Distributed Deep Learning with Apache Spark and TensorFlow

jim_dowling

Jim Dowling, Logical Clocks AB

The Cargobike Riddle

Dynamic Executors
(release GPUs when training finishes)

Spark & TensorFlow

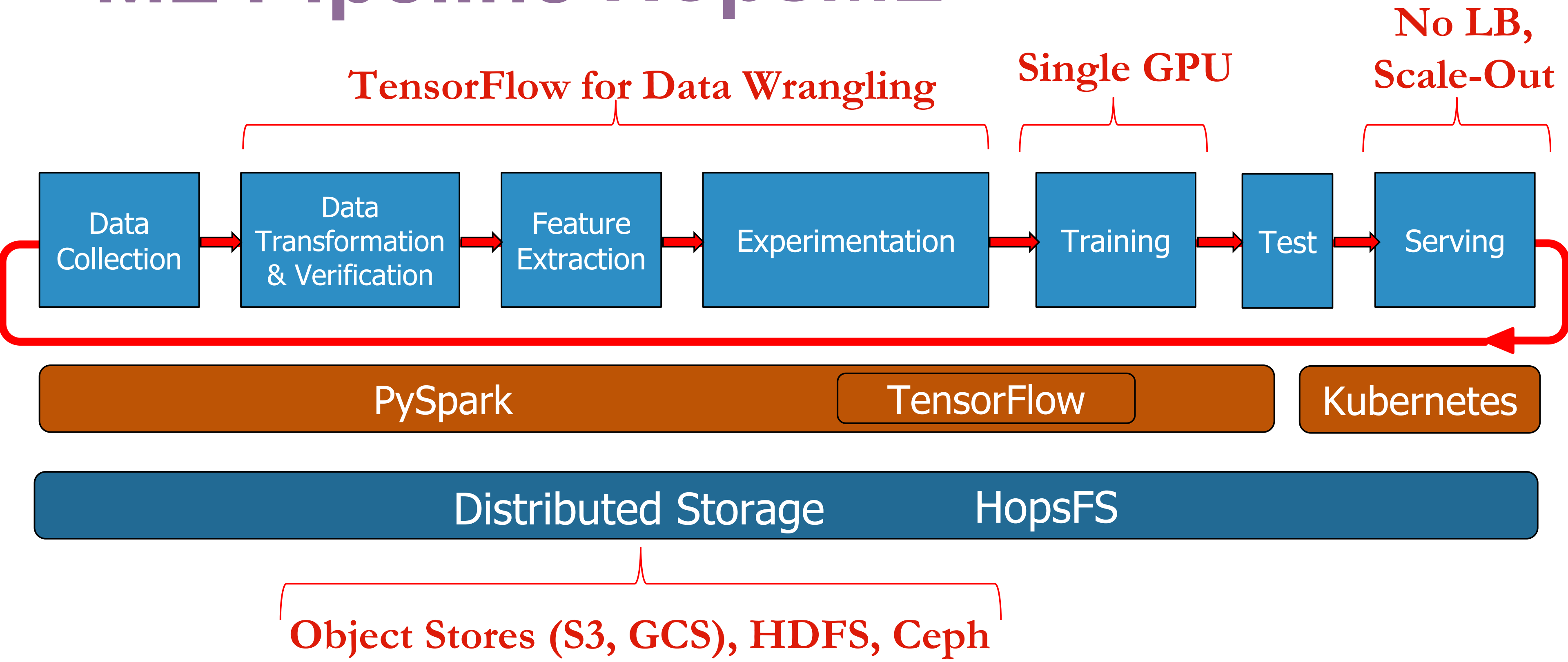
Spark
(Data Prep)

Spark Streaming
(Monitoring Models)

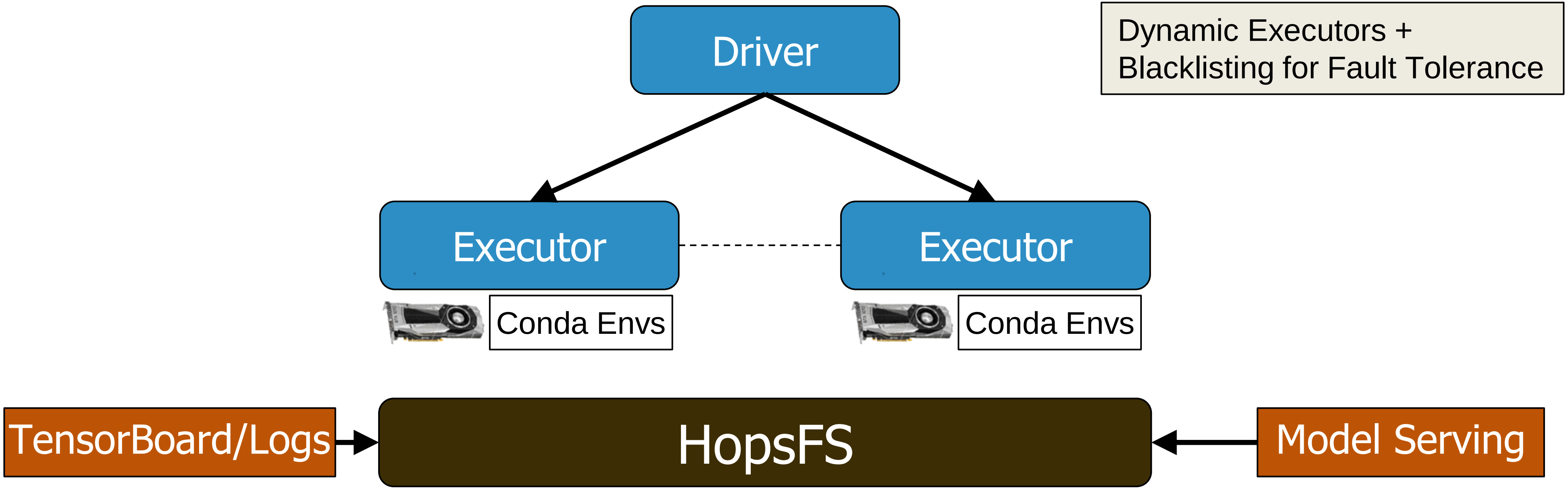
Container
(GPUs)

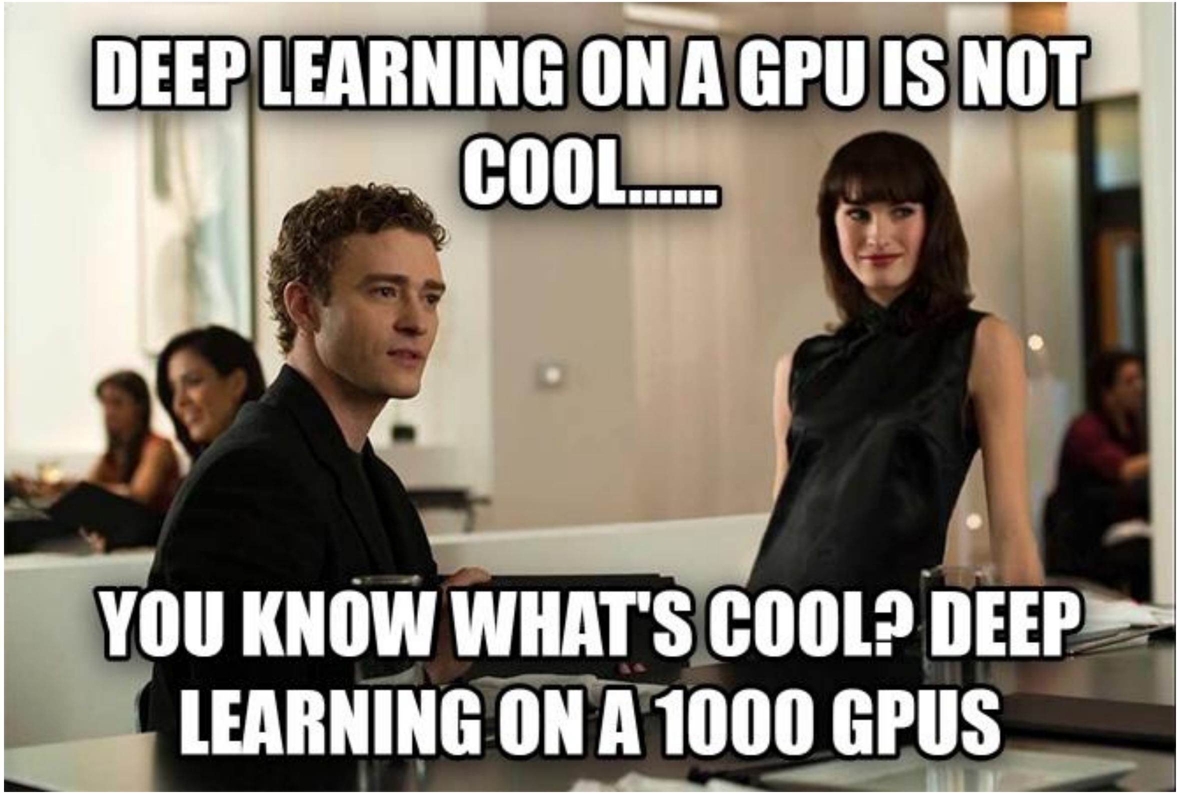
ML Pipeline HopsML

Potential Bottlenecks



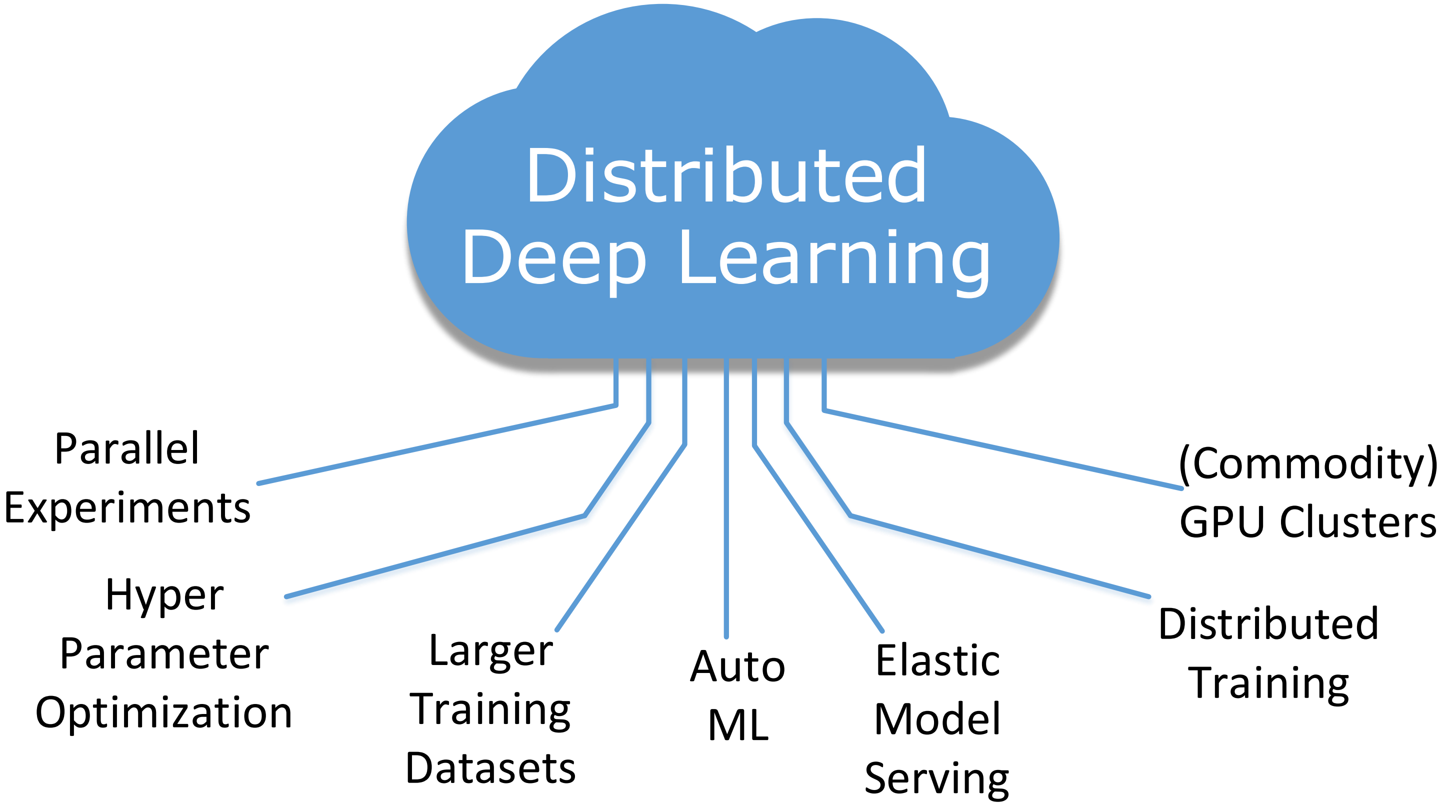
HopsML Spark/TensorFlow Arch





Why Distributed Deep Learning?

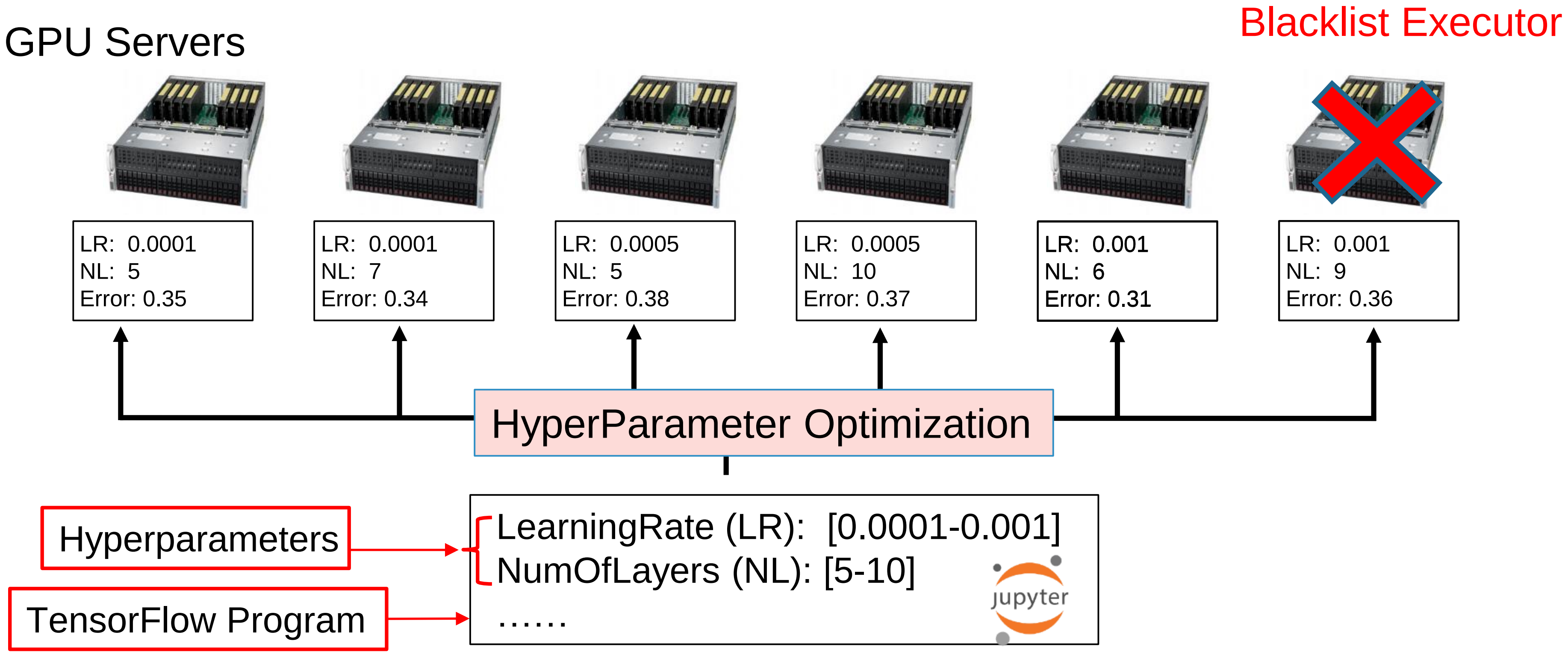
All Roads Lead to Distribution



(Because DL Theory Sucks!)

Hyperparameter Optimization

Faster Experimentation



Declarative or API Approach?

- Declarative Hyperparameters in external files
  - Vizier/CloudML (yaml)
  - Sagemaker (json)*
- API-Driven
  - Databrick's MLFlow
  - HopsML

*https://docs.aws.amazon.com/sagemaker/latest/dg/automatic-model-tuning-define-ranges.html

Google CloudML Hyperparameters

```yaml
scaleTier: CUSTOM
workerCount: 9
parameterServerCount: 3
hyperparameters:
  maxParallelTrials: 1
  params:
  - parameterName: hidden1
    type: INTEGER
    minValue: 40
    maxValue: 400
    scaleType: UNIT_LINEAR_SCALE
  - parameterName: numRnnCells
    type: DISCRETE
    discreteValues:
    - 1
    - 2
  - parameterName: rnnCellType
    type: CATEGORICAL
    categoricalValues:
    - BasicRNNCell
    - GRUCell
    - LSTMCell
```

https://cloud.google.com/ml-engine/docs/tensorflow/using-hyperparameter-tuning

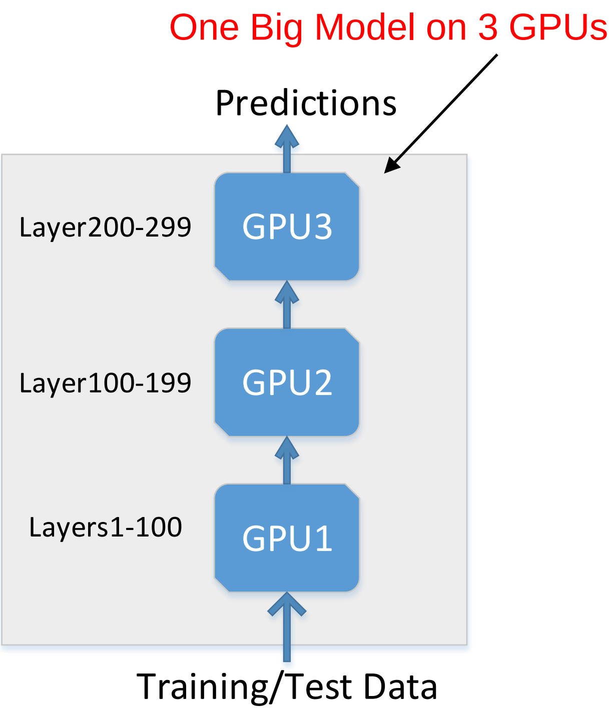
GridSearch for Hyperparameters on HopsML

```
def train(learning_rate, dropout):

  [TensorFlow Code here]

args_dict = {'learning_rate': [0.001, 0.005, 0.01],
             'dropout': [0.5, 0.6]}
experiment.launch(train, args_dict)
```

Launch 6 Spark Executors

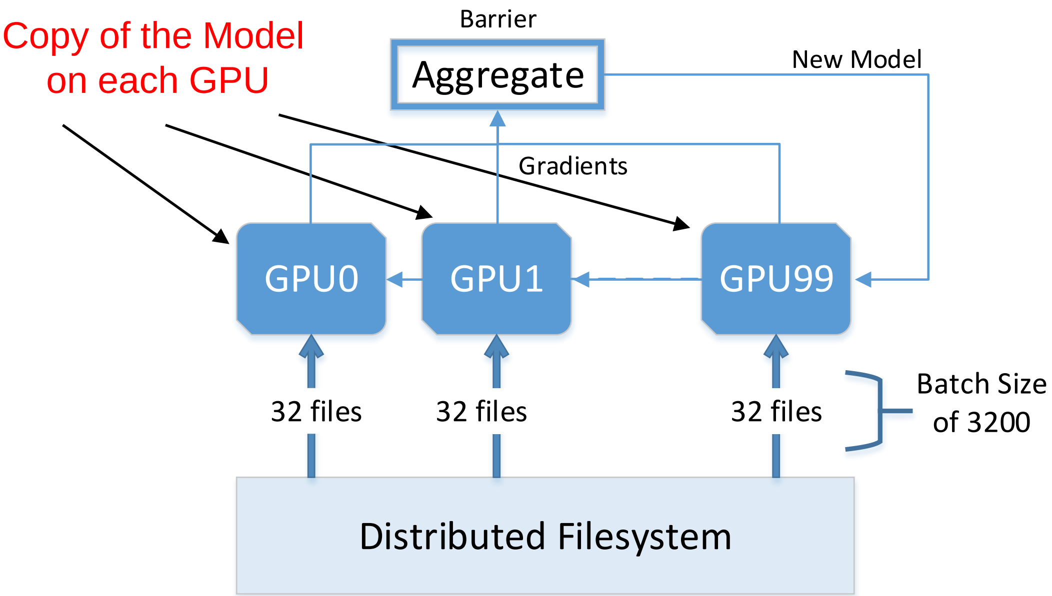
Distributed Training

Model Parallelism

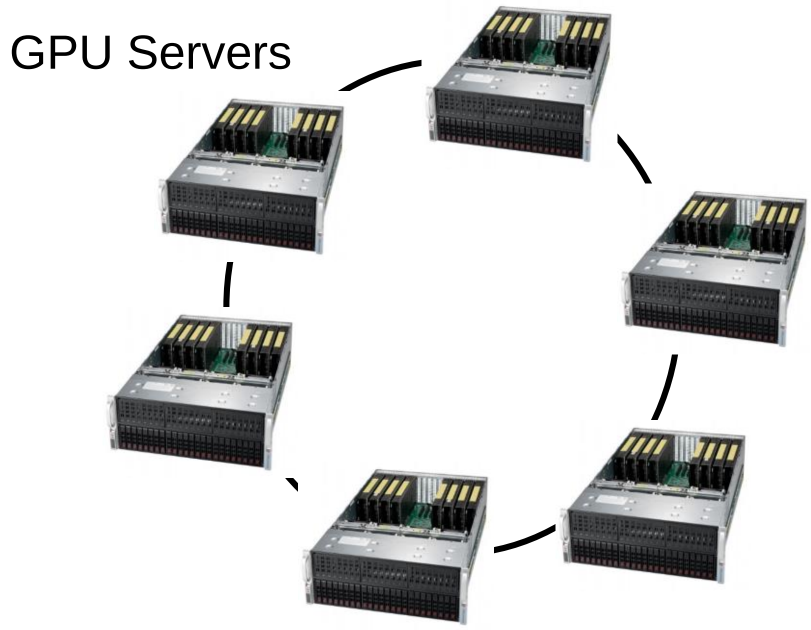


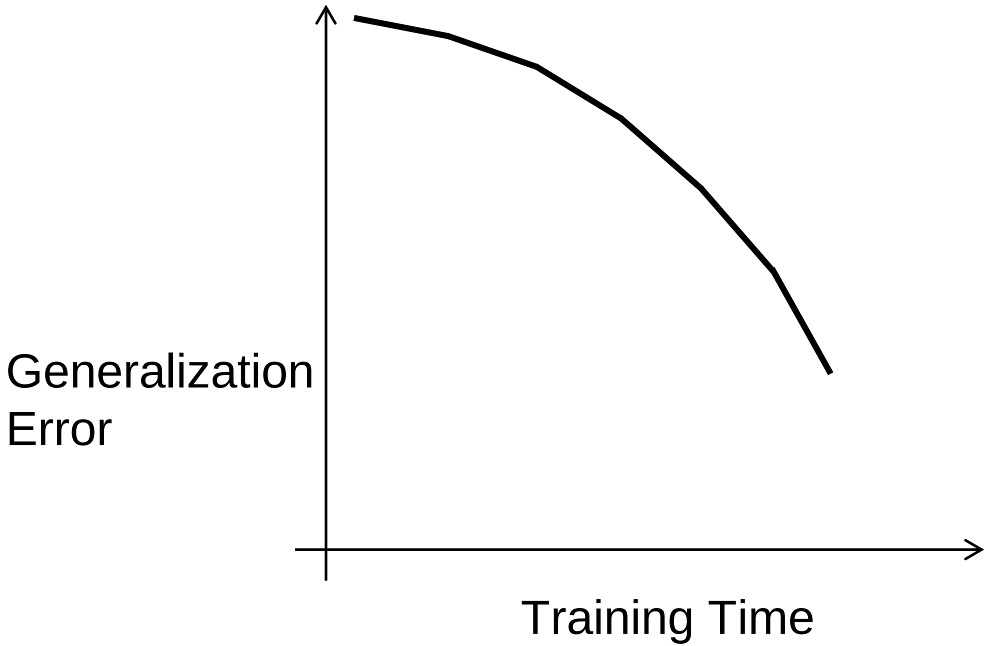
Data Parallelism

(Synchronous Stochastic Gradient Descent (SGD))



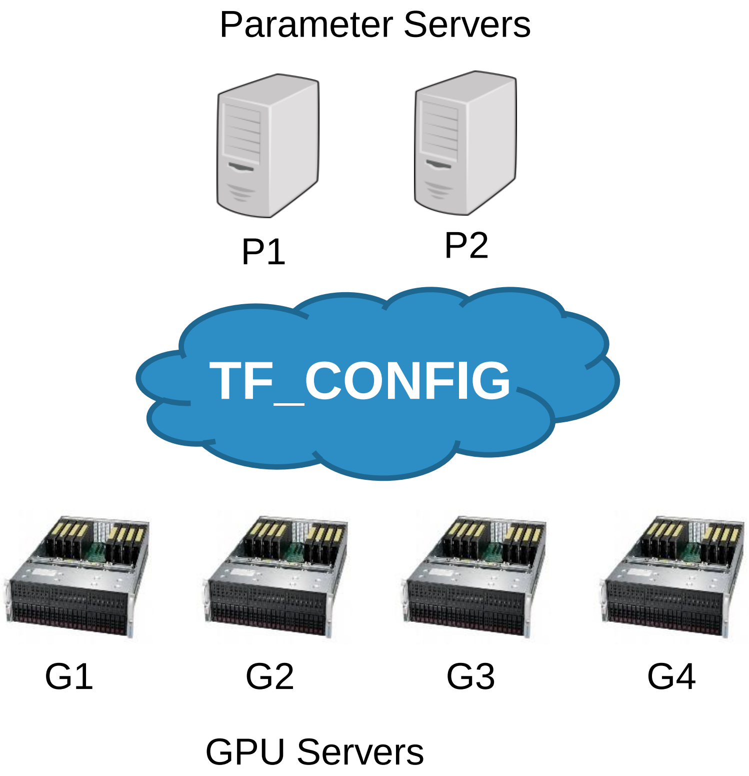
Data Parallel Distributed Training





Frameworks for Distributed Training

Distributed TensorFlow / TfOnSpark

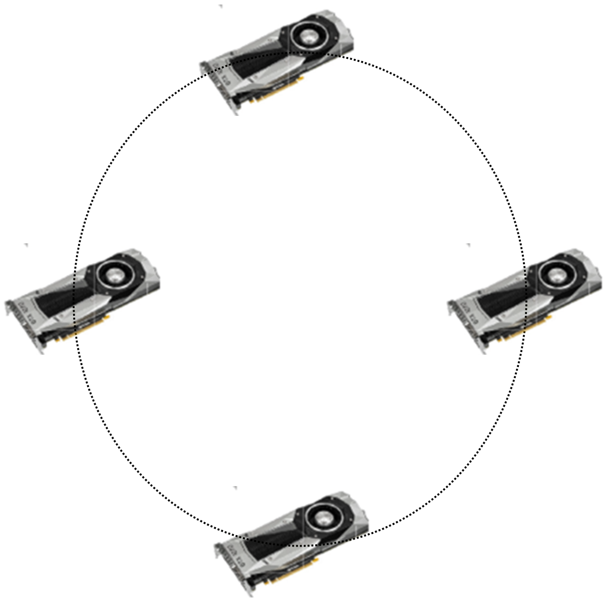


TF_CONFIG

Bring your own Distribution!

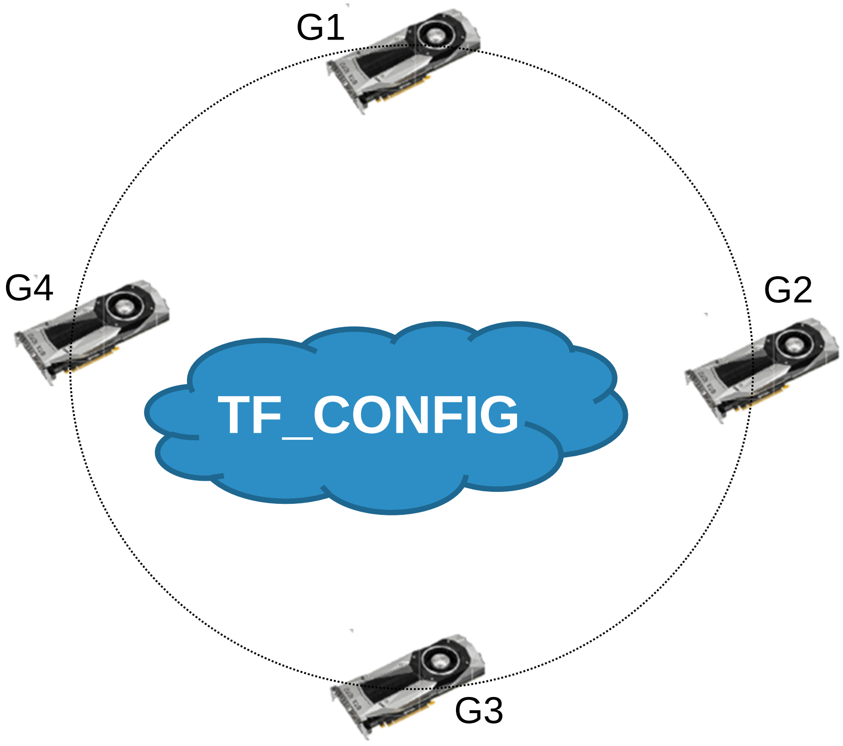
1. Start all processes for P1,P2, G1-G4 yourself
2. Collect all IP addresses in TF_CONFIG along with GPU device IDs.

RingAllReduce (Horovod)



- Bandwidth optimal
- Automatically builds the ring (MPI)
- Supported by HopsML and Databricks' HorovodEstimator

Tf CollectiveAllReduceStrategy



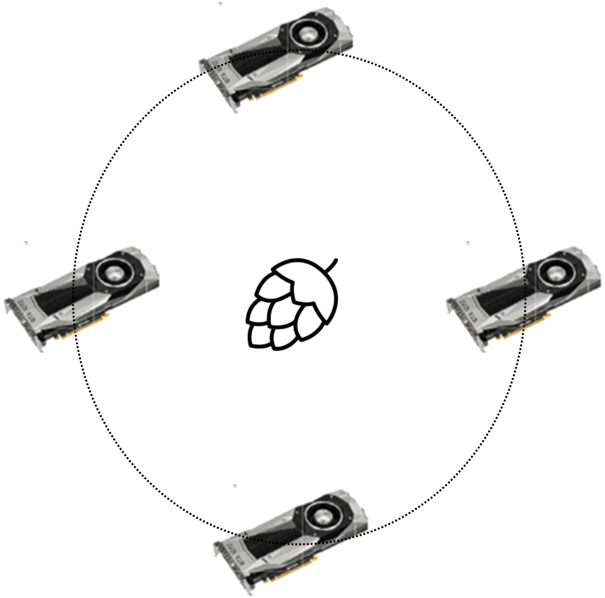
TF_CONFIG

Bring your own Distribution!

1. Start all processes for G1-G4 yourself
2. Collect all IP addresses in TF_CONFIG along with GPU device IDs.

Available from TensorFlow 1.11

HopsML CollectiveAllReduceStrategy



- Uses Spark/YARN to add distribution to TensorFlow’s CollectiveAllReduceStrategy
  - Automatically builds the ring (Spark/YARN)

https://github.com/logicalclocks/hops-util-py

CollectiveAllReduce vs Horovod Benchmark

TensorFlow: 1.11
Model: **Inception v1**
Dataset: imagenet (synthetic)
Batch size: 256 global, 32.0 per device
Num batches: 100
Optimizer Momemtum
Num GPUs: 8
AllReduce: **collective**

Step	Img/sec	total_loss
1	images/sec: 2972.4 +/- 0.0	
10	images/sec: 3008.9 +/- 8.9	
100	images/sec: 2998.6 +/- 4.3	

----

total images/sec: **2993.52**

TensorFlow: 1.7
Model: **Inception v1**
Dataset: imagenet (synthetic)
Batch size: 256 global, 32.0 per device
Num batches 100
Optimizer Momemtum
Num GPUs: 8
AllReduce: **horovod**

Step	Img/sec	total_loss
1	images/sec: 2816.6 +/- 0.0	
10	images/sec: 2808.0 +/- 10.8	
100	images/sec: 2806.9 +/- 3.9	

----

total images/sec: **2803.69**

Small Model

https://groups.google.com/a/tensorflow.org/forum/#!topic/discuss/7T05tNV08Us

CollectiveAllReduce vs Horovod Benchmark

TensorFlow:	1.11
Model:	**VGG19**
Dataset:	imagenet (synthetic)
Batch size:	256 global, 32.0 per device
Num batches:	100
Optimizer	Momemtum
Num GPUs:	8
AllReduce:	**collective**

Step	Img/sec	total_loss
1	images/sec: 634.4 +/- 0.0	
10	images/sec: 635.2 +/- 0.8	
100	images/sec: 635.0 +/- 0.5	

----

total images/sec: **634.80**

TensorFlow:	1.7
Model:	**VGG19**
Dataset:	imagenet (synthetic)
Batch size:	256 global, 32.0 per device
Num batches	100
Optimizer	Momemtum
Num GPUs:	8
AllReduce:	**horovod**

Big Model

Step	Img/sec	total_loss
1	images/sec: 583.01 +/- 0.0	
10	images/sec: 582.22 +/- 0.1	
100	images/sec: 583.61 +/- 0.2	

----

total images/sec: **583.61**

https://groups.google.com/a/tensorflow.org/forum/#!topic/discuss/7T05tNV08Us

Reduction in LoC for Dist Training

Released	Framework	Lines of Code in Hops
March 2016	DistributedTensorFlow	~1000
Feb 2017	TensorFlowOnSpark*	~900
Jan 2018	Horovod (Keras)*	~130
June 2018	Databricks’ HorovodEstimator	~100
Sep 2018	HopsML (Keras/CollectiveAllReduce)*	~100

*https://github.com/logicalclocks/hops-examples

**https://docs.azuredatabricks.net/_static/notebooks/horovod-estimator.html

HopsML CollectiveAllReduceStrategy with Keras

```
def distributed_training():
    def input_fn(): # return dataset
    model = …
    optimizer = …
    model.compile(…)
    rc = tf.estimator.RunConfig(‘CollectiveAllReduceStrategy’)
    keras_estimator = tf.keras.estimator.model_to_estimator(….)
    tf.estimator.train_and_evaluate(keras_estimator, input_fn)

experiment.allreduce(distributed_training)
```

HopsML CollectiveAllReduceStrategy

- Scale to 10s or 100s of GPUs on Hops
- Generate Tensorboard Logs in HopsFS
- Checkpoint to HopsFS
- Save a trained model to HopsFS
- Experiment History
  - Reproducible training

Add Tensorboard Support

```
def distributed_training():
  from hops import tensorboard
  model_dir = tensorboard.logdir()
  def input_fn(): # return dataset
  model = …
  optimizer = …
  model.compile(…)
  rc = tf.estimator.RunConfig(‘CollectiveAllReduceStrategy’)
  keras_estimator = keras->model_to_estimator(model_dir)
  tf.estimator.train_and_evaluate(keras_estimator, input_fn)

experiment.allreduce(distributed_training)
```

GPU Device Awareness

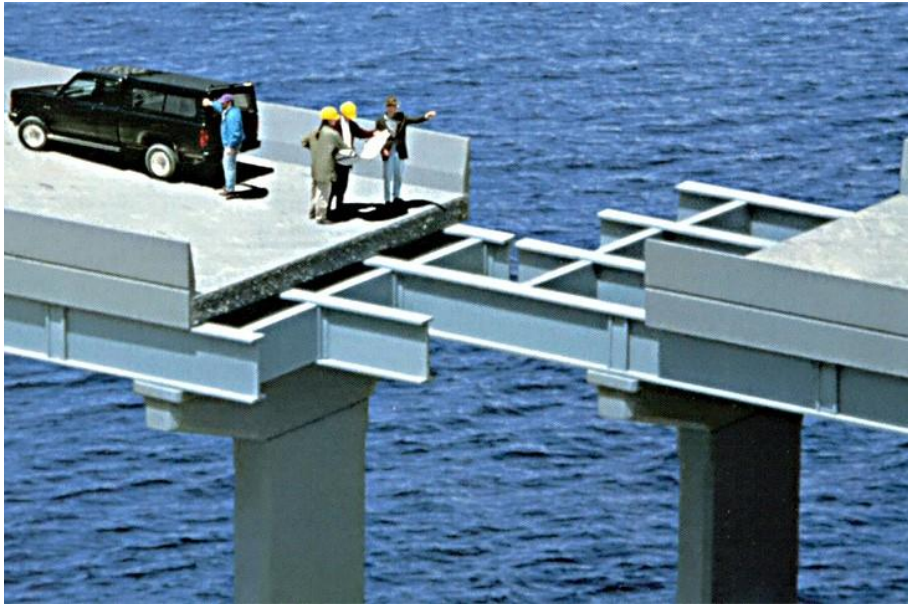
```
def distributed_training():
  from hops import devices
  def input_fn(): # return dataset
  model = …
  optimizer = …
  model.compile(…)
  est->RunConfig(num_gpus_per_worker=devices.get_num_gpus())
  keras_estimator = keras->model_to_estimator(…)
  tf.estimator.train_and_evaluate(keras_estimator, input_fn)

experiment.allreduce(distributed_training)
```

Experiment Versioning (.ipynb, conda, results)

```
def distributed_training():
  def input_fn(): # return dataset
  model = …
  optimizer = …
  model.compile(…)
  rc = tf.estimator.RunConfig(‘CollectiveAllReduceStrategy’)
  keras_estimator = keras->model_to_estimator(…)
  tf.estimator.train_and_evaluate(keras_estimator, input_fn)

notebook = hdfs.project_path()+'/Jupyter/Experiment/inc.ipynb'
experiment.allreduce(distributed_training, name='inception',
   description='A inception example with hidden layers‘,
   versioned_resources=[notebook])
```

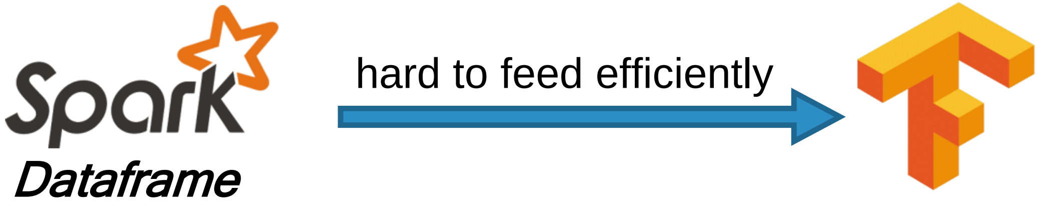
SPARK+AI SUMMIT EUROPE

Experiment Versioning/History/Reproduce



The Data Layer

The Data Layer



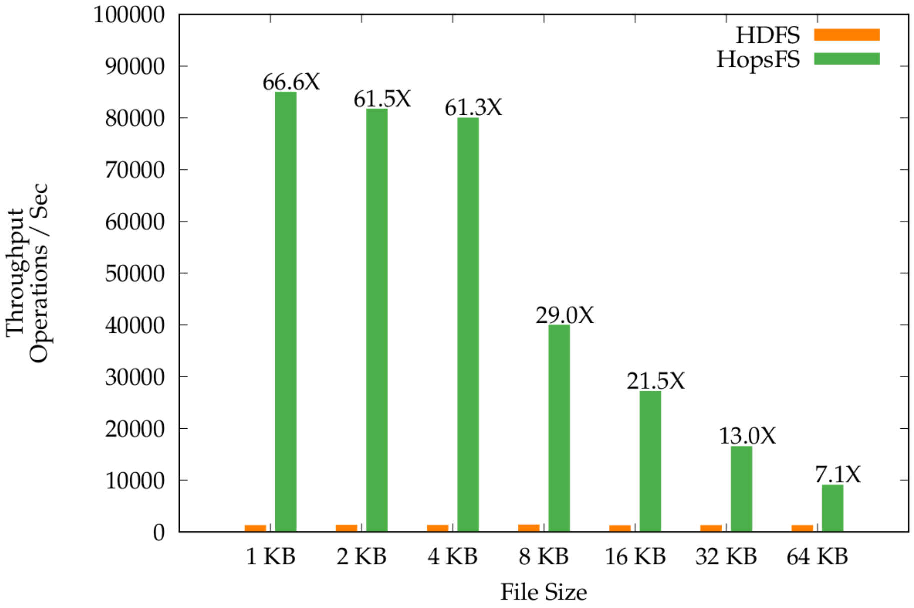
FEED_DICT is single threaded (Python GIL)

TensorFlow Dataset API does not support DFs

- Petastorm (Uber) for Parquet->TensorFlow training
- What about Datafiles (.csv, images, txt)?

HopsFS

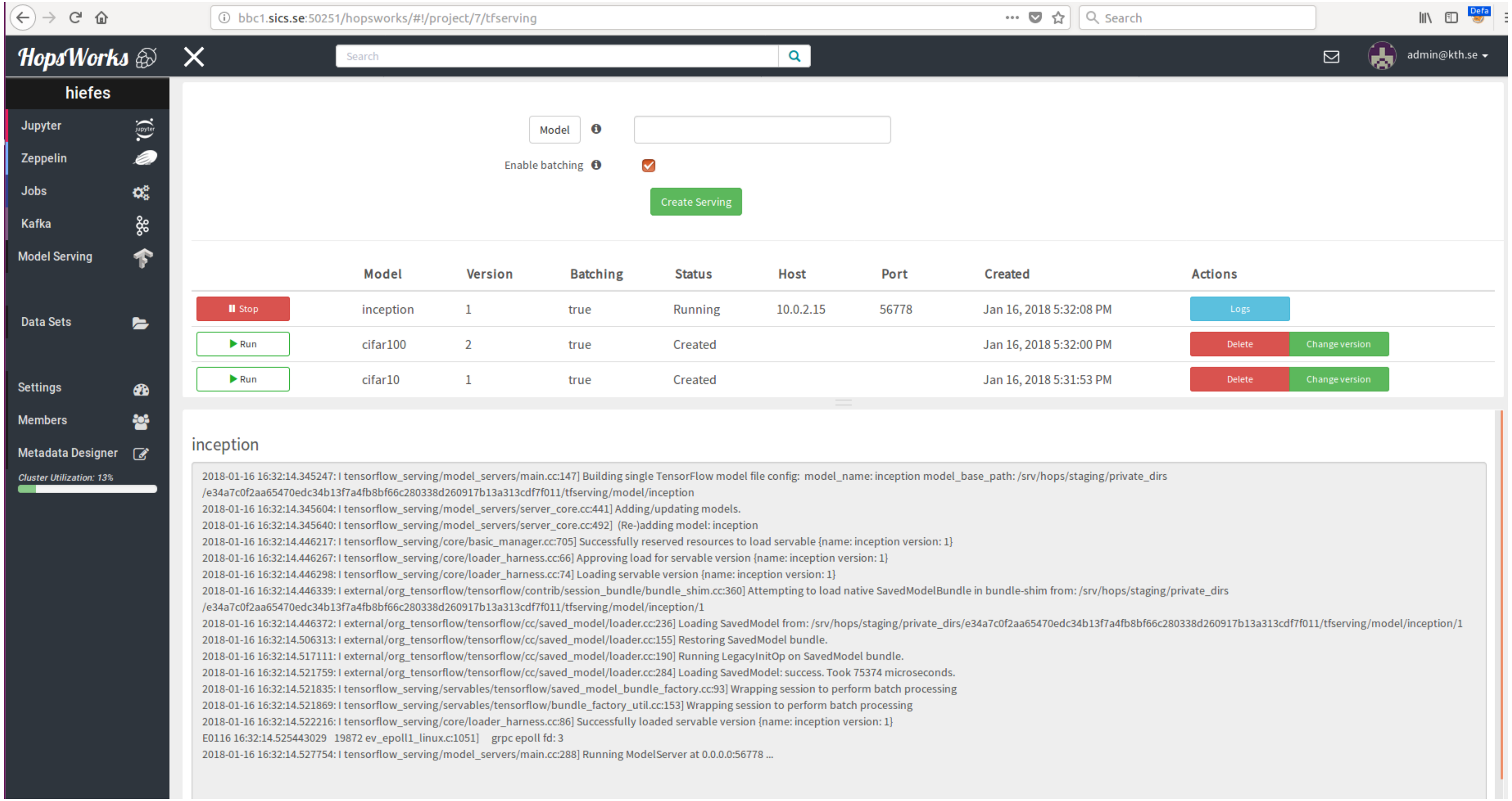
- HDFS derivative with Distributed Metadata
  - 16X HDFS throughput.
  - Winner IEEE Scale Prize 2017
- Integrates NVMe disks transparently*
  - Store small files (replicated) on NVMe hardware



a. File Write Performance

*Size Matters: Improving the Performance of Small Files in Hadoop, Middleware 2018. Niazi et al

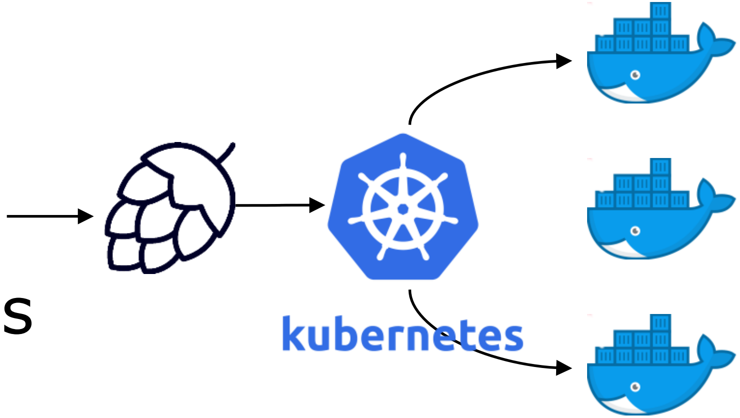
Model Serving on Kubernetes



Kubernetes Model Serving

- Elastic scaling for model serving

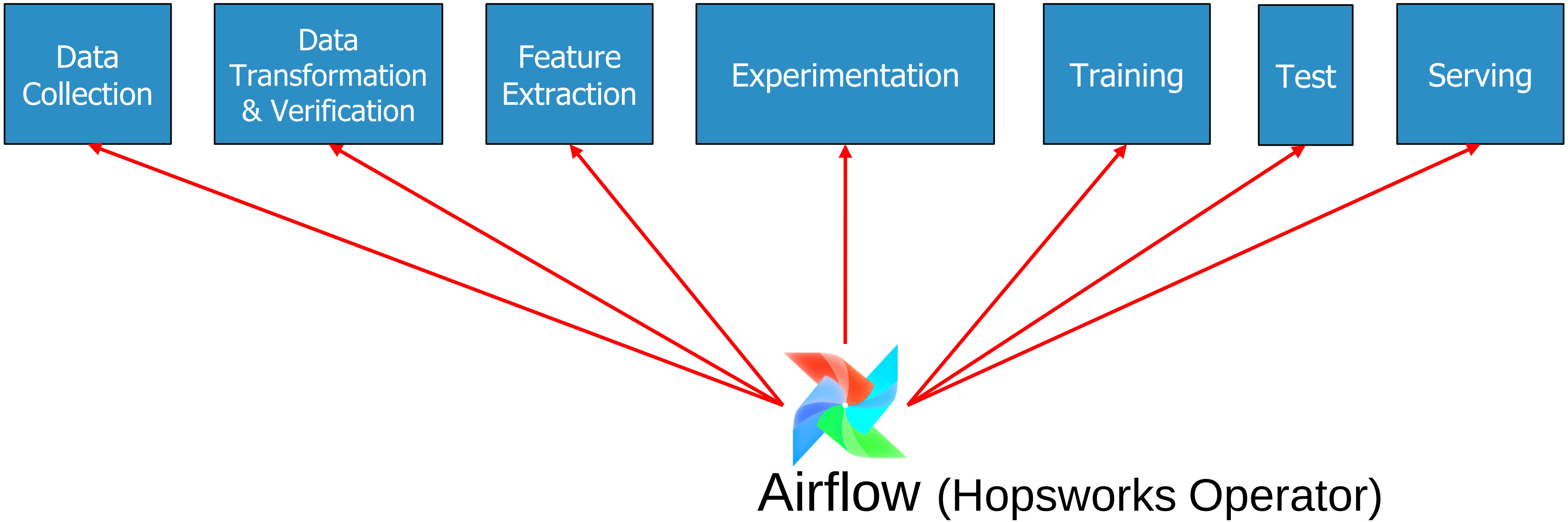
- Supports:
  - Fault tolerance
  - Rolling release new models
  - Autoscaling



Model Monitoring with Spark Streaming

- Log model inference requests/results to Kafka
- Spark monitors model performance and input data
- When to retrain?
  - If you look at the input data and use **covariant shift** to see when it deviates significantly from the data that was used to train the model on.

Orchestrating HopsML Workflows



Summary

- The future of Deep Learning is Distributed
  https://www.oreilly.com/ideas/distributed-tensorflow

- Hops is a new Data Platform with first-class support for Python / Deep Learning / ML / Data Governance / GPUs

hopshadoop

logicalclocks