

@DEANWAMPLER

CHICAGO SPARK, MAY 18, 2016

---

**SPARK 2.0**

# MEETUPS

- ▶ We need speakers and topics for future meetings!
- ▶ Thanks to Expedia for hosting...
  - ▶ ... and congratulations on your 20 year anniversary!

# GOTO CHICAGO IS NEXT WEEK!!

▶ <http://gotocon.com/>

# SPARK SUMMIT SF, JUNE 6-8

- ▶ [spark-summit.org/2016/](http://spark-summit.org/2016/)
- ▶ Discount codes:
  - ▶ Meetup16SF N% (?)
  - ▶ reynold16 20%

### ABOUT YOU...

- ▶ Who's looking?
- ▶ Who's hiring?
  
- ▶ What topics should we cover soon?



SPARK 2.0.0

---

**A NEW HOPE?**

## LIBERALLY BORROWED FROM...

- ▶ <http://go.databricks.com/apache-spark-2.0-presented-by-databricks-co-founder-reynold-xin>
- ▶ You should really view this webinar...

## WHAT DOES A 2.0.0 RELEASE MEAN?

- ▶ Some new and restructured APIs.
- ▶ Some breaking API changes?
  - ▶ They try *very hard* to avoid changing user-visible APIs.
  - ▶ Some dependency changes, e.g., Guava.
  - ▶ Changes in experimental APIs (e.g., Datasets).



## THREE MAJOR CHANGES

- ▶ Tungsten Phase 2
  - ▶ 5x-20x additional performance improvements.
- ▶ SQL 2003 and Unified DataFrames/Datasets API.
- ▶ Structured Streaming
  - ▶ Integration of DataFrames/Datasets and Streaming.



TUNGSTEN

---

**PHASE 2**

## FOR AN ADDITIONAL 10X IMPROVEMENT...

- ▶ Removing hot spots only gains a few %.
- ▶ Ask instead, if we start from scratch, what's the fastest it could be?

```
select count(*) from sales
where sku = 1234;
```

```
int count = 0;
for (Record sale: sales) {
    if (sale.sku == 1234)
        count += 1;
}
```

# FOR AN ADDITIONAL 10X IMPROVEMENT...

- ▶ Instead, databases (and Spark) use the “volcano pattern”, where each filter is actually an iterators.
  - ▶ Infrastructure adds overhead, easily 10x.

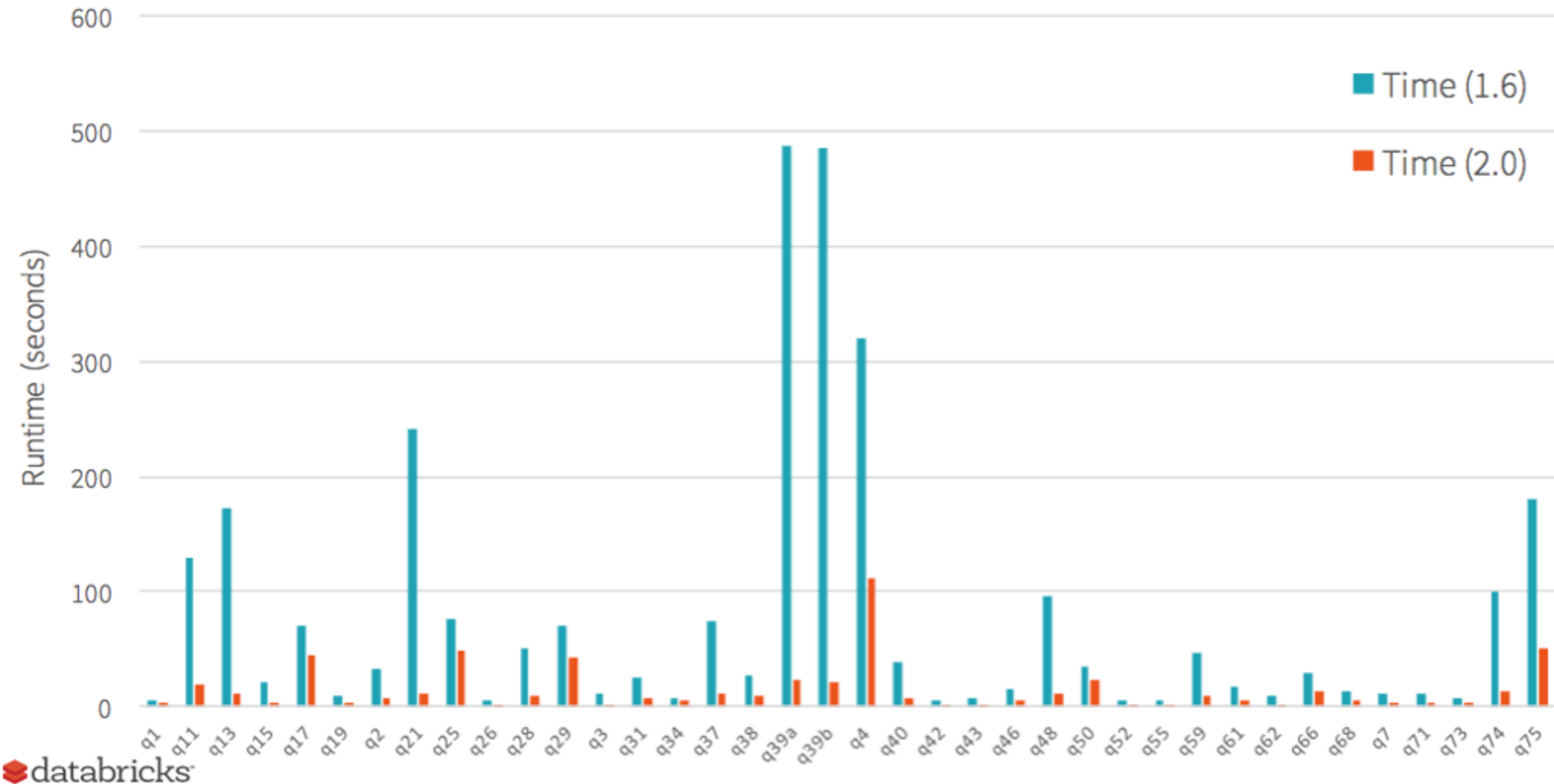
## FOR AN ADDITIONAL 10X IMPROVEMENT...

- ▶ **After** compiling the query, generate custom code!
  - ▶ No virtual function calls.
  - ▶ Put data in CPU registers, when possible.
  - ▶ Unroll loops.
  - ▶ Exploit parallelism and pipelining.

# RESULTS

Operation	Spark 1.6	Spark 2.0
filter	15ns	1.1ns
sum with/without group by	79/14ns	10.7/0.9ns
hash join	115ns	4.0ns
sort (8/16 bit entropy)	620/620ns	5.3/8.0ns
sort-merge join	750ns	700ns

# PRELIM. TPC-DS BENCHMARK (LOWER IS BETTER)



# EXAMPLE

- ▶ <http://bit.ly/1X8LKmH>
- ▶ (Example from Databricks Cloud)





**DATASET, DATAFRAME**

---

**SQL**

## SQL 2003 COMPLIANCE

- ▶ Spark SQL now supports all 99 TPC-DS benchmark queries.
- ▶ New SQL parser (with better error messages).
- ▶ Also
  - ▶ Subqueries, correlated and uncorrelated.
  - ▶ Approximate aggregate statistics.

## DATASETS VS. DATAFRAMES

- ▶ 2015: Datasets bring *field* type safety back to DataFrames.
  - ▶ Like static type safety provided by the RDD API:
    - ▶ `Dataset[T]` analogous to `RDD[T]`, where `T` is the record type.
    - ▶ In DataFrames, fields (columns) untyped => `Row` type.
- ▶ Now: `DataFrame = Dataset[Row]`

## SPARKSESSION

- ▶ `SparkSession`: The new “`SparkContext`” for `DataFrame/Dataset`.
- ▶ Entry point for ingesting data (like `SQLContext` was).
- ▶ Metadata, configuration, and cluster resource management.

## FUTURE?

- ▶ **RDD**: Will remain the low-level API.
- ▶ But **Dataset/DataFrame** will be the focus of optimizations, rich semantics.
- ▶ Higher-level libraries will be migrated to **Dataset/DataFrame**:
  - ▶ Structured streaming
  - ▶ ML pipeline replacing MLlib
  - ▶ GraphFrames

## OTHER, MISCELLANEOUS API IMPROVEMENTS

- ▶ ML pipeline coverage in all languages (Java, Python, R, as well as Scala) nearly complete.
- ▶ Improved R support:
  - ▶ Parallelizable user-defined functions in R.
  - ▶ More Models!
    - ▶ Generalized Linear Models (GLMs), Naïve Bayes, Survival Regression, K-Means.

# EXAMPLES

- ▶ <http://bit.ly/1SMPEzQ>
- ▶ <http://bit.ly/1OeqdSn>
- ▶ (Examples from Databricks Cloud)



STRUCTURED

---

**STREAMING**



### WHY STREAMING?

- ▶ Spark started as a batch mode system, but...
  - ▶ ... the mini batch model lets Spark capture data in small time windows and run batch jobs over it.
- ▶ This also lets you use business logic in both batch and streaming contexts, which is great.
- ▶ If you are not latency sensitive, mini batch lets you do fancy things:
  - ▶ “Online” training of ML models, track moving state, run SQL queries, ...

# STREAMING IS HARD

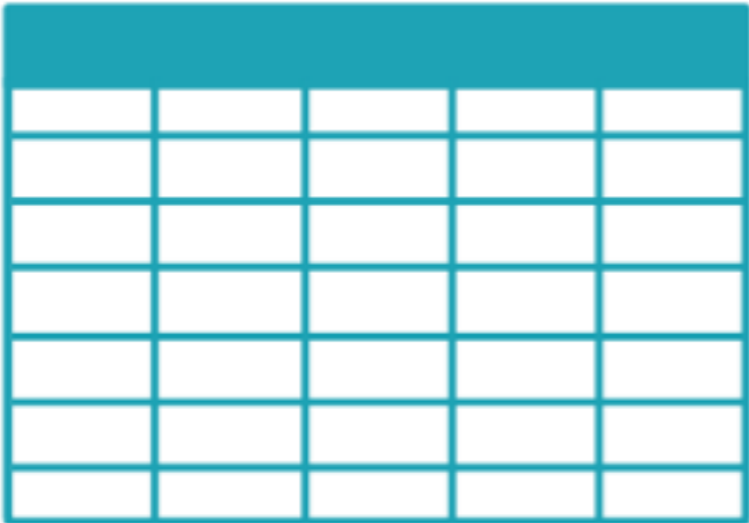
- ▶ I'm counting sales/hour, but some messages get delayed for several hours. Now what?
- ▶ Event time vs. processing time.
- ▶ What does group by or join mean in a streaming context?
- ▶ If I'm maintaining stream state and it crashes, how do I recover?

# STREAMING IS HARD

- ▶ Semantics are nontrivial:
  - ▶ <https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-101>
  - ▶ <https://www.oreilly.com/ideas/the-world-beyond-batch-streaming-102>
  - ▶ Excellent [Kafka Summit talk](#) by Frances Perry and Tyler Akidau
  - ▶ ... and others.

# GOAL: ELIMINATE THE NEED TO REASON ABOUT STREAMING

Spark 1.3  
Static DataFrames







Spark 2.0  
Infinite DataFrames







Single API !

# CLASSIC BATCH JOB

```
val logs = ctx.read.format("json").  
  open("s3://mybucket/logs")
```

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

### ... CONVERTED TO A CONTINUOUS AGGREGATION

```
val logs = ctx.read.format("json").  
  stream("s3://mybucket/logs")
```

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

- ▶ `open` changed to `stream`.
- ▶ `save` changed to `startStream`.

# SO, STRUCTURED STREAMING IS...

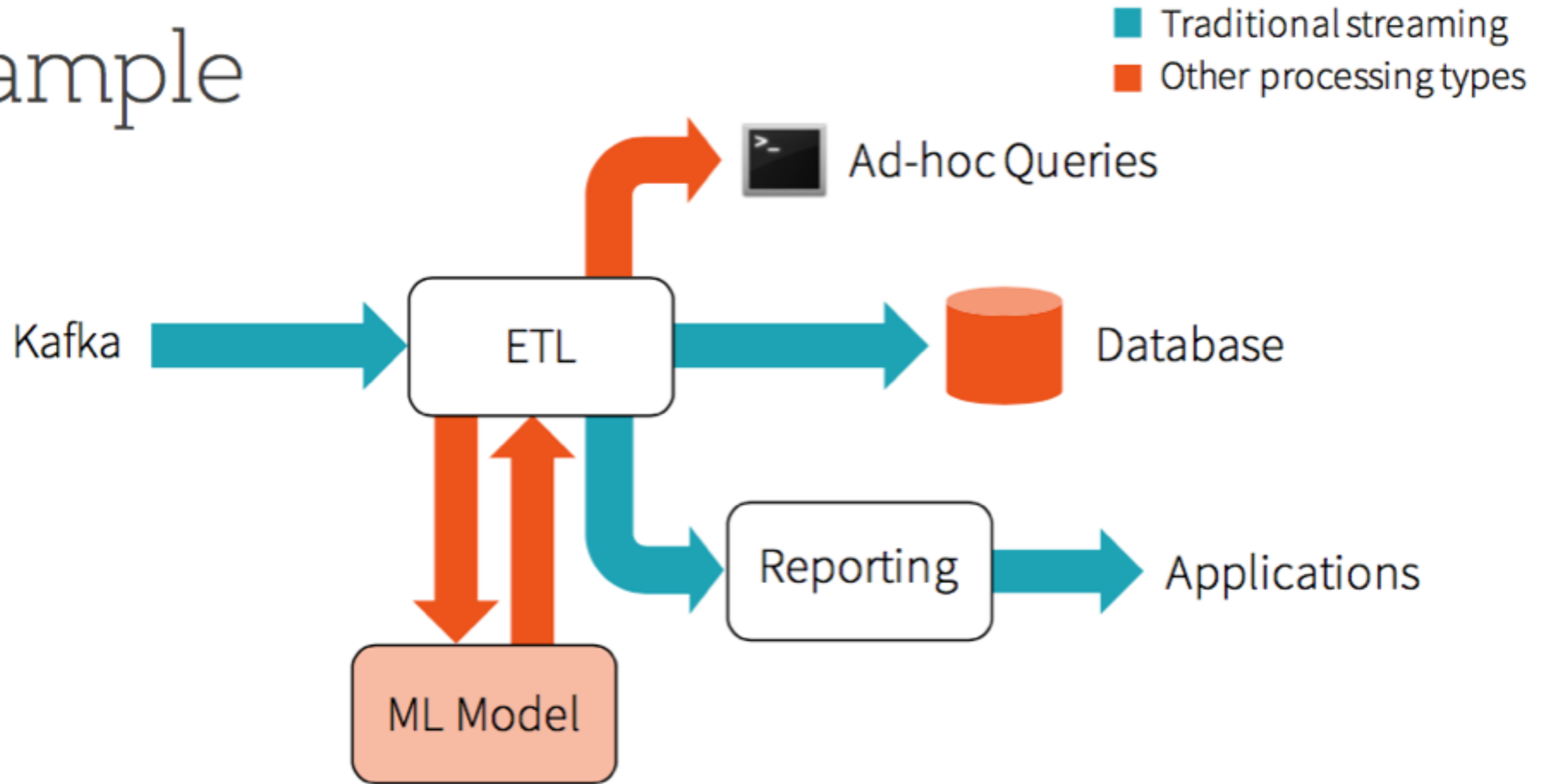
- ▶ High-level streaming API built on SparkSQL engine, rather than RDDs.
- ▶ Supports:
  - ▶ Event time vs. processing time.
  - ▶ Windowing.
  - ▶ Sessions.
  - ▶ Misc. sources and sinks.

# SO, STRUCTURED STREAMING IS...

- ▶ Queries:
  - ▶ The usual SQL-like aggregations, etc.
  - ▶ Query the stream state using JDBC.
  - ▶ Change queries at runtime.
  - ▶ Build and apply machine learning models.
- ▶ Build “continuous applications”.



# Example



Goal: end-to-end continuous applications

### FUTURE?

- ▶ There is talk of rewriting Spark Streaming to be a “true” streaming engine.
  - ▶ Low latency.
  - ▶ Full support for common streaming semantics
    - ▶ (i.e., as discussed in the Tyler Akidau blogs.)
- ▶ Needs to stay competitive with [Apache Beam](#), [Flink](#), and [Gearpump](#).



FOR MORE

---

**INFORMATION**

## RESOURCES

- ▶ Structured Streaming Strata Talk
  - ▶ <https://www.oreilly.com/learning/apache-spark-2-0--introduction-to-structured-streaming>
- ▶ 2.0 Preview:
  - ▶ code: <http://home.apache.org/~pwendell/spark-releases/spark-2.0.0-preview-bin/>
  - ▶ docs: <http://home.apache.org/~pwendell/spark-releases/spark-2.0.0-preview-docs/>
- ▶ Databricks Cloud Preview.

## RESOURCES

- ▶ [lightbend.com/fast-data](http://lightbend.com/fast-data)
- ▶ [dean.wampler@lightbend.com](mailto:dean.wampler@lightbend.com)
- ▶ [@deanwampler](https://twitter.com/deanwampler)
- ▶ [polyglotprogrammin.com/talks](http://polyglotprogrammin.com/talks)
  
- ▶ Questions?