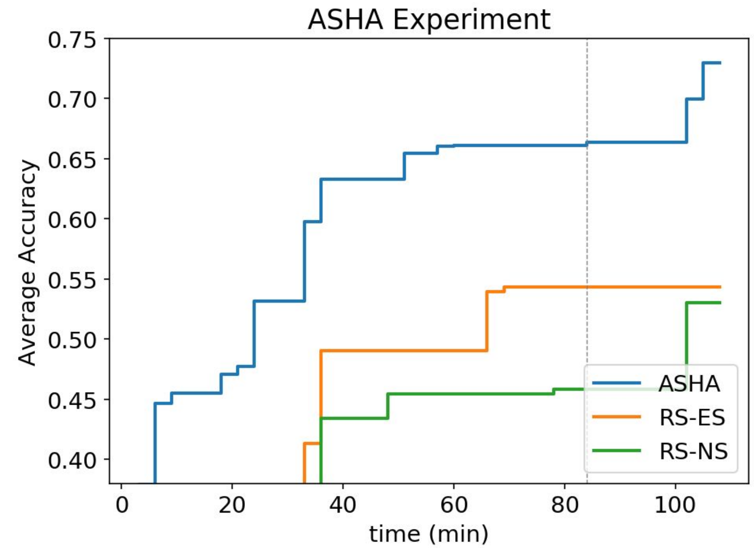
```
ablation_study.model.set_base_model_generator(base_model_generator)

def training_function(dataset_function, model_function, reporter):
   tf_dataset = dataset_function(epochs, batch_size)
   model = model_function()
   model.compile(...)

   cb = [KerasBatchEnd(reporter, metric='acc')]
   history = model.fit(tf_dataset, callbacks=cb, epochs=5, steps_per_epoch=30)

   return float(history.history['acc'][-1])
```

Results



Hyperparameter Optimization Task

	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 Task

	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)

Conclusions

- **Avoid** iterative Hyperparameter Optimization
- Black box optimization is **hard**
- State-of-the-art algorithms can be deployed **asynchronously**
- **Maggy**: platform support for automated hyperparameter optimization and ablation studies
- **Save** resources with asynchronism
- **Early stopping** for sensible models

What's next?

- More algorithms
- Distribution Transparency
- Comparability/ reproducibility of experiments
- Implicit Provenance
- Support for PyTorch

Acknowledgements

Thanks to the entire Logical Clocks Team 😊

Contributions from colleagues:

Robin Andersson @robzor92
Sina Sheikholeslami @cutlash
Kim Hammar @KimHammar1
Alex Ormenisan @alex_ormenisan

LOGICAL CLOCKS @hopsworks

HOPSWORKS

- **Maggy**
  https://github.com/logicalclocks/maggy
  https://maggy.readthedocs.io/en/latest/
- **Hopsworks**
  https://github.com/logicalclocks/hopsworks
  https://www.logicalclocks.com/whitepapers/hopsworks
- **Feature Store:** the missing data layer in ML pipelines?
  https://www.logicalclocks.com/feature-store/