
Spark + AI Summit
@rxin
databricks

Lester

$1m NETFLIX Prize

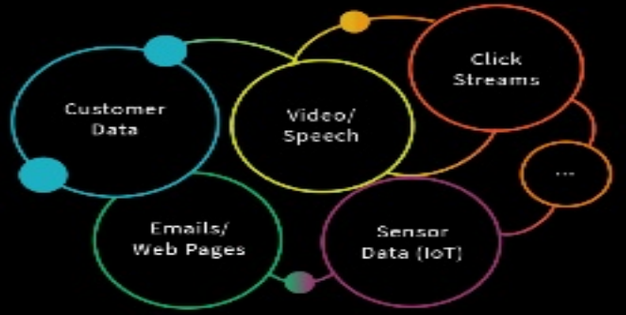
anonymized movie rating dataset

best recommendation algorithm wins

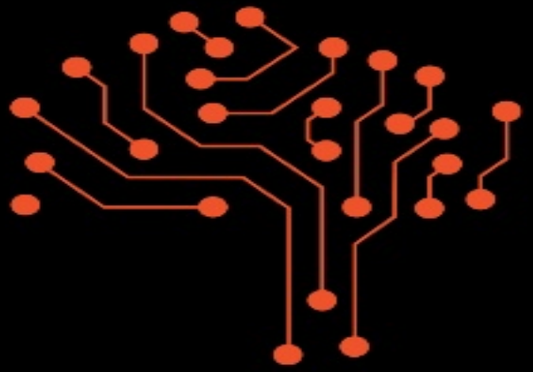
Lester

Matei

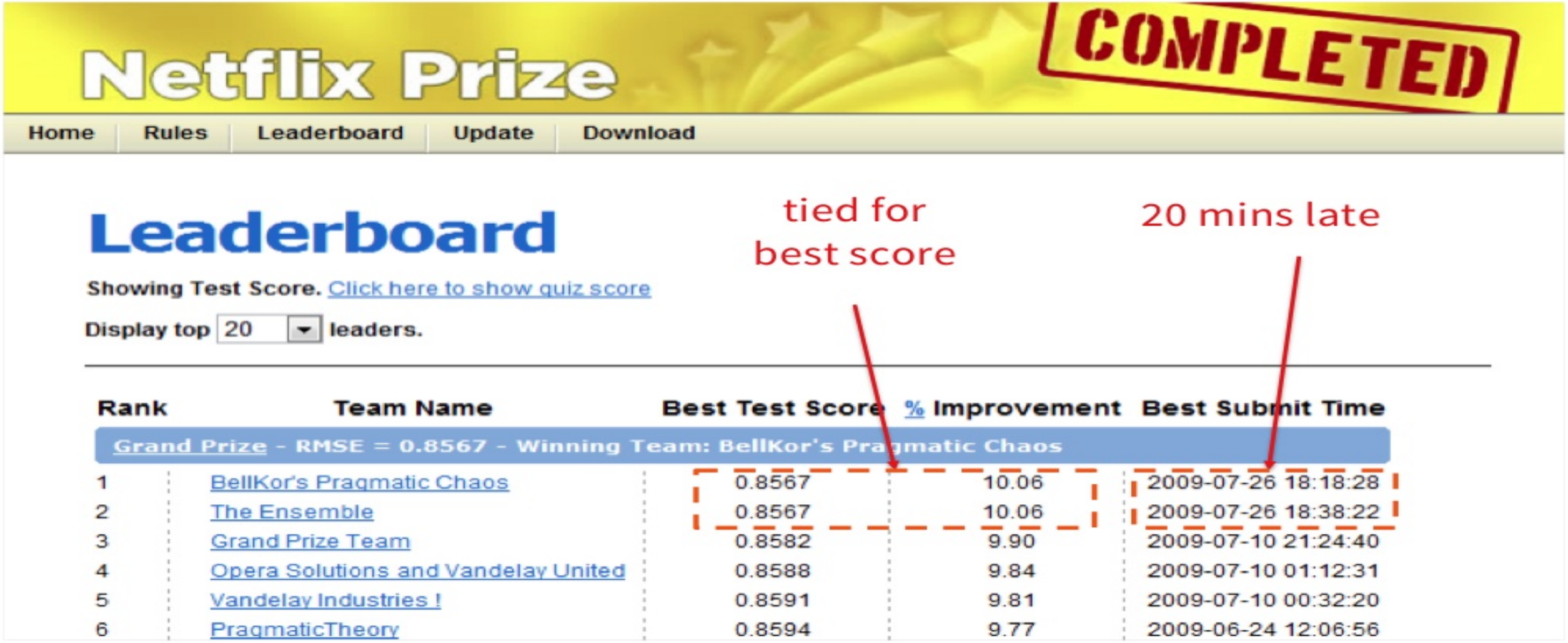
The first unified analytics engine in 600 lines of code...



APACHE Spark™

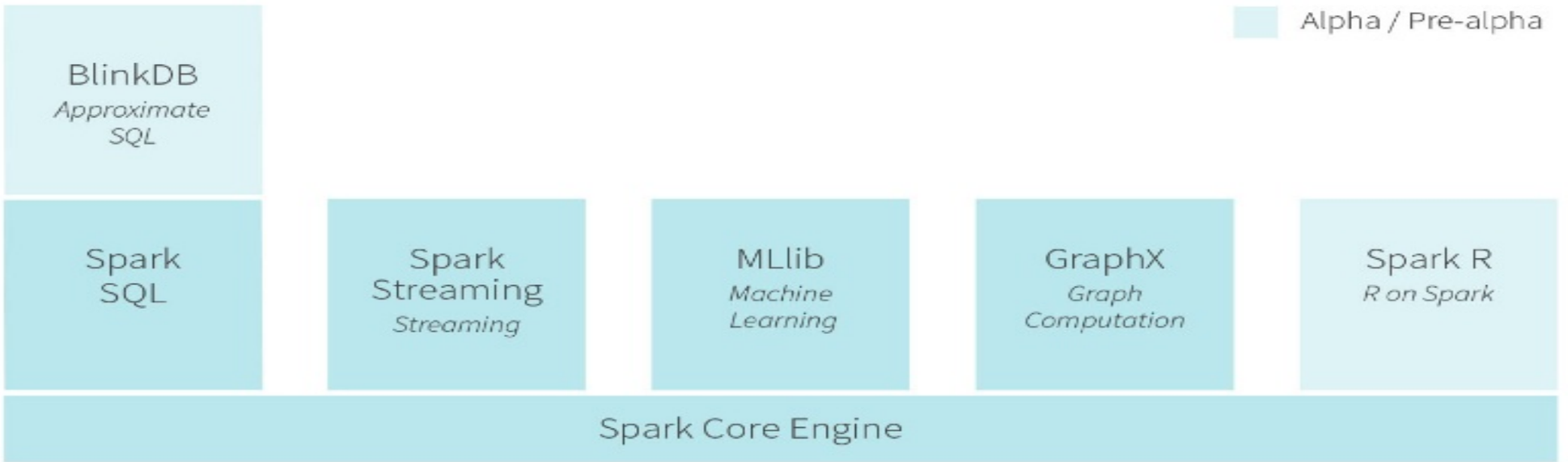


Big Data | Machine Learning


Netflix Prize
COMPLETED
Home
Rules
Leaderboard
Update
Download
Leaderboard
tied for best score
20 mins late
Showing Test Score. Click here to show quiz score
Display top 20 leaders.
Rank
Team Name
Best Test Score
% Improvement
Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos
1 BellKor's Pragmatic Chaos 0.8567 10.06 2009-07-26 18:18:28
2 The Ensemble 0.8567 10.06 2009-07-26 18:38:22
3 Grand Prize Team 0.8582 9.90 2009-07-10 21:24:40
4 Opera Solutions and Vandelay United 0.8588 9.84 2009-07-10 01:12:31
5 Vandelay Industries ! 0.8591 9.81 2009-07-10 00:32:20
6 PragmaticTheory 0.8594 9.77 2009-06-24 12:06:56


NETFLIX
2009
DATE 09.21.09
PAY TO THE ORDER OF BellKor's Pragmatic Chaos
$ 1,000,000 00
AMOUNT ONE MILLION
00/100
FOR The Netflix Prize
Reed Hastings

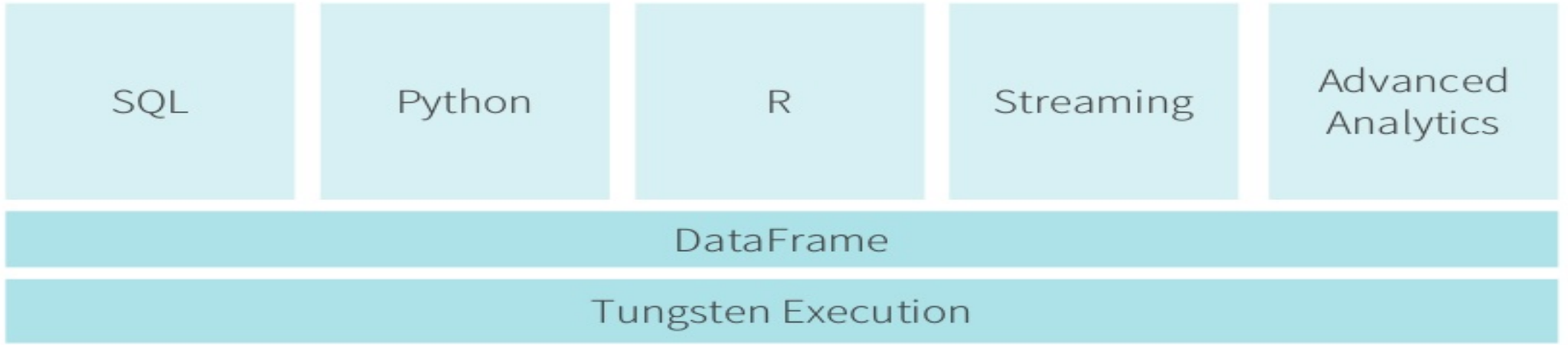
Apache Spark 1.0 (2014)



Fast and general engine for distributed data processing

databricks

DataFrame + Tungsten (2015)

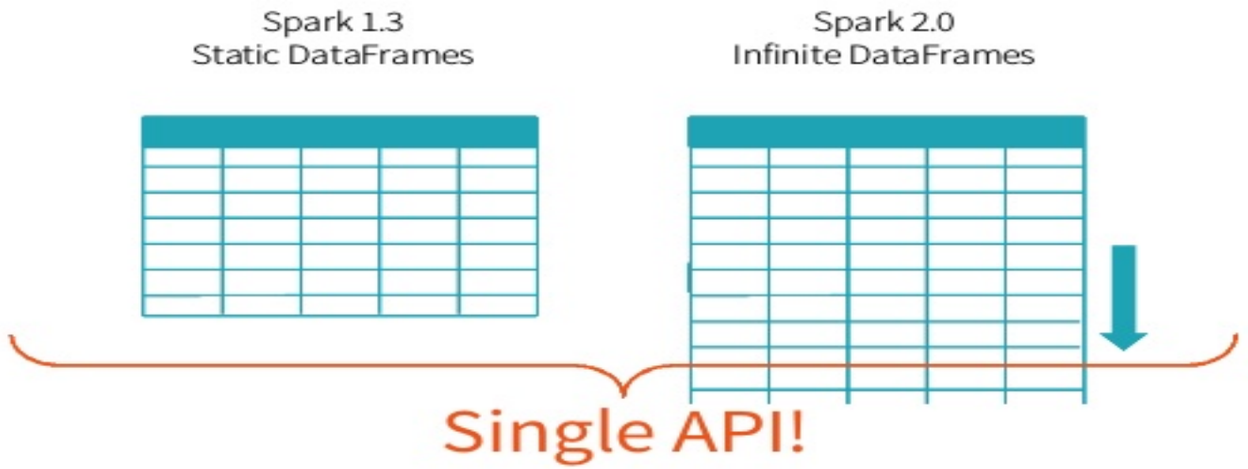


“Apache Spark is the Taylor Swift of big data software.”

- Derrick Harris, Fortune, Sep 2015

Structured Streaming (2016)

The simplest way to perform analytics is not having to *reason* about streaming



databricks

Continuous Processing (2017)

sub-milliseconds streaming

A new execution mode that follows fully pipelined execution.

- Streaming execution ***without microbatches***
- Supports async checkpointing ~1ms latency
- no changes required for user code

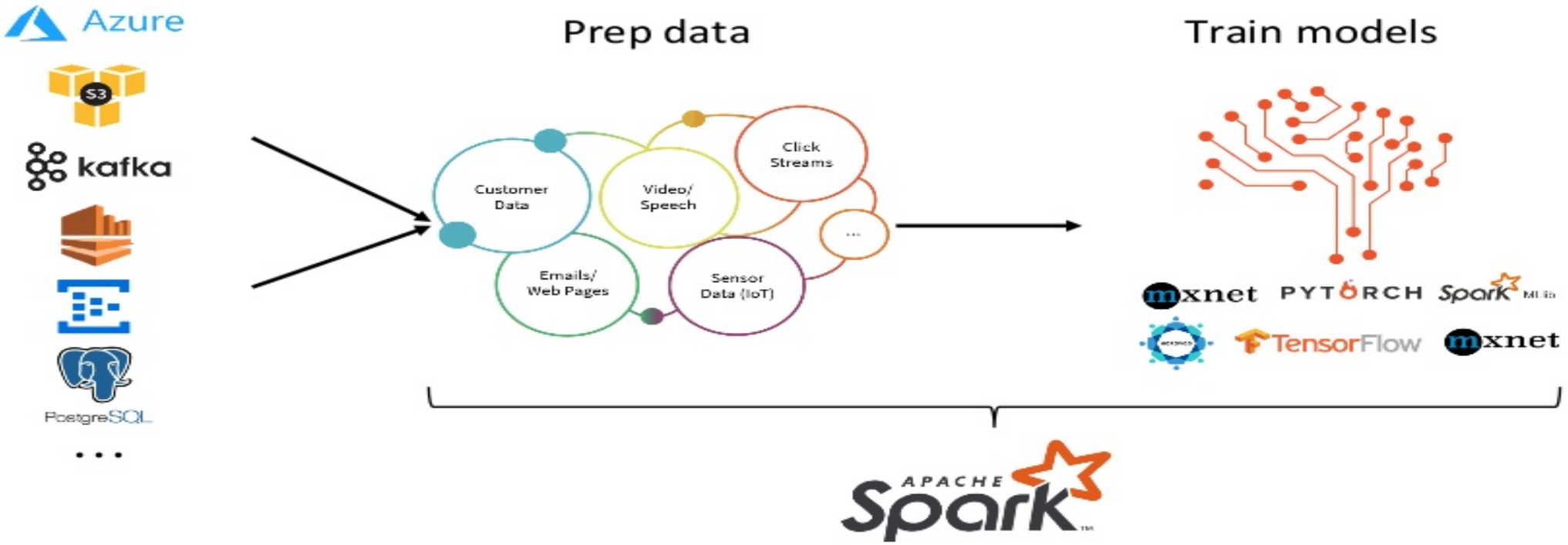
Proposal available at https://issues.apache.org/jira/browse/SPARK-20928

100,000,000,000,000

streaming records processed
on Databricks in 2018

Explosion of ML Frameworks

Embracing ML ecosystem as 1st-class citizens



databricks

Two Challenges in Supporting ML Frameworks in Spark

1

Data exchange:

need to push data in high throughput between Spark and ML frameworks

2

Execution model:

fundamental incompatibility between Spark (embarrassingly parallel) vs ML frameworks (gang scheduled)


Introducing
Project Hydrogen
databricks

Data Exchange

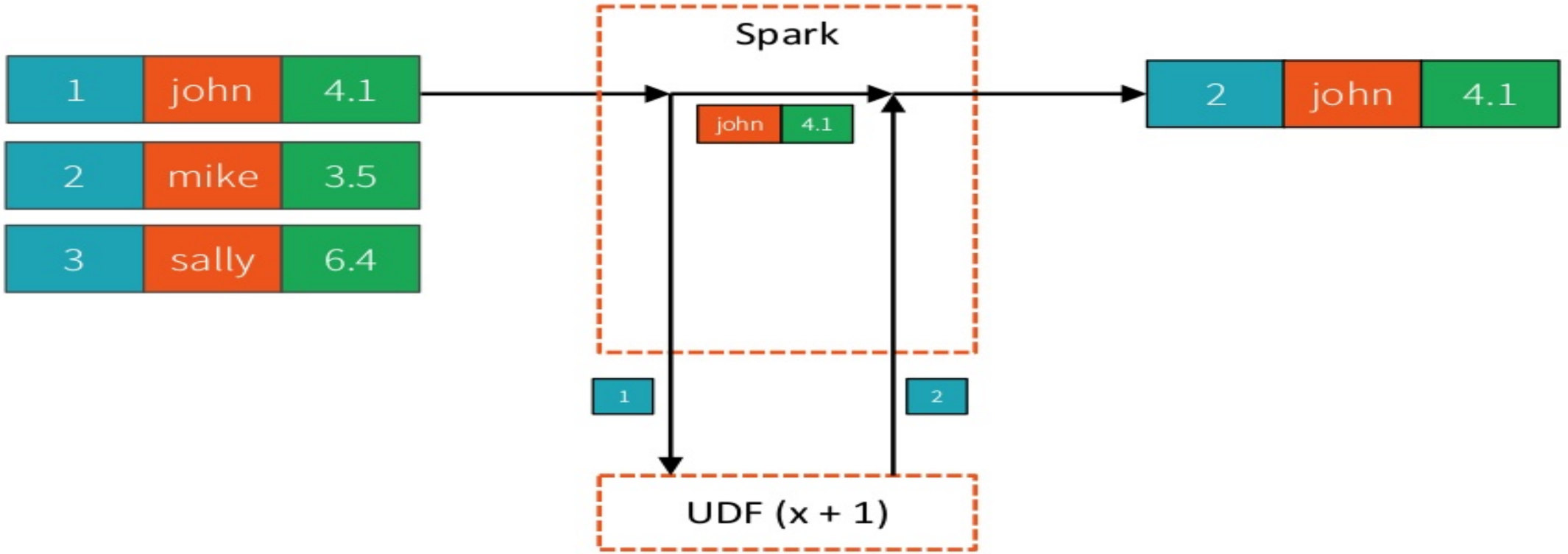
Execution Model

User-Defined Functions (UDFs)

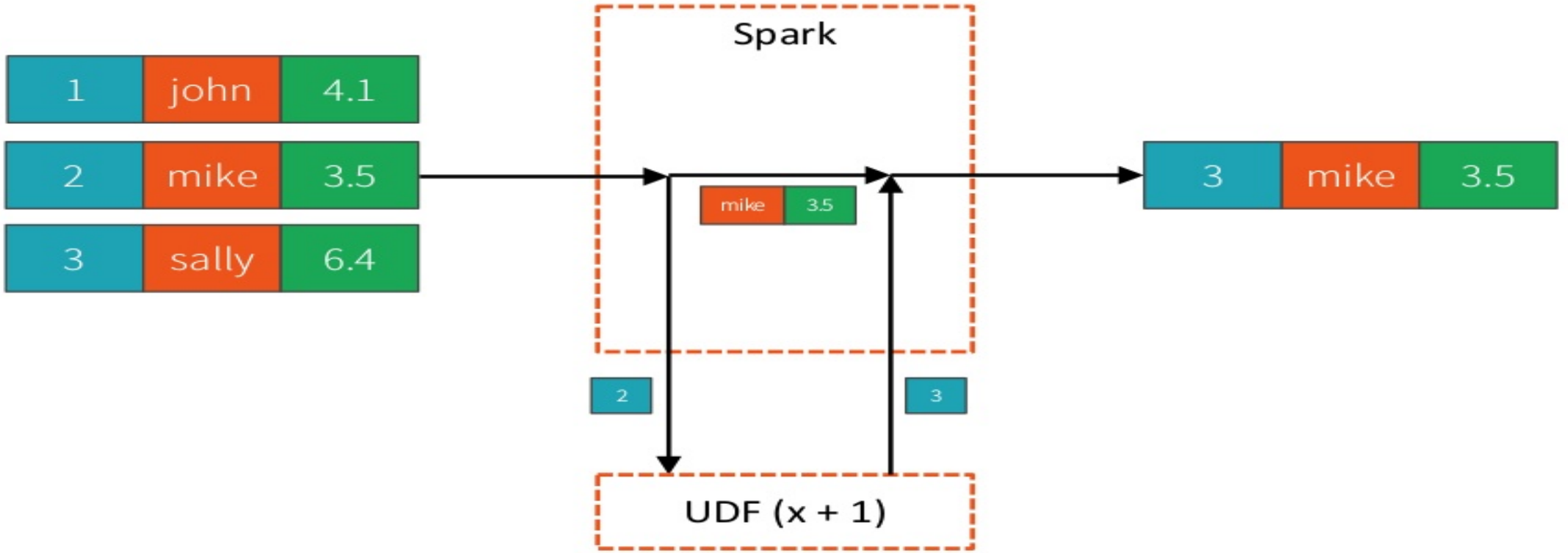
Allows executing arbitrary code, often used for integration with ML frameworks

Example: prediction on data using TensorFlow

Row-at-a-time Data Exchange



Row-at-a-time Data Exchange

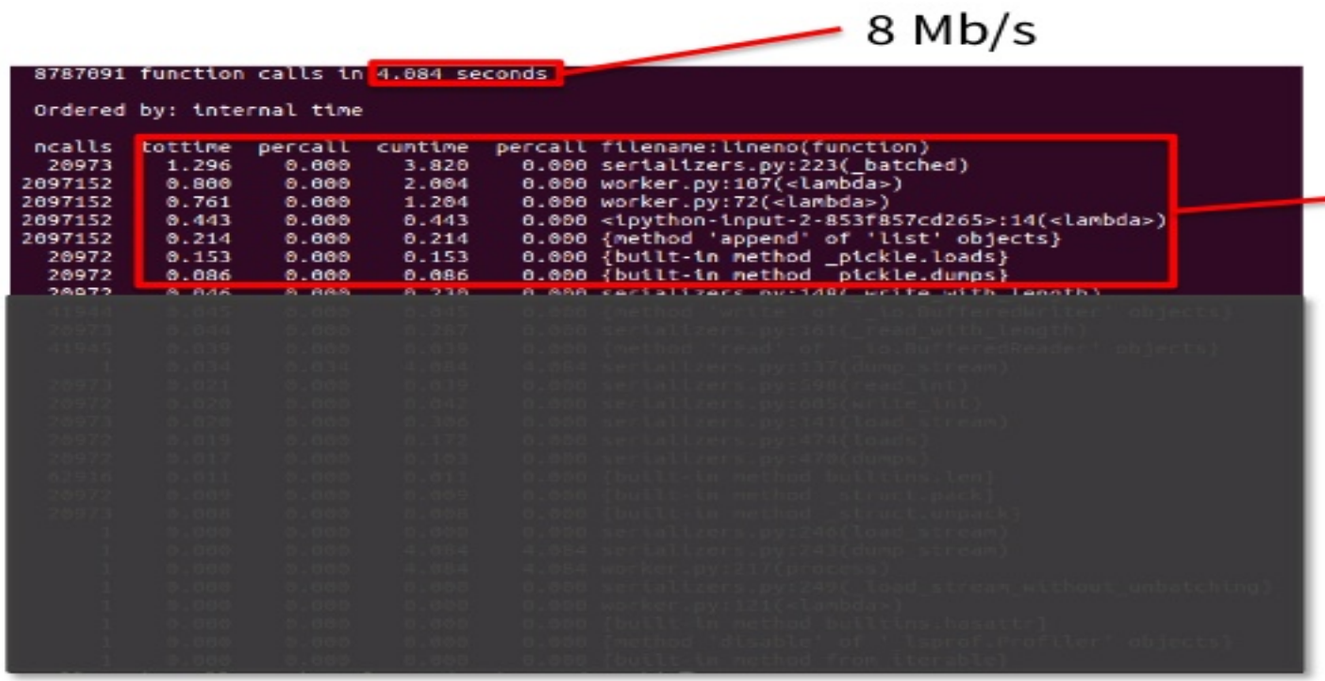


databricks

Row-at-a-time Data Exchange

Profile UDF

lambda x: x + 1



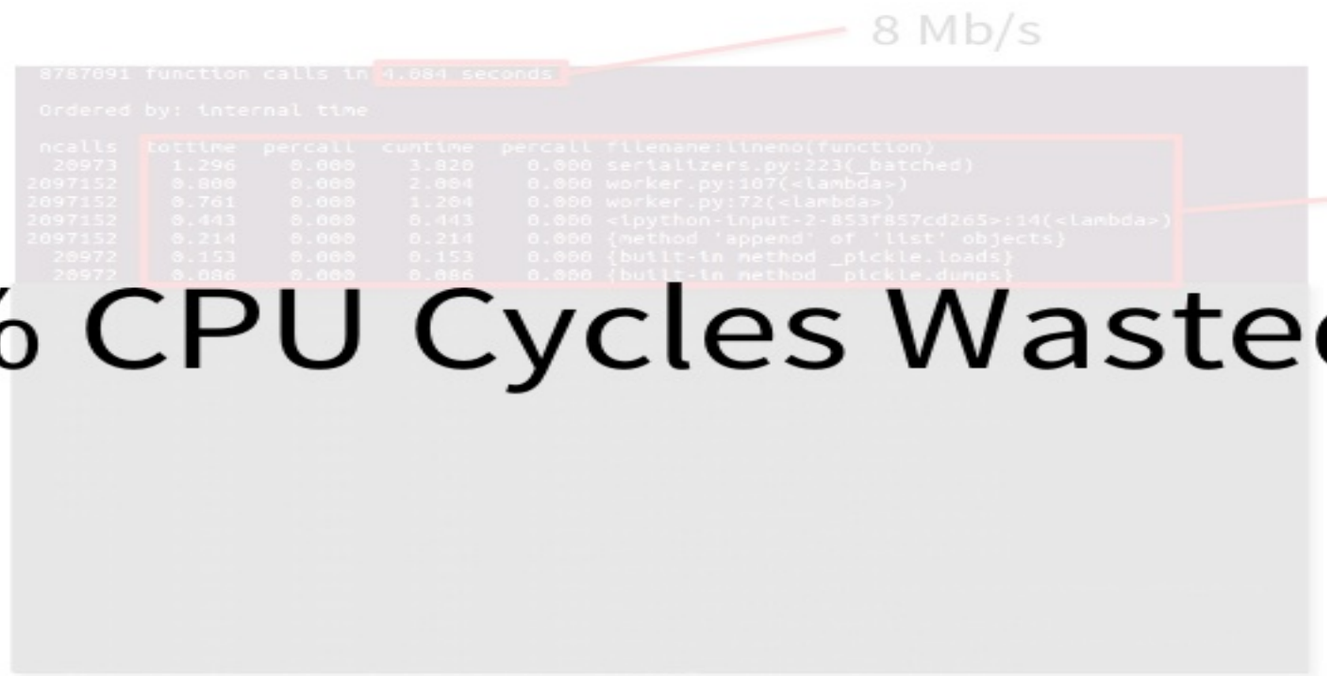
databricks

Source: Li Jin, Pandas UDF: Scalable Analysis with Python and PySpark

Row-at-a-time Data Exchange

Profile UDF

lambda x: x + 1

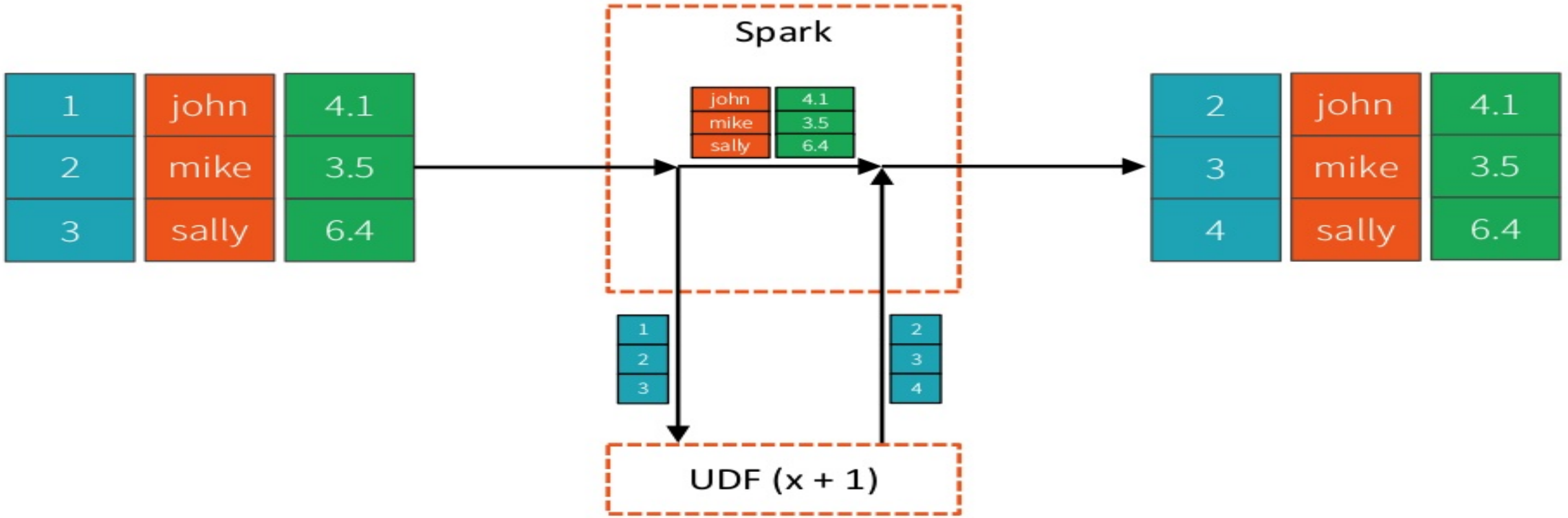


92% CPU Cycles Wasted!!!

databricks

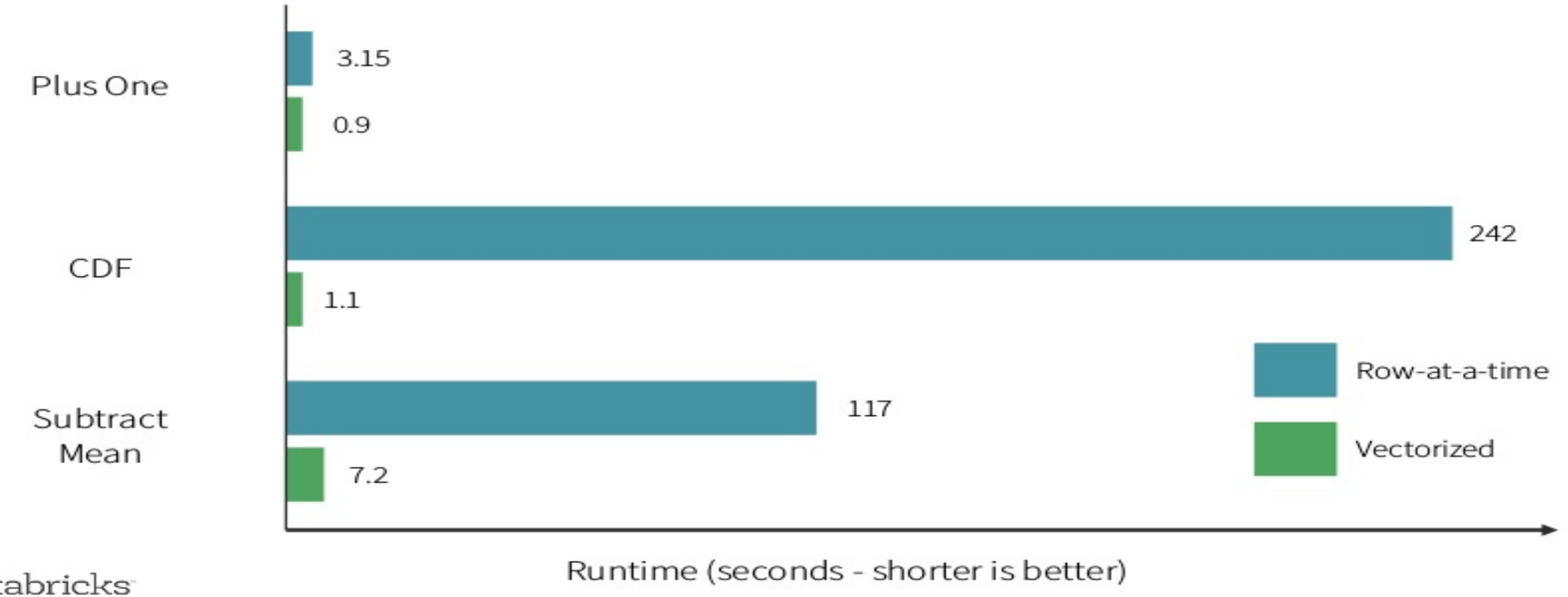
Source: Li Jin, Pandas UDF: Scalable Analysis with Python and PySpark

Vectorized Data Exchange



databricks

Performance - 3 to 240X faster



databricks

Data Exchange

Execution Model

Execution Models

Spark

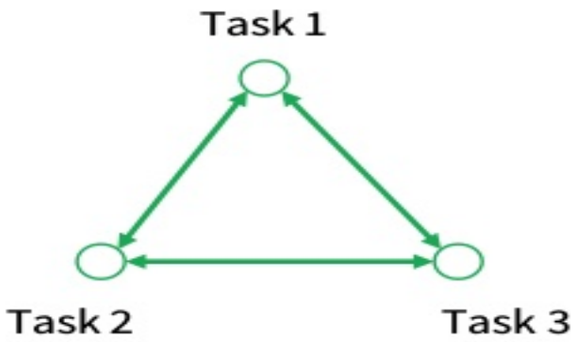
Tasks are independent of each other

Embarrassingly parallel & massively scalable

Distributed ML Frameworks

Complete coordination among tasks

Optimized for communication



What if a task *crashes*?

Spark

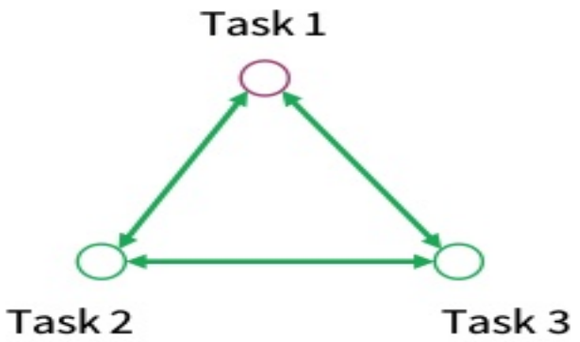
Tasks are independent of each other

Embarrassingly parallel & massively scalable

Distributed ML Frameworks

Complete coordination among tasks

Optimized for communication



Incompatible Execution Models

Spark

Tasks are independent of each other

Embarrassingly parallel & massively scalable

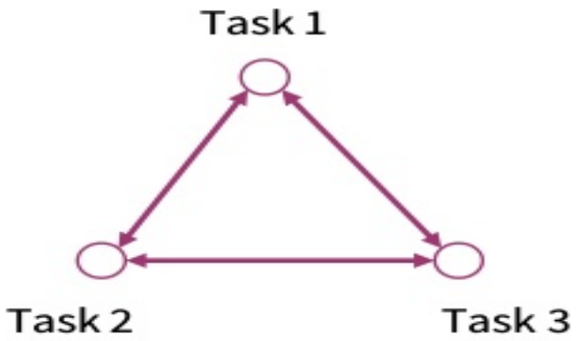
If a task crashes, rerun that one

Distributed ML Frameworks

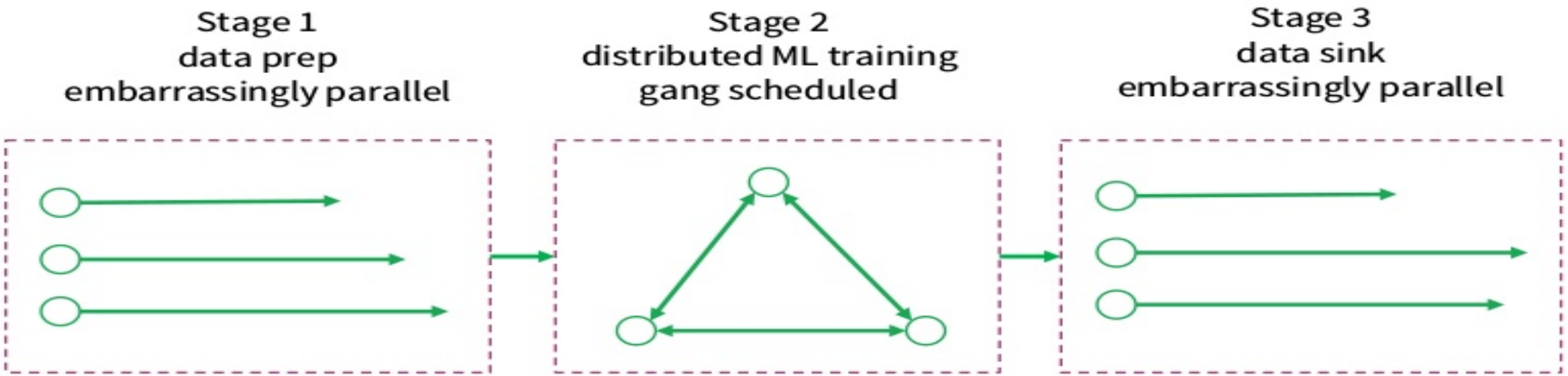
Complete coordination among tasks

Optimized for communication

If a task crashes, must rerun all tasks



Unifying Execution Models with Barrier



tasks “all or nothing”
to reconcile fundamental incompatibility
between Spark and distributed ML frameworks

Project Hydrogen

10 to 100X Faster

Data Exchange

Unify Spark + ML

Execution Model

Timeline

Spark 2.3 (Spring 2018): Basic vectorized UDFs (SPARK-21190)

Spark 2.4 (Fall 2018): Barrier scheduling (SPARK-24374), and more vectorized UDFs support (SPARK-22216)

Spark 3.0 (2019): GA and standard format for data exchange (SPARK-24579)

Session Talk Highlights

Project Hydrogen: Unifying State-of-the-Art AI and Big Data in Apache Spark

Tim Hunter, Databricks

Xiangrui Meng, Databricks

Apache Spark on K8S and HDFS Security

Ilan Filonenko, Bloomberg

Experience Of Optimizing Spark SQL When Migrating from MPP Database

Yucai Yu, eBay

Yuming Wang, eBay

databricks