Trends for Big Data and Apache Spark in 2017

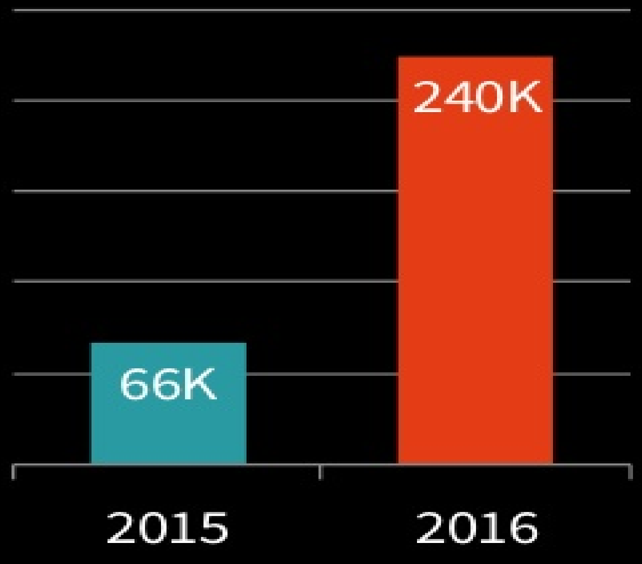
Matei Zaharia
@matei_zaharia

databricks

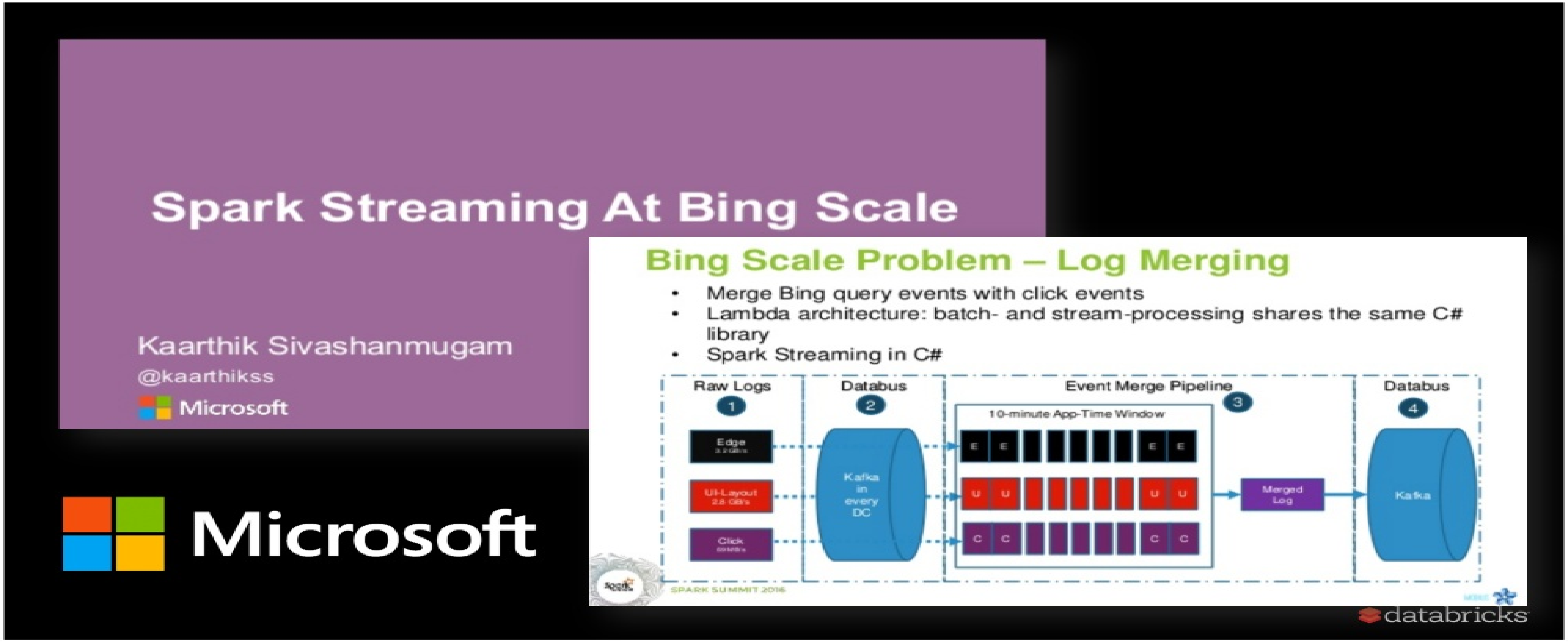
2016: A Great Year for Spark

- Spark 2.0: stabilizes structured APIs, 10x speedups, SQL 2003 support
- Structured Streaming
- 3.6x growth in meetup members

Spark Meetup Members



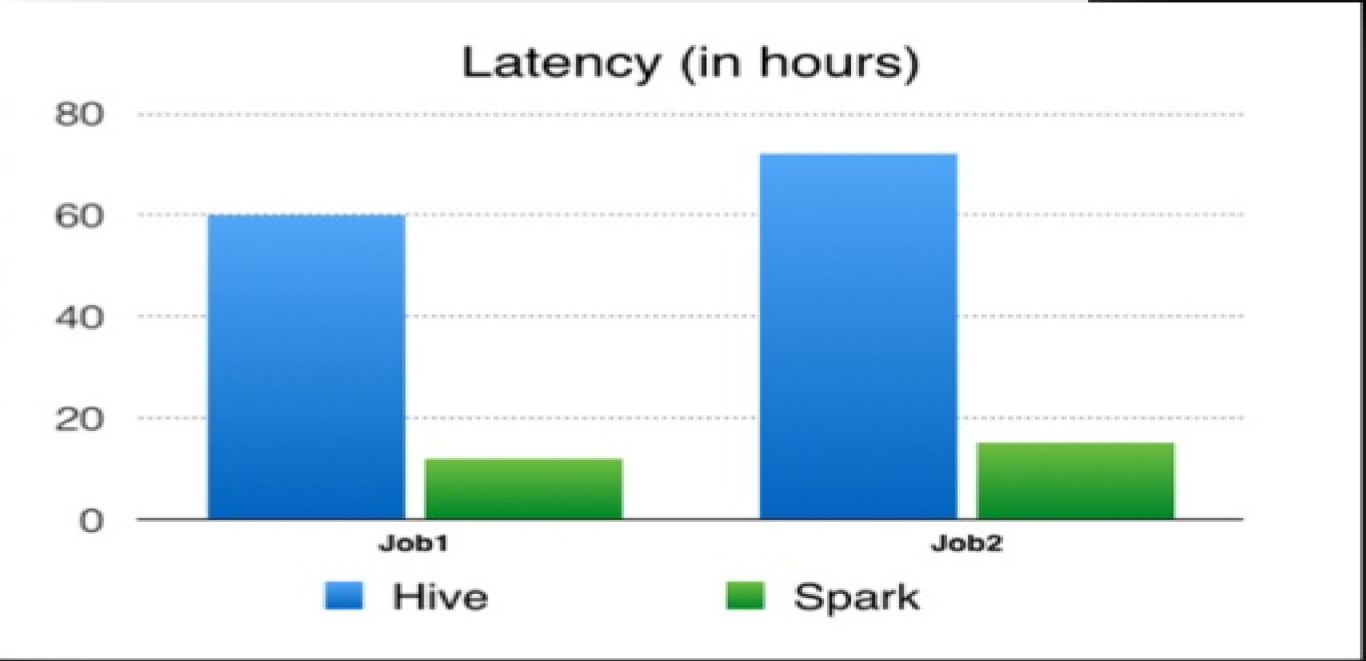
databricks


Spark Streaming At Bing Scale
Kaarthik Sivashanmugam
@kaarthikss
Microsoft
Microsoft
Bing Scale Problem – Log Merging
• Merge Bing query events with click events
• Lambda architecture: batch- and stream-processing shares the same C# library
• Spark Streaming in C#
Raw Logs
1
Databus
2
Event Merge Pipeline
3
Databus
4
10-minute App-Time Window
Edge
UI-Layout
Click
Kafka in every DC
Merged Log
Kafka
SPARK SUMMIT 2016
databricks

Apache Spark @Scale: A 60 TB+ production use case

Sital Kedia Shuojie Wang Avery Ching

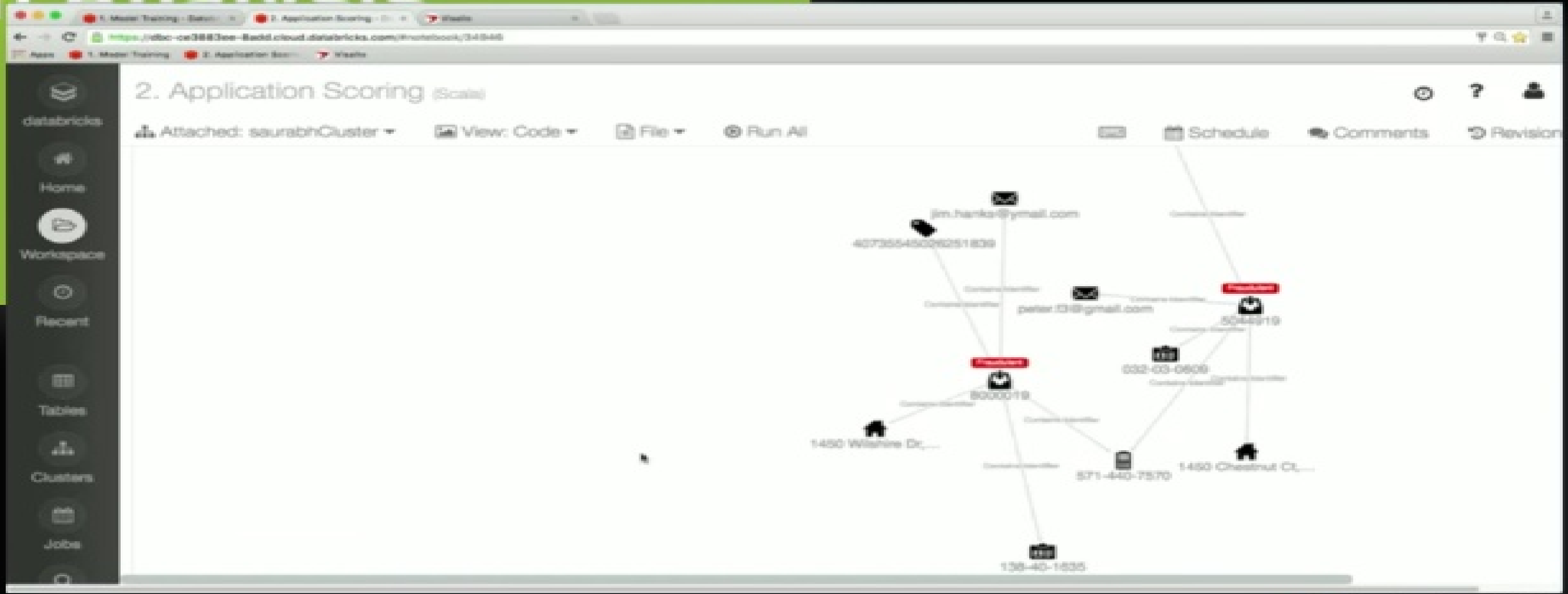
Facebook often uses analytics for data-driven decis
product growth has pushed our analytics engines tc
a single query. Some of our batch analytics is execu
(contributed to Apache Hive by Facebook in 2009) a
implementation. Facebook has also continued to gr
against several internal data stores, including Hive.



facebook

Credit Fraud Prevention with Spark and Graph Analysis

Chris D'Agostino
VP Technology, Capital One
@chrisdagostino



Capital One

databricks

This Talk

What are the new trends for big data apps in 2017?

Work to address them at Databricks + elsewhere

Three Key Trends

1. **Hardware:** compute bottleneck
2. **Users:** democratizing access to big data
3. **Applications:** production apps

databricks

Three Key Trends

1. **Hardware:** compute bottleneck
2. **Users:** democratizing access to big data
3. **Applications:** production apps

databricks

Hardware Trends

	2010
Storage	100 MB/s (HDD)
Network	1Gbps
CPU	~3GHz

databricks

Hardware Trends

	2010	2017
Storage	100 MB/s (HDD)	1000 MB/s (SSD)
Network	1Gbps	10Gbps
CPU	~3GHz	~3GHz

Hardware Trends

	2010	2017	
Storage	100 MB/s (HDD)	1000 MB/s (SSD)	**10x**
Network	1Gbps	10Gbps	**10x**
CPU	~3GHz	~3GHz	☹

Response: simpler but more parallel devices (e.g. GPU, FPGA)

databricks

Summary

In 2005-2010, I/O was the name of the game

- Network locality, compression, in-memory caching

Now, compute efficiency matters even for data-intensive apps

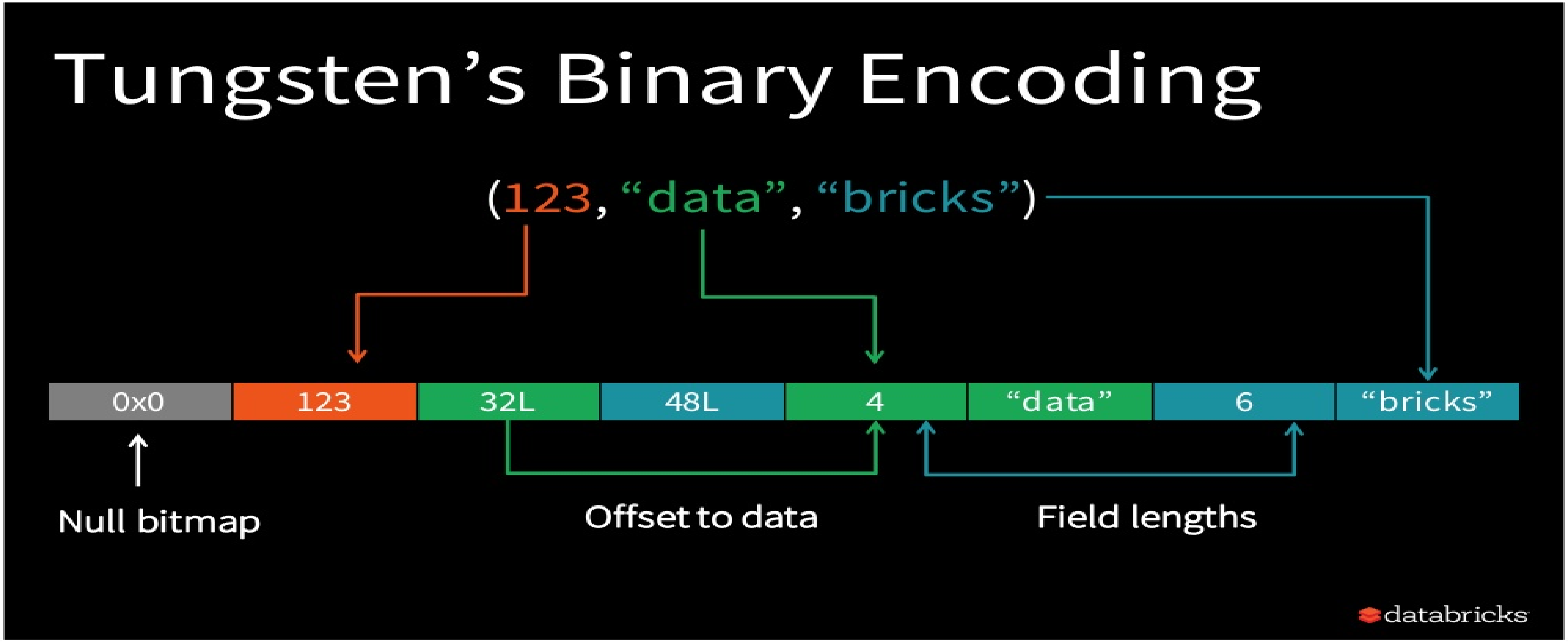
- And harder to obtain with more types of hardware!

databricks

Spark Effort: Project Tungsten

Optimize Apache Spark's CPU and memory usage, via:

- Binary storage format
- Runtime code generation


Tungsten's Binary Encoding
(123, "data", "bricks")
0x0
123
32L
48L
4
"data"
6
"bricks"
Null bitmap
Offset to data
Field lengths
databricks

Tungsten's Code Generation

DataFrame/SQL Code

```
df.where(df("year") > 2015)
```

Logical Expressions

```
GreaterThan(year#234, Literal(2015))
```

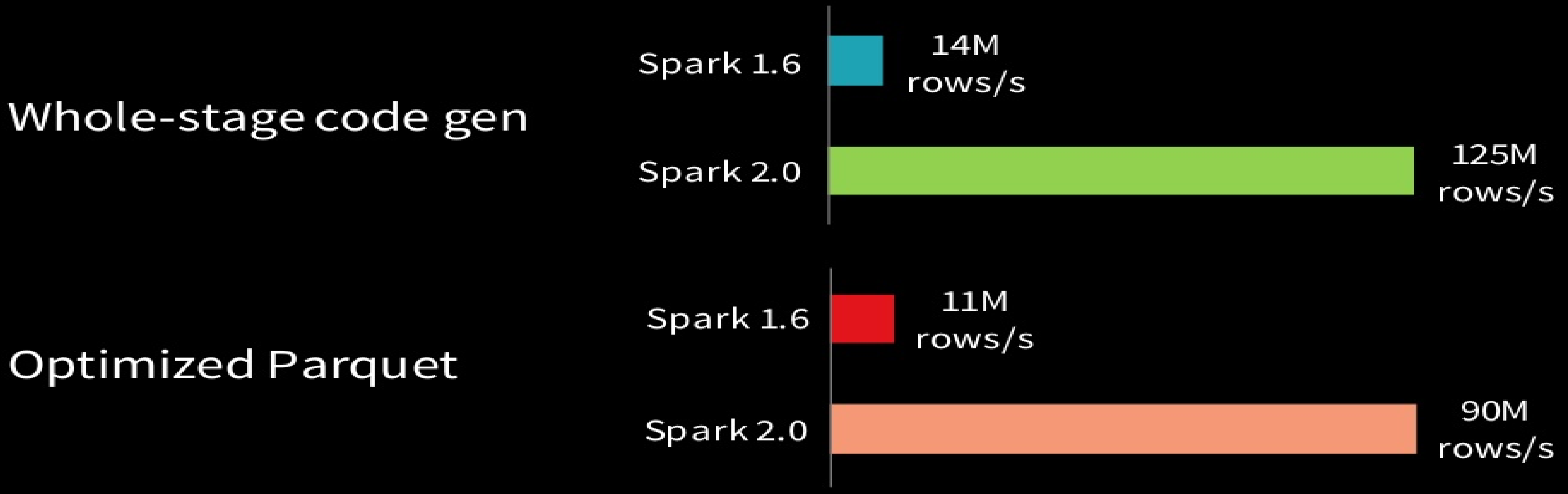
Java Bytecode

```
bool filter(Object baseObject) {
   int offset = baseOffset + bitSetWidthInBytes + 3*8L;
   int value = Platform.getInt(baseObject, offset);
   return value > 2015;
}
```

compiles to pointer arithmetic

databricks

Impact of Tungsten



databricks

Ongoing Work

Standard binary format to pass data to external code

- Either existing format or Apache Arrow (SPARK-19489, SPARK-13545)
- Binary format for data sources (SPARK-15689)

Integrations with deep learning libraries

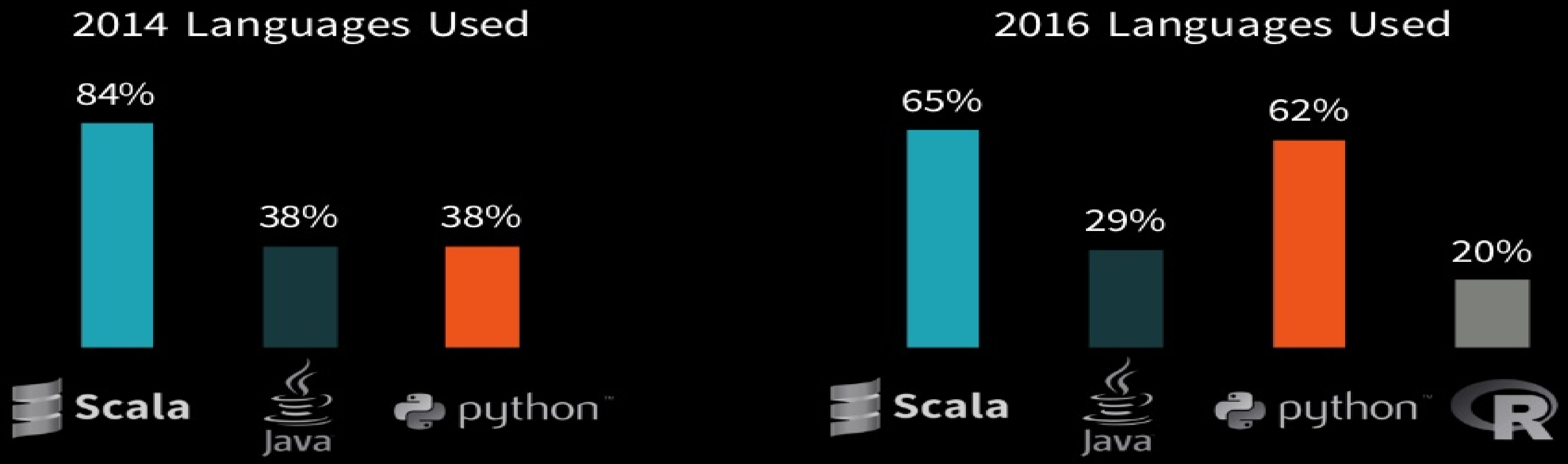
- Intel BigDL, Databricks TensorFrames (see talks today)

Accelerators as a first-class resource

Three Key Trends

1. **Hardware:** compute bottleneck
2. **Users:** democratizing access to big data
3. **Applications:** production apps

Languages Used for Spark



+ Everyone uses SQL (95%)

databricks

Our Approach

Spark SQL

- Spark 2.0: more of SQL 2003 than any other open source engine

High-level APIs based on single-node tools

- DataFrames, ML Pipelines, PySpark, SparkR

databricks

Ongoing Work

Next generation of SQL and DataFrames

- Cost-based optimizer (SPARK-16026 + many others)
- Improved data sources (SPARK-16099, SPARK-18352)

Continue improving Python/R (SPARK-18924, 17919, 13534, ...)

Make Spark easier to run on a single node

- Publish to PyPI (SPARK-18267) and CRAN (SPARK-15799)
- Optimize for large servers
- As convenient as Python `multiprocessing`

databricks

Three Key Trends

1. **Hardware:** compute bottleneck
2. **Users:** democratizing access to big data
3. **Applications:** production apps

databricks

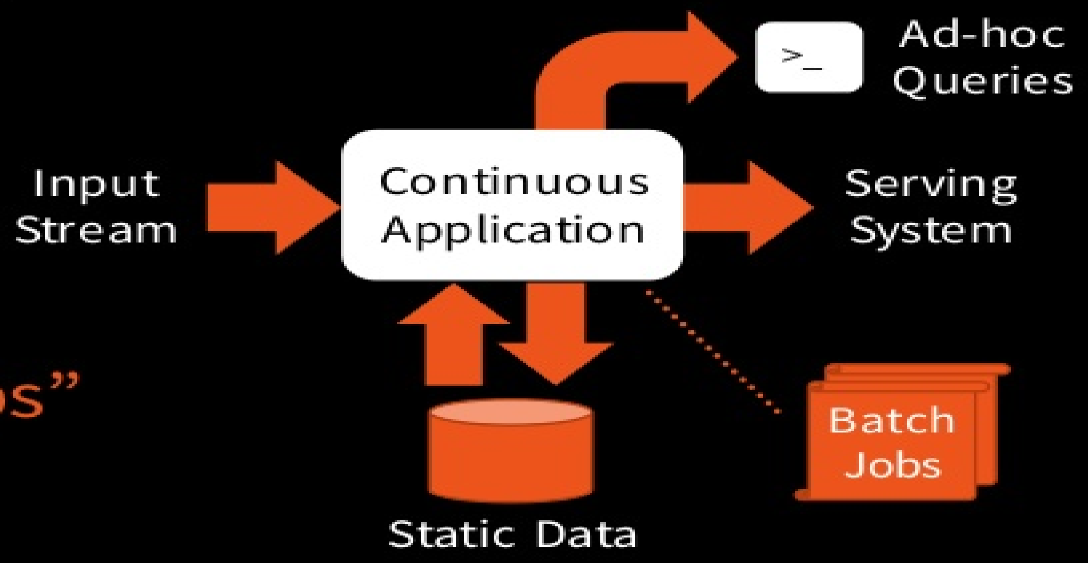
Big Data in Production

Big data is moving from offline analytics to production use

- Incorporate new data in seconds (streaming)
- Power low-latency queries (data serving)

Currently very hard to build: separate streaming, serving & batch systems

Our goal: single API for “continuous apps”



databricks

Structured Streaming

High-level streaming API built on DataFrames

- Transactional I/O to files, databases, queues
- Integration with batch & interactive queries

databricks

Structured Streaming

High-level streaming API built on DataFrames
- Transactional I/O to files, databases, queues
- Integration with batch & interactive queries

API: incrementalize an existing DataFrame query

Example batch job:

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

logs.groupBy("userid", "hour").avg("latency")
    .write.format("parquet")
    .save("s3://stats")
```

databricks

Structured Streaming

High-level streaming API built on DataFrames
- Transactional I/O to files, databases, queues
- Integration with batch & interactive queries

API: incrementalize an existing DataFrame query

Example as streaming:

```
logs = ctx.readStream.format("json").load("s3://logs")

logs.groupBy("userid", "hour").avg("latency")
    .writeStream.format("parquet")
    .start("s3://stats")
```

databricks

Early Experience

databricks	Running in our analytics pipeline since second half of 2016
viacom	Powers real-time metrics for properties including Nickelodeon and MTV
iPass	Monitors 1000s of WiFi access points

Ongoing Work

Integrations with more systems

- JDBC source and sink (SPARK-19478, SPARK-19031)
- Unified access to Kafka (SPARK-18682)

New operators

- mapWithState operator (SPARK-19067)
- Session windows (SPARK-10816)

Performance and latency

Three Key Trends

1. **Hardware:** compute bottleneck
2. **Users:** democratizing access to big data
3. **Applications:** production apps

All integrated in Apache Spark 2.0

databricks

Thanks

Enjoy Spark Summit!

databricks