Building realtime BI Systems with Kafka, Spark and Kudu

Ruhollah Farchtchi
Zoomdata

SPARK SUMMIT EAST 2017

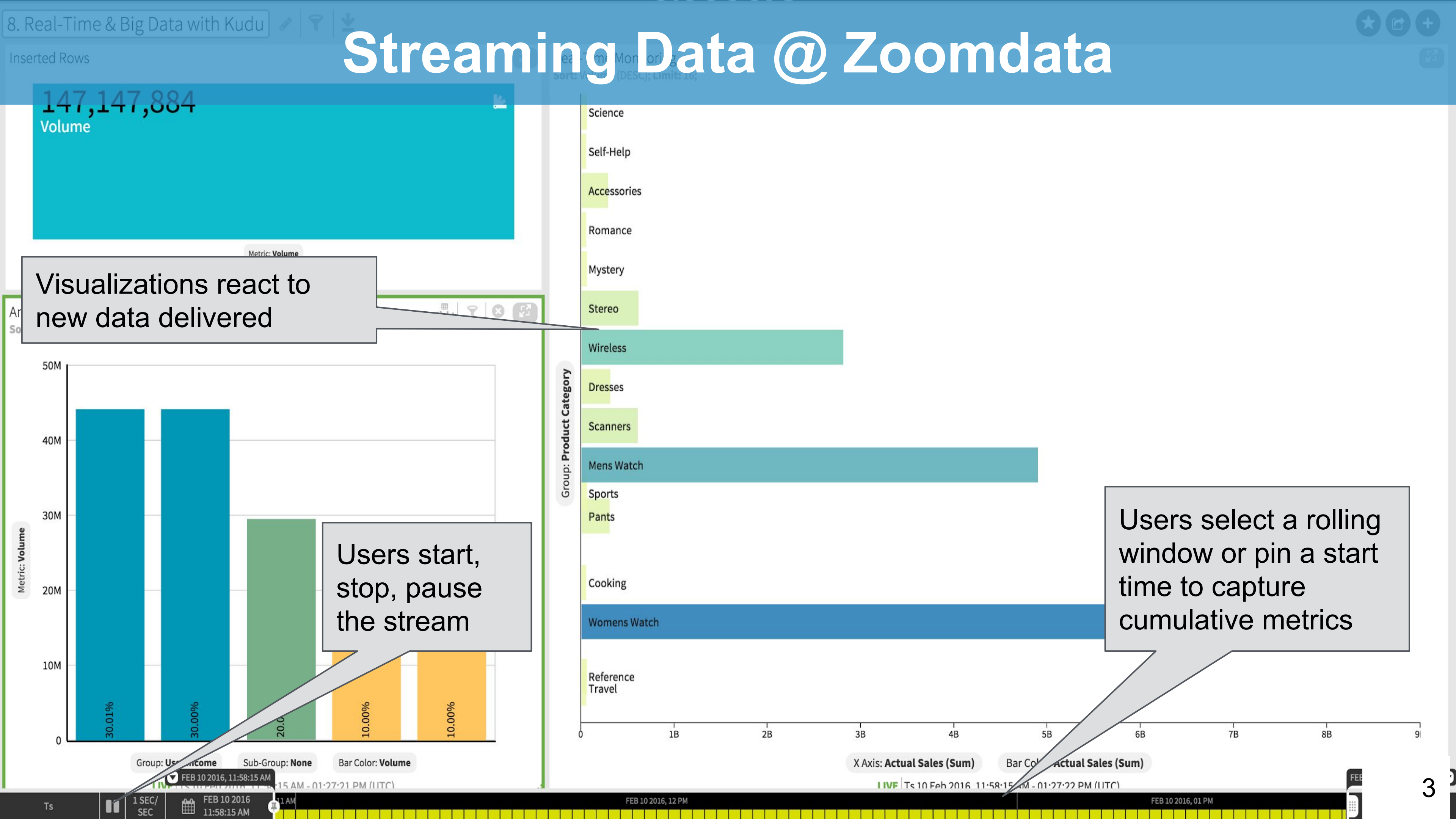
Drivers for Streaming Data

Data Freshness

Time to Analytic

Business Context

Streaming Data @ Zoomdata

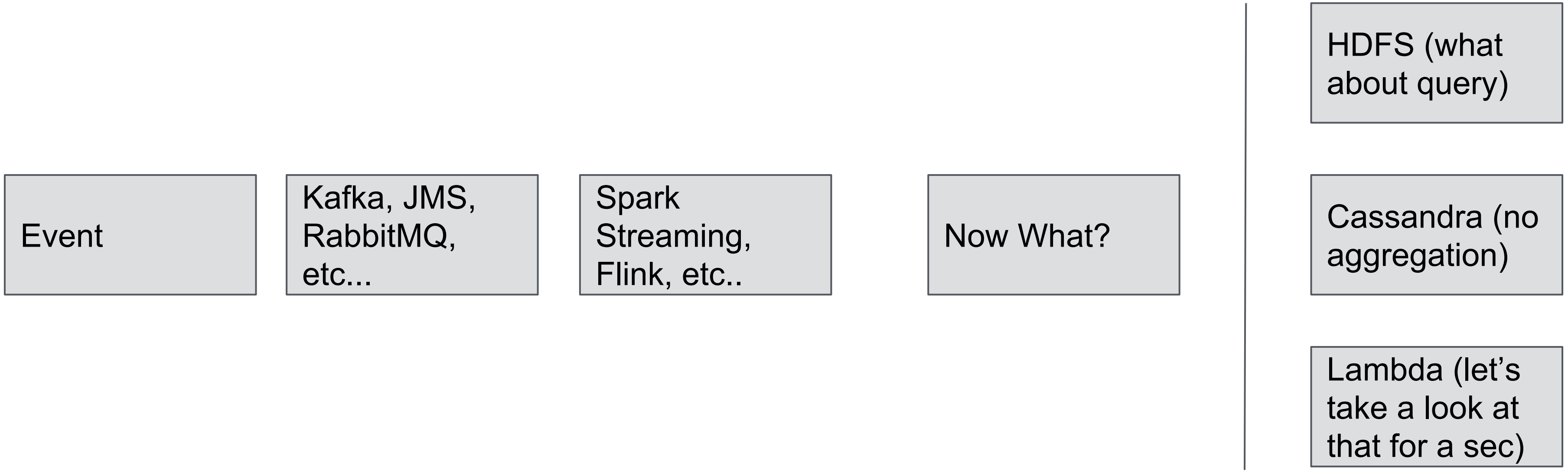


Visualizations react to new data delivered

Users start, stop, pause the stream

Users select a rolling window or pin a start time to capture cumulative metrics

Typical Streaming Architectures


Event
Kafka, JMS, RabbitMQ, etc...
Spark Streaming, Flink, etc..
Now What?
HDFS (what about query)
Cassandra (no aggregation)
Lambda (let's take a look at that for a sec)

SPARK SUMMIT EAST 2017

Lambda

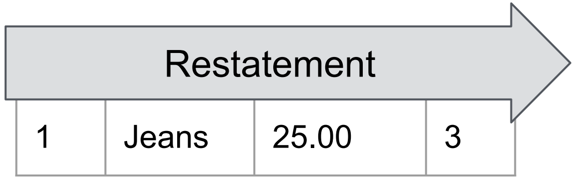
- Stream data to hdfs
- Keep some in avro
- Do your own compactions to parquet / orc
- Expose via impala, sparksql, or other

OR

- Impala avro partition (speed)
- With history in parquet
- Union compatible schema
- Project as single table via view
- Works ok… still doing a lot of manual data management

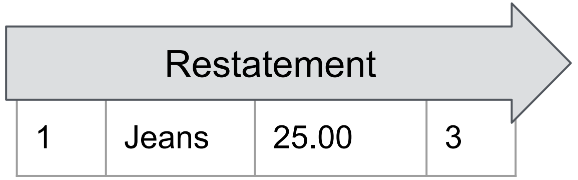
Oh.. and what happens to noncommutative operations like Distinct Count?

Restatements … yeah we went there



Txn ID	Item	Price	Quantity	Partition
1	Jeans	25.00	2	2016-05-30
2	Shirt	10.00	1	2016-05-31
3	Skirt	20.00	1	2016-06-01

Restatements … how you do it



Txn ID	Item	Price	Quantity	Partition
1	Jeans	25.00	2	2016-05-30
2	Shirt	10.00	1	2016-05-31
3	Skirt	20.00	1	2016-06-01

General Algorithm

- Figure out which partition(s) are affected
- Recompute affected partition(s) with restated data
- Drop/replace existing partition(s) with new data

Enter Kudu

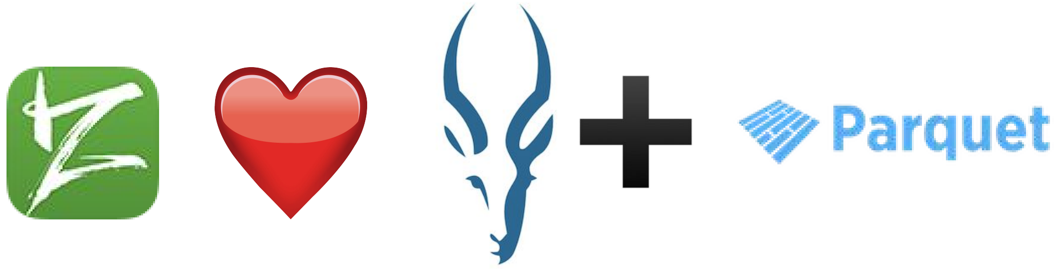
What is Kudu?

- **Kudu is an open source storage engine for structured data which supports low-latency random access together with efficient analytical access patterns.** (Source: http://getkudu.io/kudu.pdf)

Why do you care?

- **It makes management of streaming data for ad-hoc analysis MUCH easier**
- **Bridges the mental gap from random access to append only**

Why does Zoomdata care?



Impala + Kudu: Performance

Nearly the same performance as Parquet for many similar workloads

Simplified data management model

Can handle a new class of streaming use cases and workloads

Impala + Kudu: Performance

Nearly the same performance as Parquet for many similar workloads

Simplified data management model

Can handle a new class of streaming use cases and workloads

Great… let's just use Kudu from now on:
- We can ingest data with great write throughput
- Support analytic queries
- Support random access writes

What's not to love?

Ship It!

SPARK SUMMIT EAST 2017

There’s a catch...

… it’s your data model

Good news! If you have figured this out with HDFS and Parquet, you’re not too far off.

Things to consider:

- **Access pattern and partition scheme (similar to partitioning data parquet)**
  - Has a big role to play in parallelism of your queries
- **Cardinality of your attributes**
  - Affects what type of column encoding you decide to use
- **Key structure**
  - You get only one, use it wisely

More on this can be found at : http://getkudu.io/docs/schema_design.html

Let’s put it all together

I have a fruit stand

I sell my fruits via phone order to remote buyers

My transactions look something like:

Orders(<u>orderID,orderTS</u>,fruit,price,customerID, customerPhone,customerAddress)

SPARK SUMMIT EAST 2017

Impala DDL for Kudu

Key

```
CREATE EXTERNAL TABLE `strata_fruits_expanded` (
`_ts` BIGINT,
`_id` STRING,
```

```
TBLPROPERTIES(
  'storage_handler' =
'com.cloudera.kudu.hive.KuduStorageHandler',
  'kudu.table_name' = 'strata_fruits_expanded',
  'kudu.master_addresses' = '10.xxx.xxx.xxx:7051',
  'kudu.key_columns' = '_ts, _id'
);
```

Key

Impala DDL for Kudu

```
CREATE EXTERNAL TABLE `strata_fruits_expanded` (
`_ts` BIGINT,
`_id` STRING,
`fruit` STRING,
`country_code` STRING,
`country_area_code` STRING,
`phone_num` STRING,
`message_date` BIGINT,
`price` FLOAT,
`keyword` STRING
)
```

```
DISTRIBUTE BY HASH (_ts) INTO 60 BUCKETS
TBLPROPERTIES(
  'storage_handler' =
'com.cloudera.kudu.hive.KuduStorageHandler',
  'kudu.table_name' = 'strata_fruits_expanded',
  'kudu.master_addresses' = '10.xxx.xxx.xxx:7051',
  'kudu.key_columns' = '_ts, _id'
);
```

attributes

attributes

Low cardinality attributes -- things I want to group by -- are great candidates for dictionary encoding

Impala DDL for Kudu

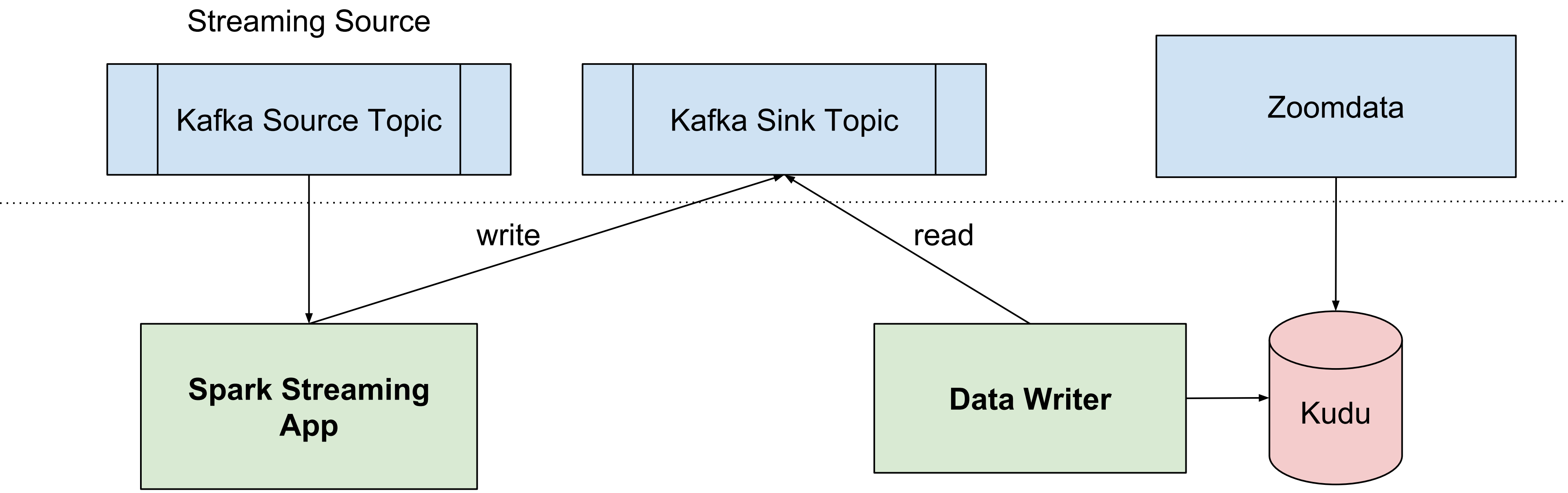
```
CREATE EXTERNAL TABLE `strata_fruits_expanded` (
`_ts` BIGINT,
`_id` STRING,
`fruit` STRING,
`country_code` STRING,
`country_area_code` STRING,
`phone_num` STRING,
`message_date` BIGINT,
`price` FLOAT,
`keyword` STRING
)
DISTRIBUTE BY HASH (_ts) INTO 60 BUCKETS
TBLPROPERTIES(
  'storage_handler' =
'com.cloudera.kudu.hive.KuduStor
  'kudu.table_name' = 'strata_fr
  'kudu.master_addresses' = '10.
  'kudu.key_columns' = '_ts, _id
);
```

Partition Scheme

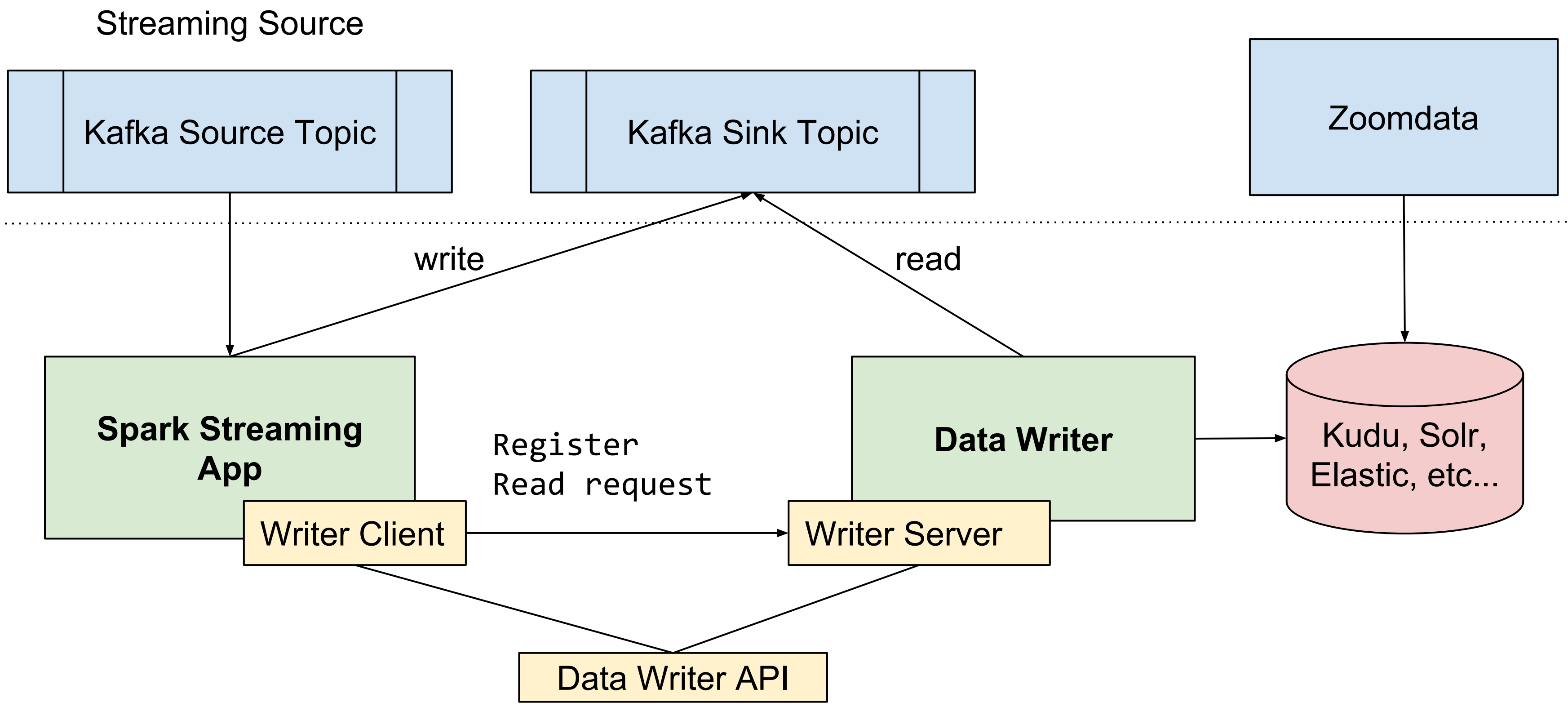
How you distribute your data directly impacts your ability to process in parallel as well as any predicate push-down type of operations Kudu can perform

For large tables, such as fact tables, aim for as many tablets as you have cores in the cluster -- but figure out what else you are running as well.

Let’s see it in action….



Let’s see it in action… not actually that simple



Special Thanks

Anton Gorshkov: For his original streaming with kafka fruit stand demo

The Cloudera Kudu Team: Specifically Todd Lipcon for all the insight into Kudu optimization

Nexmo: For use of their SaaS SMS service in this demo

SPARK SUMMIT EAST 2017

Thank You.

www.zoomdata.com

SPARK SUMMIT EAST 2017