SPARK+AI SUMMIT 2018

Avoiding Performance Potholes: Scaling Python for Data Science on Spark

Garren Staubli, Blueprint Technologies

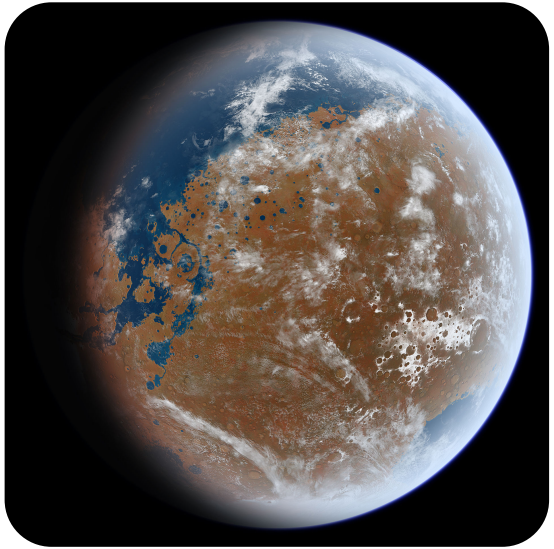
#Py9SAIS

GARRENS.COM/potholes

My Background

2011
Curiosity Launched

Started in Data



2015
Water on Mars

Started with Spark

2018
North meets South

Stream-Stream Joins + ML

Overview – fo real

PySparky,
not Malarkey

RDD no mo
DF fo sho

Package
Schmackage

Don't go
chasing UDFalls

PySparky

Spark application (Driver)

```
spark = SparkSession.builder\
        .appName(name="Stable Genius")\
        .master("yarn")\
        .getOrCreate()
```

SparkSession

Master (Cluster Manager)

Slave (Worker)

Executor (JVM)

Task

Task

Slave (Worker)

Executor (JVM)

Task

Task

python™

python™

Packages

1 New Library

2 Install

3 Attach to cluster(s)

New Library

Language: Upload Python Egg or PyPI

Install PyPi Package

You can specify a package name with an optional version specification

PyPi Name: PyPi Package (e.g. simplejson or simplejson==3.8.0)

Install Library

Upload Egg

Library Name: Library Name

Egg File: Drop library egg here to upload

Create Library

Packages

Others:

1. Each node: pip install

```
PYSPARK_PYTHON=… PYSPARK_DRIVER_PYTHON=… pyspark app.py
```

2. --py-files

```
pyspark --py-files numpy.zip app.py
```

Scala Packages in PySpark?

Scala Packages

Spark-Packages.org

spark-nlp (homepage)

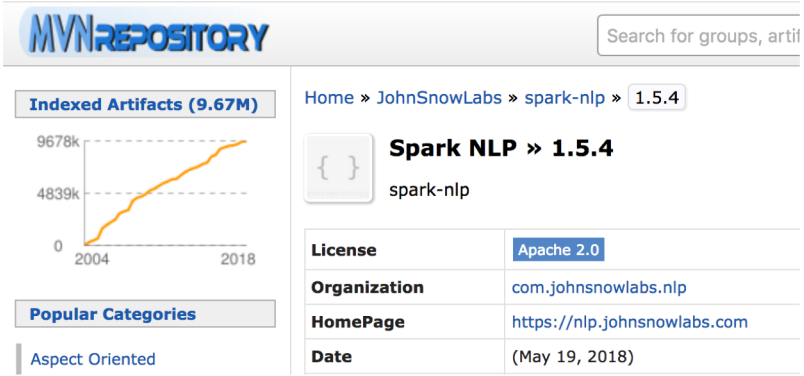
Natural Language Processing Library for Apache Spark.

@JohnSnowLabs / ★★★★★ (4)

```
--packages JohnSnowLabs:spark-nlp:1.5.4

spark = SparkSession.builder
.config("spark.jars.packages",
    "JohnSnowLabs:spark-nlp:1.5.4")
```

MVNrepository.com



Python & Spark Interplay

```
pandas_df = df.toPandas()
```

Innocuous, right?

1. Forces evaluation
2. Collects to driver
3. Serializes to memory

Python & Spark Interplay


```
df.toPandas()
df.collect()
df.take()
```

Uses **driver** memory

Get DataFrame memory usage

1. `df.cache().count()`

Check Spark UI

RDDs

ID	RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size on Disk
17	*(1) Range (0, 1000, step=1, splits=8)	Memory Deserialized 1x Replicated	8	100%	3.1 KB	0.0 B

`getRDDStorageInfo` not supported in PySpark

Python & Spark Interplay


```
df.toPandas()
df.collect()
df.take()
```

Uses **driver** memory

Get DataFrame memory usage

2
```
df.sample(fraction=0.01)\
  .cache().count()
```

Size in Memory
105.1 MB

x100 = 10.25GB

⚠ Avoid SizeEstimator – for RDDs not DF! ⚠

RDD no mo | DF fo sho

Before Spark 2.X	After Spark 2.X
SparkContext (sc)	SparkSession (“spark” variable)
Resilient **D**istributed **D**ataset	DataFrame (Python) Dataset (Scala/Java)
DStreams	Structured Streaming
MLlib (RDD-based)	Spark.ML (aka MLlib DF-based)

DataFrame

O(PI~~ES~~)

Optimized

Parallelized

Immutable

Not applicable to Python

~~**E**ncoded~~

~~**S**trongly Typed~~

^ Only for Scala & Java ^

DataFrame

O(PI)
Optimized

Tungsten

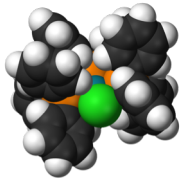
Memory Management

Binary Processing

Cache-Aware Computation

Code Generation

Catalyst



Analysis (Abstract Syntax Tree)

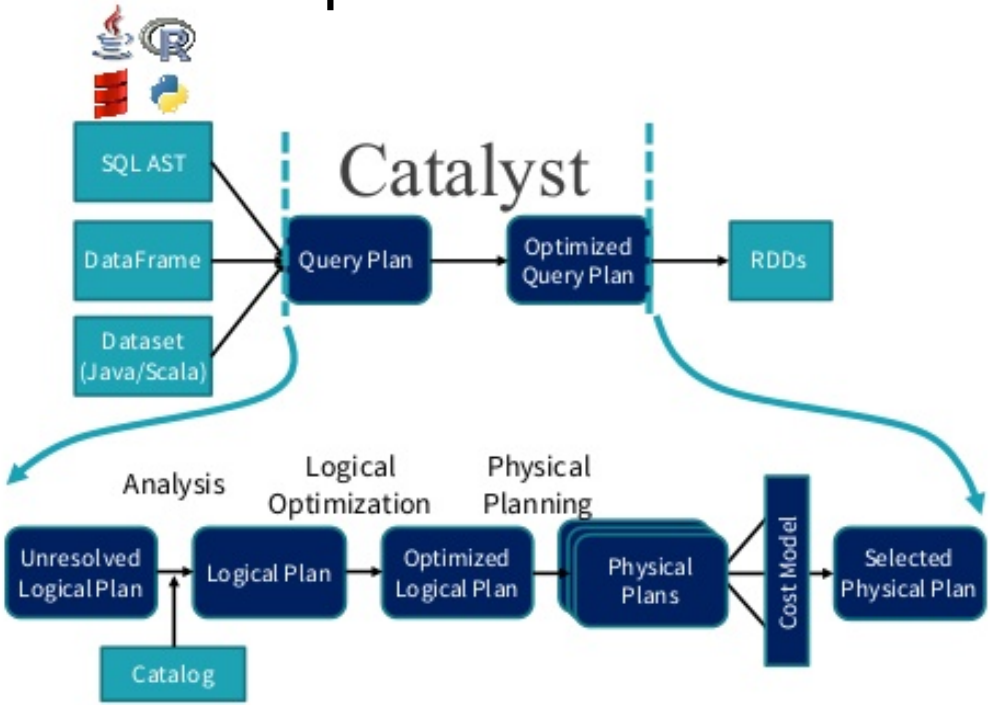
Logical Planning

Physical Planning

Code Generation

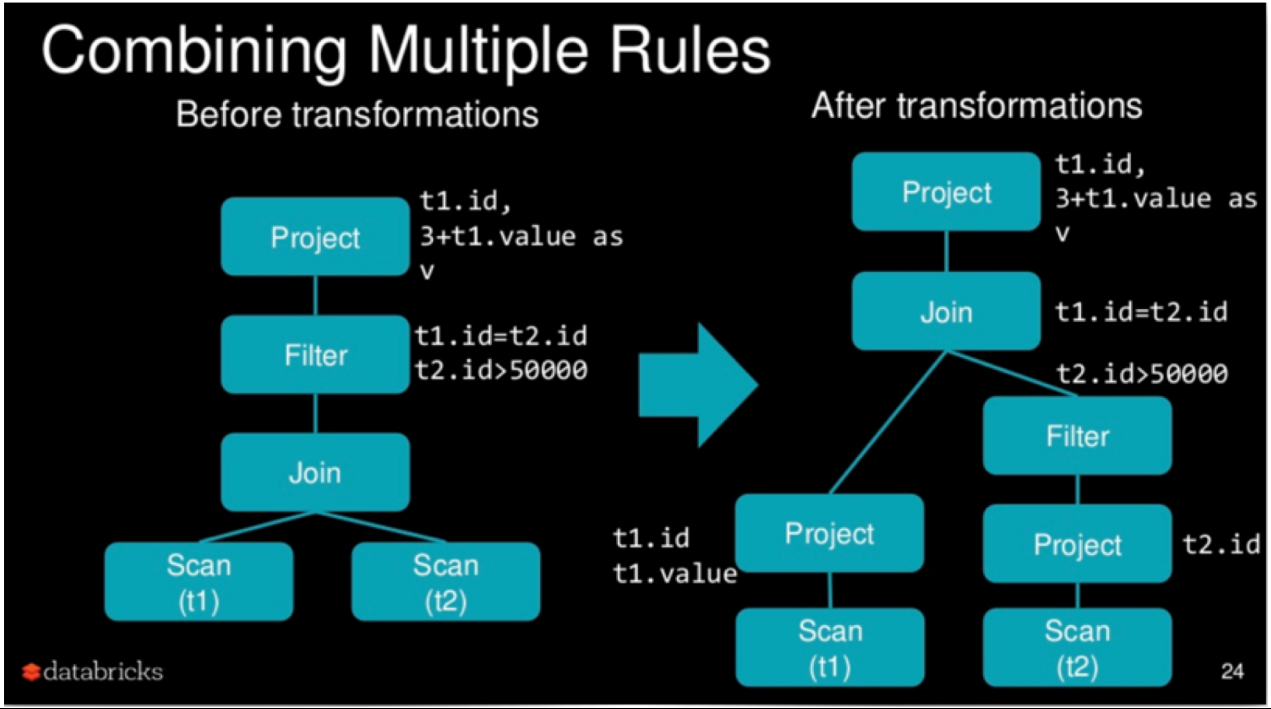
DataFrame

O(PI)
Optimized



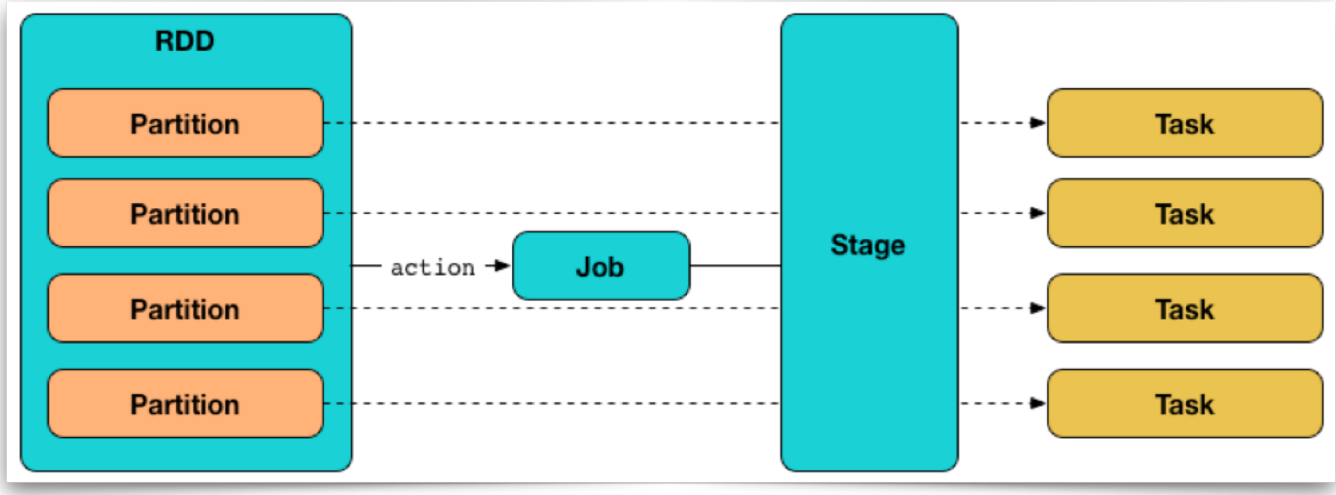
DataFrame

O(PI)
Optimized



DataFrame

O(PI)
Parallelized



DataFrame

O(PI)
Parallelized

Partition

- Logical split of data
- Maps to HDFS/Cassandra
- Up to executor memory limit

Task

- Individual unit of execution

DataFrame

O(PI)

Immutable

```
1. d = {'foo': 'bar'}
2. print(d['foo']) # => 'bar'
3. d['foo'] = 'qux'
4. print(d['foo']) # => 'qux'
```

```
1. df.foreach(print) # Row(foo='qux')
2. df['baz'] = 'qux'
3. # TypeError: 'DataFrame' object does not support item assignment
```

```
5. df.withColumn("baz", lit("qux"))
6. df.foreach(print) # Row(foo='qux')
```

```
8. new_df = df.withColumn("baz", lit("qux"))
9. new_df.foreach(print) # Row(foo='qux', baz='qux')
```

UDFalls

```
ts_to_epoch = F.udf(
   lambda t: int(t.strftime("%s")))
```

```
df.select(ts_to_epoch(df.tpep_pickup_datetime))\
   .distinct().count()
```

Pure Python

Executed in JVM

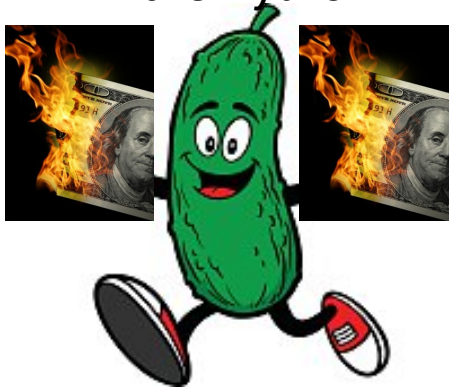
```
df.select(F.unix_timestamp(df.tpep_pickup_datetime))\
   .distinct().count()
```

UDFalls

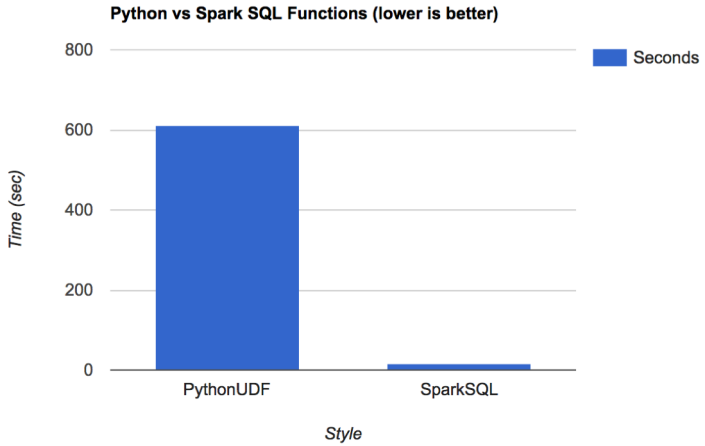
1 Project [**pythonUDF0#33 AS <lambda>**(id)#26]

Executed in JVM → 30-40x faster

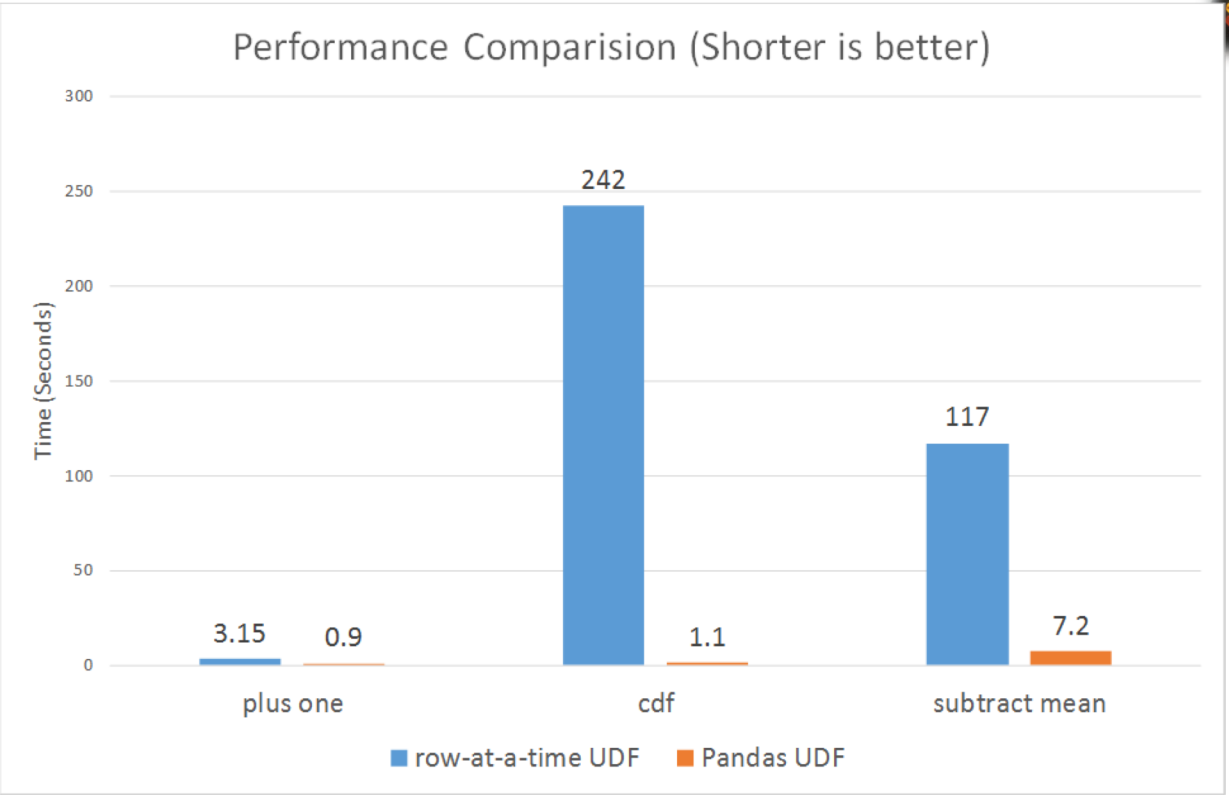
Pure Python



2 Project [unix_timestamp(id#0L)]



Vectorized UDFs



Vectorized UDFs

```
from scipy import stats
from pyspark.sql.functions import udf
```

→ `pandas_udf`

```
@udf('double')
def cdf(v):
    return float(stats.norm.cdf(v))
df.withColumn('cumulative_probability', cdf(df.v))
```

→ `@pandas_udf('double', PandasUDFType.SCALAR)`

→ `return pd.Series(stats.norm.cdf(v))`

Vectorized UDFs

```
import pandas as pd
from scipy import stats

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

df.withColumn('cumulative_probability’, pcdf(df.v))
```


EXCELLENT
MY WORK HERE IS DONE.
quickmeme.com

Blueprint™

bpcs.com

NOW HIRING

Data Engineers & Scientists