



Spark + AI Summit

@rxin





Lester

\$1m **NETFLIX** Prize

anonymized movie rating dataset
best recommendation algorithm wins

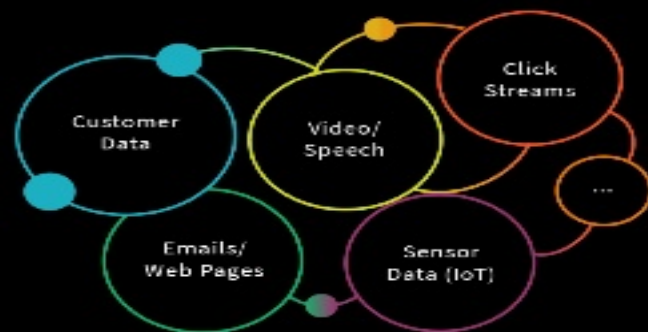


Lester

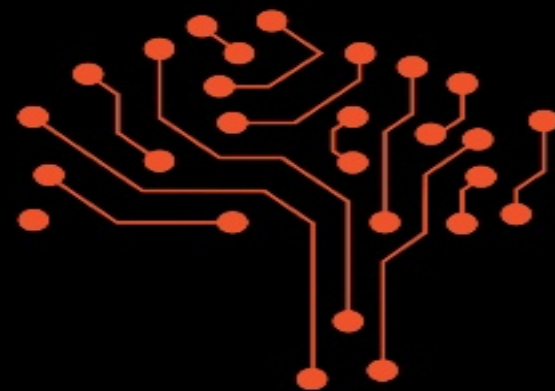


Matei

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

Showing Test Score. [Click here to show quiz score](#)

Display top leaders.

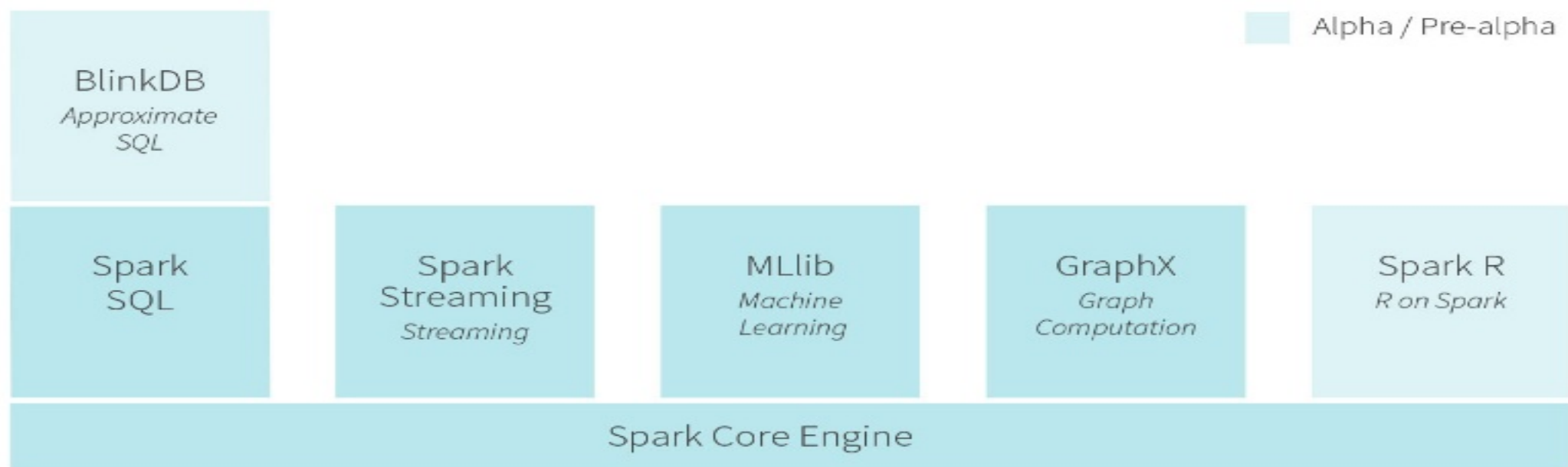
Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera 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

tied for
best score

20 mins late

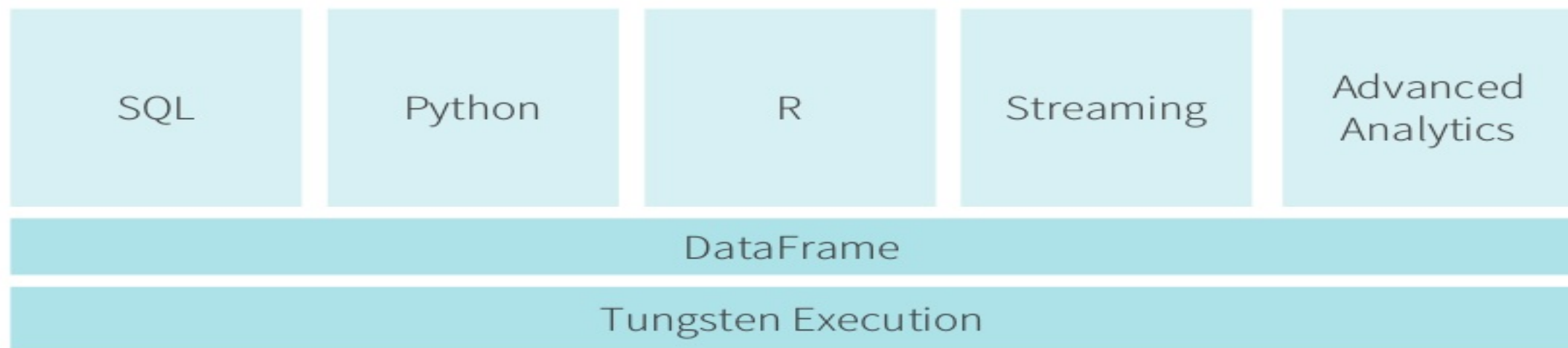


Apache Spark 1.0 (2014)



Fast and general engine for distributed data processing

DataFrame + Tungsten (2015)



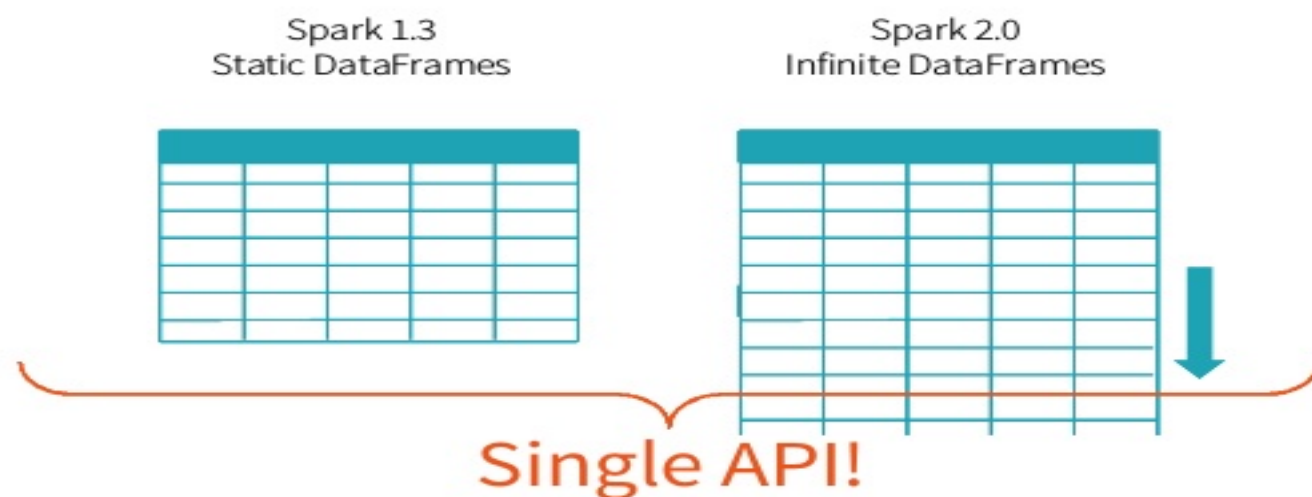
“Apache Spark is the Taylor Swift
of big data software.”

- Derrick Harris, Fortune, Sep 2015



Structured Streaming (2016)

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



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

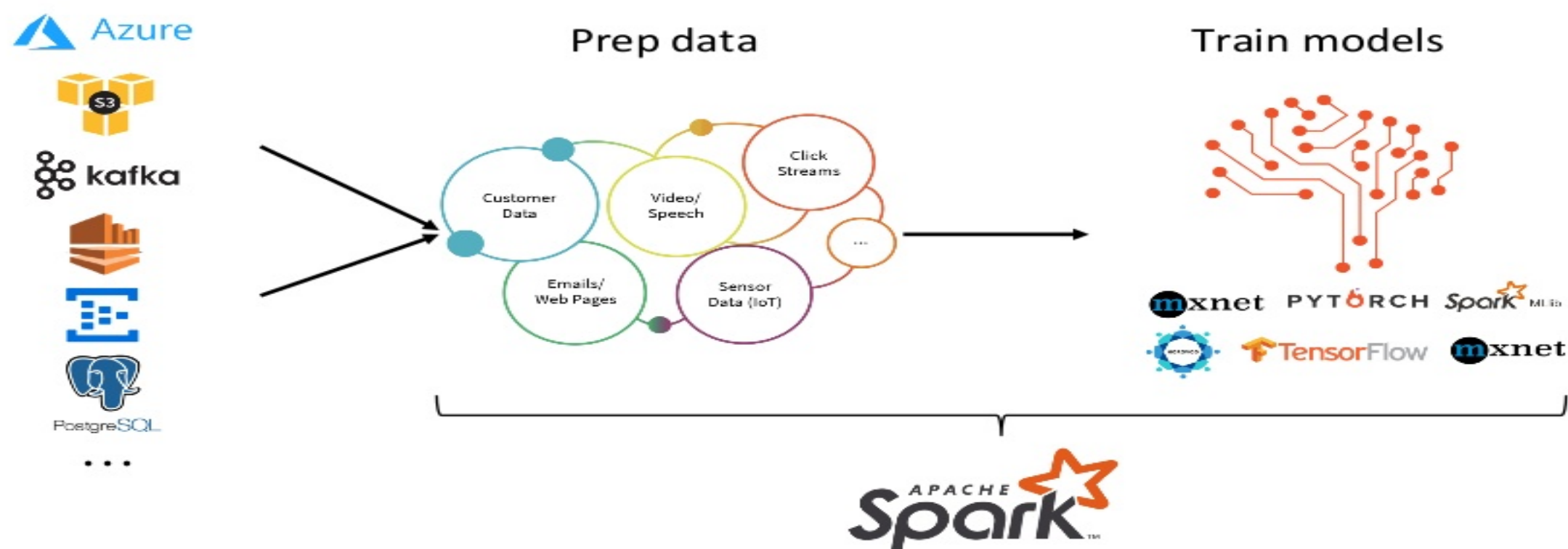
Explosion of ML Frameworks



...



Embracing ML ecosystem as 1st-class citizens



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



Data Exchange

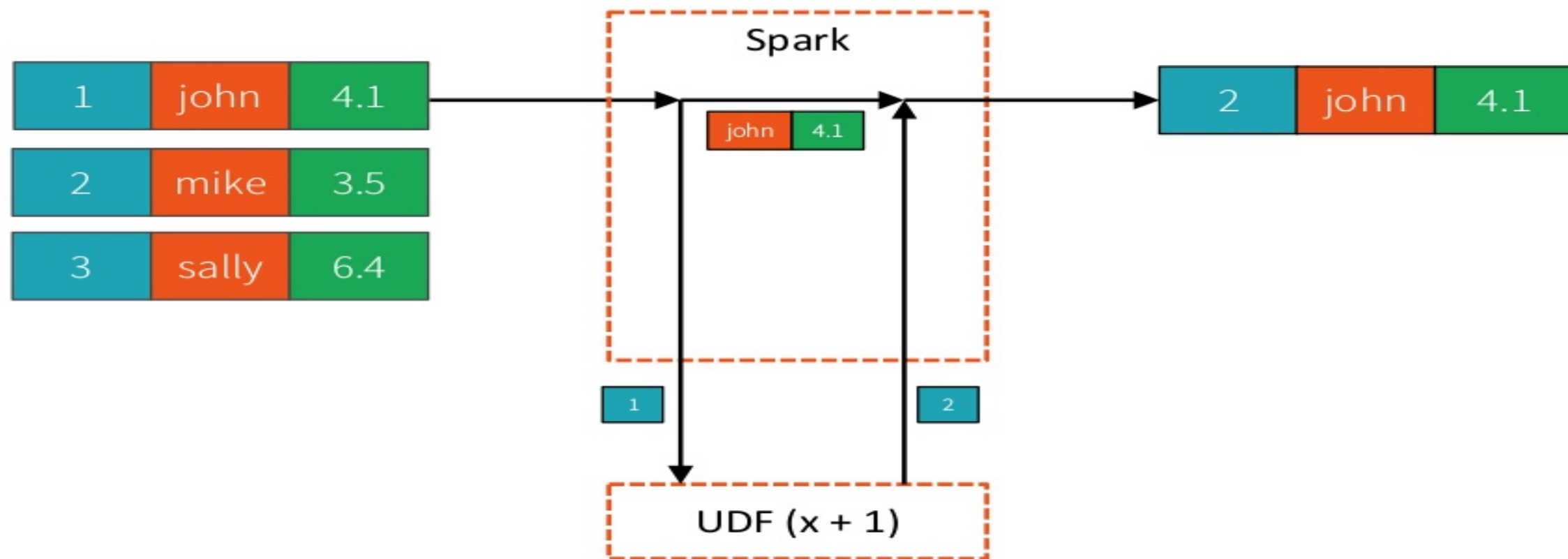
Execution Model

User-Defined Functions (UDFs)

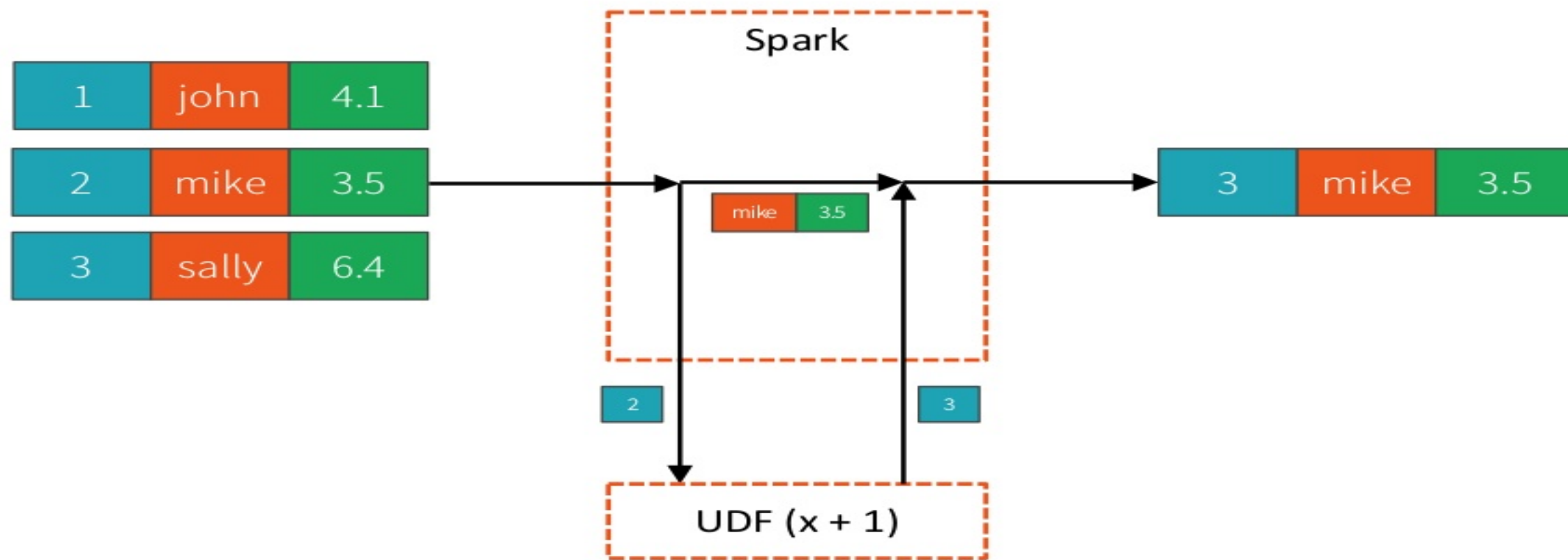
Allows executing arbitrary code, often used for integration with ML frameworks

Example: prediction on data using TensorFlow

Row-at-a-time Data Exchange



Row-at-a-time Data Exchange



Row-at-a-time Data Exchange

Profile UDF

lambda x: x + 1

8787091 function calls in 4.084 seconds

8 Mb/s

Ordered by: internal time

ncalls	tottime	percall	cuntime	percall	filename:lineno(function)
20973	1.296	0.000	3.820	0.000	serializers.py:223(<batched>)
2097152	0.000	0.000	2.004	0.000	worker.py:107(<lambda>)
2097152	0.761	0.000	1.204	0.000	worker.py:72(<lambda>)
2097152	0.443	0.000	0.443	0.000	<ipython-input-2-853f857cd265>:14(<lambda>)
2097152	0.214	0.000	0.214	0.000	{method 'append' of 'list' objects}
20972	0.153	0.000	0.153	0.000	{built-in method _pickle.loads}
20972	0.086	0.000	0.086	0.000	{built-in method _pickle.dumps}
20972	0.046	0.000	0.046	0.000	serializers.py:148(write_with_length)
20972	0.045	0.000	0.045	0.000	{method 'write' of '_io.BufferedWriter' objects}
20973	0.044	0.000	0.044	0.000	serializers.py:161(read_with_length)
41945	0.039	0.000	0.039	0.000	{method 'read' of '_io.BufferedReader' objects}
1	0.034	0.000	0.034	0.000	serializers.py:137(dump_stream)
20973	0.021	0.000	0.019	0.000	serializers.py:598(read_int)
20972	0.020	0.000	0.042	0.000	serializers.py:605(write_int)
20973	0.020	0.000	0.300	0.000	serializers.py:141(load_stream)
20972	0.019	0.000	0.172	0.000	serializers.py:474(load)
20972	0.017	0.000	0.103	0.000	serializers.py:478(dump)
82310	0.011	0.000	0.011	0.000	{built-in method builtins.len}
20972	0.009	0.000	0.009	0.000	{built-in method struct.pack}
20973	0.008	0.000	0.008	0.000	{built-in method struct.unpack}
1	0.008	0.000	0.008	0.000	serializers.py:246(load_stream)
1	0.008	0.000	4.004	4.004	serializers.py:243(dump_stream)
1	0.008	0.000	4.004	4.004	worker.py:217(process)
1	0.008	0.000	0.008	0.000	serializers.py:249(load_stream_without_unbatching)
1	0.008	0.000	0.008	0.000	worker.py:121(<lambda>)
1	0.008	0.000	0.008	0.000	{built-in method builtins.hasattr}
1	0.008	0.000	0.008	0.000	{method 'disable' of 'cProfile.Profile' objects}
1	0.008	0.000	0.008	0.000	{built-in method __iter__ of 'list' objects}

92% in data exchange

Row-at-a-time Data Exchange

Profile UDF

lambda x: x + 1

8787091 function calls in 4.084 seconds

Ordered by: internal time

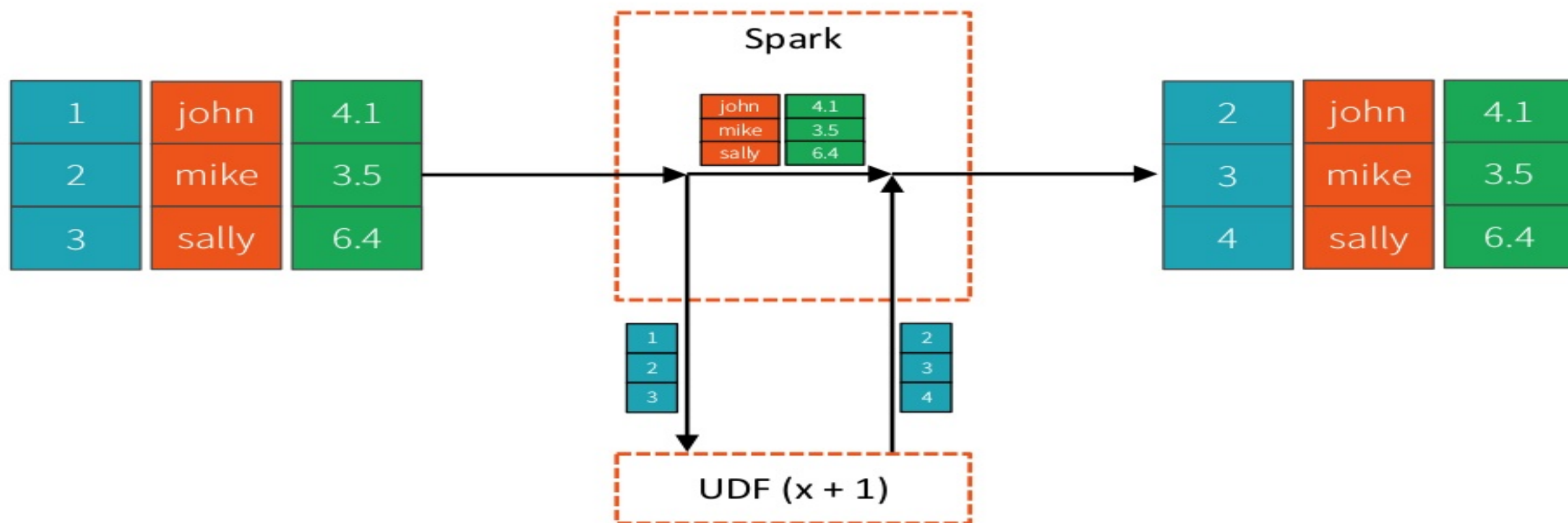
ncalls	tottime	percall	cuntime	percall	filename:lineno(function)
20973	1.296	0.000	3.820	0.000	serializers.py:223(_batched)
2097152	0.000	0.000	2.004	0.000	worker.py:107(<lambda>)
2097152	0.761	0.000	1.204	0.000	worker.py:72(<lambda>)
2097152	0.443	0.000	0.443	0.000	<ipython-input-2-853f857cd265>:14(<lambda>)
2097152	0.214	0.000	0.214	0.000	{method 'append' of 'list' objects}
20972	0.153	0.000	0.153	0.000	{built-in method _pickle.loads}
20972	0.086	0.000	0.086	0.000	{built-in method _pickle.dumps}

8 Mb/s

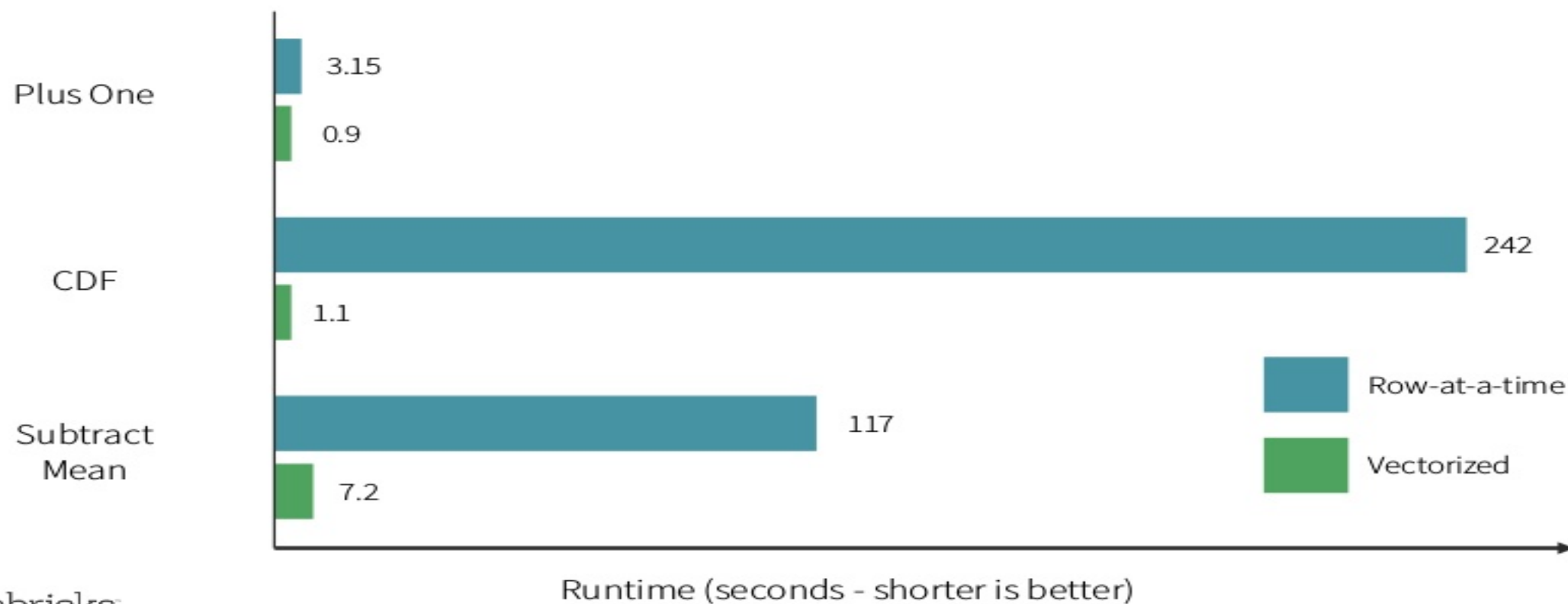
92% in
data exchange

92% CPU Cycles Wasted!!!

Vectorized Data Exchange



Performance - 3 to 240X faster



Data Exchange

Execution Model

Execution Models

Spark

Tasks are independent of each other

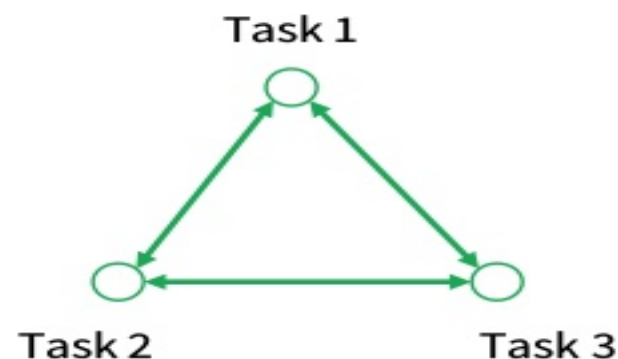
Embarrassingly parallel & massively scalable



Distributed ML Frameworks

Complete coordination among tasks

Optimized for communication



What if a task *crashes*?

Spark

Tasks are independent of each other

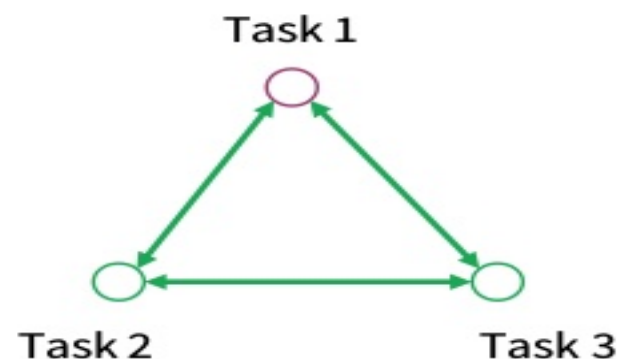
Embarrassingly parallel & massively scalable



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

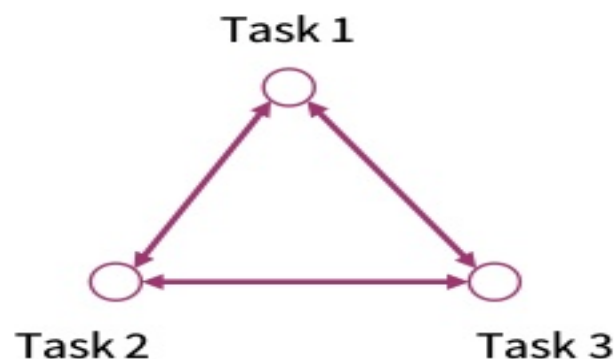


Distributed ML Frameworks

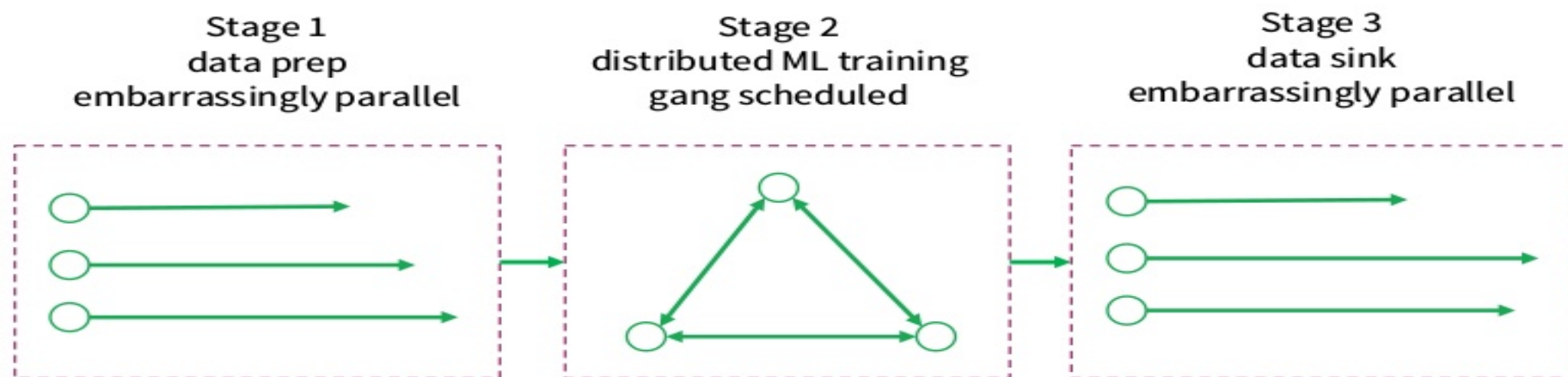
Complete coordination among tasks

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