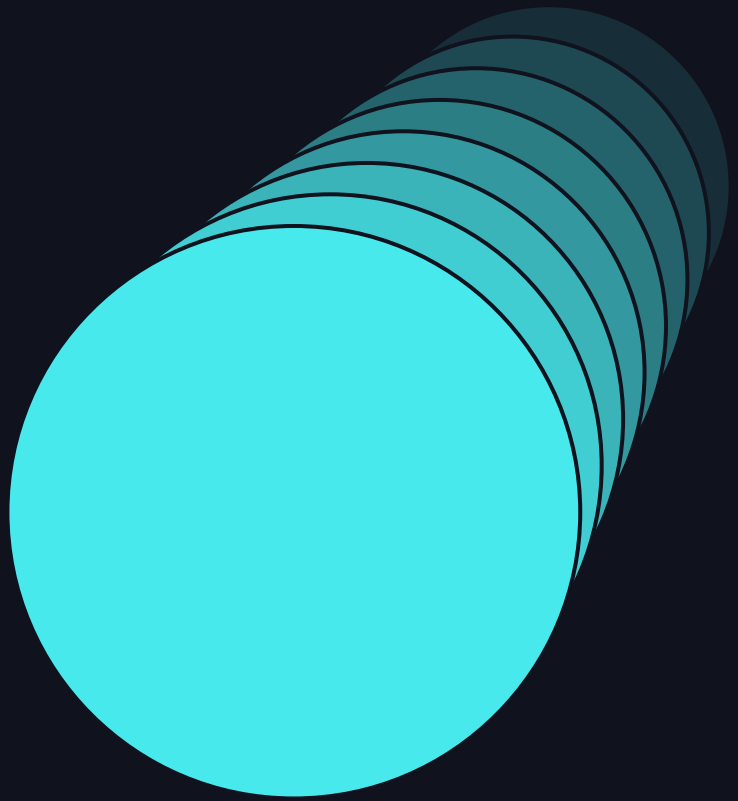
DATA+AI SUMMIT BY databricks

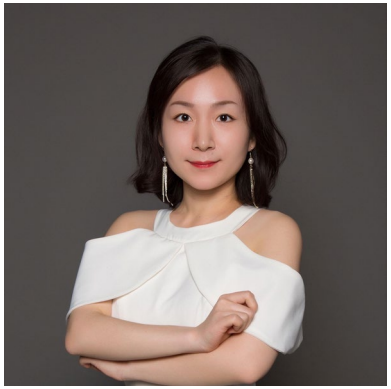
pandas on Spark: Simplicity of pandas with efficiency of Spark



Xinrong Meng
Matthew Powers CFA

Speakers

Xinrong Meng



- Senior Software Engineer @ Databricks
- Apache Spark PMC Member and Committer
- One of the major contributors of Pandas on Spark

Matthew Powers, CFA

- Staff Developer Advocate @ Databricks
- Spark / Delta Lake blogger, OSS contributor
- Quite active on LinkedIn recently

Agenda

What is Pandas on Spark 

How Does Pandas on Spark Work 

Beyond Pandas 

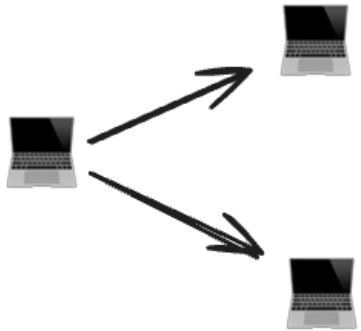
What’s new 

Simple example

Cluster

PS Benchmark

Runtime	DBR 15.2 • Spark 3.5.0 • Scala 2.12
Driver	d3.2xlarge • 64 GB • 8 Cores
Workers (2)	d3.2xlarge • 128 GB • 16 Cores



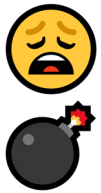
Read data with pandas

pandas fails to read a 48 GB Parquet dataset

Last execution failed 2 Python

```
1  import pandas as pd
2
3  pdf = pd.read_parquet("/dbfs/mnt/performance-datasets/2018TPC/tpcds-2.4/sf10000_parquet/web_returns")  # 48G
```

```
Fatal error: The Python kernel is unresponsive.
--------------------------------------------------------------------------------
The Python process exited with exit code 137 (SIGKILL: Killed). This may have been caused by an OOM error. Check your command's memory usage.
```



Read data with pandas on Spark

pandas on Spark successfully reads a 48 GB Parquet dataset

```
02:29 PM (21s)                                                    3

import pyspark.pandas as ps

psdf = ps.read_parquet("dbfs:/mnt/performance-datasets/2018TPC/tpcds-2.4/sf10000_parquet/web_returns")  # 48G

(2) Spark Jobs
```

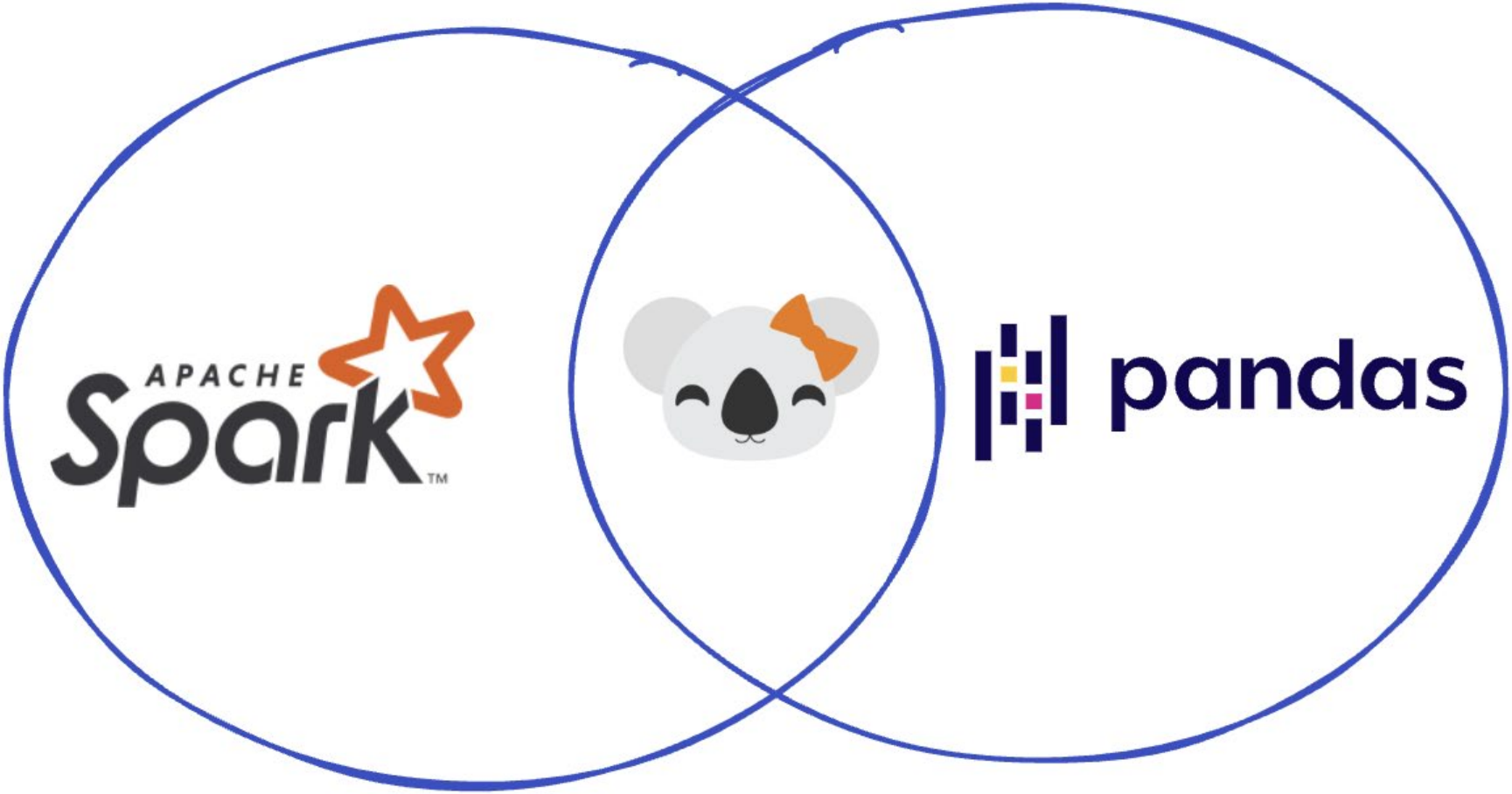
Operations

```
02:32 PM (<1s)                                                    5

psdf.dtypes

wr_returned_time_sk          int32
wr_item_sk                   int32
wr_refunded_customer_sk      int32
```

Intro to Pandas on Spark



Apache Spark

De facto engine for large-scale unified analytics

Integration across ecosystems

Works on a single node or a cluster

PySpark for Python users

pandas

- Popular Python DataFrame library
- Integrated into Python data science ecosystem
- Works on a single node only (single core)

pandas on Spark

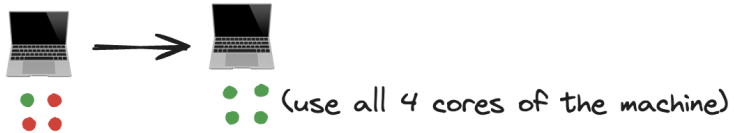
Implements pandas API on top of Spark

2019, announced as Koalas

2021, integrated into PySpark (pyspark.pandas)

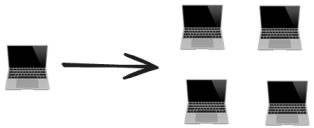
How you can scale

1: Use all the cores of a machine



2: Use a bigger computer

3: Scale out to many computers



4: Use a subset of the data

What is pandas on Spark?

- Allows you to write Spark code with pandas syntax
- Performs similarly to PySpark, Scala Spark, or SQL
- Great for users that like pandas syntax
- Can be used instead of pandas OR in conjunction with pandas (they place together nicely)
- Usually faster than pandas for large datasets

You can query large datasets with pandas on Spark

```
1  len(ps.read_table("shared.default.h2o_groupby"))
```

▸ (4) Spark Jobs

```
7109566279
```

```
1  df = ps.read_table("shared.default.h2o_groupby")[["id1", "id2", "v3"]]
2  df.query("id1 == 'id094'").groupby("id2").sum().head(3)
```

	v3
id2	
id089	3.620359e+07
id080	3.656286e+07
id087	3.619828e+07

Command took 11.39 seconds -- by matthew.powers@databricks.com at 5/21/2024, 8:23:40 PM on Matthew Powers's Cluster

pandas vs PySpark syntax

PySpark syntax differs from pandas

pandas

```
import pandas as pd


pdf = pd.read_parquet("x.parquet")

pdf_filtered = pdf.loc[pdf["quantity"] > 10, ["data",
"quantity"]]

pdf_filtered
```

PySpark

```
df = spark.read.format("parquet").load("x.parquet")

df_filtered = df.select("date",
"quantity").filter("quantity > 10")

df_filtered.show()
```

pandas vs pandas on Spark syntax

Same syntax with pandas on Spark!

pandas

```
import pandas as pd

pdf = pd.read_parquet("x.parquet")

pdf_filtered = pdf.loc[pdf["quantity"] > 10, ["data",
"quantity"]]

pdf_filtered
```

Pandas on Spark

```
import pyspark.pandas as ps

pdf = ps.read_parquet("x.parquet")

pdf_filtered = pdf.loc[pdf["quantity"] > 10, ["data",
"quantity"]]

pdf_filtered
```

Faster for single node queries (sometimes)

1: Parquet file with 1 billion rows like this

id1	id2	id3	id4	id5	id6	v1	v2	v3
id016	id046	id0000109363	88	13	146094	4	6	18.8377
id039	id087	id0000466766	14	30	111330	4	14	46.7973
id047	id098	id0000307804	85	23	187639	3	5	47.5773

2: Run a query with pandas

```
df = pd.read_parquet(
    "G1_1e9_1e2_0_0.parquet",
    columns=["id1", "id2", "v3"],
    filters=[("id1", ">", "id098")],
    engine="pyarrow",
)
df.query("id1 > 'id098'").groupby("id2").sum().head(3)
```

275 seconds

3: Run the same query with pandas on Spark

```
import pyspark.pandas as ps

df = ps.read_parquet("G1_1e9_1e2_0_0.parquet")[
    ["id1", "id2", "v3"]
]
df.query("id1 > 'id098'").groupby("id2").sum().head(3)
```

→ 62 seconds

Details of single node benchmark

- 1 billion row dataset in single Parquet file
  - 31 GB on disk
- 2020 Macbook M1 with 64GB of RAM
- Spark 3.5

Manual pandas optimizations can be dangerous

1: Parquet file with 1 billion rows like this

id1	id2	id3	id4	id5	id6	v1	v2	v3
id016	id046	id0000109363	88	13	146094	4	6	18.8377
id039	id087	id0000466766	14	30	111330	4	14	46.7973
id047	id098	id0000307804	85	23	187639	3	5	47.5773

2: pandas on Spark syntax is nice

```
import pyspark.pandas as ps

df = ps.read_parquet("G1_1e9_1e2_0_0.parquet")[
    ["id1", "id2", "v3"]
]
df.query("id1 > 'id098'").groupby("id2").sum().head(3)
```

3: pandas syntax is verbose and error-prone

```
df = pd.read_parquet(
    "G1_1e9_1e2_0_0.parquet",
    columns=["id1", "id2", "v3"],
    filters=[("id1", "==", "id001")],
    engine="pyarrow",
)
df.query("id1 > 'id098'").groupby("id2").sum().head(3)
```

filters are wrong, so query result is wrong 😱🫠

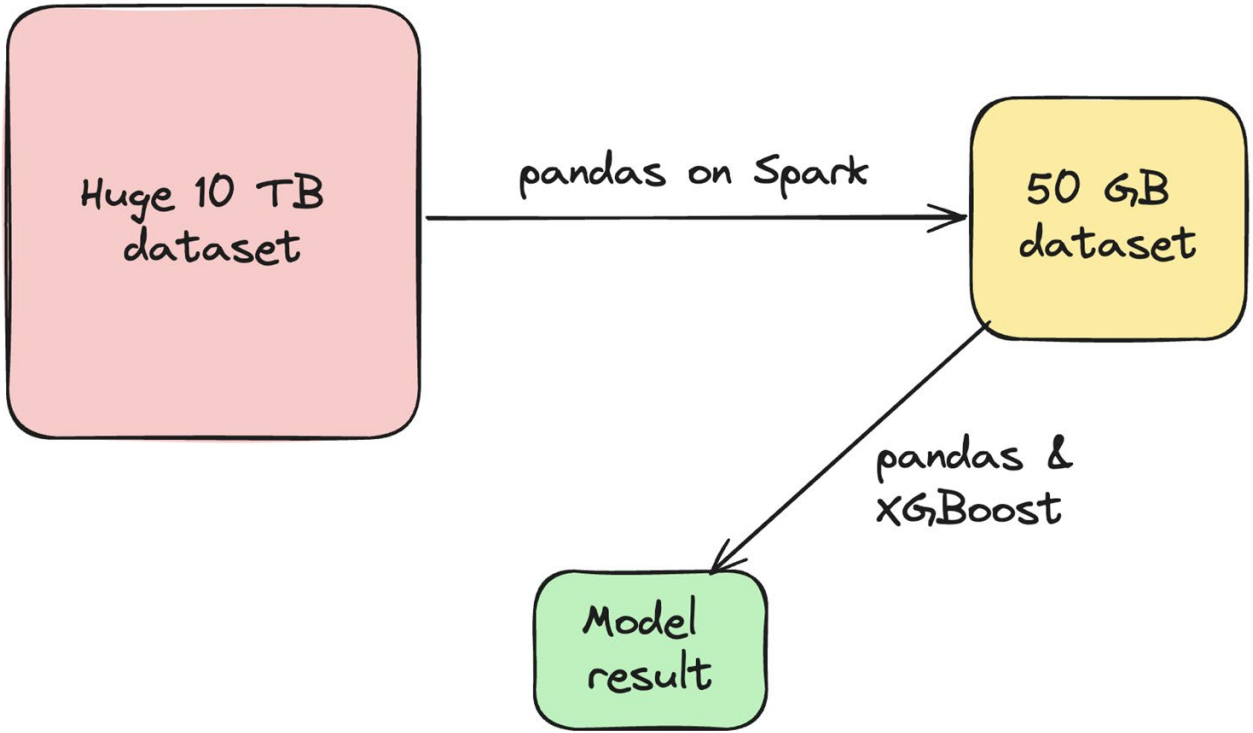
Easy to convert pandas on Spark DataFrames to pandas DataFrames

```
1  pdf = df.query("id1 == 'id094'").groupby("id2").sum().to_pandas()
2  pdf.head(3)
```

	v3
id2	
id089	3.620359e+07
id080	3.656286e+07
id087	3.619828e+07

```
Command took 13.01 seconds -- by matthew.powers@databricks.com at 5/21/2024, 8:23:06 PM on Matthew Powers's Cluster
```

Great production use case for pandas on Spark

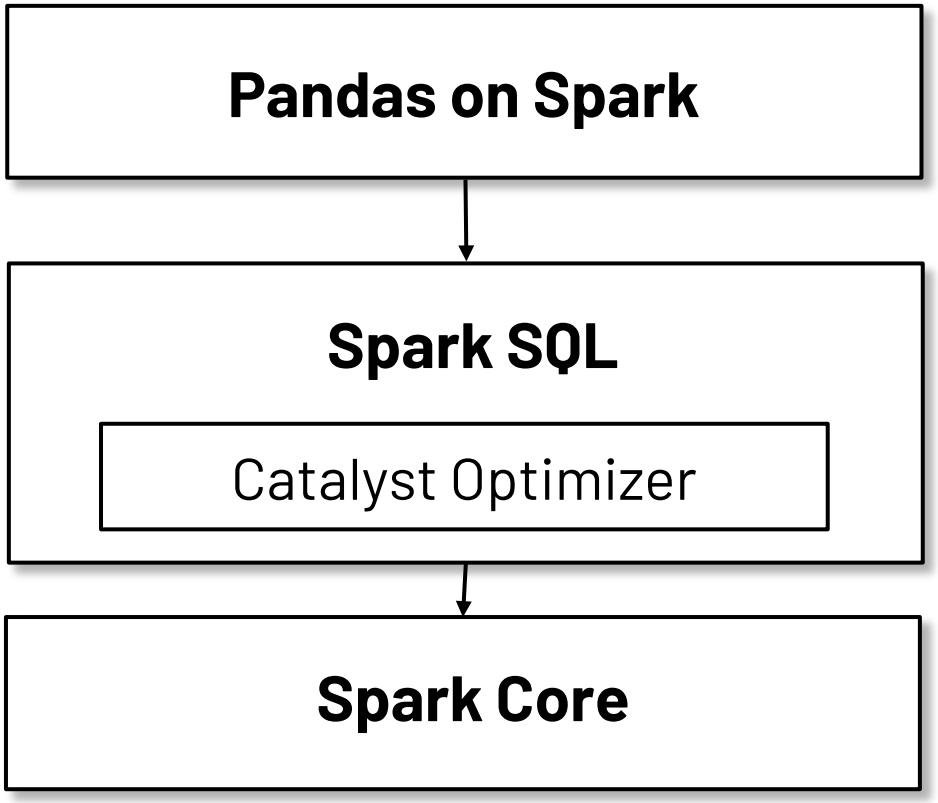


Advantages of pandas on Spark

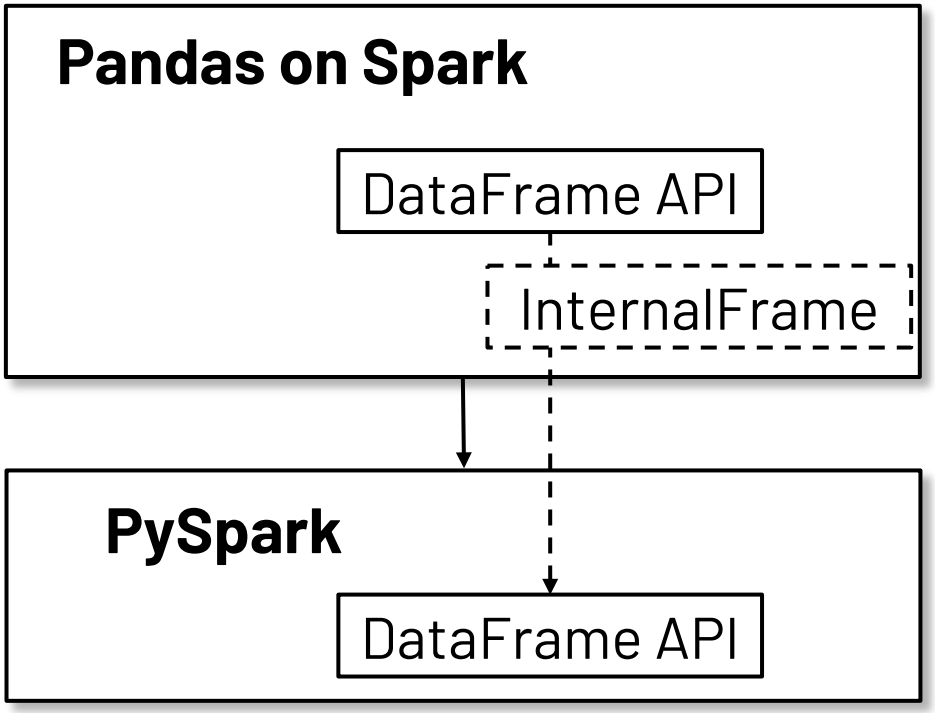
- Familiar pandas syntax for pandas coders
- Potentially faster performance on single node queries
- Ability to query large datasets
- Can query larger than memory datasets
- Great interop with pandas

How Does Pandas on Spark Work

Architecture



Architecture - DataFrame

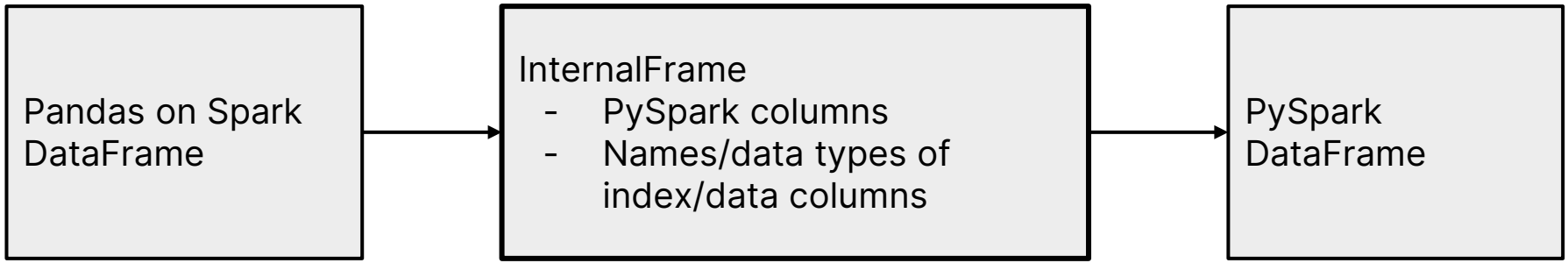


DataFrame Comparison

	Pandas on Spark	PySpark
Compliance	pandas DataFrame	Relations (tables)
Identifier	Rows, Columns	Columns
Mutability	Mutable	Immutable
Execution Engine	Spark SQL's	Spark SQL's

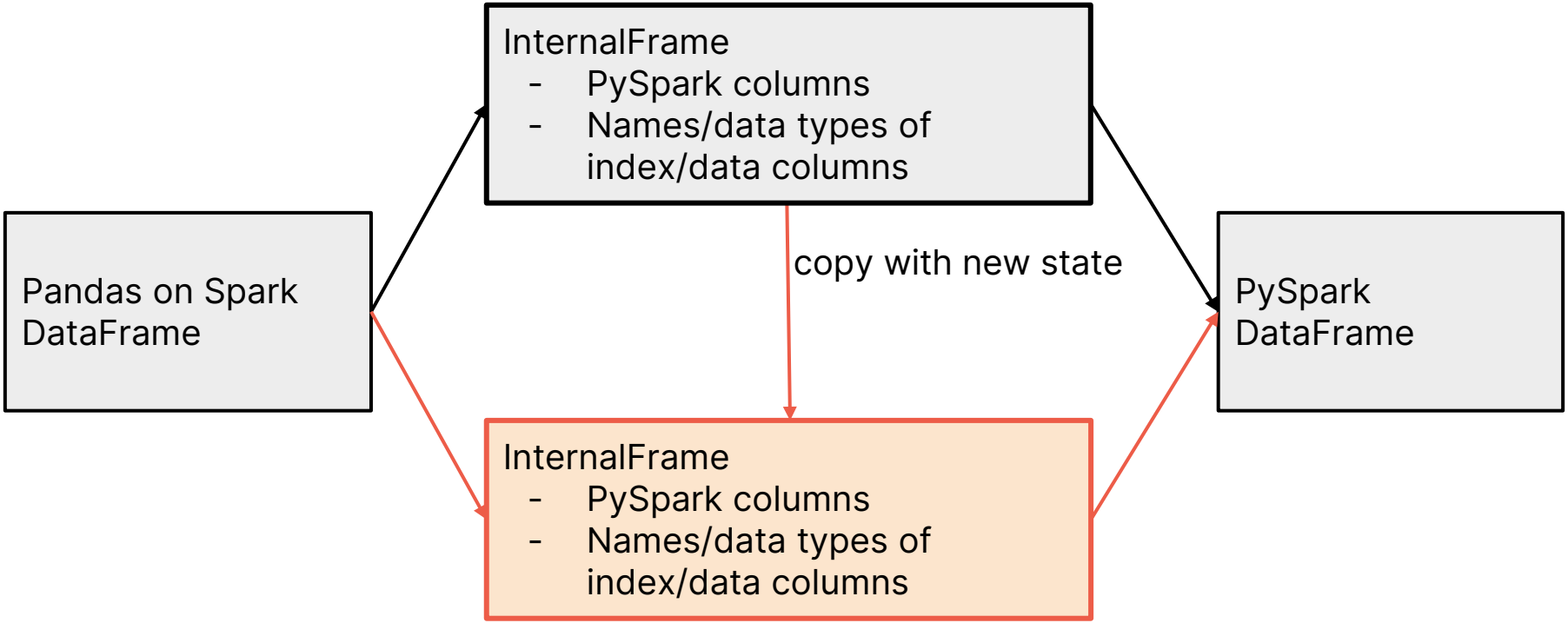
InternalFrame

Bridging the DataFrames



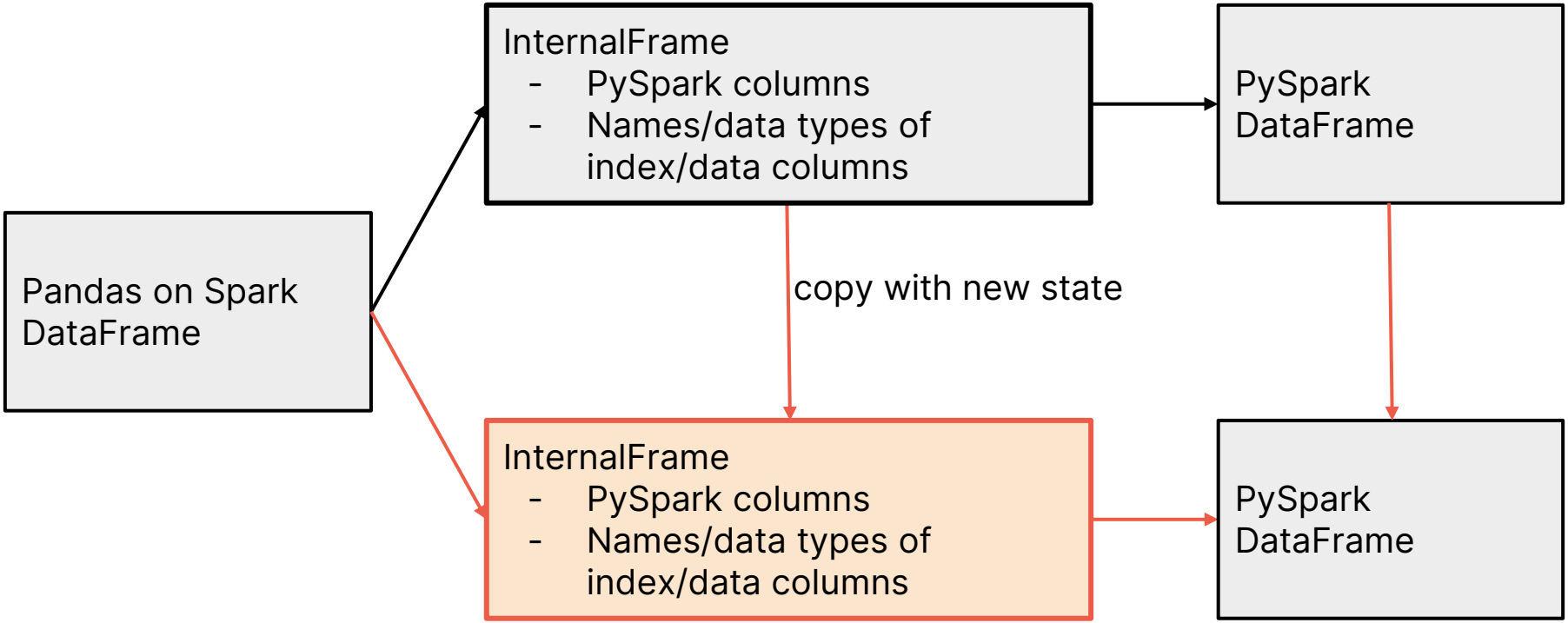
InternalFrame

When an API call happens



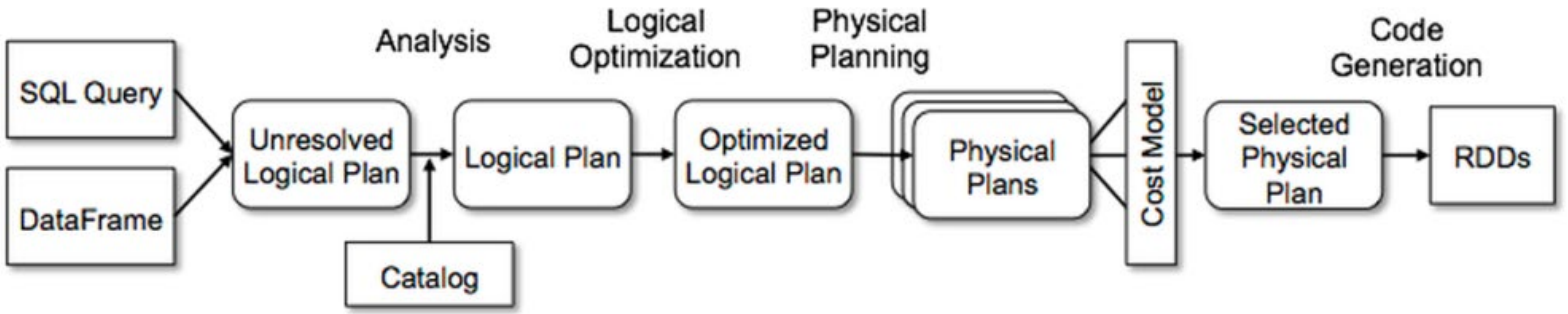
InternalFrame

When an API call happens



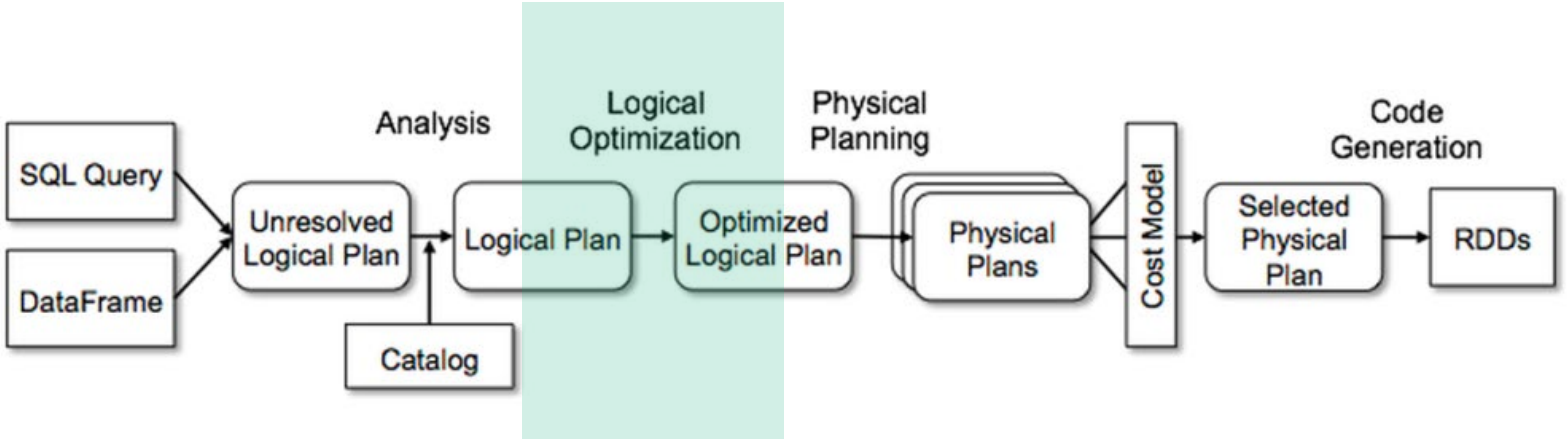
Catalyst Optimizer

A framework for representing trees and applying rules to manipulate them



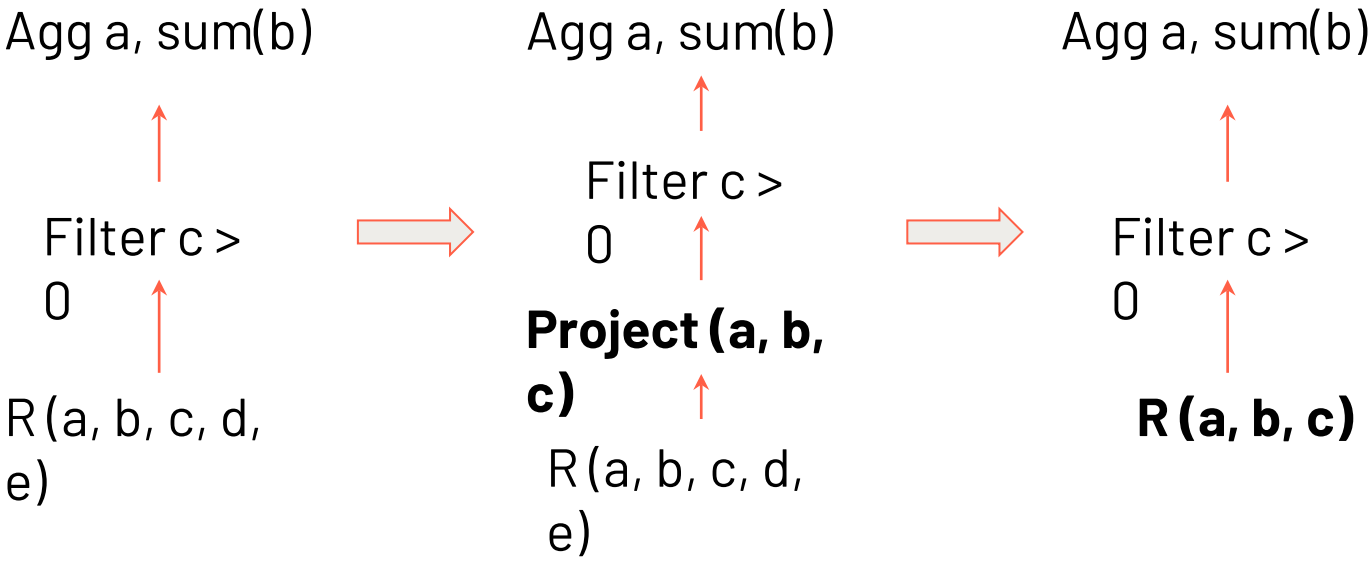
Catalyst Optimizer

Logical Optimization



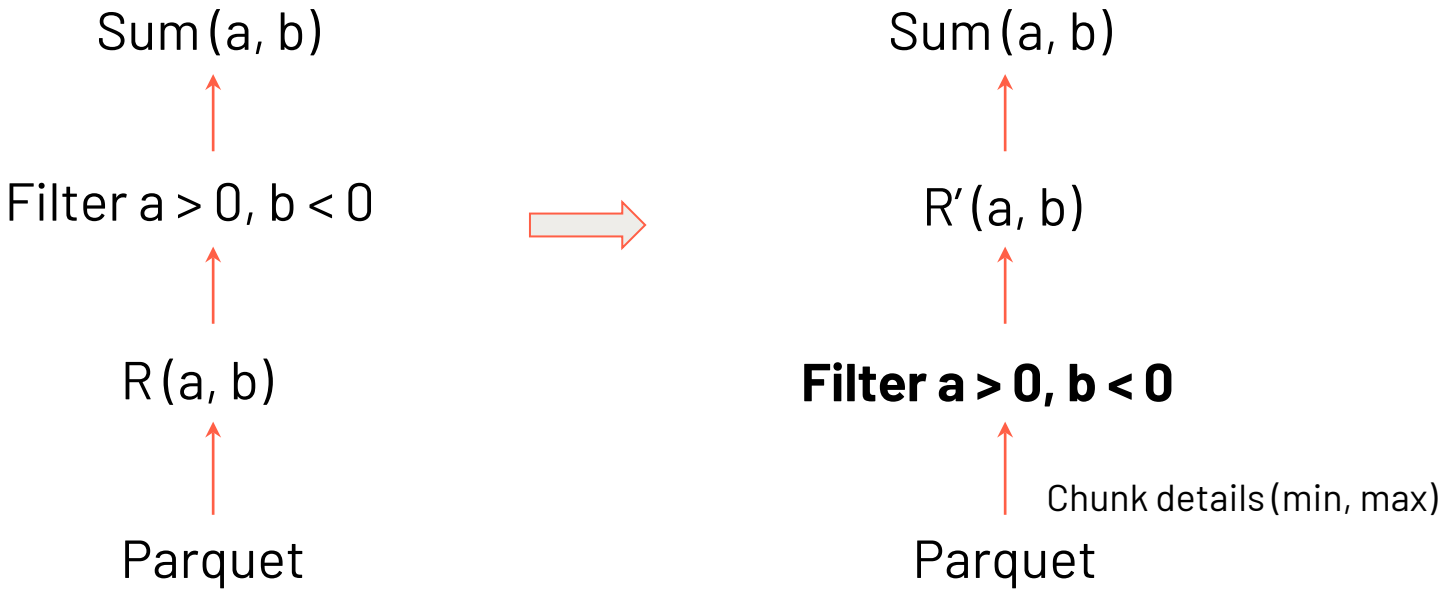
Rule - Column Pruning

Read the necessary columns only



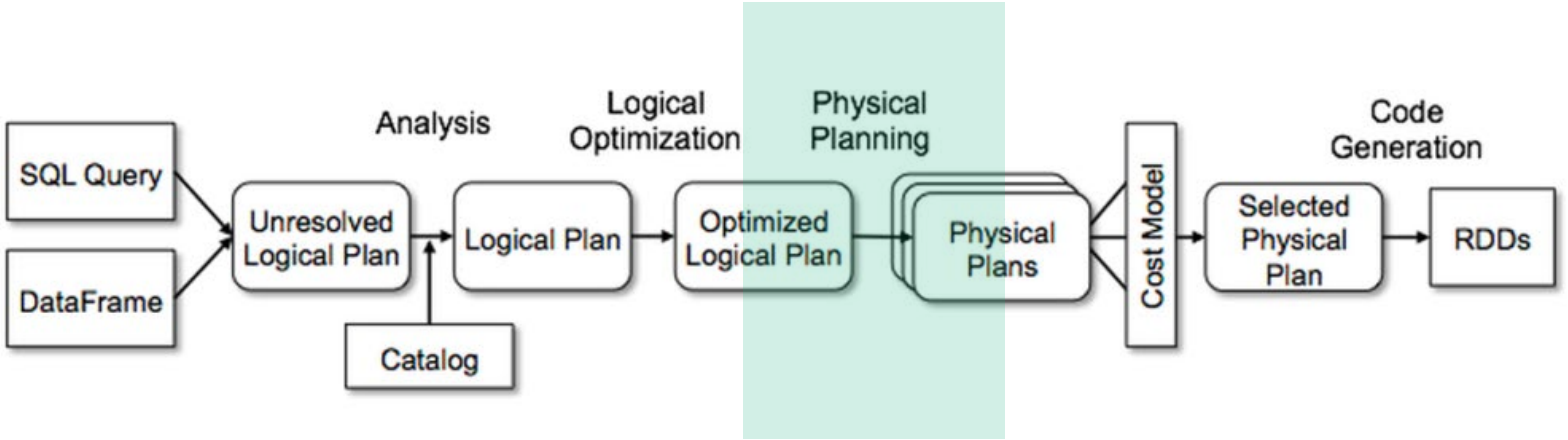
Rule - Predicate Pushdown

Push the filtering down to the data source



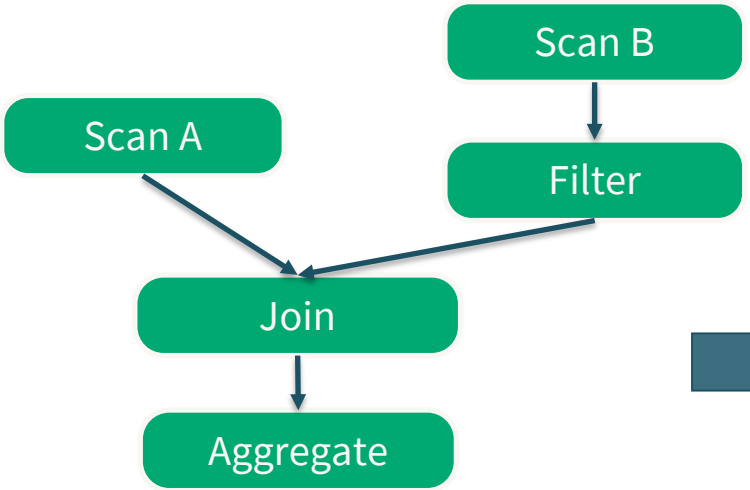
Catalyst Optimizer

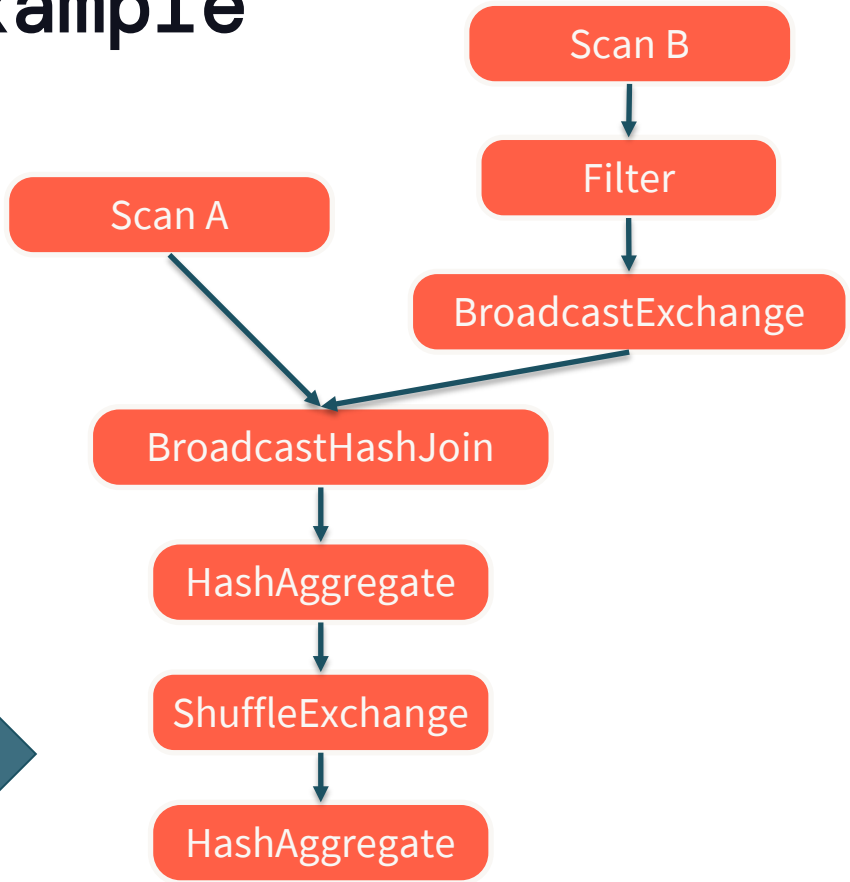
Physical Planning



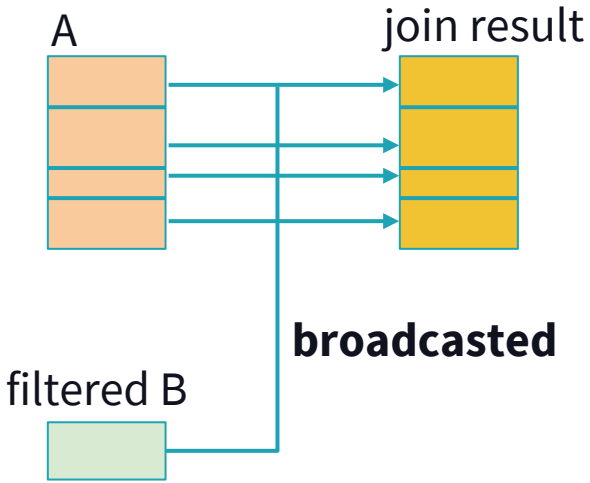
Physical Planning Example

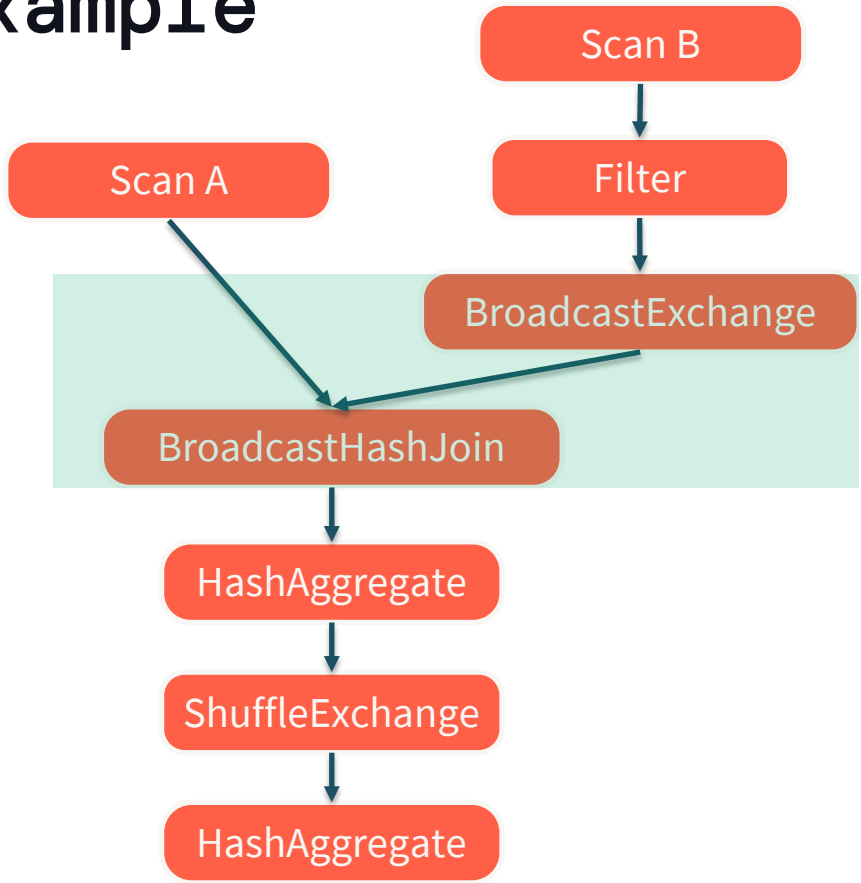
```
SELECT a1, sum(b1) FROM A
JOIN B ON A.key = B.key
WHERE b1 < 1000 GROUP BY a1
```





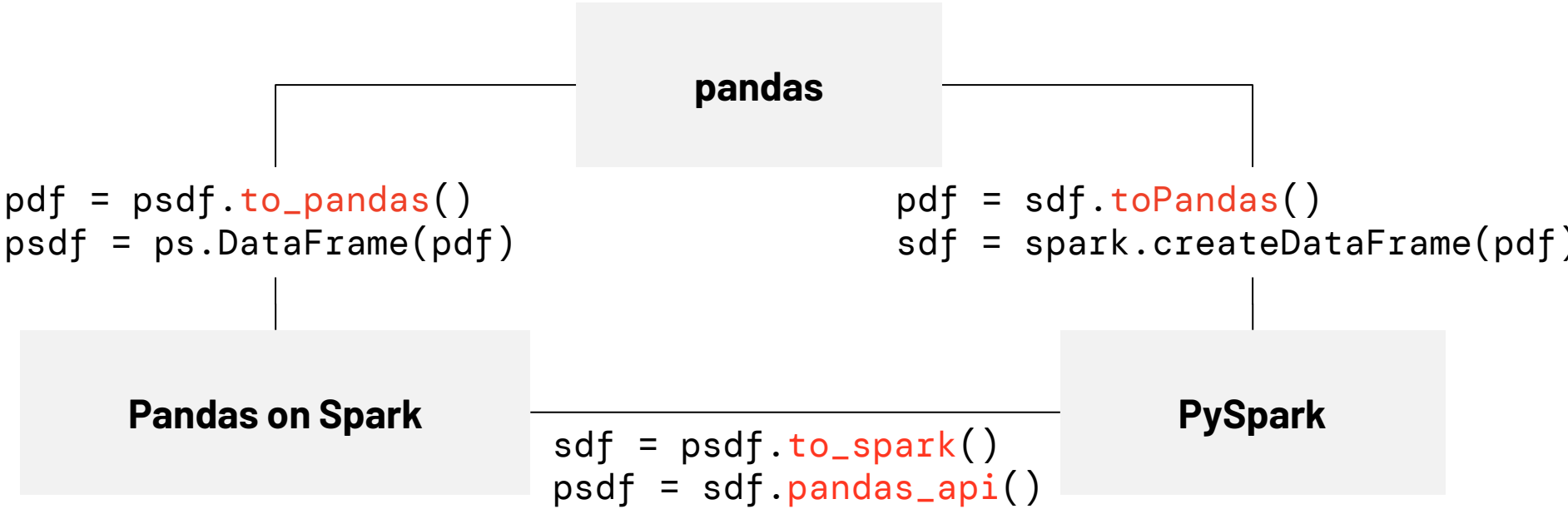
Physical Planning Example





Beyond Pandas

DataFrame Conversion



Query Data via SQL

ps.sql

```
>>> import pyspark.pandas as ps
>>> psdf = ps.DataFrame({"a": [1, 3, 5]})
>>> # Query via SQL
... ps.sql("SELECT count(*) AS num FROM {df}", df=psdf)
```

Check Execution Plans

ps.DataFrame.spark.explain

```
>>> import pyspark.pandas as ps
>>> psdf = ...
>>> psdf_small = ...
>>> joined_psdf = psdf.join(psdf_small, lsuffix='_left',
rsuffix='_right')

>>> joined_psdf.spark.explain()
== Physical Plan ==
+- Project ...
   +- SortMergeJoin
...
```

Check Execution Plans - Cont.

ps.DataFrame.spark.hint

```
>>> import pyspark.pandas as ps
>>> psdf = ...
>>> psdf_small = …

>>> joined_psdf = psdf.join(psdf_small.spark.hint('broadcast'),
lsuffix='_left', rsuffix='_right')

>>> joined_psdf.spark.explain()
== Physical Plan ==
+- Project ...
   +- BroadcastHashJoin
...
```

Machine Learning Utilities

ps.mlflow.load_model

```
>>> from pyspark.pandas.mlflow import load_model
>>> run_info = client.search_runs(exp_id)[-1].info
>>> model = load_model("runs:/{run_id}/model".format(run_id=run_info.run_id))
>>> prediction_df = ps.DataFrame({"x1": [2.0], "x2": [4.0]})
>>> prediction_df["prediction"] = model.predict(prediction_df)
>>> prediction_df
    x1   x2  prediction
0  2.0  4.0    1.355551
```

What's New

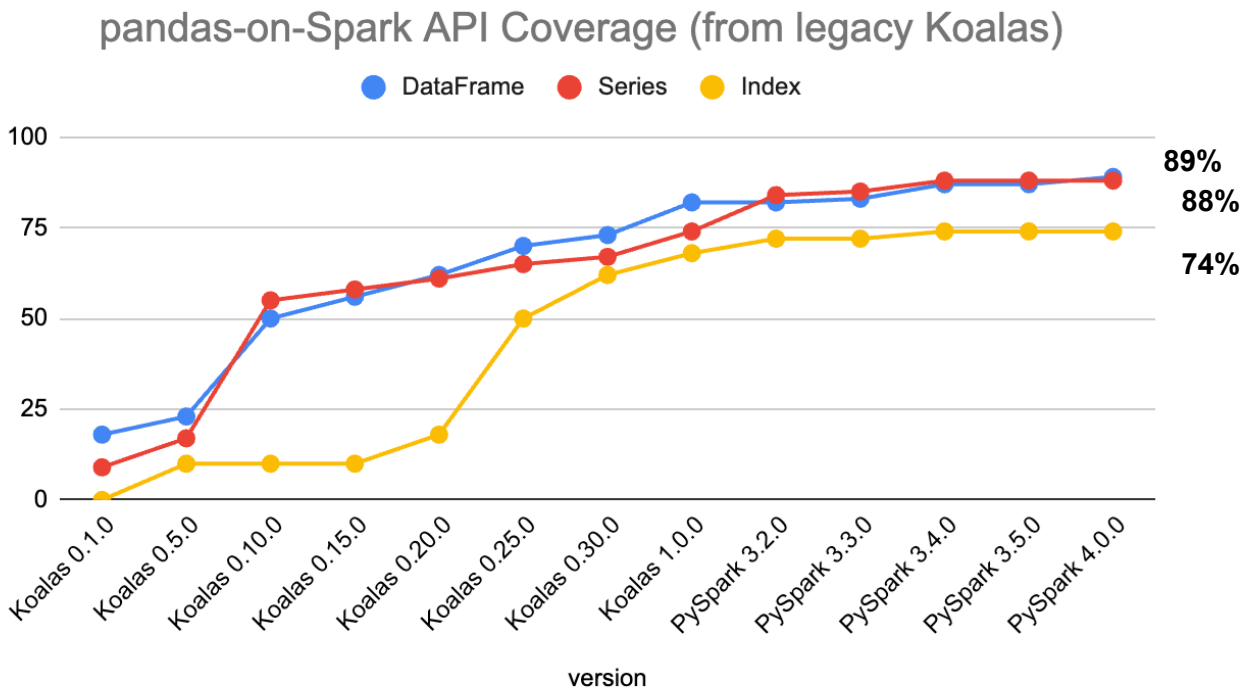
Pandas 2.x Support

API change parity with Pandas 2.2.2

Backwards incompatible API changes

-
-

Pandas API Coverage



Supported pandas API

Fallback Mechanism

“Defaulting to pandas” method for non -supported APIs

Fallback Mechanism

Option compute.pandas_fallback

2 minutes ago (1s) 10 Python

```
ps.set_option("compute.pandas_fallback", True)
psdf.asfreq(freq='30s')
```

```
/databricks/spark/python/pyspark/pandas/utils.py:1026: PandasAPIOnSparkAdviceWarning: `asfreq` is executed in fallback mode. It loads partial data into the driver's memory to infer the schema, and loads all data into one executor's memory to compute. It should only be used if the pandas DataFrame is expected to be small.
  warnings.warn(message, PandasAPIOnSparkAdviceWarning)
/databricks/spark/python/pyspark/pandas/utils.py:1026: PandasAPIOnSparkAdviceWarning: If the type hints is not specified for `groupby.apply`, it is expensive to infer the data type internally.
  warnings.warn(message, PandasAPIOnSparkAdviceWarning)
```

	0
2024-06-01 00:00:00	0
2024-06-01 00:00:30	1
2024-06-01 00:01:00	2
2024-06-01 00:01:30	3

Join the Spark community

Open, welcoming, and fun

- Join the user or dev mailing lists
- Help us answer questions on StackOverflow
- Follow Apache Spark on LinkedIn
- Perhaps a community Discord coming soon…

Thank You