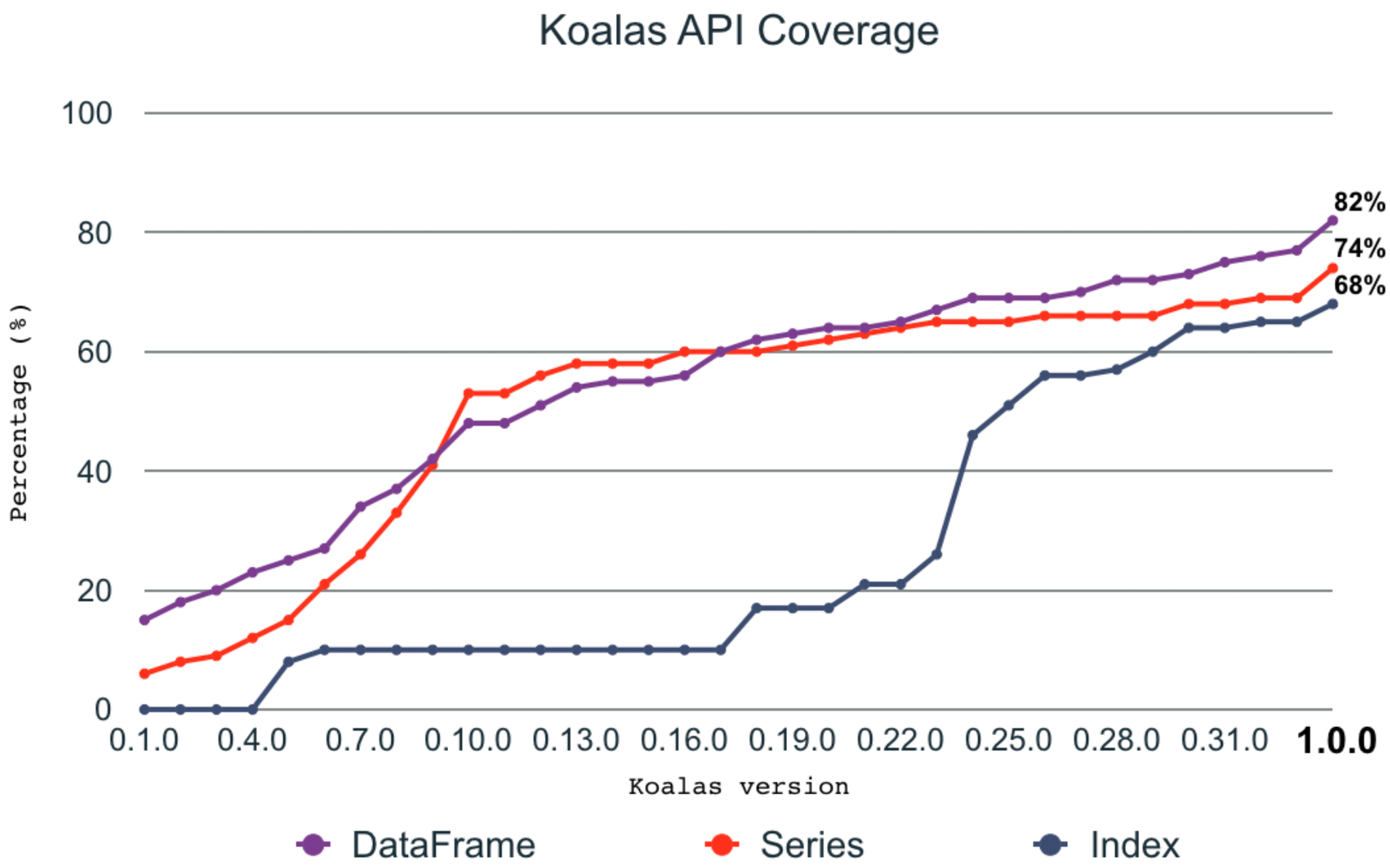
CS 649 Big Data: Tools and Methods
Spring Semester, 2022
Doc 11 Spark Panda API
Feb 15, 2022

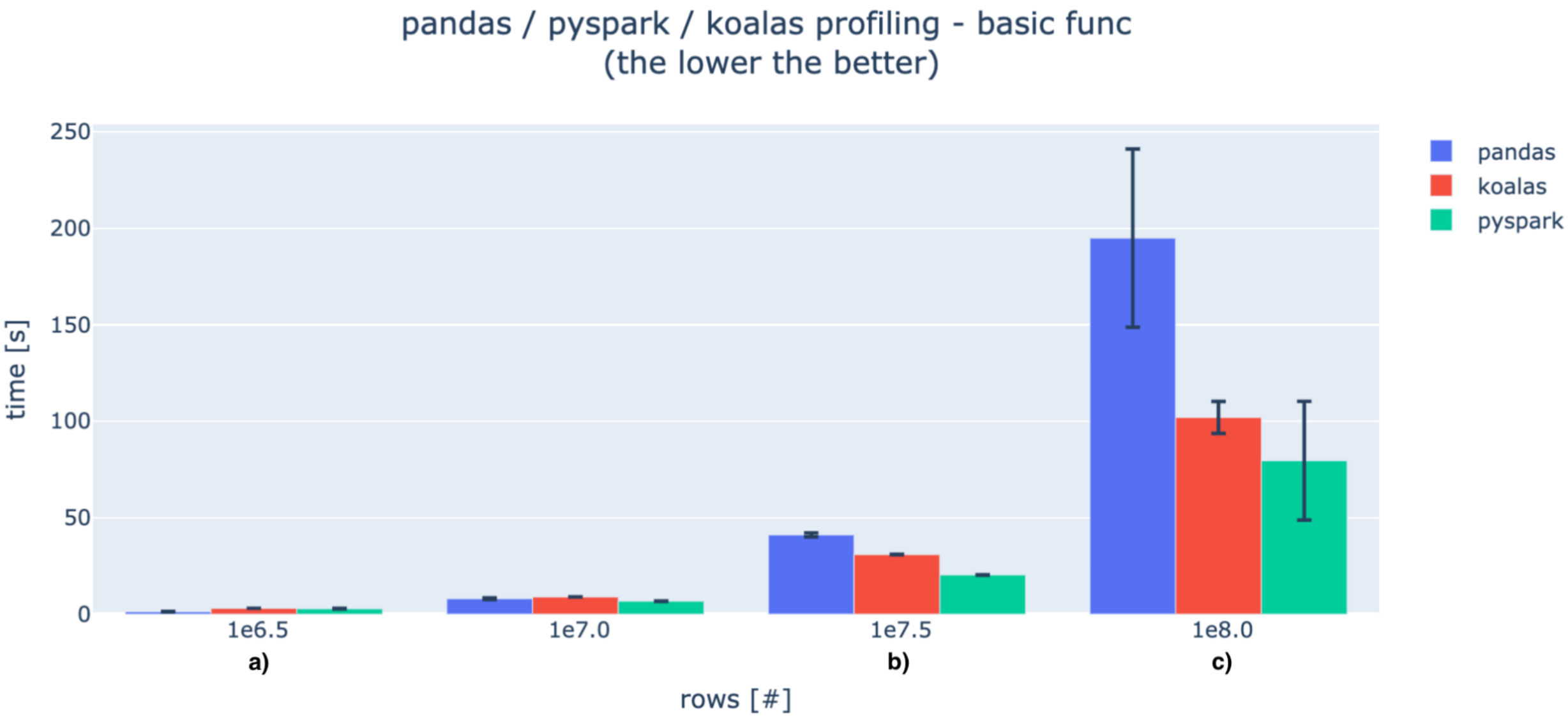
Panadas API on Spark

Project started as additional package: Koalas

Koala 1.0 released 2020



Performance



https://databricks.com/blog/2019/08/22/guest-blog-how-virgin-hyperloop-one-reduced-processing-time-from-hours-to-minutes-with-koalas.html Accessed 2/14/22

Pandas API Can be Shorter than PySpark

PySpark

```
from pyspark.sql.functions import pandas_udf, PandasUDFType
from pyspark.sql.types import *
import pyspark.sql.functions as F
schema = StructType([
 StructField("pod_id", StringType()),
 StructField("trip_id", StringType()),
 StructField("distance_miles", DoubleType()),
 StructField("travel_time_sec", DoubleType())
])
@pandas_udf(schema, PandasUDFType.GROUPED_MAP)
def calculate_distance_from_speed( gdf ):
 gdf = gdf.sort_values('timestamp')
 print(gdf)
 gdf['time_diff'] = gdf['timestamp'].diff()
 return pd.DataFrame({
  'pod_id':[gdf['pod_id'].iloc[0]],
  'trip_id':[gdf['trip_id'].iloc[0]],
  'distance_miles':[ (gdf['time_diff']*gdf['speed_mph']).sum()],
  'travel_time_sec': [ gdf['timestamp'].iloc[-1]-gdf['timestamp'].iloc[0] ]
})
sdf = spark_df.groupby("pod_id","trip_id").apply(calculate_distance_from_speed)
sdf = sdf.withColumn('distance_km',F.col('distance_miles') * 1.609)
sdf = sdf.withColumn('avg_speed_mph',F.col('distance_miles')/ F.col('travel_time_sec') / 60.0)
sdf = sdf.withColumn('avg_speed_kph',F.col('avg_speed_mph') * 1.609)
sdf = sdf.orderBy(sdf.pod_id,sdf.trip_id)
sdf.summary().toPandas()
```

Pandas API (Koalas)

```
import databricks.koalas as ks
def calc_distance_from_speed_ks( gdf ) -> ks.DataFrame[ str, str, float , float]:
 gdf = gdf.sort_values('timestamp')
 gdf['meanspeed'] = (gdf['timestamp'].diff()*gdf['speed_mph']).sum()
 gdf['triptime'] = (gdf['timestamp'].iloc[-1] - gdf['timestamp'].iloc[0])
 return gdf[['pod_id','trip_id','meanspeed','triptime']].iloc[0:1]

kdf = ks.from_pandas(df)
results = kdf.groupby(['pod_id','trip_id']).apply( calculate_distance_from_speed
# due to current limitations of the package, groupby.apply() returns c0 .. c3 col
results.columns = ['pod_id', 'trip_id', 'distance_miles', 'travel_time_sec']
# spark groupby does not set the groupby cols as index and does not sort ther
results = results.set_index(['pod_id','trip_id']).sort_index()
results['distance_km'] = results['distance_miles'] * 1.609
results['avg_speed_mph'] = results['distance_miles'] / results['travel_time_sec']
results['avg_speed_kph'] = results['avg_speed_mph'] * 1.609
results.describe()
```

Dataframe

```
import pyspark.pandas as ps
data = {'a': [1, 2, 3, 4, 5, 6],
        'b': [100, 200, 300, 400, 500, 600],
        'c': ["one", "two", "three", "four", "five", "six"]}

sample_df = ps.DataFrame(data, index=[10, 20, 30, 40, 50, 60])
sample_df
```

	a	b	c
10	1	100	one
20	2	200	two
30	3	300	three
40	4	400	four
50	5	500	five
60	6	600	six

Converting from Pandas

```
import pandas as pd
data = {'a': [1, 2, 3, 4, 5, 6],
        'b': [100, 200, 300, 400, 500, 600],
        'c': ["one", "two", "three", "four", "five", "six"]}

df = pd.DataFrame(data, index=[10, 20, 30, 40, 50, 60])
pdf = ps.from_pandas(df)
pdf
```

Converting

```
import pandas as pd
data = {'a': [1, 2, 3, 4, 5, 6],
        'b': [100, 200, 300, 400, 500, 600],
        'c': ["one", "two", "three", "four", "five", "six"]}

df = pd.DataFrame(data, index=[10, 20, 30, 40, 50, 60])

pdf = ps.from_pandas(df)

df_copy = pdf.to_pandas()
sdf = pdf.to_spark()
```

```
type(pdf)        pyspark.pandas.frame.DataFrame

type(sdf)        pyspark.sql.dataframe.DataFrame

type(df)         pandas.core.frame.DataFrame
```

Some Method have Different Semantics

```
ps.DataFrame({'a': [1, 2, 3], 'b': [4, 5, 6]}).count() # Pandas API
```

```
a    3
b    3
dtype: int64
```

number of non-NA/null entries for each column/row

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()
spark.createDataFrame([[1, 4], [2, 5], [3, 6]], schema=["a", "b"]).count()
```

```
3
```

number of retrieved rows, including rows containing null

```
# Create a pandas DataFrame
pdf = pd.DataFrame({'A': np.random.rand(5),
                    'B': np.random.rand(5)})
# Create a pandas-on-Spark DataFrame
psdf = ps.DataFrame({'A': np.random.rand(5),
                     'B': np.random.rand(5)})
# Create a pandas-on-Spark DataFrame by passing a pandas DataFrame
psdf = ps.DataFrame(pdf)
psdf = ps.from_pandas(pdf)
```

	A	B
0	0.124840	0.150538
1	0.295045	0.089399
2	0.277994	0.628313
3	0.534681	0.588598
4	0.242770	0.653700

psdf.head(2)

	A	B
0	0.461542	0.444230
1	0.573106	0.786139

psdf.describe()

	A	B
count	5.000000	5.000000
mean	0.295066	0.422110
std	0.149515	0.277633
min	0.124840	0.089399
25%	0.242770	0.150538
50%	0.277994	0.588598
75%	0.295045	0.628313
max	0.534681	0.653700

psdf.sort_values(by='B')

	A	B
1	0.295045	0.089399
0	0.124840	0.150538
3	0.534681	0.588598
2	0.277994	0.628313
4	0.242770	0.653700

psdf.transpose()

	2	3	0	1	4
A	0.277994	0.534681	0.124840	0.295045	0.24277
B	0.628313	0.588598	0.150538	0.089399	0.65370

psdf['A']

```
0    0.461542
1    0.573106
2    0.219727
3    0.587838
4    0.934723
Name: A, dtype: float64
```

II

psdf.loc[1:2]

	A	B
1	0.573106	0.786139
2	0.219727	0.231118

psdf.iloc[:3, 1:2]

	B
0	0.444230
1	0.786139
2	0.231118

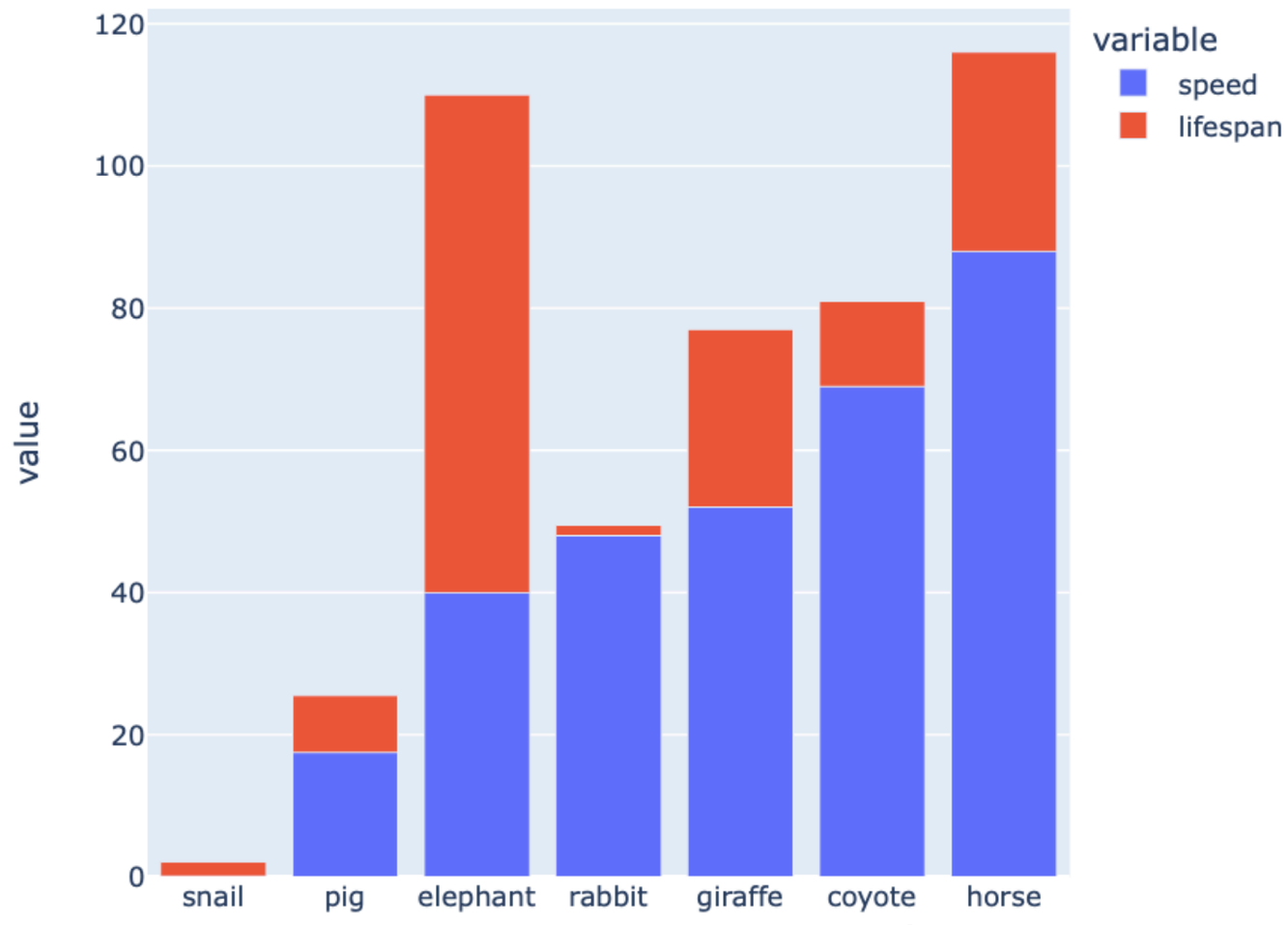
```
psdf = ps.DataFrame({'A': [2, 1, 2, 4, 2],
                     'B': np.random.rand(5),
                    'C': [100, 200, 300, 400, 500]}
                     )
psdf
```

	A	B	C
0	2	0.359469	100
1	1	0.180497	200
2	2	0.726065	300
3	4	0.000165	400
4	2	0.978933	500

```
psdf.groupby('A').sum()
```

	B	C
A		
2	2.064466	900
1	0.180497	200
4	0.000165	400

```
speed = [0.1, 17.5, 40, 48, 52, 69, 88]
lifespan = [2, 8, 70, 1.5, 25, 12, 28]
index = ['snail', 'pig', 'elephant',
         'rabbit', 'giraffe', 'coyote', 'horse']
psdf = ps.DataFrame({'speed': speed,
                     'lifespan': lifespan}, index=index)
psdf.plot.bar()
```



```
psdf = ps.DataFrame({'A': np.random.rand(5000),
                     'B': np.random.rand(5000),
                     'C': np.random.rand(5000)}
                     )

psdf.transpose()
```

```
ValueError: Current DataFrame has more then the given limit 1000 rows. Please set
'compute.max_rows' by using 'pyspark.pandas.config.set_option' to retrieve to retrieve more
than 1000 rows. Note that, before changing the 'compute.max_rows', this operation is
considerably expensive.
```

compute.max_rows

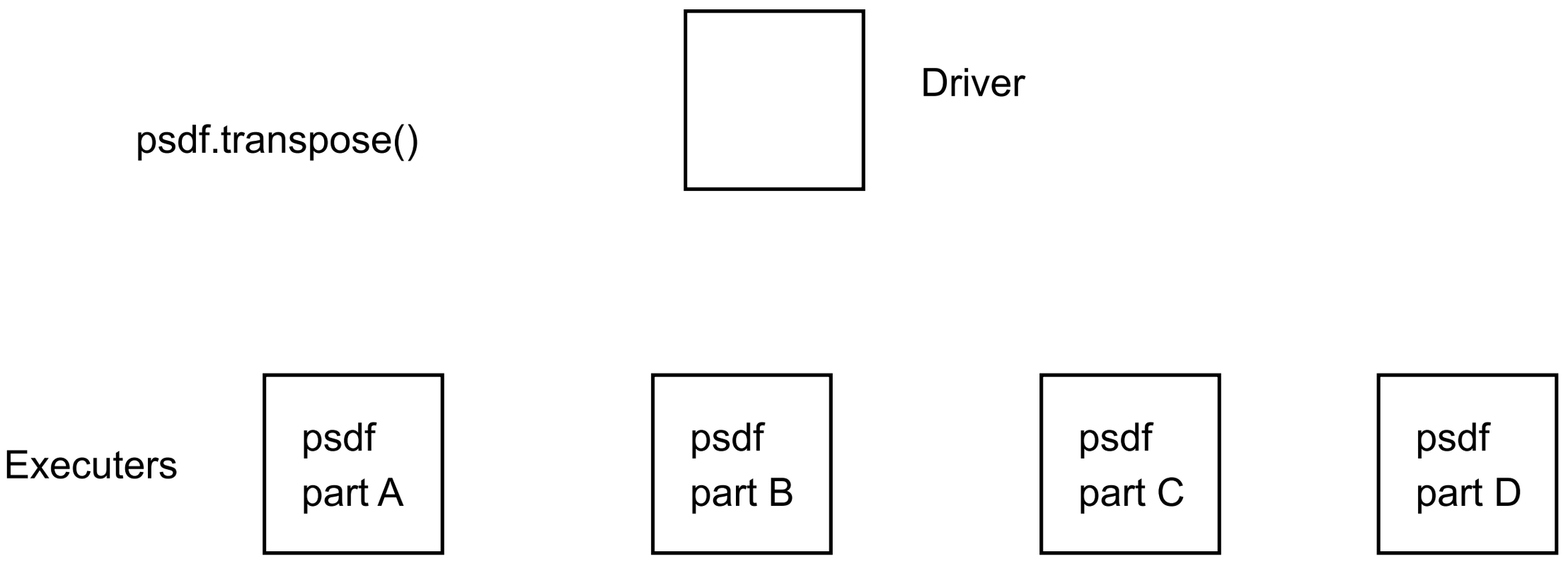
‘compute.max_rows’ sets the limit of the current Pandas API DataFrame.

Set None to unlimit the input length.

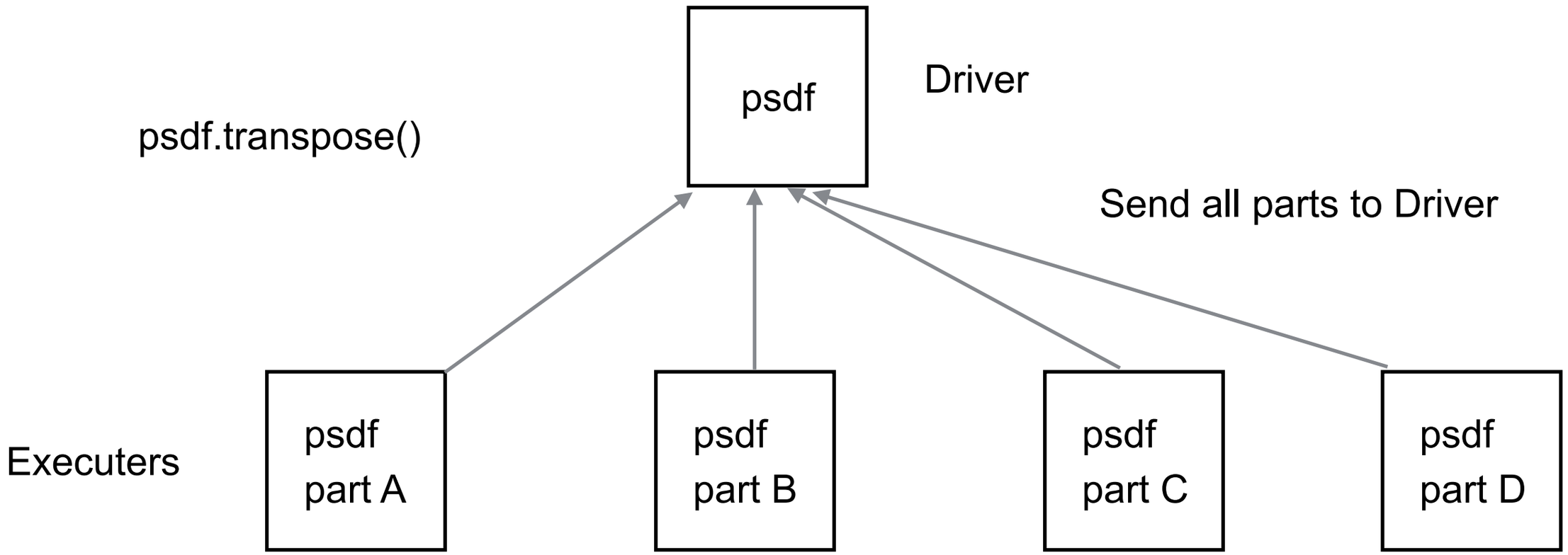
When the limit is set, it is executed by the shortcut by collecting the data into the driver, and then using the pandas API.

If the limit is unset, the operation is executed by PySpark. Default is 1000.

compute.max_rows - What this Means

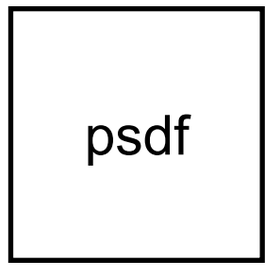


compute.max_rows - What this Means



compute.max_rows - What this Means

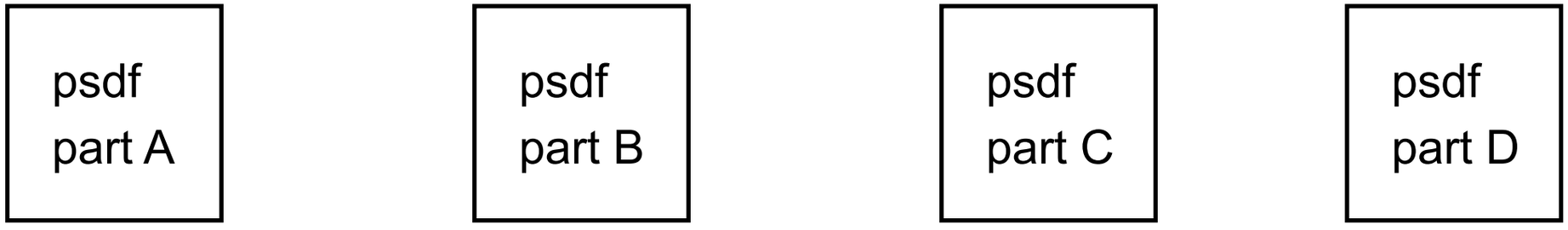
psdf.transpose()



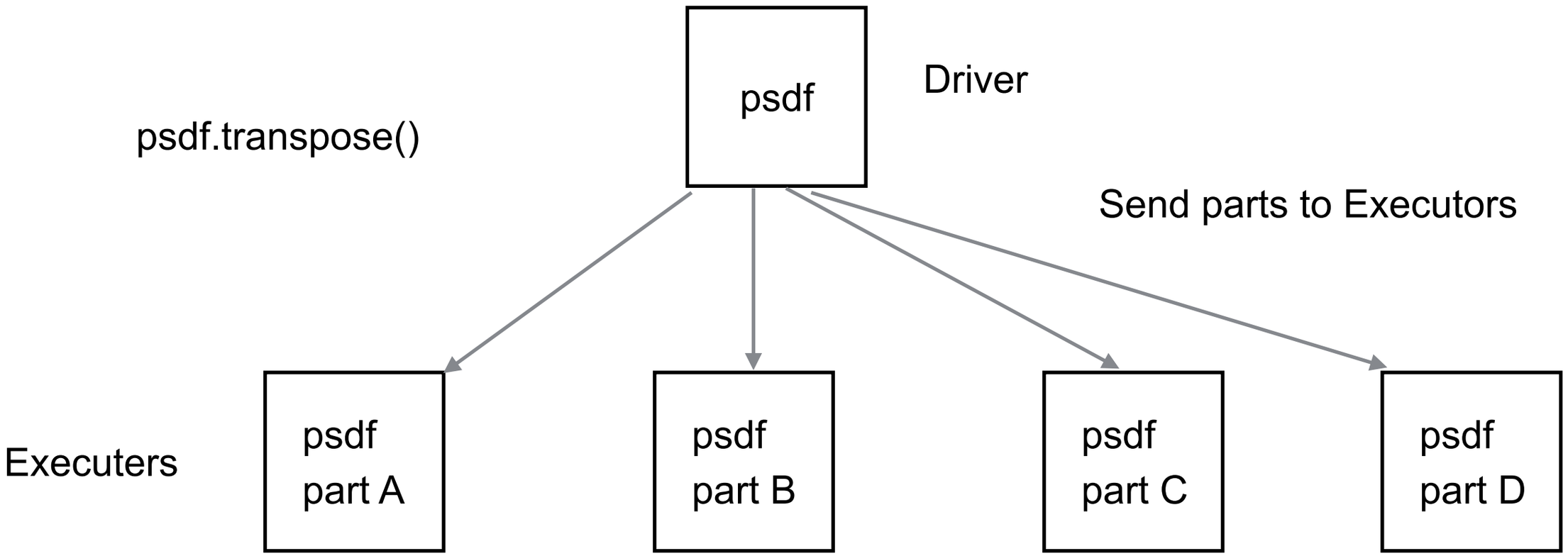
Driver

- Convert to Pandas Dataframe
- Use Pandas to do transpose
- Convert transpose to Pandas API format

Executers



compute.max_rows - What this Means



Other Things to Avoid

__iter__

list comprehension
generator expression

__iter__

Requires all data to be on one machine

Pandas min, max, sum use __iter__

```
import pandas as pd
import numpy as np
df = pd.Series(np.random.rand(5000))
%timeit max(df)
```

299 **µs** ± 473 ns per loop

```
import pyspark.pandas as ps
import numpy as np
psdf = ps.Series(np.random.rand(5000))

max(psdf)
```

Error

```
%timeit psdf.max()
```

370 **ms** ± 131 ms per loop

The slowest run took 4.01 times longer than the fastest. This could mean that an intermediate result is being cached.

Pandas API vs PySpark

If you are starting a new project

Use PySpark

If you are new to Spark

Best to use PySpark

If have existing code in Python

Use Pandas API on Spark