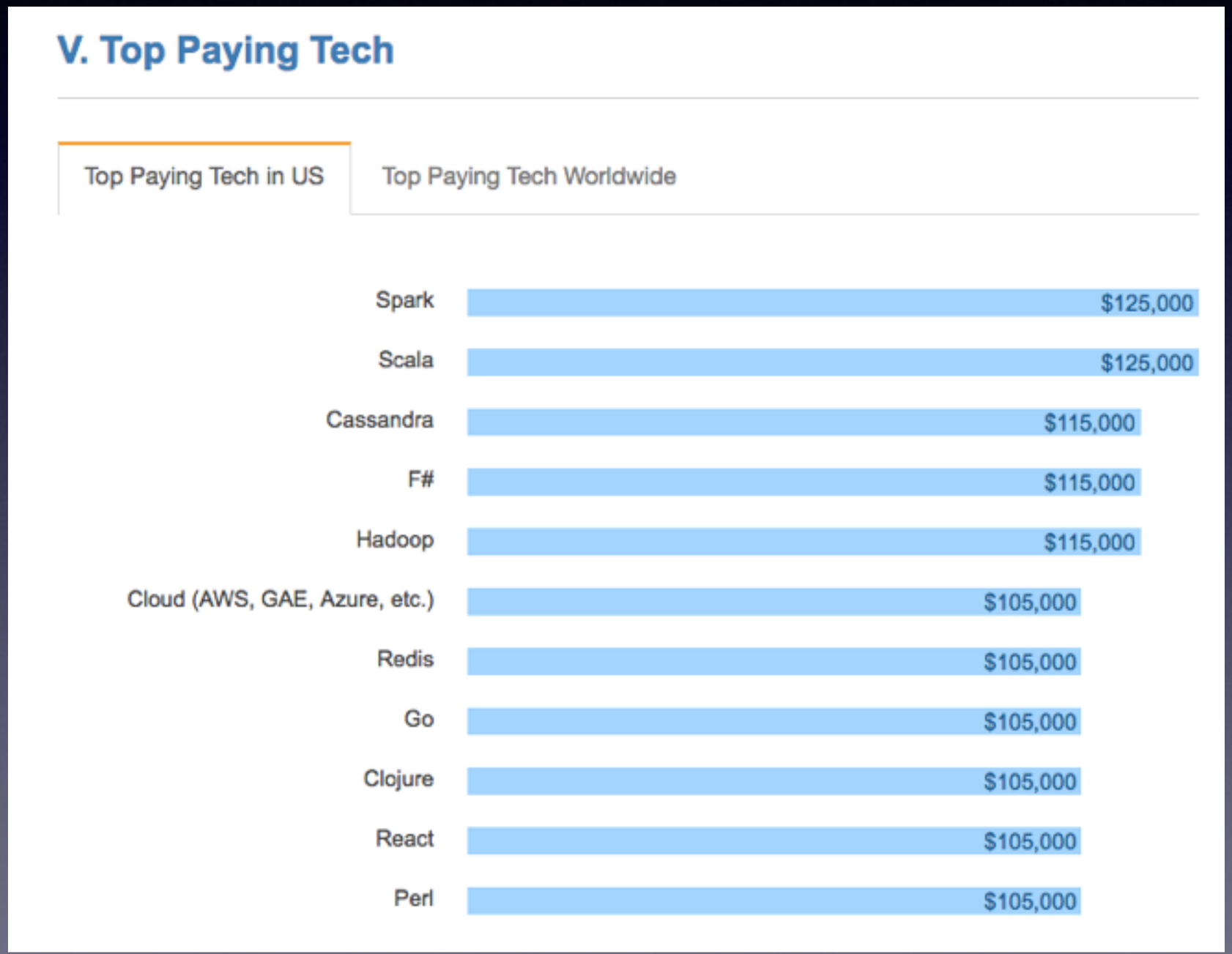
Spark 2.0

陈 超@CrazyJvm
work@qiniu
2016/06/05

Q: What is Spark?

A: Spark is a fast and general engine for large-scale data processing.

Spark == well paid



What's New

- Unifying Datasets & DataFrames
- Structured Streaming
- Tungsten Phase 2

Unifying Datasets & DataFrames

DataFrame & Dataset

- DataFrames: collections of rows with a schema
- Datasets: add static types

DataFrame & Dataset

- Running based on Tungsten
- Spark 2.0 : DataFrame = Dataset[Row]
- Basic API for other components

```
type DataFrame = Dataset[Row]
```

	SQL	DataFrame	DataSet
Syntax Errors	Runtime	Compile Time	Compile Time
Analysis Errors	Runtime	Runtime	Compile Time

DataFrames, Datasets and SQL share the same optimization/execution pipeline

what about RDD

- remain as low-level API
- New RDD: Dataset & DataFrame

New Entry Point

- SparkSession = SparkContext for DF&DS

```
SparkSession.builder()
  .master(“local”)
  .appName("Word Count”)
  .config("spark.some.config.option", “some-value")
  .getOrCreate()
```

Structured Streaming

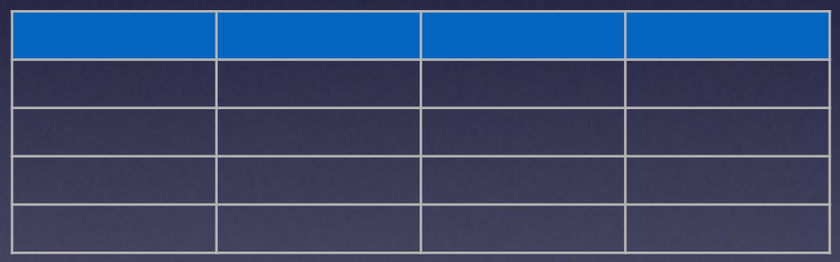
Streaming

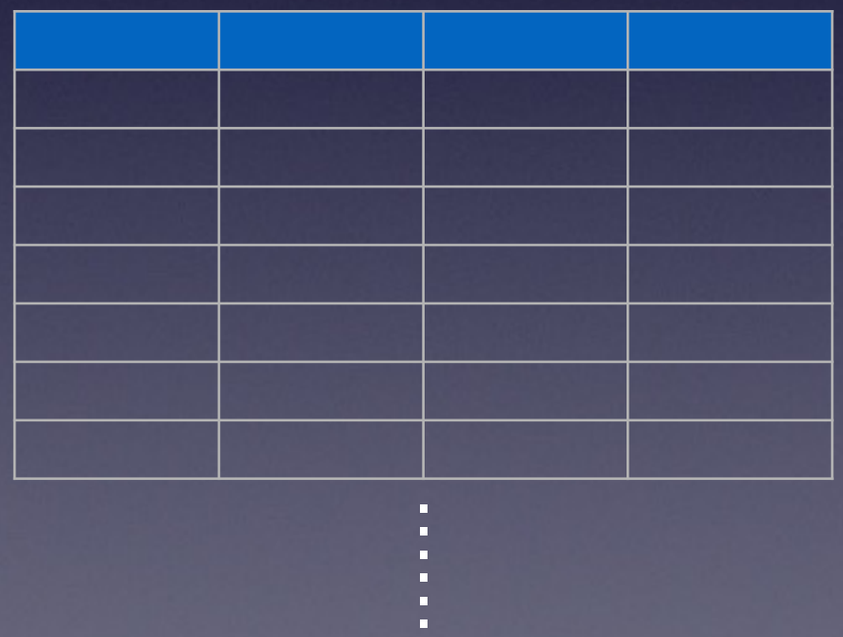
- Spark Streaming
- Flink
- Storm/JStorm
- Heron
- ……

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

how to achieve

- Static DataFrame ======> Infinite DataFrame





share the same API

show me the code

batch

```
logs = ctx.read.format(“json”).open(“…”)

logs.groupBy(logs.user_id).agg(sum(logs.time))
  .write.format(“jdbc”)
  .save(“jdbc:mysql//…”)
```

streaming

```
logs = ctx.read.format(“json”).stream(“…”)

logs.groupBy(logs.user_id).agg(sum(logs.time))
  .write.format(“jdbc”)
  .startStream(“jdbc:mysql//…”)
```

why there’s no interval in streaming code?

Structured Streaming

- High level streaming API built on Spark SQL engine
  - Declarative API that extends DataFrame/Datasets
  - Event time, windowing, sessions, source & sink
- Support interactive & batch queries
  - Aggregate data in a stream, then serve using JDBC
  - Change queries at runtime
  - Build and apply ML models

Structured Streaming

- Input source
- Query
- Trigger
- Output mode

Output mode

- Append
- Delta
- Update-in-place
- Complete

```
sealed trait OutputMode

case object Append extends OutputMode
case object Update extends OutputMode
```

Execution

- Run queries in the way of streaming
- Differences between Spark and Flink?※

Tungsten Phase 2

Phase 1

- Perform manual memory management instead of relying on Java objects
- Code generation for expression evaluation
- cache conscious sorting

CPU cycles waste

- Making virtual function calls
- R/W intermediate data to CPU cache or memory

Optimize

- Emit optimized bytecode at runtime in order to collapse the entire query into a single function
- Eliminating virtual function calls
- Leveraging CPU registers for intermediate data.

how to achieve

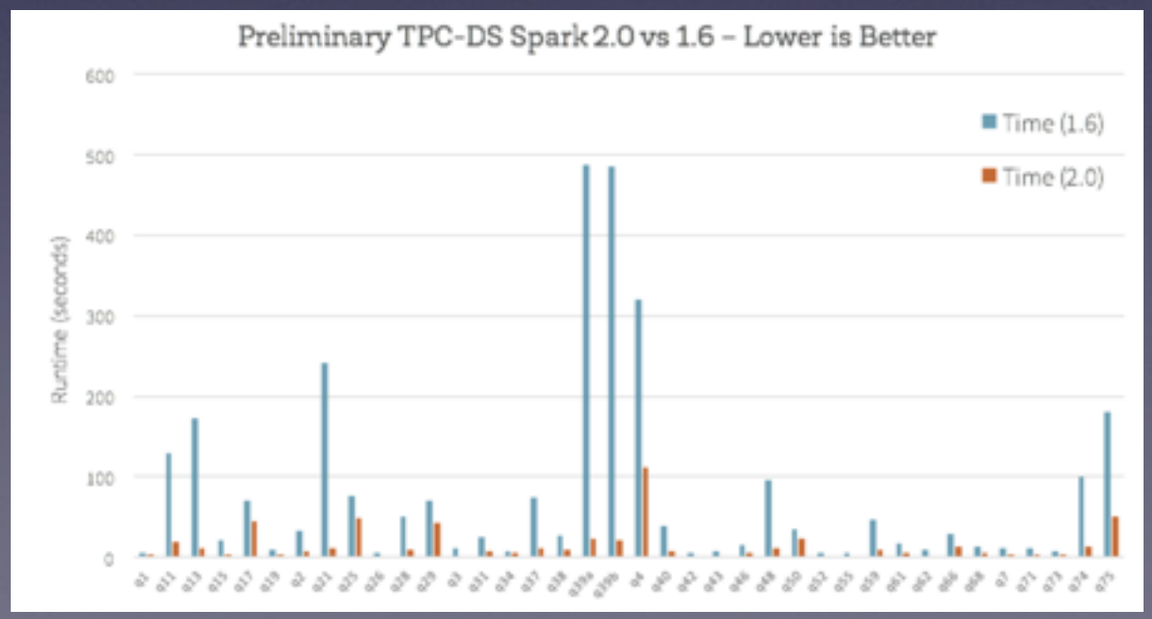
- Whole-stage code generation
- Vectorization

Benchmark

process a row on one core for some of the operators in Spark 1.6 vs. Spark 2.0
cost per row (single thread)

PRIMITIVE	SPARK 1.6	SPARK 2.0
filter	15ns	1.1ns
sum w/o group	14ns	0.9ns
sum w/ group	79ns	10.7ns
hash join	115ns	4.0ns
sort (8-bit entropy)	620ns	5.3ns
sort (64-bit entropy)	620ns	40ns
sort-merge join	750ns	700ns

TPC-DS



Others

ML pipeline API

- DataFrame-based Machine Learning API emerges as the primary ML API

ML pipeline persistence

- Machine learning pipeline persistence: Users can now save and load machine learning pipelines and models across all programming languages supported by Spark.

We are hiring!

The End

Thanks