Polystores for Real

REFLECTIONS FROM MICROSOFT

Alekh Jindal

Has the needle moved?

Microsoft then …

Externally

SQL Server

Single stop shop for all data management needs

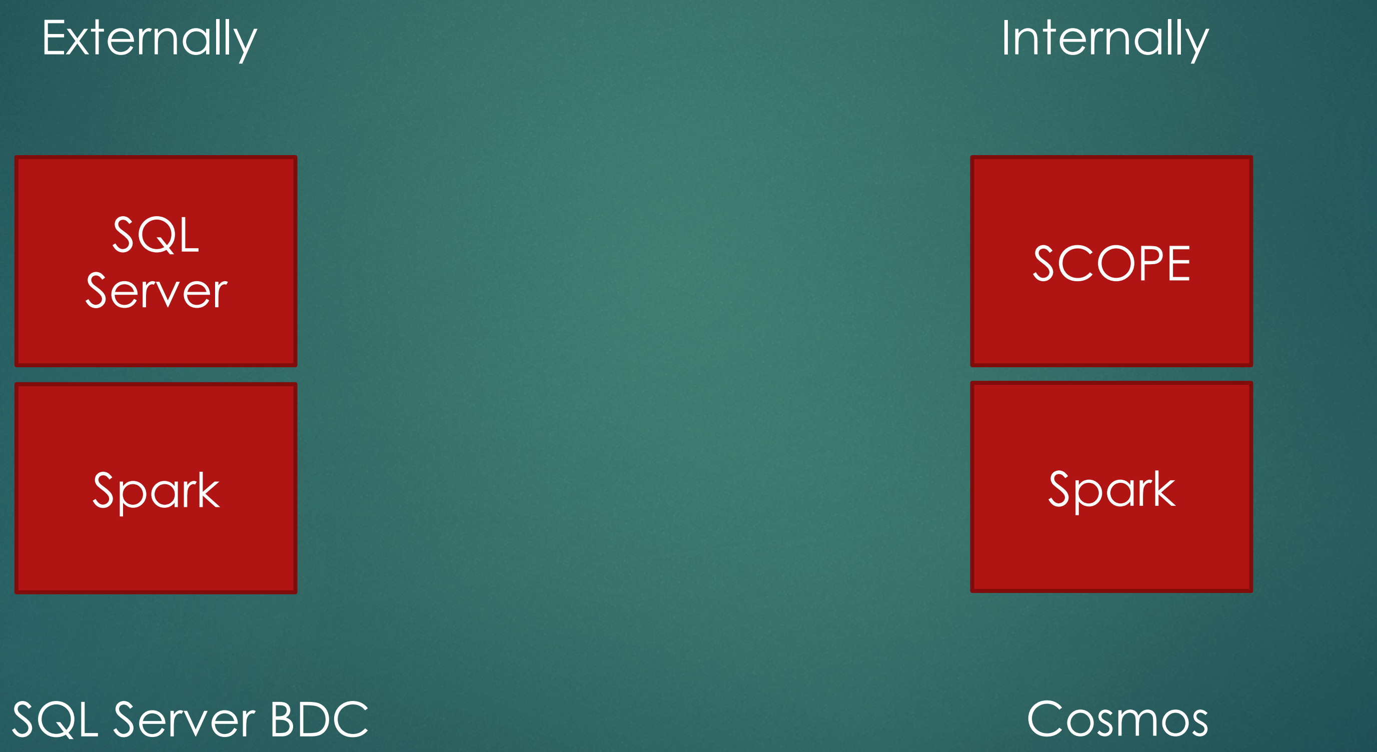
Internally

SCOPE

Single stop shop for all big data needs

Giant Monoliths are not the future!

… Microsoft now



Externally

SQL Server

Spark

SQL Server BDC

Internally

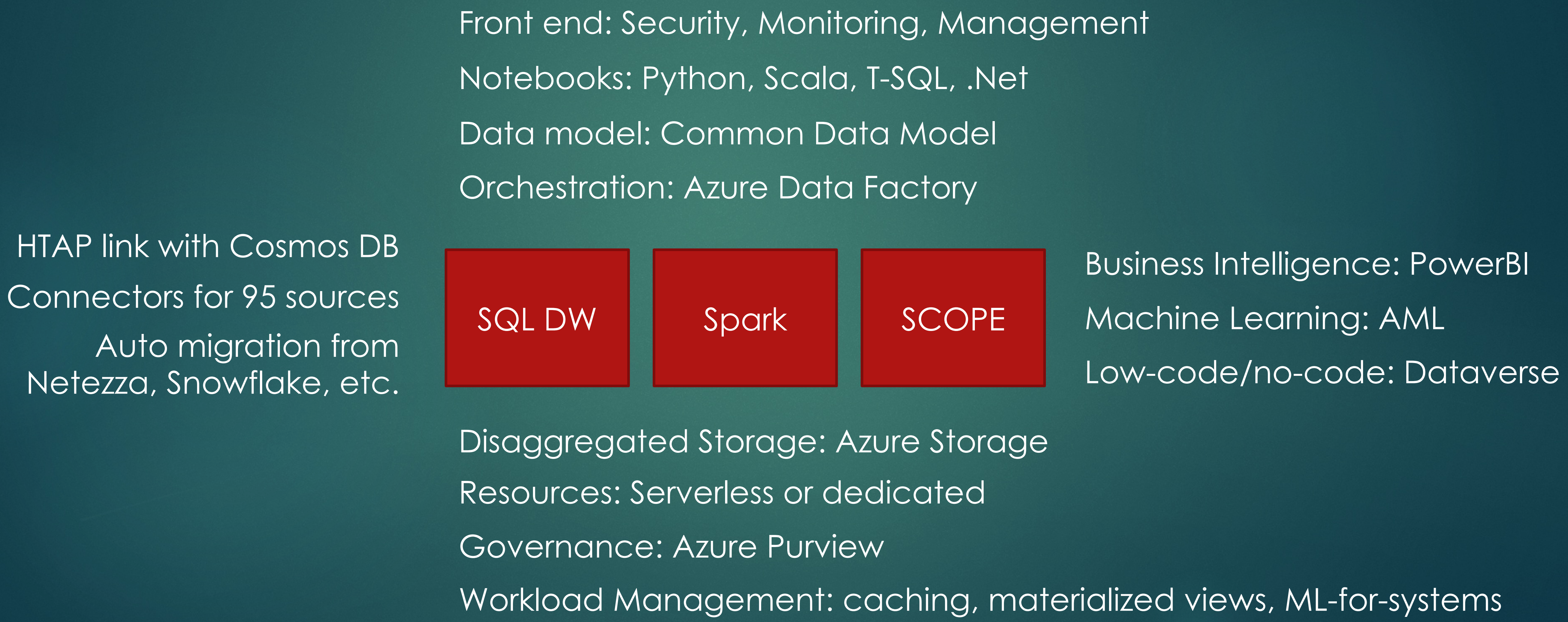
SCOPE

Spark

Cosmos

Towards more inclusive data platforms

Azure Synapse: a unified world for analytics



Front end: Security, Monitoring, Management

Notebooks: Python, Scala, T-SQL, .Net

Data model: Common Data Model

Orchestration: Azure Data Factory

HTAP link with Cosmos DB

Connectors for 95 sources

Auto migration from Netezza, Snowflake, etc.

SQL DW

Spark

SCOPE

Business Intelligence: PowerBI

Machine Learning: AML

Low-code/no-code: Dataverse

Disaggregated Storage: Azure Storage

Resources: Serverless or dedicated

Governance: Azure Purview

Workload Management: caching, materialized views, ML-for-systems

An engine-inclusive platform

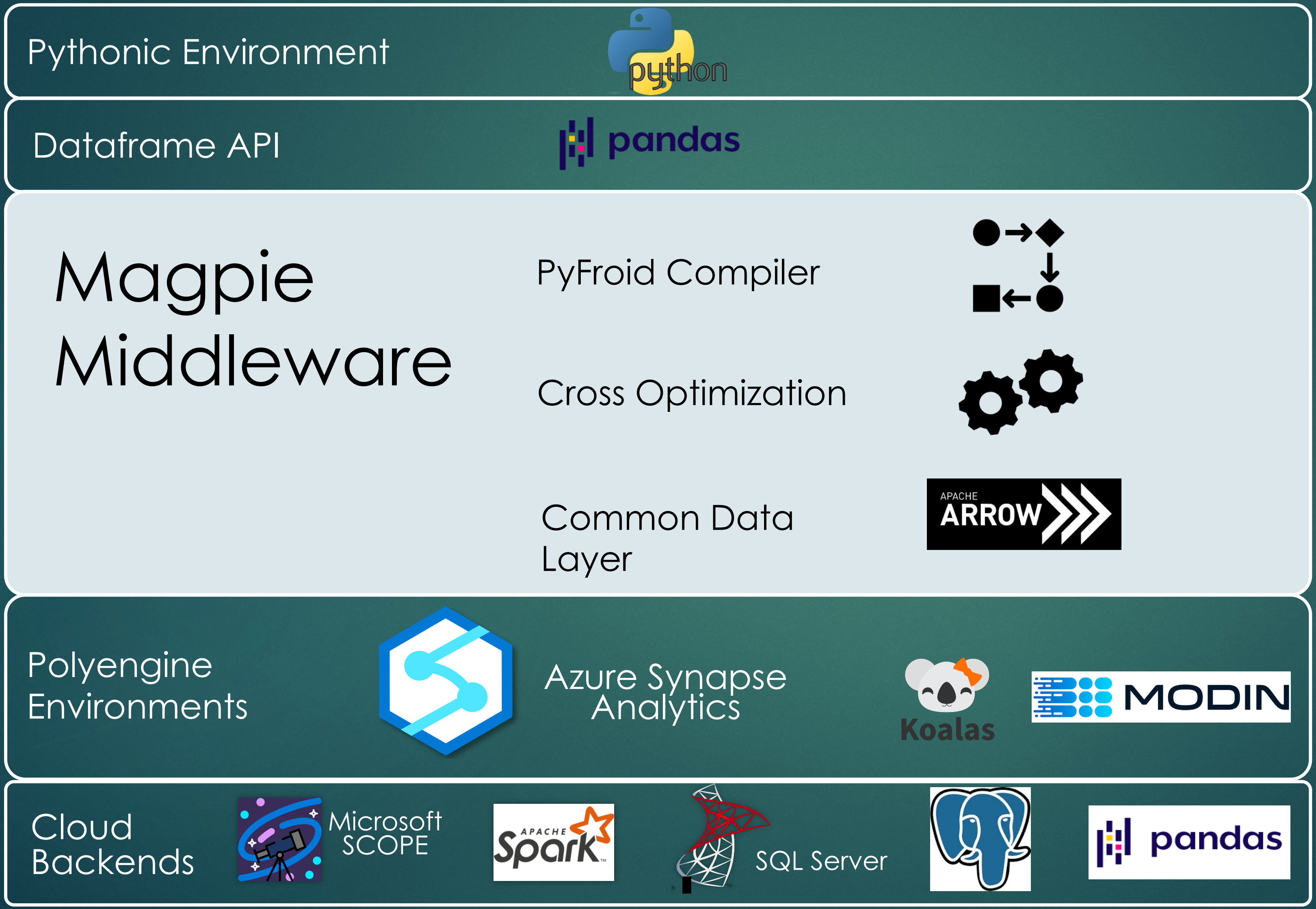
- All engines are welcome!
- Tightly integrated ecosystem
- Decoupling common functionality into separate layers
- Polystores => Polyengines

Can data platforms be engine-agnostic?

Challenges

- Users need to:
  - Be aware of the polyengines
  - Carefully pick their engines
  - Operate the chosen set of engines
- Can we:
  - Interoperate?
  - Move data efficiently?
  - Pick the best engine for each application?
- Should users really care about the various polyengines?

Scenario: data science at cloud-scale



Pythonic Environment

Dataframe API

Magpie Middleware

PyFroid Compiler

Cross Optimization

Common Data Layer

Polyengine Environments

Azure Synapse Analytics

Cloud Backends

Microsoft SCOPE

SQL Server

Familiar Python surface

Ongoing standardization

Batching Pandas into large query expressions

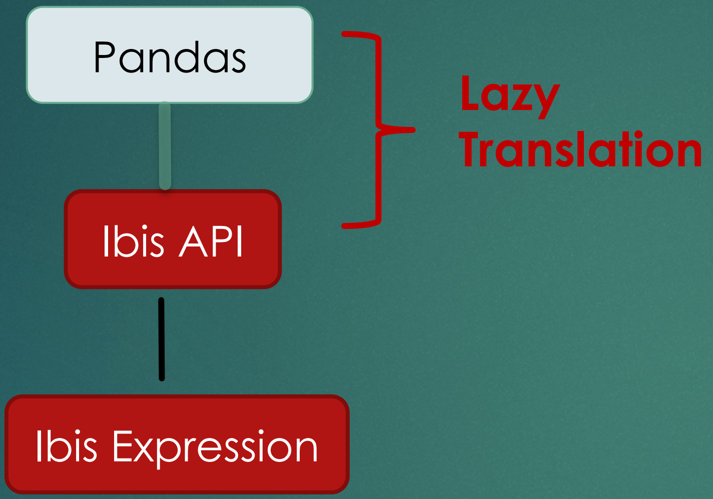
Backend selection using past workloads

Cache commonly seen dataframes

Multi-backend environments and libraries

Cloud backends

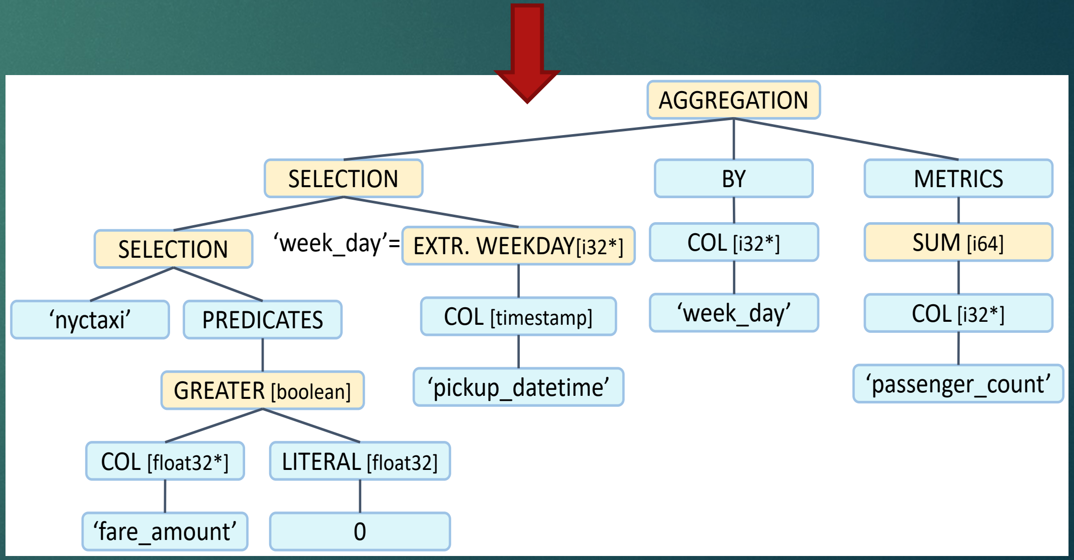
Compiling Pandas using Ibis



The number of taxi trips per
weekday over the NYC Taxi dataset

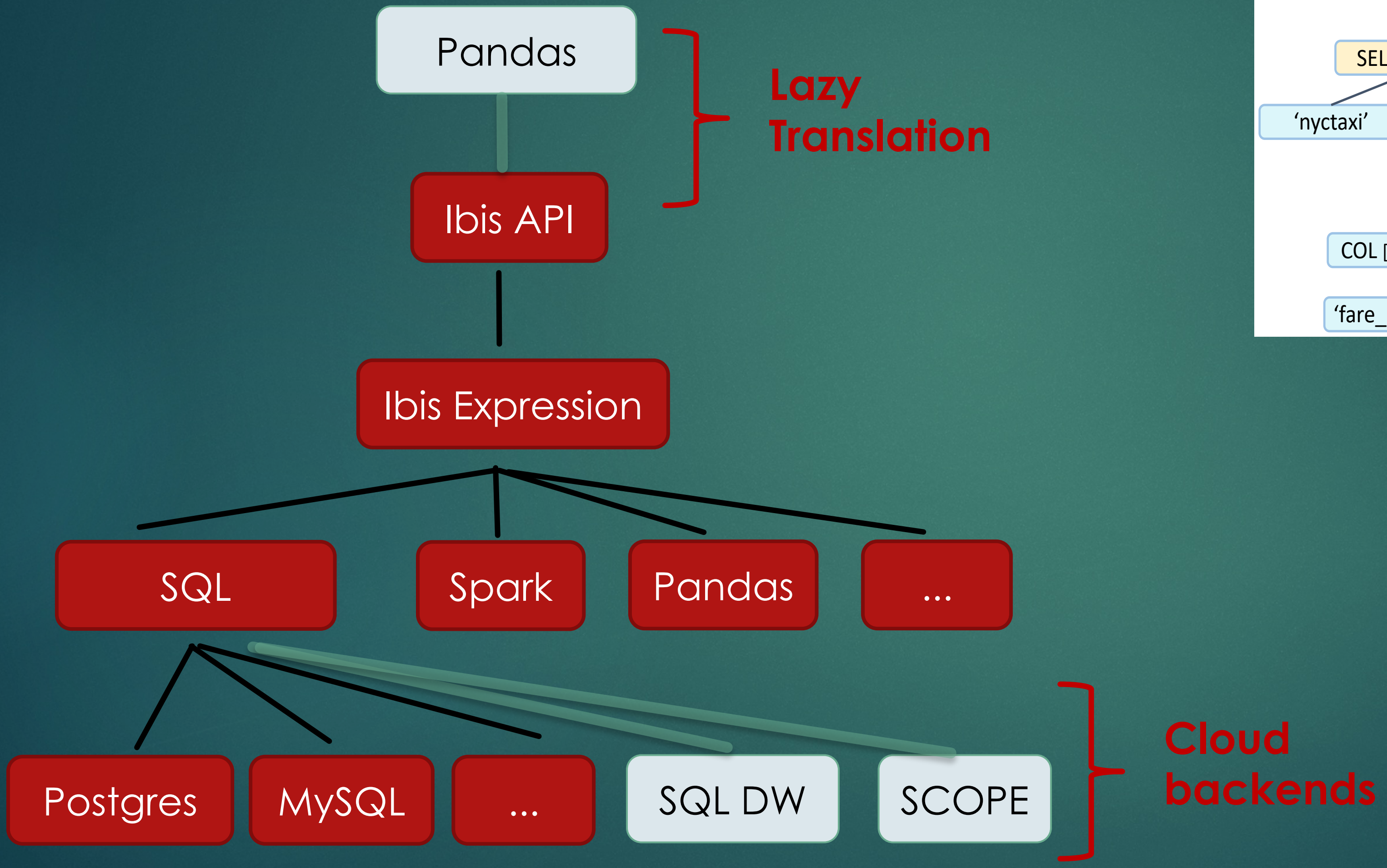
```
import pyfroid.pandas as pd                          # vs import pandas as pd
df = pd.read_sql(‘nyctaxi’, con)                     # fetch data
df = df[df.fare_amount > 0]                          # filter bad rows
df[‘day’] = df.pickup_datetime.dt.dayofweek              # add features
df = df.groupby([‘day’])[‘passenger_count’].sum()    # aggregation
print(df)                                         # use dataframe
```

Pandas Dataframe Program



Intermediate Representation

Compiling Pandas using Ibis


Pandas

Ibis API

Lazy Translation

Ibis Expression

SQL

Spark

Pandas

...

Postgres

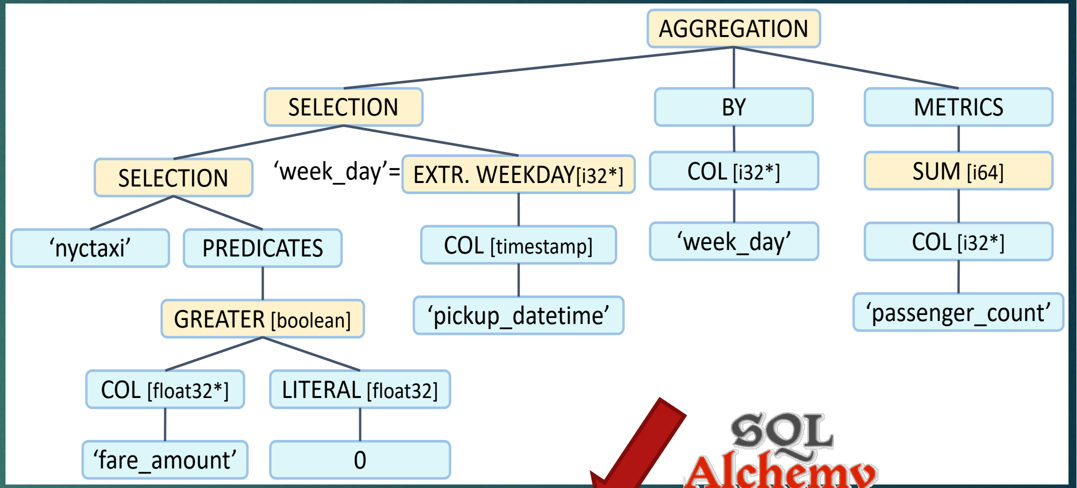
MySQL

...

SQL DW

SCOPE

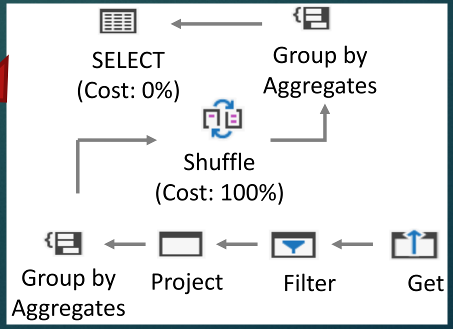
Cloud backends



SQL Alchemy

```
 SELECT DATEPART(WEEKDAY, pickup_datetime) AS
day,
        SUM(passenger_count)
 FROM nyctaxi WHERE fare_amount > 0
 GROUP BY DATEPART(WEEKDAY, pickup_datetime)
```

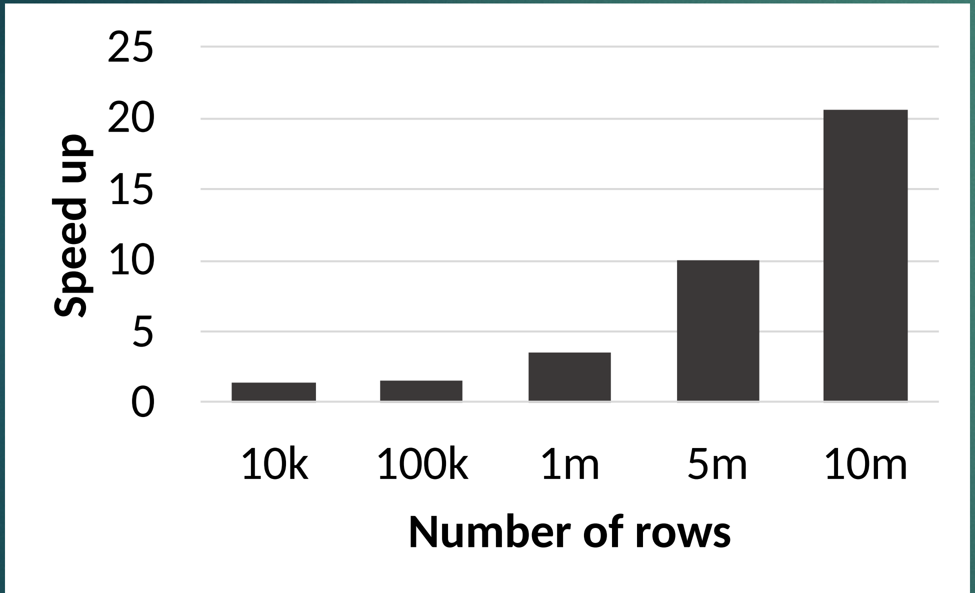
T-SQL Statement



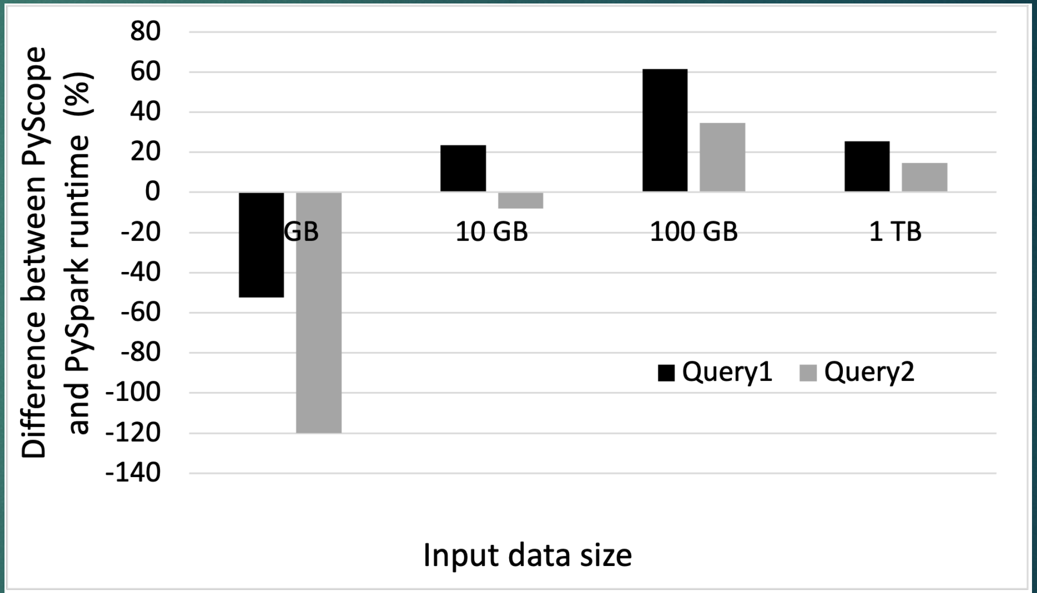
SQL DW Execution Plan

Backend Selection Decisions

Speed-up using SQL DW

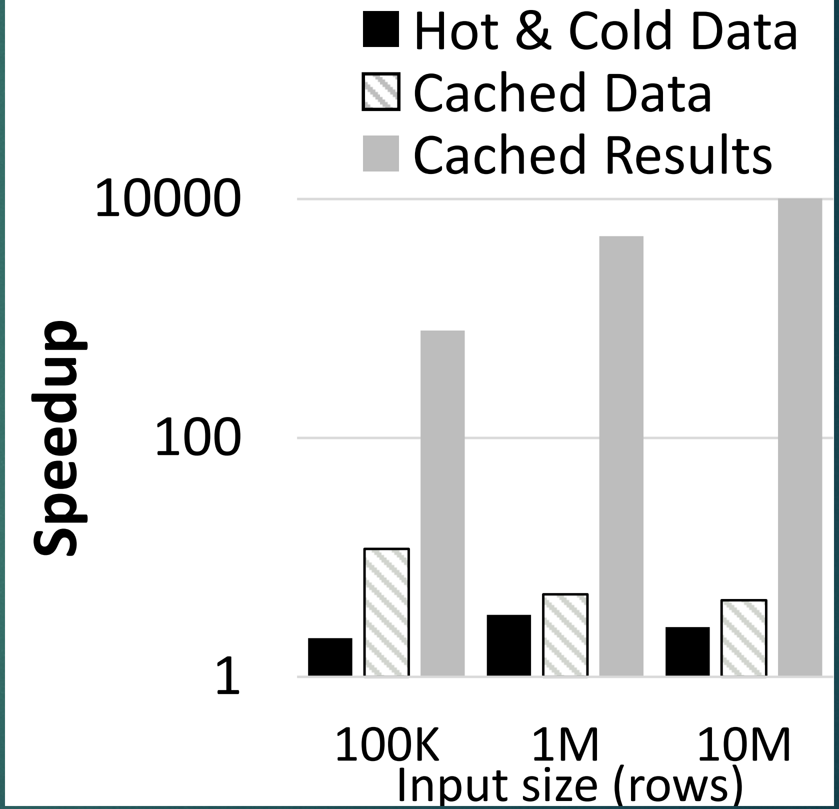


Scale-out using SCOPE/Spark



Data Movement and Caching

- Different data sources together
  - Combing hot data from SQL DW with cold data from Cosmos
  - Cache more stationary cold data
  - 2-3x speedups
- Different data scientists together
  - Collaboration on same datasets
  - Cache frequently accessed
    - Datasets: 4-11x speedup
    - Dataframes: 800-3800x speedup



Remarks

- Polystores have come a long way from academia to industry
  - Evidence of engine-inclusive platforms
- Example: Azure Synapse provides
  - Polyengines
  - Tightly integrated
  - Common functionality abstracted out
- Question: can the next level be engine-agnostic?
  - Do users really need to be aware of and learn numerous engines?
  - Can we make their easier with better cost and performance?
  - E.g., bringing data scientists to cloud-scale