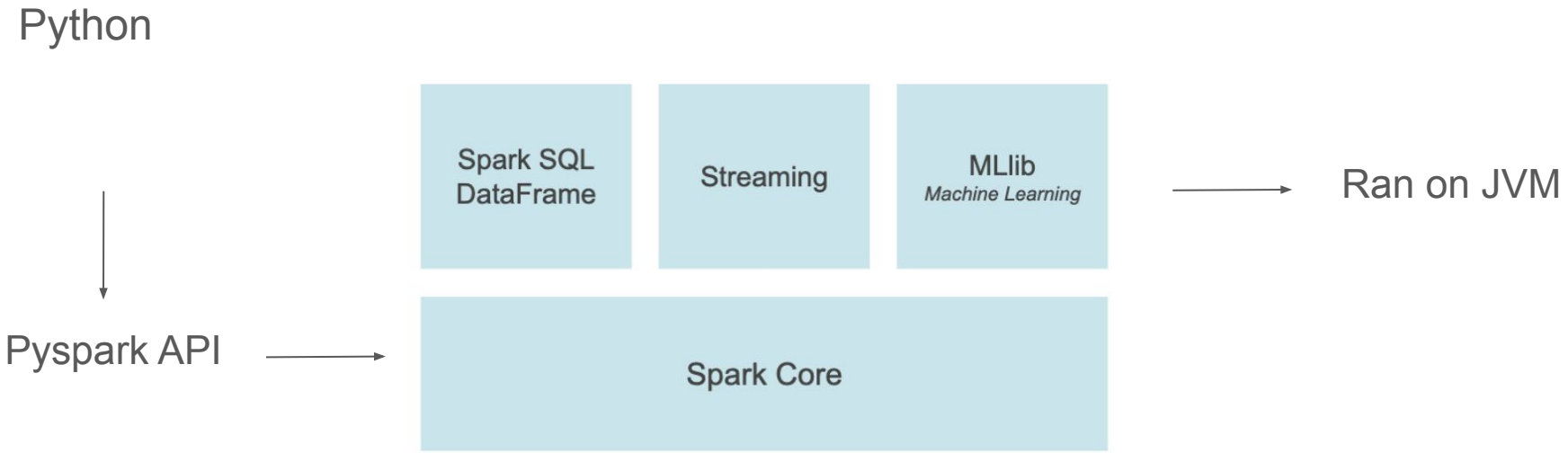
Simplifying Testing of Spark Applications

Megan Yow

fugue

Demo

How Pyspark actually works



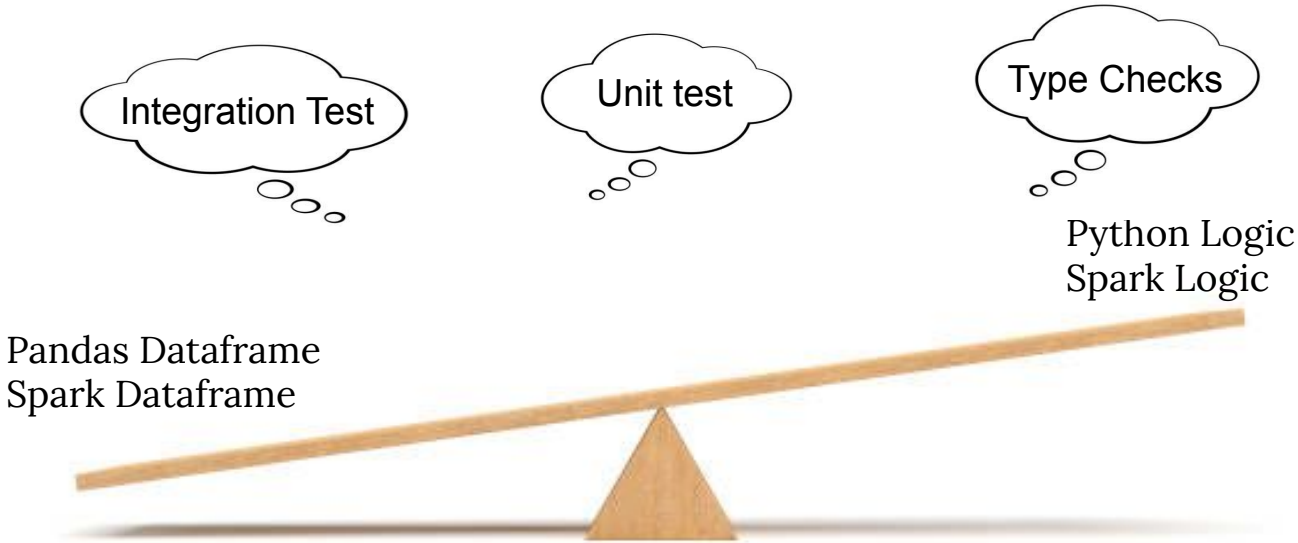
Pyspark is not python native

A Balance of what you want

Python Logic
Spark Logic

Pandas Dataframe
Spark Dataframe

A Balance of what you want



Native Execution Trumps Cluster Spin Up

	Pandas Dataframe	**Spark RDD/ Dataframe**
Python Logic	• No cluster spin up **<u>Easiest to test</u>**	• Have to spin up cluster • Have to convert spark dataframe to pandas (may be expensive) • Use UDF to wrap python functionality **<u>Cumbersome to test (may be expensive)</u>**
Spark Logic	• May have to spin up cluster • May have to convert pandas dataframe to spark dataframe **<u>Cumbersome to test (may be expensive)</u>**	• Have to spin up cluster **<u>OK to test, can spin up local spark cluster</u>**

Native Execution Trumps Cluster Spin Up

	Pandas Dataframe	Spark RDD/ Dataframe
Python Logic	• No cluster spin up **Easiest to test**	• Have to spin up cluster • Have to convert spark dataframe to pandas (may be expensive) • Use UDF to wrap python functionality **Cumbersome to test (may be expensive)**
Spark Logic	• May have to spin up cluster • May have to convert pandas dataframe to spark dataframe **Cumbersome to test (may be expensive)**	• Have to spin up cluster **OK to test, can spin up local spark cluster**

UDFs, Pandas UDFs

Evolution of Python UDFs #1

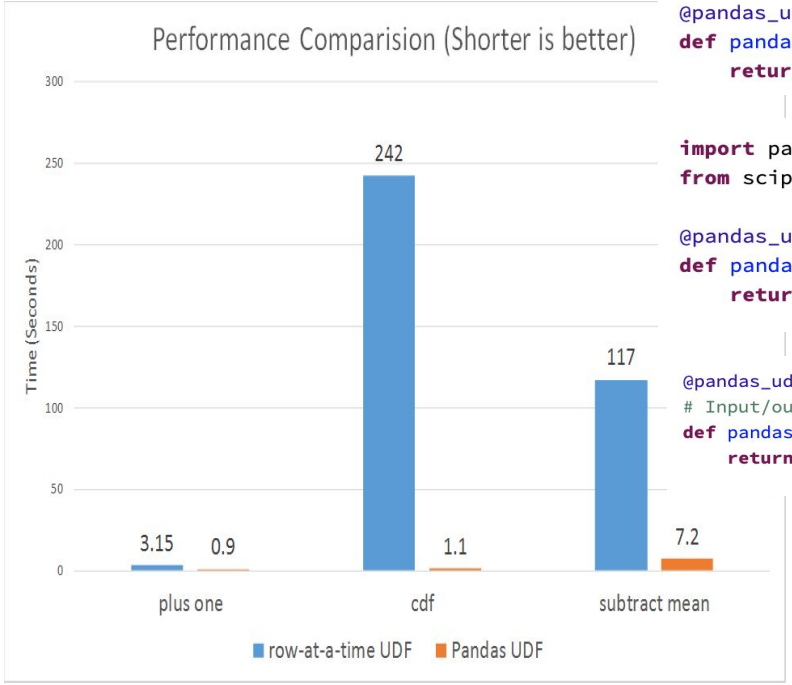
Testing your python logic would encounter less time

Python API UDFs (Spark 0.7)

to

Pandas UDFs (Spark 2.3)

Pandas_UDF outperforms the vanilla UDF




```python
@pandas_udf("double", PandasUDFType.SCALAR)
def pandas_plus_one(v):
    return v + 1
```

```python
import pandas as pd
from scipy import stats

@pandas_udf('double', PandasUDFType.SCALAR)
def pandas_cdf(v):
    return pd.Series(stats.norm.cdf(v))
```

```python
@pandas_udf(df.schema, PandasUDFType.GROUPED_MAP)
# Input/output are both a pandas.DataFrame
def pandas_subtract_mean(pdf):
    return pdf.assign(v=pdf.v - pdf.v.mean())
```

Evolution of Python UDFs #2

Your tests can take a python native type hint!

PandasUDFType (Spark 2.3) to Python Type Hints (Spark 3.0)

```
27
28  @pandas_udf('long', PandasUDFType.SCALAR)
29  def pandas_plus_one(v):
30      # `v` is a pandas Series
31      return v + 1  # outputs a pandas Series
```

```
# New type of Pandas UDF in Spark 3.0.
@pandas_udf('long', PandasUDFType.SCALAR_ITER)
def pandas_plus_one(itr):
    # `iterator` is an iterator of pandas Series.
    return map(lambda v: v + 1, itr)  # outputs an iterator of pandas Series.
```

```
@pandas_udf("id long", PandasUDFType.GROUPED_MAP)
def pandas_plus_one(pdf):
    # `pdf` is a pandas DataFrame
    return pdf + 1  # outputs a pandas DataFrame
```

```
def pandas_plus_one(v: pd.Series) -> pd.Series:
    return v + 1
```

```
def pandas_plus_one(itr: Iterator[pd.Series]) -> Iterator[pd.Series
    return map(lambda v: v + 1, itr)
```

```
def pandas_plus_one(pdf: pd.DataFrame) -> pd.DataFrame:
    return pdf + 1
```

Evolution of Python UDFs #3

Familiar syntax and natively executable functions work with Spark's API!

```
# mapInPandas
from typing import Iterator
import pandas as pd

df = spark.createDataFrame([(1, 21), (2, 30)], ("id", "age"))

def pandas_filter(iterator: Iterator[pd.DataFrame]) -> Iterator[pd.DataFrame]:
    for pdf in iterator:
        yield pdf[pdf.id == 1]

df.mapInPandas(pandas_filter, schema=df.schema).show()
```

Yet.. it may be cumbersome to test

```python
from typing import Iterator, Any, Union
from pyspark.sql.types import StructType
from pyspark.sql.types import StructField
from pyspark.sql.types import IntegerType
from pyspark.sql import DataFrame, SparkSession

def predict_wrapper(
    dfs: Iterator[pd.DataFrame],
    model_path:str
) -> Iterator[pd.DataFrame]:
    for df in dfs:
        yield predict(df, model_path)

def run_predict(
    df: DataFrame, model_path:str
) -> DataFrame:
    schema = StructType(list(df.schema.fields))
    schema.add(
        StructField("pred", IntegerType()))
    return df.mapInPandas(
        lambda dfs:predict_wrapper(dfs, model_path),
        schema=schema
    )

if isinstance(input_df, pd.DataFrame):
    sdf = spark_session.createDataFrame(input_df)
else:
    sdf = input_df

result = run_predict(spark_session, sdf, model_path)
```

```python
from fugue import transform

result = transform(
    input_df, predict, schema="*,pred:int",
    params=dict(model_path=some_path),
    engine=spark_session
)
```

Python execution engine

Spark execution engine

Databricks and databricks-connect

- Databricks founded by the original creators of Apache Spark
- Allows developers to spin up their own spark clusters
- Databricks-connect library that allows you to run your pyspark on the a databricks spark cluster remotely

Step 1: Install the client

1. Uninstall PySpark. This is required because the `databricks-connect` package conflicts with PySpark. For details, see Conflicting PySpark installations.

Bash Copy

```
pip uninstall pyspark
```

Introducing Fugue!

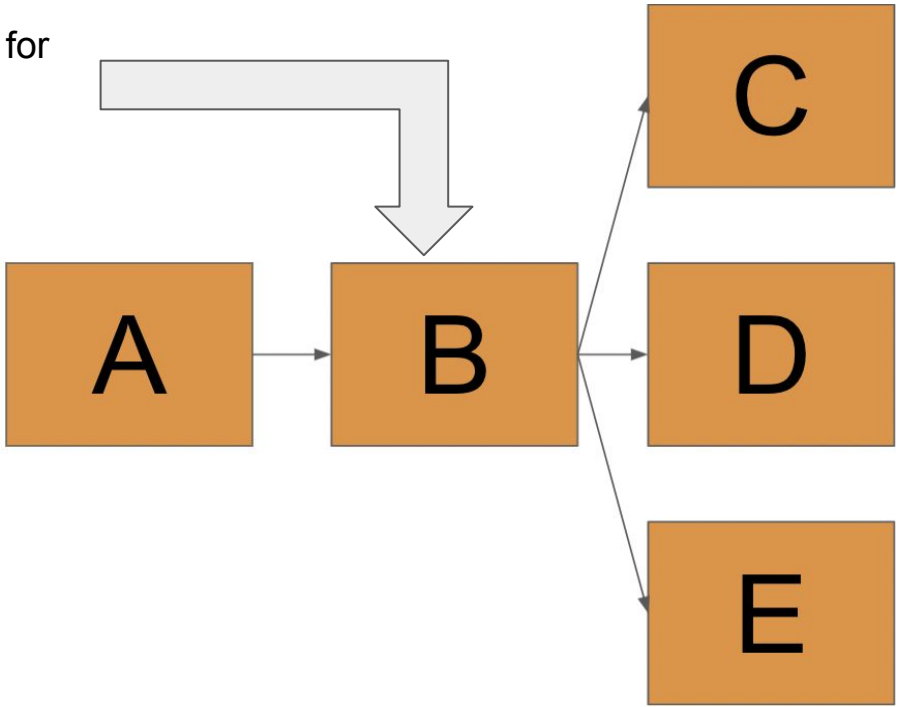
- An abstraction framework that allows users write code in native Python or Pandas, and then port it over to Spark and Dask
- Enables users to keep their native python logic for spark testable
- No need for cluster spin up unless warranted

The guarantee on consistency is beneficial for developing your spark applications with Fugue

	Pandas	**Spark**
Joining	NULL joins with NULL	NULL records are not joined (they are joined if NaN is used, which is different from NULL)
Sorting	NULLs are put at the end of the column for both ascending and descending	NULLs are treated as biggest value, meaning it is at the bottom when ascending and top when descending
Groupby	NULLs are dropped from the groupby	NULLs are kept in the groupby
Representation	Pandas represents NULLs with None, numpy.NaN and more recently pd.NA	Spark can use None and NaN, but not pd.NA. NaN can lead to some typing issues occasionally.

Fugue will also save you development and execution time

Computed 3 times for
C, D and E



Fugue's schema validation at compile time allows you to fail fast!

```
# schema: *, all_toxic:float
def make_new_col(df: pd.DataFrame) -> pd.DataFrame:
    #df['all_toxic'] = df[['toxic', 'severe_toxic']].max(axis=1)
    df['all_toxic'] = 'hello'
    return df
```

✓ 0.4s

Expressing your schema will speed up your development time.

```
ValueError: could not convert string to float: 'hello'

The above exception was the direct cause of the following exception:

FugueDataFrameInitError                   Traceback (most recent call last)
/var/folders/66/7g4zs6w171b_8tm20wrhxtbm0000gn/T/ipykernel_20662/4278460511.py in <module>
----> 1 train_df_new = transform(train_df, make_new_col, engine=NativeExecutionEngine)

~/Documents/GitProjects/OSS_testability/spark_fugue/lib/python3.9/site-packages/fugue/interfaceless.py in transform(df, using, schema, params, partition, ignore_errors, engine, engine_conf)
     64         ignore_errors=ignore_errors or [],
     65     ).yield_dataframe_as("result")
---> 66     result = dag.run(engine, conf=engine_conf)["result"]
     67     if isinstance(df, (DataFrame, Yielded)):
     68         return result
```

Contact Us

Github

- https://github.com/fugue-project/fugue

Fugue Tutorials

- https://fugue-project.github.io/tutorials/

Slack

- https://slack.fugue.ai/

Spotify Insights

We’re Hiring!
https://jobs.lever.co/spotify

Thank you!

Extra Slides

pandas_udfs are great for extending spark functionality but can be confusing to use

```
27
28  @pandas_udf('long', PandasUDFType.SCALAR)
29  def pandas_plus_one(v):
30      # `v` is a pandas Series
31      return v + 1  # outputs a pandas Series
32
33  # this works
34  spark.range(10).select(pandas_plus_one("id")).show()
35  # this works
36  spark.range(10).withColumn('id', pandas_plus_one('id')).show()
37  # this fails, type has to be GROUPED_MAP
38  spark.range(10).groupBy('id').apply(pandas_plus_one).show()
39
```

```
# New type of Pandas UDF in Spark 3.0.
@pandas_udf('long', PandasUDFType.SCALAR_ITER)
def pandas_plus_one(itr):
    # `iterator` is an iterator of pandas Series.
    return map(lambda v: v + 1, itr)  # outputs an iterator of pandas Series.

# this works
spark.range(10).select(pandas_plus_one("id")).show()
# this works
spark.range(10).withColumn('id', pandas_plus_one('id')).show()
# this fails, type has to be GROUPED_MAP
spark.range(10).groupBy('id').apply(pandas_plus_one).show()
```

```
@pandas_udf("id long", PandasUDFType.GROUPED_MAP)
def pandas_plus_one(pdf):
    # `pdf` is a pandas DataFrame
    return pdf + 1  # outputs a pandas DataFrame

# this fails
spark.range(10).select(pandas_plus_one("id")).show()
# this fails
spark.range(10).withColumn('id', pandas_plus_one('id')).show()
# this fails
spark.range(10).groupby('id').apply(pandas_plus_one('id')).show()
# this works
spark.range(10).groupby('id').apply(pandas_plus_one).show()
```

fugue

Benefits of testing!

	Cost for development	**Speed of development**	**Risk of production**
Less testing	Unpredictable	Unpredictable	High
More testing	Low	High	Low