Koalas

Pandas API on Apache Spark

Ben Sadeghi

Databricks Solutions Architect

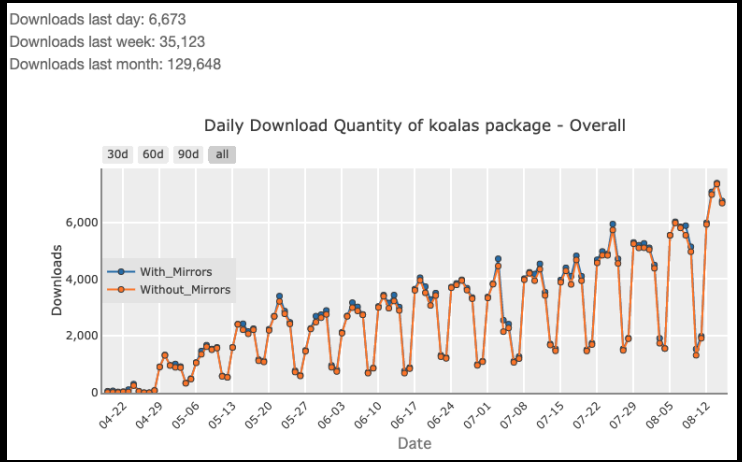
databricks

Use Cases and Goals

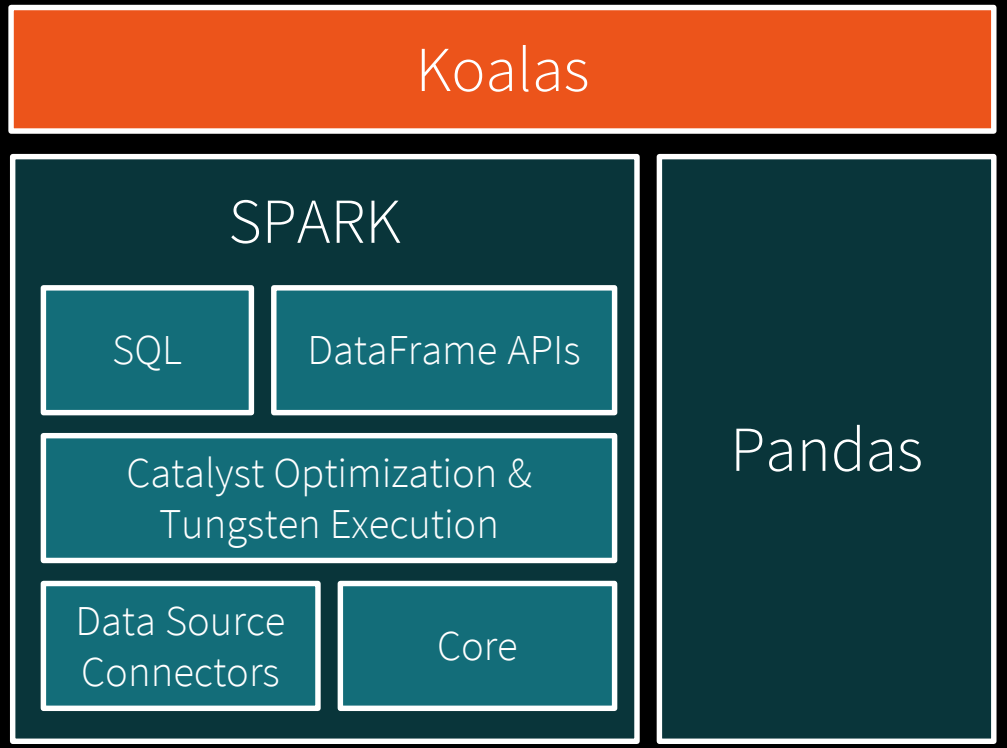
- **Better scale** the breadth of Pandas to big data
  - Pandas is single-threaded & memory bound
  - Spark is fully distributed (CPU & RAM)

- **Reduce friction** by unifying big data environment

- Has been quickly adopted
  - 300+ patches merged since April 2019
  - 6k+ daily downloads



Koalas Architecture



A lean API layer

A Few Callouts to the Koalas Design Principles

Be ‘Pythonic’

- snake_case rather than camelCase
- NumPy
- Docs and style follow PyData projects

Pandas First Mentality

- Function found in both follow the same naming conventions
- Functions in Spark with a Pandas equivalent will be implemented with the Pandas alias (‘Pandas First’)
- Functions found in Pandas that are appropriate for distributed datasets will become available in Koalas
- Functions only found in Spark that control distribution will become available in Koalas
  - ex: cache()

Guardrails

- Methods in Koalas are safe to **perform at scale**
- To maintain safe methods at scale, the following will not be implemented in Koalas
  - Capabilities that are fundamentally not parallelizable
  - Capabilities that require materializing the entire working set in a single node memory
- Exceptions
  - DataFrame.to_pandas()
  - DataFrame.to_numpy()

API Differences

Pandas

- Born of need + batteries included: providing APIs for common tasks
- Type system from NumPy
- Be Pythonic

PySpark

- Abstraction: tasks are implemented by primitives composition
- Type system from ANSI SQL
- Consistent with Scala DataFrame APIs

Pandas DataFrame vs Spark DataFrame

	Pandas DataFrame	Spark DataFrame
Column	df[‘col’]	df[‘col’]
Mutability	Mutable	Immutable
Add a column	df[‘c’] = df[‘a’] + df[‘b’]	df.withColumn(‘c’, df[‘a’] + df[‘b’])
Rename columns	df.columns = [‘a’,’b’]	df.select(df[‘c1’].alias(‘a’), df[‘c2’].alias(‘b’))
Value count	df[‘col’].value_counts()	df.groupBy(df[‘col’]).count() .orderBy(‘count’, ascending = False)

A Short Example

Pandas

```
import pandas as pd
df = pd.read_csv("my_data.csv")

df.columns = [‘x’, ‘y’, ‘z1’]

df[‘x2’] = df.x * df.x
```

PySpark

```
df = (spark.read
    .option("inferSchema", "true")
    .option("comment", True)
    .csv("my_data.csv"))

df = df.toDF(‘x’, ‘y’, ‘z1’)

df = df.withColumn(‘x2’, df.x*df.x)
```

A Short Example

Pandas

```
import pandas as pd
df = pd.read_csv("my_data.csv")

df.columns = [‘x’, ‘y’, ‘z1’]

df[‘x2’] = df.x * df.x
```

Koalas

```
import databricks.koalas as ks
df = ks.read_csv("my_data.csv")

df.columns = [‘x’, ‘y’, ‘z1’]

df[‘x2’] = df.x * df.x
```

Demo

Appendix

Blog Post: Reducing Processing Time from Hours to Minutes with Koalas
databricks.com/blog/2019/08/22/guest-blog-how-virgin-hyperloop-one-reduced-processing-time-from-hours-to-minutes-with-koalas.html

Want to contribute? (Apache 2.0 License)
github.com/databricks/koalas

Thank you!

/BenSadeghi

LinkedIn, GitHub, Twitter