DATA+AI SUMMIT EUROPE
FORMERLY SPARK+AI SUMMIT

Scaling Machine Learning with Apache Spark

Holly Smith

Niall Turbitt

#DataTeams #DataAISummit

About

Holly Smith

Senior Consultant at Databricks

- Professional Services and Training
- Experience
  - Credit Risk & Decisioning
  - Mobile Banking
  - Forecasting & Optimisation
- BSc Mathematics University of Greenwich

databricks

About

Niall Turbitt

Senior Data Scientist at Databricks

- Professional Services and Training
- Experience
    - e-Commerce
    - Supply Chain and Logistics
    - Recommender Systems & Personalisation
- MS Statistics University College Dublin
- BA Mathematics & Economics Trinity College Dublin

Outline

- Motivation
- Spark Architecture Recap
- Paradigms of ML on Spark:
  - Training & Tuning
    - Spark MLlib
    - Pandas Function APIs
    - Hyperopt
  - Inference
    - Pandas UDFs

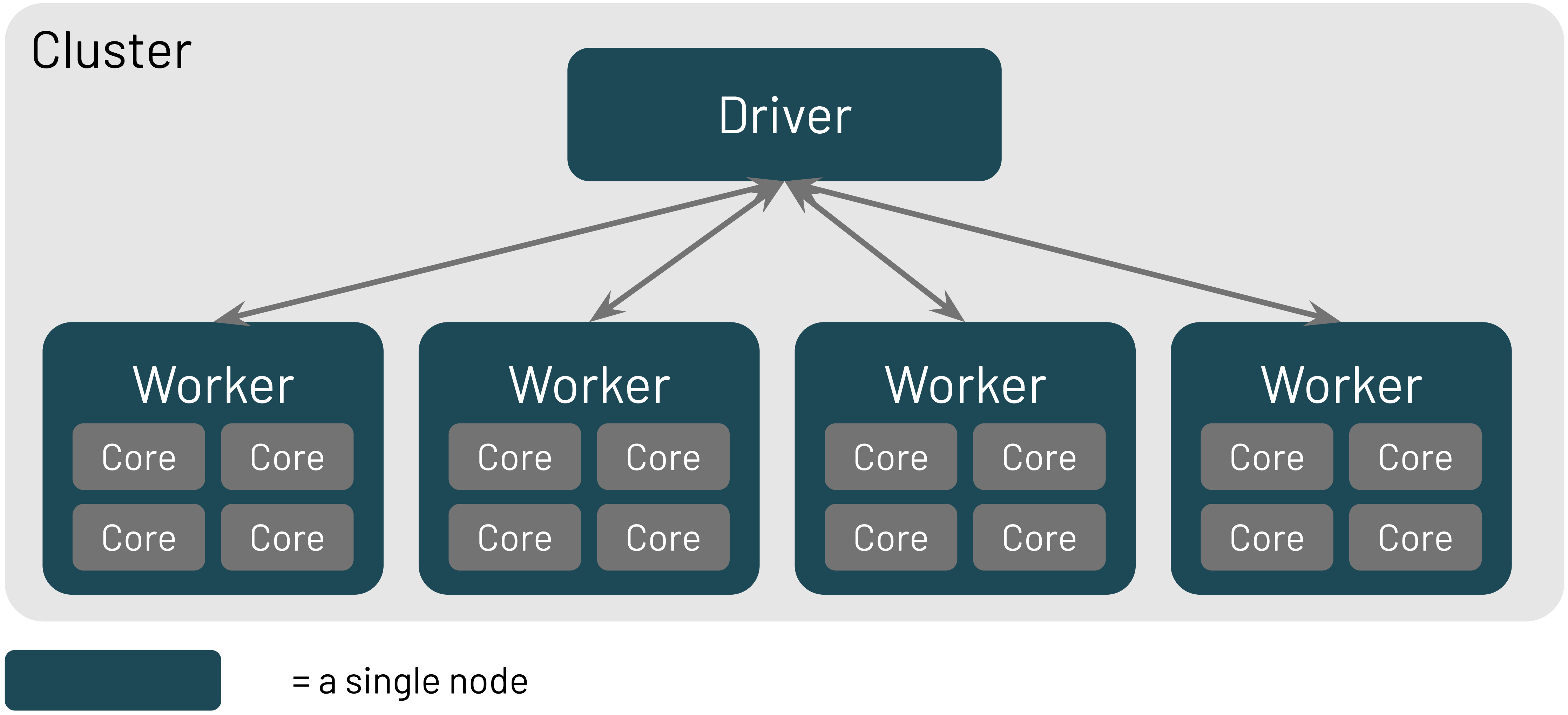
Motivation

- Confusion around where and when to use Spark for Machine Learning
- Spark is powerful: harness the full potential of Spark for Machine Learning
- Fast moving environment
  - Use new APIs and techniques before they become mainstream

Motivation

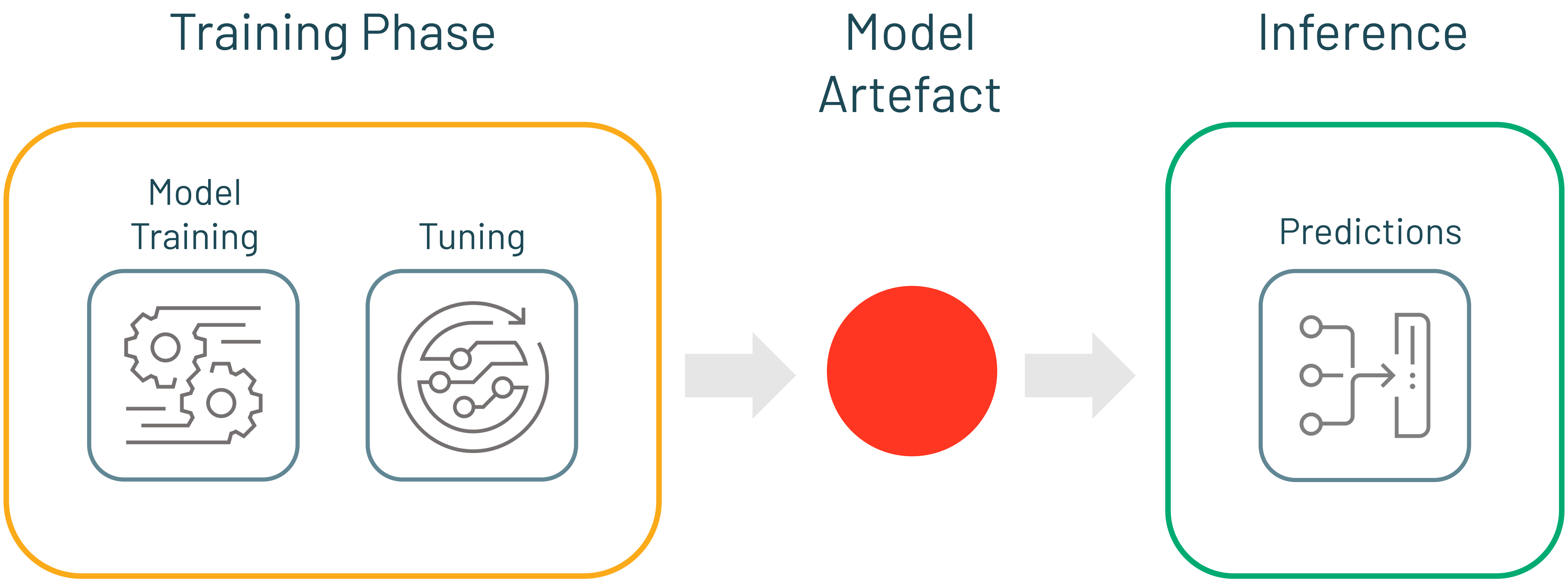
- Confusion around where and when to use Spark for Machine Learning
- Spark is powerful: harness the full potential of Spark for Machine Learning
- Fast moving environment
  - Use new APIs and techniques before they become mainstream

Refresher: Spark Architecture

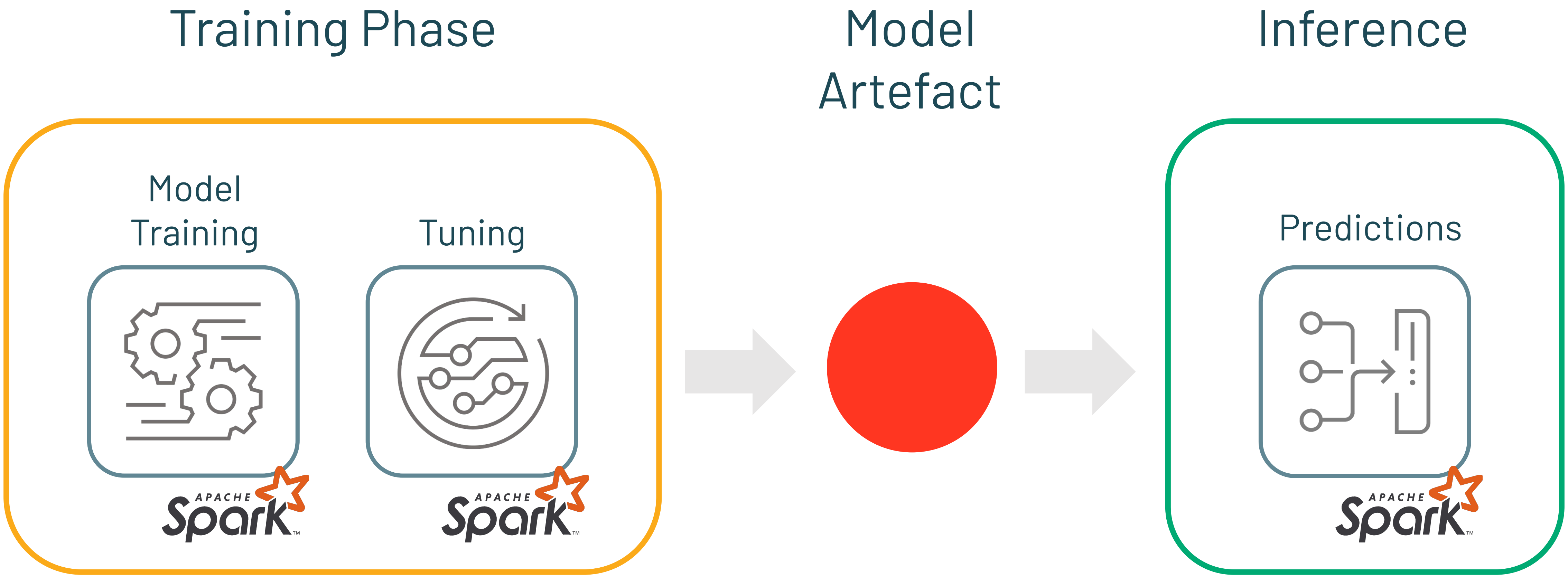


ML on Spark

The ML Lifecycle



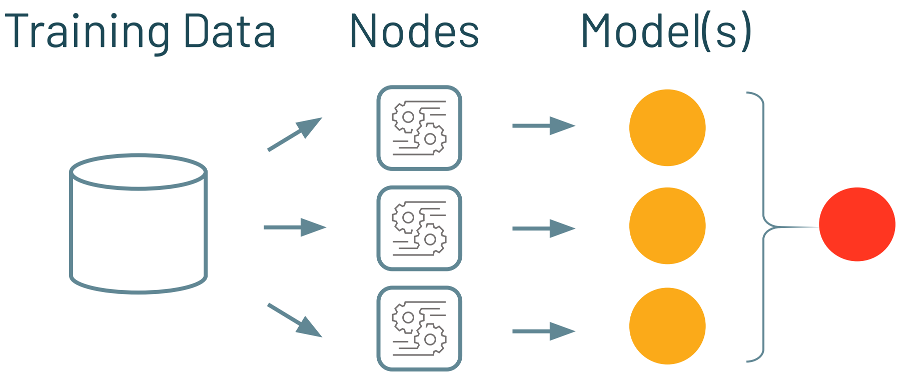
The ML Lifecycle



ML Training & Tuning on Spark

Distributed
ML Library

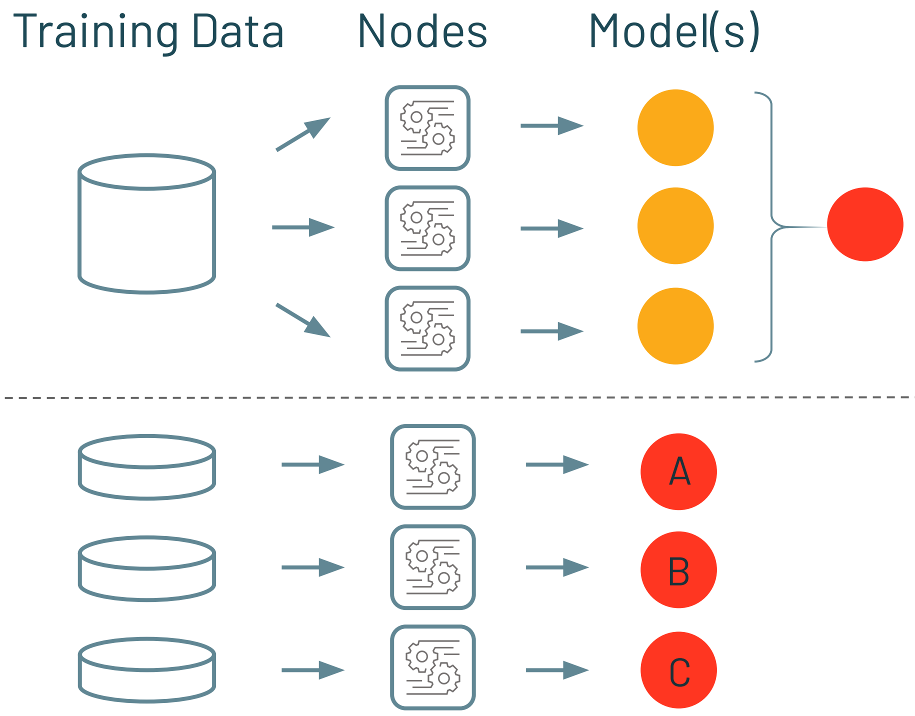
Data Parallel
Training



Training a single model across multiple nodes

Creates one model artefact

ML Training & Tuning on Spark



Distributed ML Library

Data Parallel Training

Training a single model across multiple nodes

Creates one model artefact

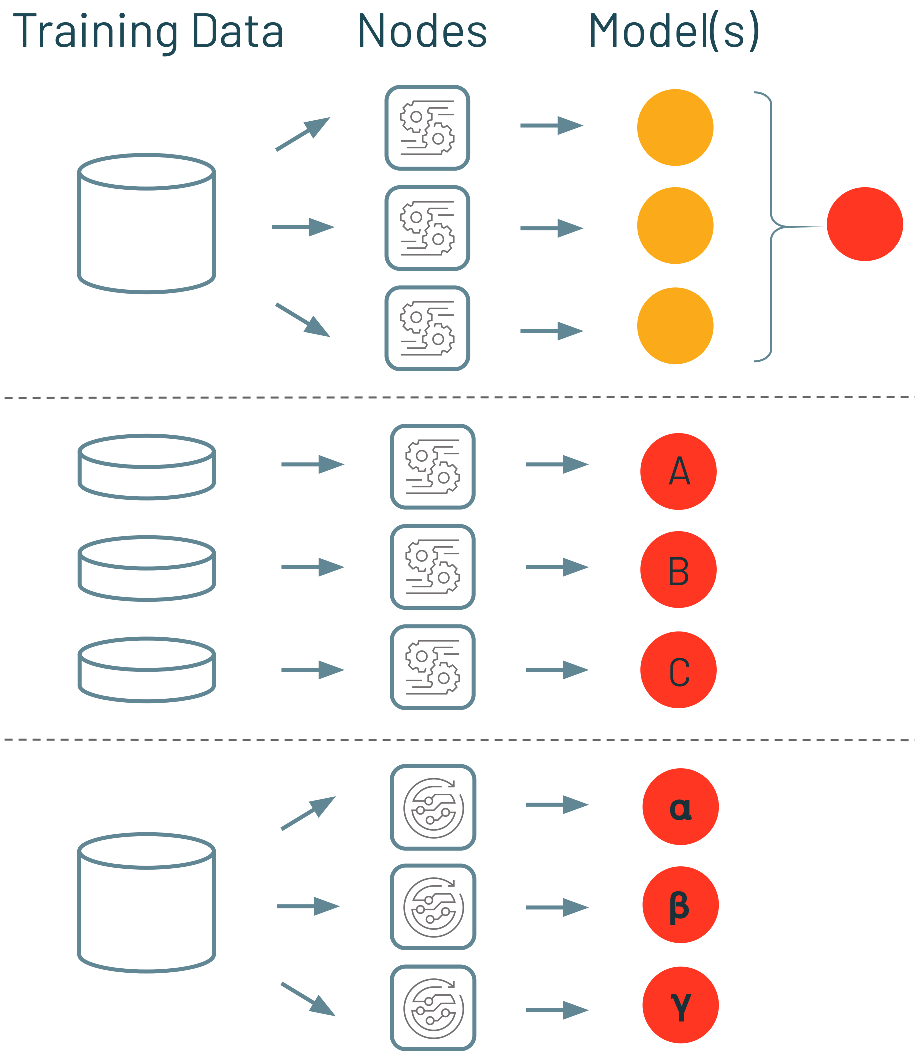
Single Node ML Library

Parallel Model Training

Training multiple unique versions of a model in parallel

Sending different groups of data to each node, creating a model per group

ML Training & Tuning on Spark



Distributed ML Library

Data Parallel Training

Training a single model across multiple nodes

Creates one model artefact

Single Node ML Library

Parallel Model Training

Training multiple unique versions of a model in parallel

Sending different groups of data to each node, creating a model per group

Combination

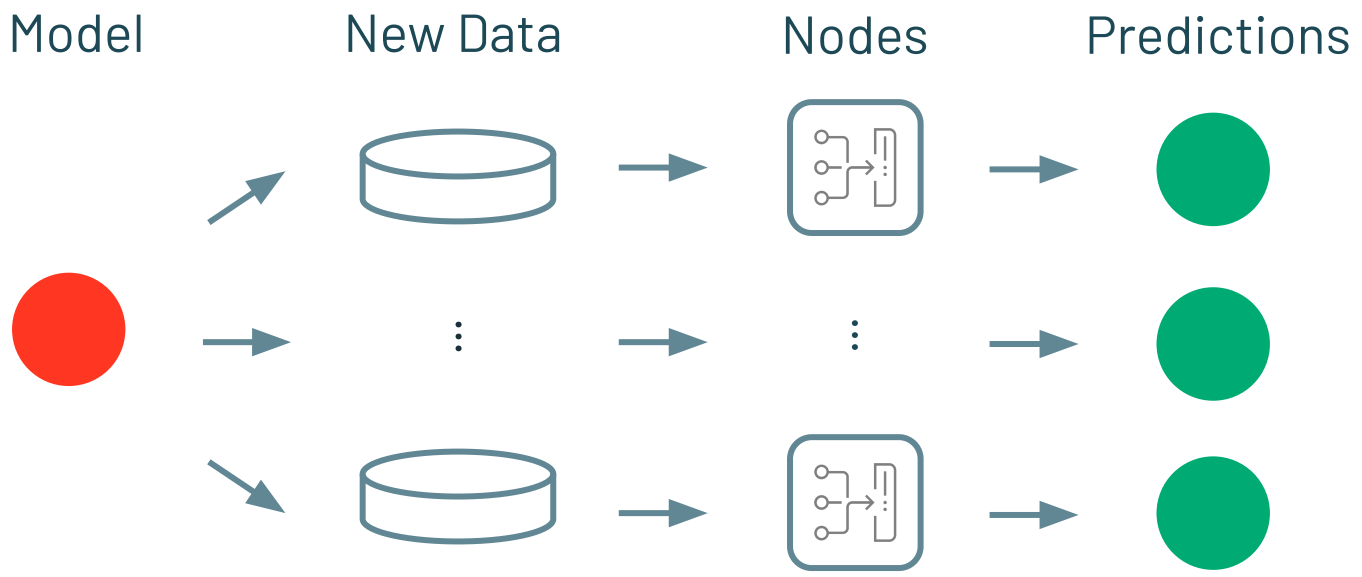
Parallel Hyperparameter Optimisation

Evaluating multiple hyperparameter configurations in parallel

ML Inference on Spark

For both distributed and single node ML libraries:

1. Take trained model
2. Distribute new instances
3. Apply model in parallel



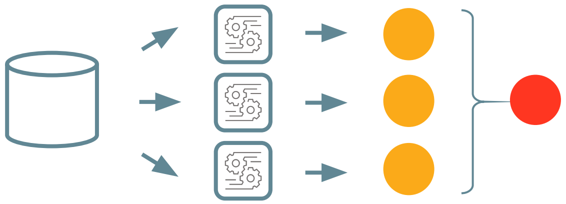
ML Project Considerations

- Data Dependent
- Compute Resources Available
- Single machine vs distributed computing
- Inference: Deployment Requirements

	Throughput	Latency	Example
Batch	High	Hours to days	Customer churn prediction
Streaming	Medium	Seconds to minutes	Predictive maintenance
Real-time	Low	Milliseconds	Fraud detection

Spark MLlib

Parallelising Single-Model Training



Distributed ML Library

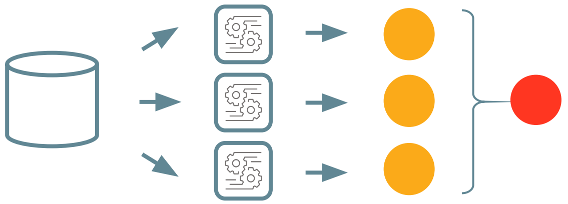
- Spark's Machine Learning Library
  - ML algorithms
  - Featurization
  - Pipelines
- MLlib vs sklearn
- A note on terminology:

What is meant by "MLlib"

`Spark.mllib` RDD based API **Maintenance Mode**	`Spark.ml` Dataframe based API **Recommended**

Spark MLlib

Parallelising Single-Model Training



Distributed ML Library

- Spark's Machine Learning Library
  - ML algorithms
  - Featurization
  - Pipelines
- MLlib vs sklearn
- A note on terminology:

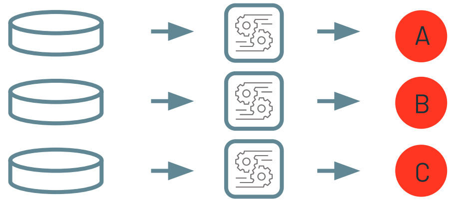
```
from pyspark.ml.regression import LinearRegression

train_df = spark.read…
test_df = spark.read…

lr = LinearRegression().fit(train_df)
predictions = lr.transform(test_df)
```

What is meant by "MLlib"

Spark.mllib RDD based API	Spark.ml Dataframe based API
Maintenance Mode	**Recommended**

Pandas Function API - Grouped Map

Parallelising training of independent models

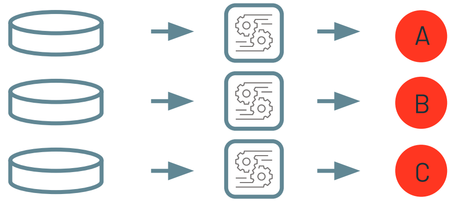


Single Node ML Library

- `DataFrame.groupby().applyInPandas()` **NEW**
- Directly apply a Python native function against a Spark DataFrame as if each group is a Pandas DataFrame
- “split-apply-combine” pattern:
  - Split data into groups
  - Apply function on each group
  - Combine results into new Spark DataFrame

Pandas Function API - Grouped Map

Parallelising training of independent models



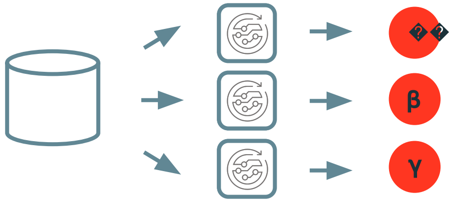
Single Node ML Library

- `DataFrame.groupby().applyInPandas()` **NEW**
- Directly apply a Python native function against a Spark DataFrame as if each group is a Pandas DataFrame
- "split-apply-combine" pattern:
  - Split data into groups
  - Apply function on each group
  - Combine results into new Spark DataFrame

```
def train_model(pd.Dataframe)->pd.DataFrame:

    # fit single-node model
    sklearn_model.fit()
    ...

    return pandas_df


spark_df.groupBy("device_id")
        .applyInPandas(train_model,
                       schema=return_schema)
```

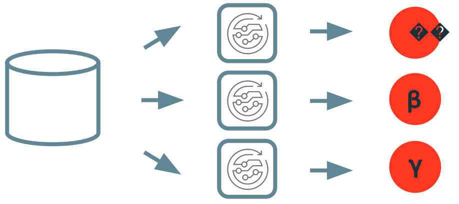
Hyperopt

Parallelising Hyperparameter Optimisation



Combination

- Open source hyperparameter optimisation package
  - Enables either serial or parallel optimisation over provided search spaces
  - Can tune both distributed and single node libraries
  - HOWEVER: distributed training and distributed tuning don't mix
- Bayesian based approach
  - Adaptively selects new hyperparameter settings to explore based on prior results
  - Enables exploration of the hyperparameter space in an intelligent way
  - Allows a wider search space with more hyperparameters

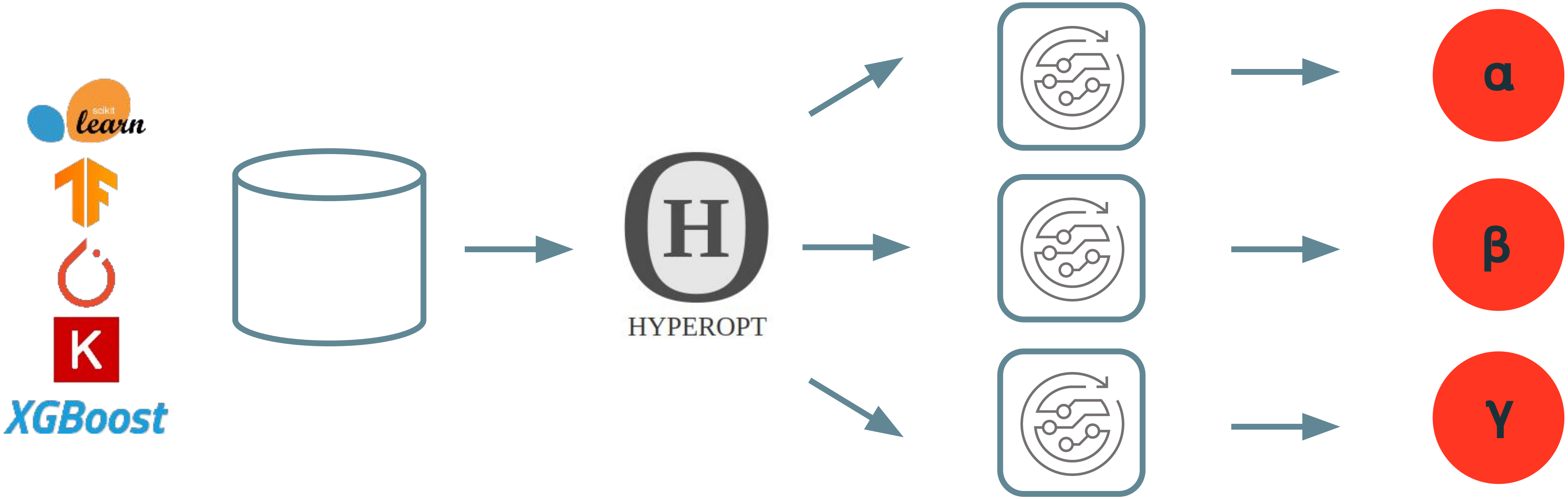


Combination

Hyperopt

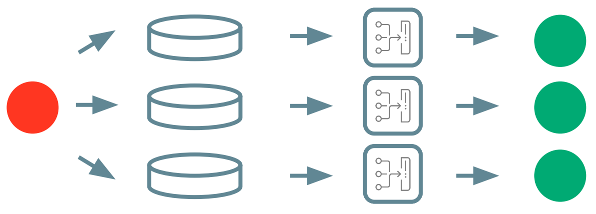
Parallelising Hyperparameter Optimisation

- Distributed Hyperopt with a single node library
  - SparkTrials



Pandas Scalar Iterator UDF

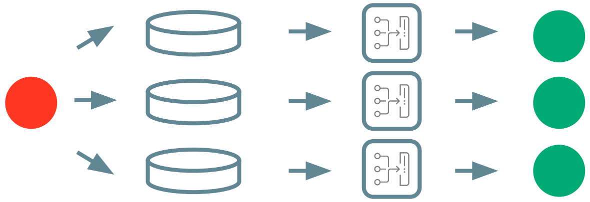
Distributing inference



- Pandas UDFs can accept an iterator of `pandas.Series` or `pandas.DataFrame`
- Spark DataFrame is split into batches and the function called for each batch
- Iterator negates the need to repeatedly load the same model for every batch in the same Python worker process

Pandas Scalar Iterator UDF

Distributing inference



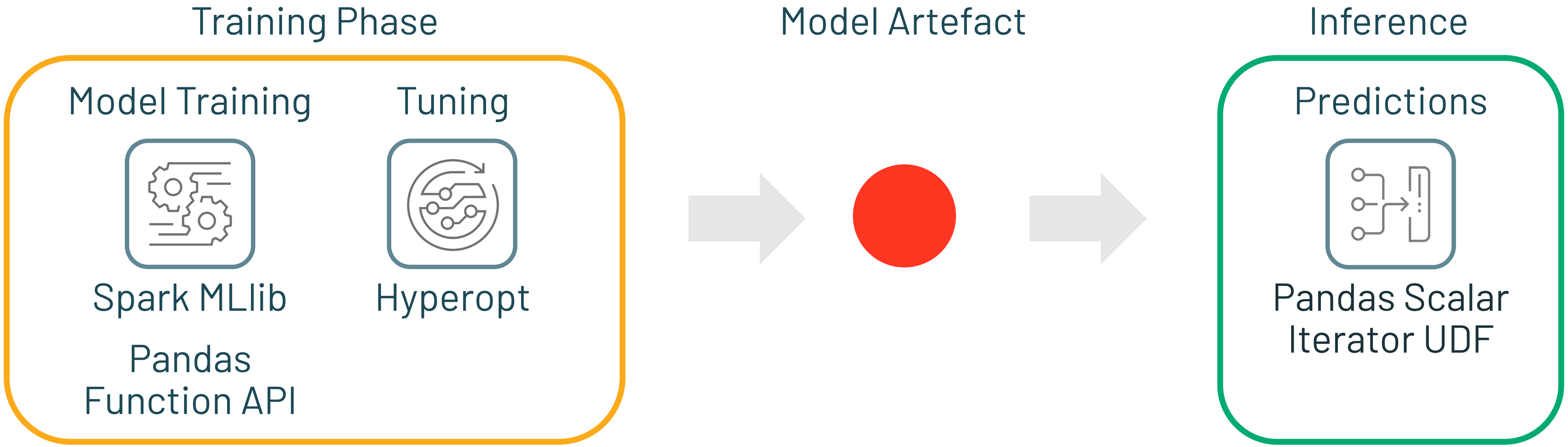
Inference

- Pandas UDFs can accept an iterator of `pandas.Series` or `pandas.DataFrame`
- Spark DataFrame is split into batches and the function called for each batch
- Iterator negates the need to repeatedly load the same model for every batch in the same Python worker process

```
@pandas_udf
def predict_udf(iterator):

    # load model
    model = ...

    for features in iterator:
        yield pd.Series(model.predict(features))


spark_df.withColumn("prediction",
                    predict_udf(*input_cols))
```

Conclusion

Distributing workloads allows you to scale, either by using libraries that are multi or single node to suit your project.



Resources

Notebook: bit.ly/scaling_ml_spark_2020
Pandas UDF Blog post - bit.ly/Pandas_UDF
Docs:

- MLlib - bit.ly/ML_lib
- Hyperopt - bit.ly/hyperopt_spark
- Pandas Grouped Map - bit.ly/grouped_map

Feedback

Your feedback is important to us.

Don't forget to rate and review the sessions.