REAL TIME ANALYTICS WITH SPARK AND KAFKA

NIXON PATEL, Chief Data Scientist, Brillio LLC.

Predictive Analytics & Business Insights

September 15th, 2015

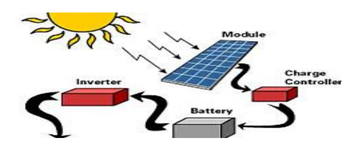
AGENDA

Who Am I?	Kafka as Distributed, Fault tolerant RT Broker
What is Real-Time?	Real-Time Data Streaming Analytics Architecture
What is Spark?	What is Spark Streaming?
Spark Overview.	Spark Streaming Terminology.
Why Spark?	Discretized Stream Processing (DStream)
Why Spark Over Hadoop MapReduce?	Windowed DStream
How Does Spark Work?	Mapped DStream
What is Spark Streaming?	Network Input DStream
What is Kafka?	Architecture For RT Analytics with SS And Kafka
What is Mllib?	Technology Stack For Demo
What is Lambda Architecture?	Analytics Model
Data Flow in Lambda Architecture	Inferences Drawn With Tableau Dashboard
Spark Streaming as Real-Time Micro-Batching	

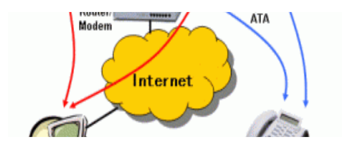
WHO AM I?

- Chief Data Scientist @ Brillio
- SVP - Business Head Emerging Technologies (SMAC) @ Collabera 2014
- Also, A Serial & Parallel Entrepreneur 1994-2013

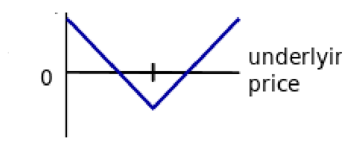
YantraSoft, A Speech Products Company 2009-2013
Bhrigus, A Speech & ERP Solutions Company



YantraeSolar, A 5 MW Solar Photovoltaic plant to generate Electricity from Solar Energy 2009-20013



eComServer, A Global Solutions & Products Company for Telecom, IVR and Web Servers 1998-2002



Objectware, A Boutique Company Specialized in providing Design of High End Derivatives Trading Systems 1994-1998

IBM - Research Engineer, Research Triangle Park, NC 1991-1994

IIT BTech (Hons) -Chemical Engineering
NJIT MS Computer Science
Rutgers Master of Business and Science in Analytics--Ongoing

brillio

WHAT IS "REAL TIME"(RT)?

Like there is no true "Unstructured" data so there is no "RT". Only "Near Real Time" (NRT)

NRT systems can respond to data as it receives it without persisting in a DB ("Present")

"Present" could mean different for different scenarios-

- Options Trader it is in Milliseconds, Ecommerce Site it is Attention Span

IT IS ABOUT

- ABILITY TO MAKE BETTER DECISIONS & TAKE MEANINGFUL ACTIONS AT THE RIGHT TIME
- DETECTING FRAUD WHILE SOMEONE IS SWIPING A CREDIT CARD
- TRIGGERING AN OFFER WHILE THE SHOPPER IS STANDING IN CHECKOUT LINE
- PLACING AN AD ON A WEBSITE WHILE SOMEONE IS READING A SPECIFIC ARTICLE
- COMBING & ANALYZING DATA SO YOU CAN TAKE RIGHT ACTION AT RIGHT TIME & RIGHT P

WHAT IS SPARK?

Brief Relevant Historical Information Of Spark

Map Reduce Paper 2004

Spark Paper by AMP Lab

Spark 1.4.1 Release July 15, 2015

2002

2004

2006

2008

2010

2012

2014

2015

Map Reduce @ Google 2002

2006 Hadoop @ Yahoo

2008 Hadoop Summit

Apache Spark Top Level

Spark 1.5 Preview Release in

SPARK OVERVIEW

Spark Stack

Spark SQL	Spark Streaming	Mlib (machine learning)	GraphX (graph)
Apache Spark			

- Spark SQL

  For SQL and unstructured data processing

- MLib

  Machine Learning Algorithms

- GraphX

  Graph Processing

- Spark Streaming

  Stream processing of live data stream

Runs Everywhere

Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, and S3.

You can run Spark using its standalone cluster mode, on EC2, on Hadoop YARN, or on Apache Mesos. Access data in HDFS,Cassandra, HBase, Hive

Speed

Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing.

Ease of Use

Write applications quickly in Java, Scala, Python, R.

Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it *interactively* from the Scala, Python and R shells.

Generality

Combine SQL, streaming, and complex analytics.

Spark powers a stack of libraries including SQL and DataFrames, Mlib for machine learning, Graph X, and Spark Streaming. You can combine these libraries seamlessly in the same application.

WHY SPARK

Limitations of MapReduce Model

- MR Programming Model is very Complex & has high overhead in launch of new M/R
- Performance bottlenecks as Streams needs multiple transformation
- Most Machine Learning Algorithms are iterative as multiple iterations improves results
- With Disk based approach each iteration's output is written to disk

Spark unifies Batch, Real-Time (Interactive) & Iterative apps into a single Framework

Spark's lazy evaluation of lineage graph reduces wait states with better pipelining

Spark's Optimized heap use of large memory spaces

Use of Functional Programming model makes creation and maintenance of apps simple

HDFS friendly, Unified Toolset, Pipelined Oriented

WHY SPARK OVER HADOOP MAPREDUCE?

Hadoop Execution Flow

Input Data on Disk → MR1 → Tuples (on Disk) → MR2 → Tuples (on Disk) → MR3 → Tuples (on Disk) → MR4 → Output Data on Disk

Spark execution flow

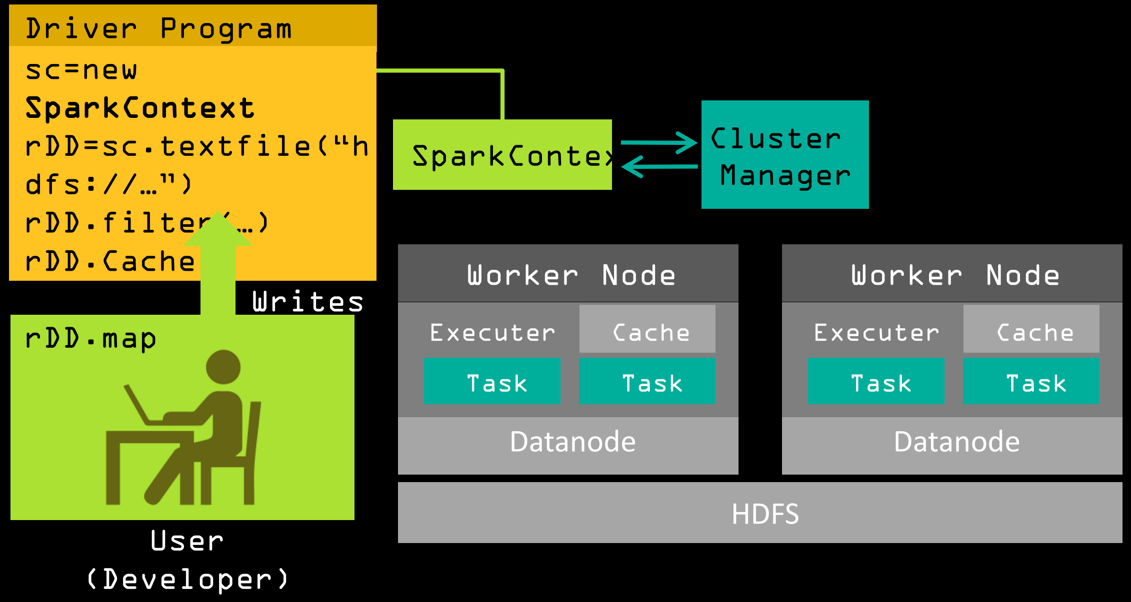
Input Data on Disk → t1 → RDD1 (in memory) → t2 → RDD2 (in memory) → t3 → RDD3 (in memory) → t4 → Output Data on Disk

	Hadoop MR Record	Spark Record	Spark 1 PB
Data Size	102.5 TB	100 TB	1000 TB
Elapsed Time	72 mins	23 mins	234 mins
# Nodes	2100	206	190
# Cores	50400 physical	6592 virtualized	6080 virtualized
Cluster Disk throughput	3150 GB/s (est.)	618 GB/s	570 GB/s
Sort Benchmark Daytona Rules	Yes	Yes	No
Network	Dedicated data center, 10Gbps	Virtualized (EC2)10Gbps network	Virtualized (EC2) 10Gbps network
Sort rate	1.42 TB/min	4.27 TB/min	4.27 TB/min
Sort rate/node	0.67 GB/min	20.7 GB/min	22.5 GB/min

large-scale-sorting.html

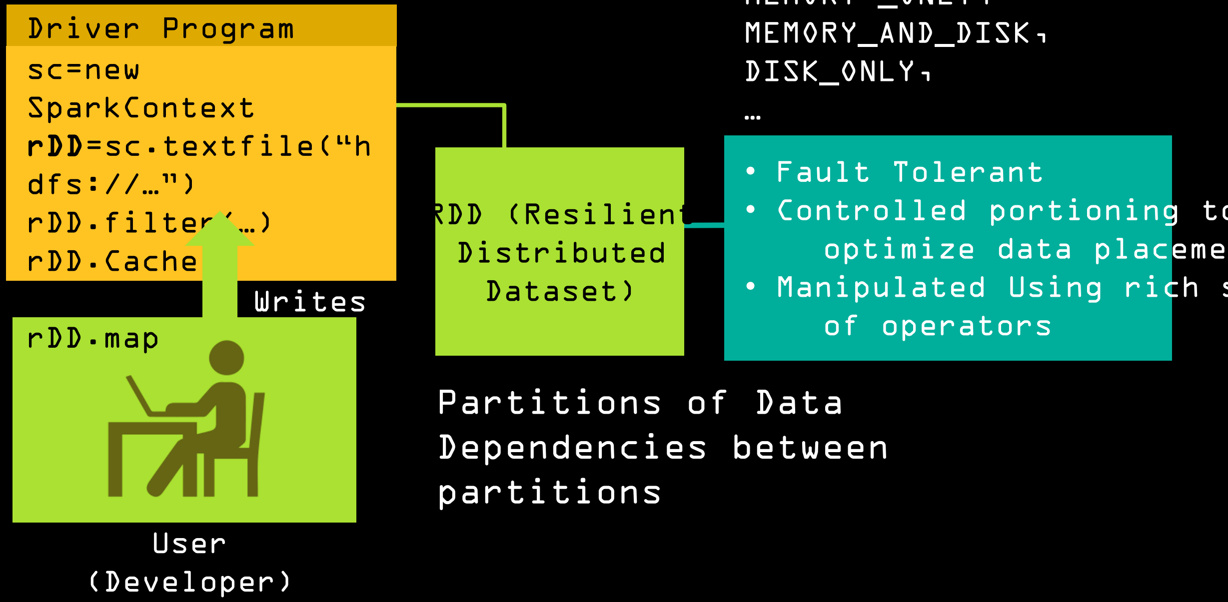
HOW SPARK WORKS

How Spark Works- SparkContext



How Spark Works- RDD

Storage Types:
MEMORY _ONLY,
MEMORY_AND_DISK,
DISK_ONLY,
…



How Spark Works - RDD Operations

Driver Program
- Create a new dataset from an existing one.
- Lazy in nature, executed only when some action is performed.
- Example -
  - Map(func)

Action
- Returns a value or exports data after performing a computation.
- Example -
  - Count()
  - Reduce(func)
  - Collect
  - Take()

Persistence
- Caching dataset in-memory for future operations.
- Store on disk or RAM or mixed
  - Persist()
  - Cache()

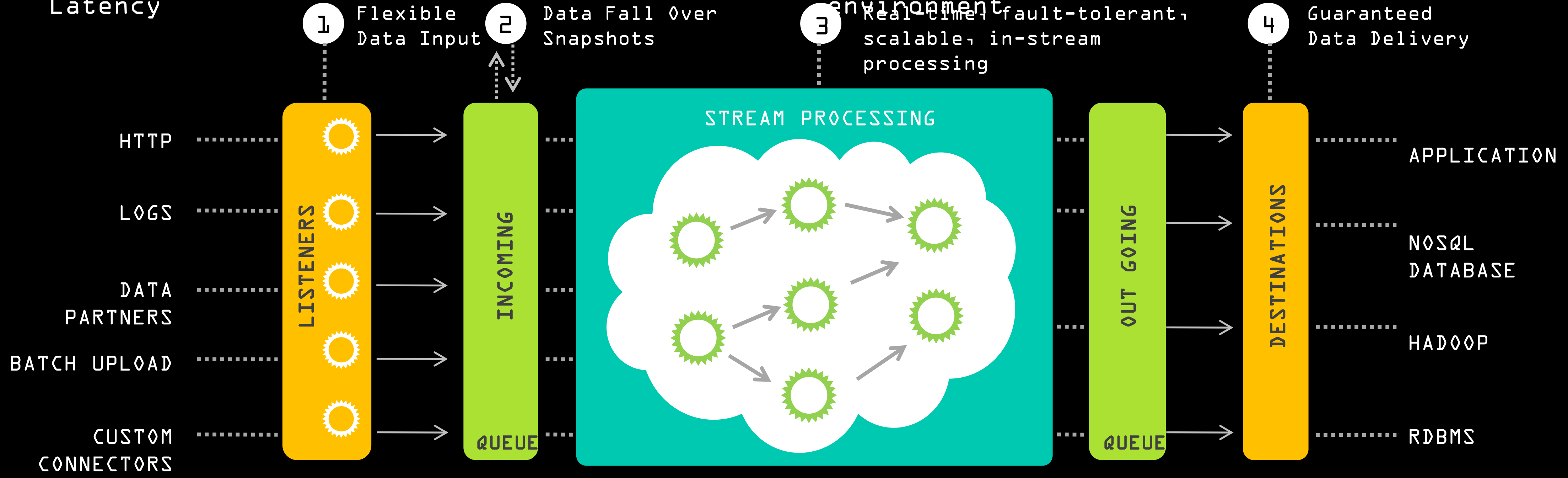
RDD- Resilient Distributed Dataset

- A big collection of data having following properties
- Immutable
- Lazy evaluate
- Cacheable
- Type Inferred

WHAT ARE THE CHALLENGES OF A REAL TIME STREAMING DATA PROCESSING?

- RTSD Processing challenges are Complex
- Collection of RT events coming @ Millions/Second
- Needs events correlation using Complex Event Proc
- Needs to Handle events with Low Latency

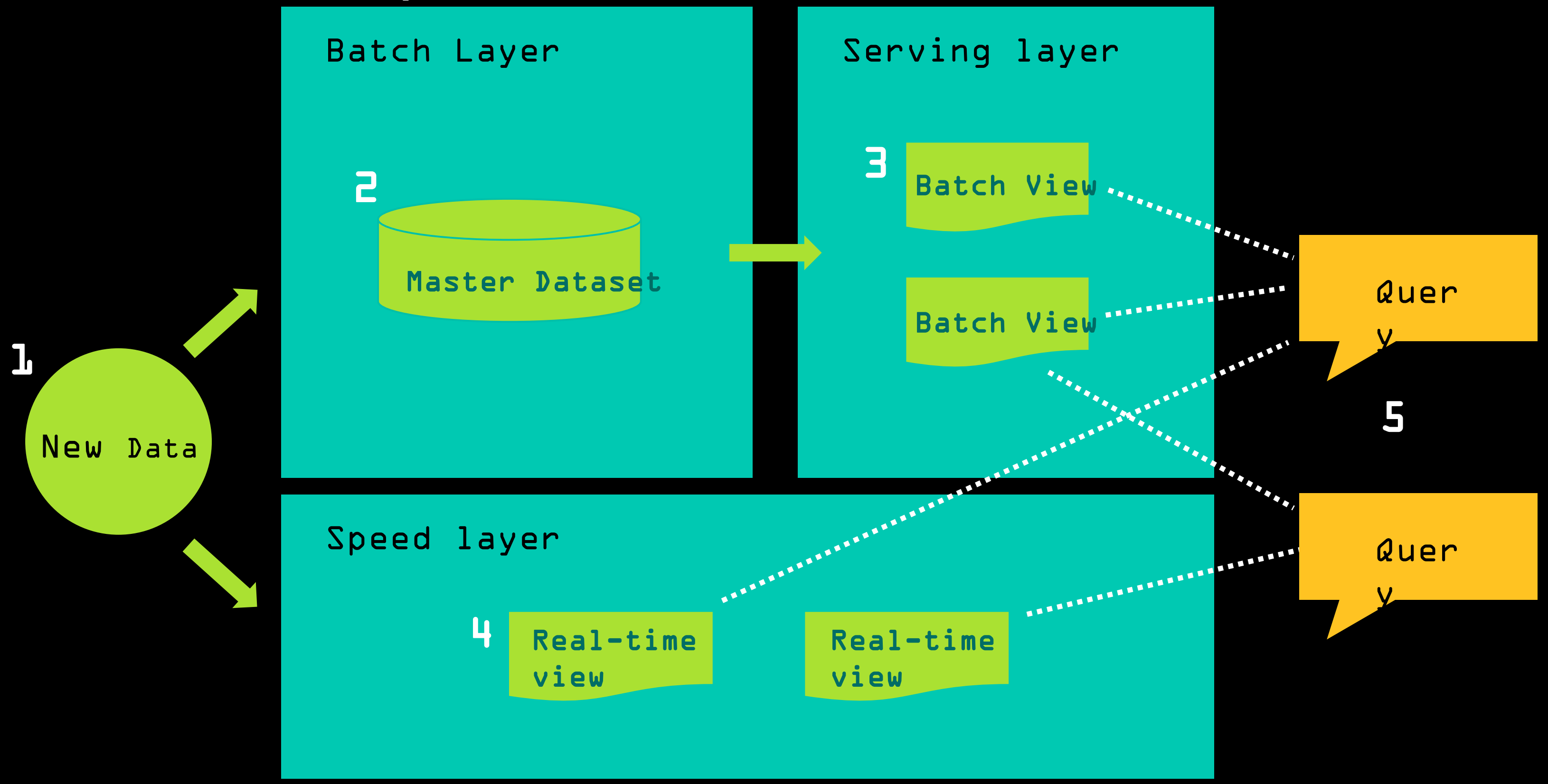
- Besides Vol & Var needs to handle Vel & Ver
- Needs Parallel processing of data being collected
- Needs Fault Tolerance
- Needs to handle data in distributed environment



http://preview.tinyurl.com/o983fmw

LAMBDA ARCHITECTURE*

LA satisfies the needs for a robust system that is fault-tolerant, both against hardware failures and human mistakes, being able to serve a wide range of workloads and use cases, and in which low-latency reads and updates are required. The resulting system should be linearly scalable, and it should scale out rather than up.



* http://lambda-architecture.net/ (Nathan Marz)

brillio

DATA FLOW IN LAMBDA ARCHITECTURE

1. All data entering the system is dispatched to both the batch layer and the speed layer for processing.

2. The batch layer has two functions:
   - managing the master dataset (an immutable,
     append-only set of raw data),
   - to pre-compute the batch views.

3. The serving layer indexes the batch views so that they can be queried in low-latency, ad-hoc way.

4. The speed layer compensates for the

5. Any incoming query can be answered by merging results from batch views and real-time view

- **Seems Like a Complex Solution…**
- **Besides, there is a general notion that Real-Time Layer is prone to FAILURE!**
- Why?
- Computation happens in memory
- A system Crash will erase the state of the Real-Time System
  - Bad Deployment
  - Out Of Memory
- Delayed upstream data will give inaccurate metrics

SPARK STREAMING AS RT MICRO-BATCHING SYSTEM COMES TO RESCUE

- Since the Real Time layer only compensates for the last few hours of data, everything the real-time layer computes is eventually overridden by the batch layer. So if you make a mistake or something goes wrong in the real-time layer, the batch layer will correct it. -Nathan Marz http://preview.tinyurl.com/nugnz6t Big Data: Principles and best practices of scalable real-time data systems
- Spark Streaming, A Micro Batch Layer on top of Core Spark corrects the fault tolerance issue
- Spark Streaming Provides Efficient and fault-tolerant state full stream processing
- Integrates with Spark's batch and interactive processing

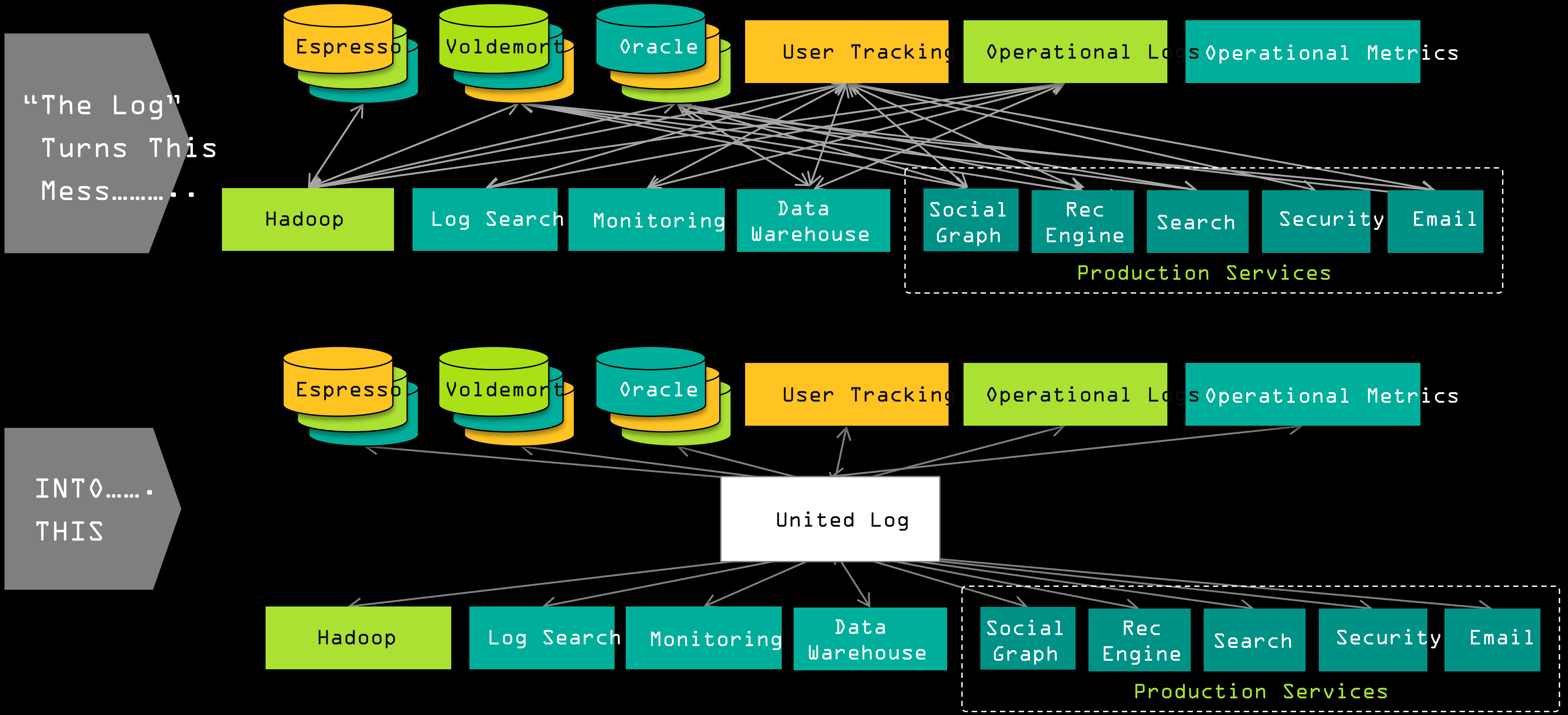
Wait A minute…

- Has the issue raised in component #1 of the Lambda Architecture addressed?

Not Yet!!!!

APACHE KAFKA COMES FOR RESCUE FOR FAULT #1 IN LA

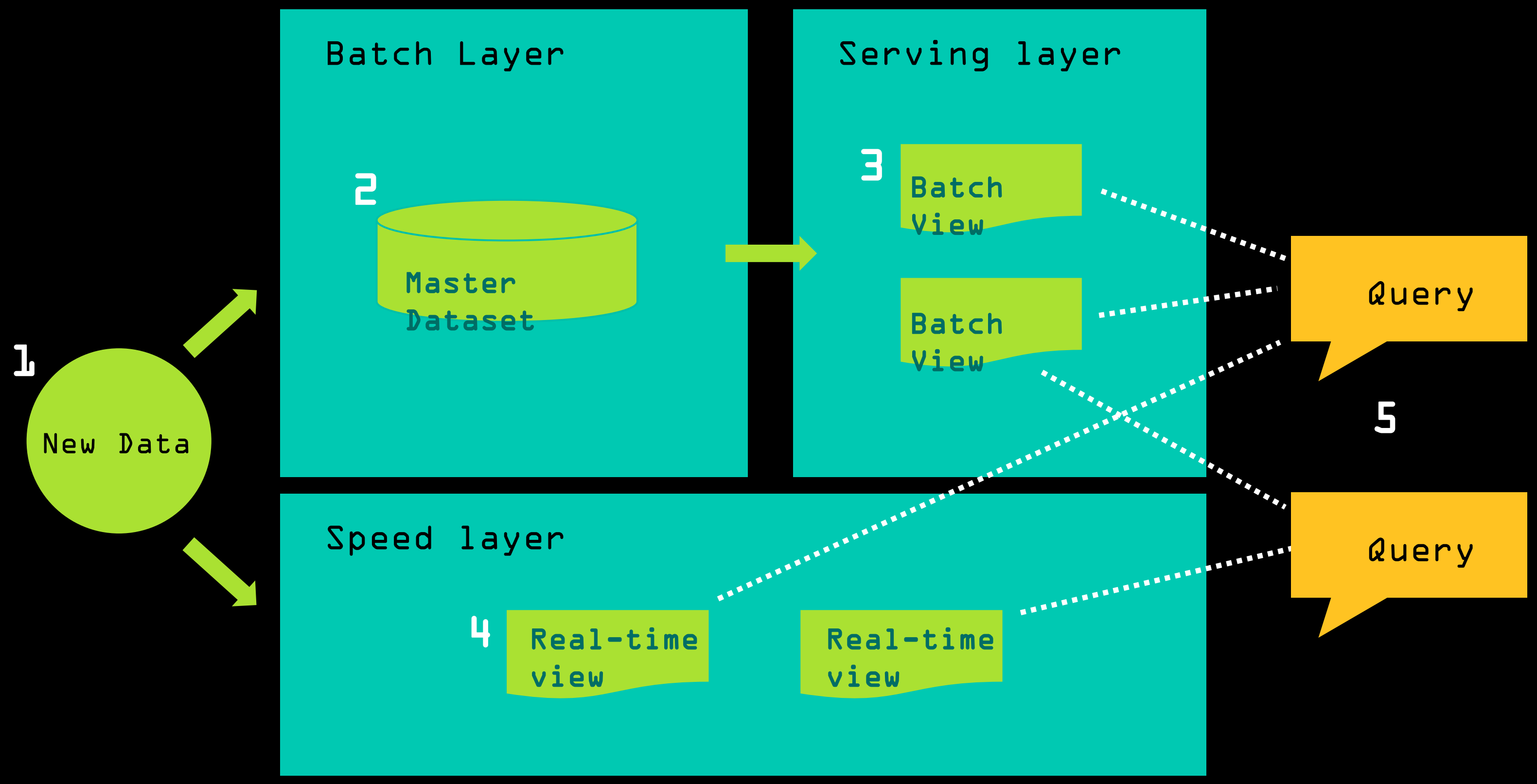
This is very well resolved by “The Log” from Jay Kreps
http://preview.tinyurl.com/qc43s5j



COMPONENTS OF REAL TIME DATA STREAMING ANALYTICS ARCHITECTURE

Lambda Architecture Stack

1. Unified Log - Apache Kafka
2. Batch Layer - Hadoop for Storage
3. Serving Layer - MySQL, Cassandra, NoSQL or other KV Stores
4. Real-Time Layer - Spark Streaming
5. Visualization Layer -- Tableau



brillio

WHAT IS SPARK STREAMING ?

- Extends Spark for doing large scale stream processing
- Scales to 100s of nodes and achieves second scale latencies
- Efficient and fault-tolerant state full stream processing
- Integrates with Spark's batch and interactive processing
- Provides a simple batch-like API for implementing complex algorithms
- These batches are called Discrete Stream (DStreams)

Credit: http://spark.apache.org/

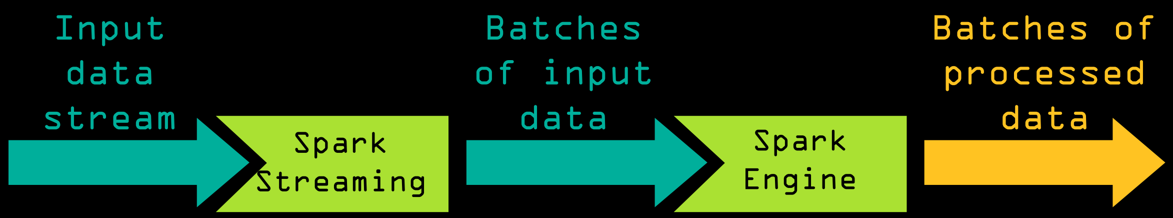
SPARK STREAMING TERMINOLOGY

DStreams

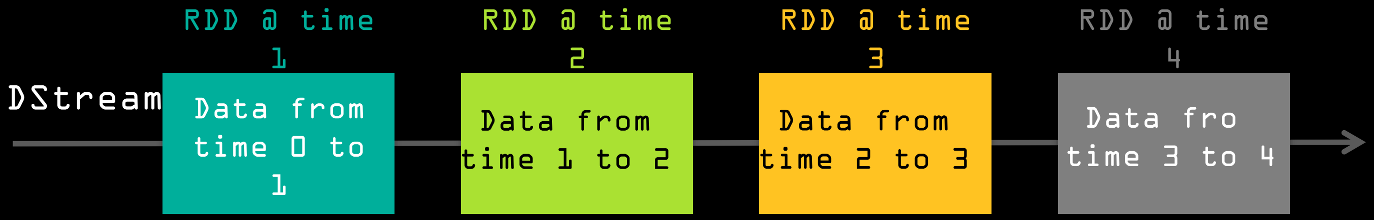
- A Sequence of Mini-Batches, where each mini- batch is represented as a Spark RDD
- Defined by its Input Source - Kafka, Twitter, HDFS, Flume, TCP, Sockets Akka Actor
- Defined by a time window called the Batch Interval
- Each RDD in Stream contains records

Each batch of DStream is replicated as RDD	RDDs are replicated in cluster for fault tolerance
Each DS operation result in RDD transformation	There are APIs to access these RDDs directly
RDDs from Batch and Stream can be combined	

Streaming Flow



DStream to RDD

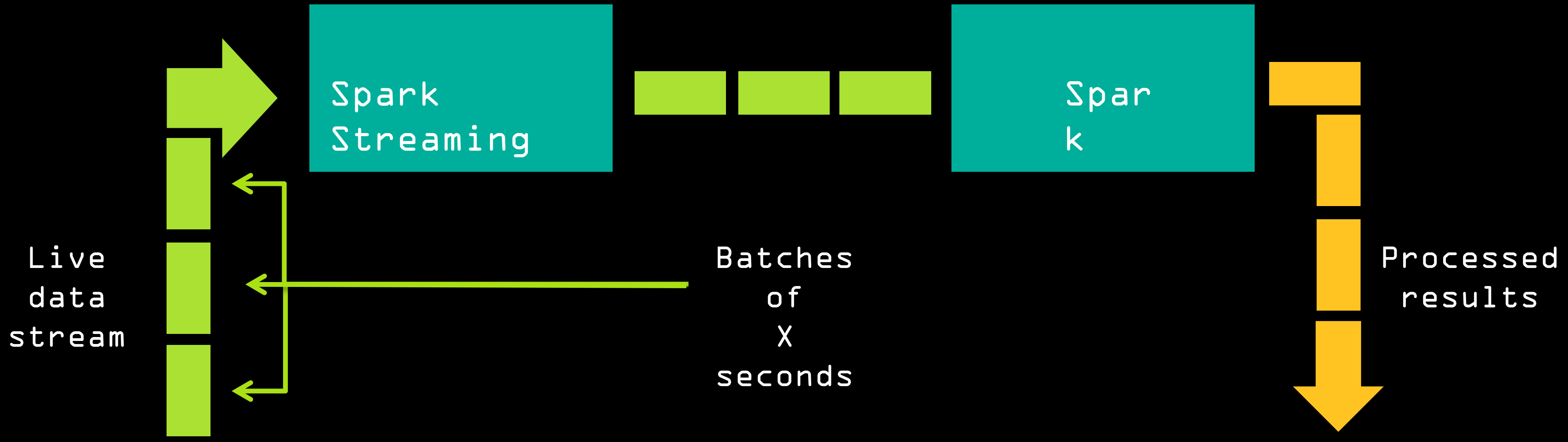


brillio

DSTREAM PROCESSING

Run a streaming computation as a series of very small, deterministic batch jobs

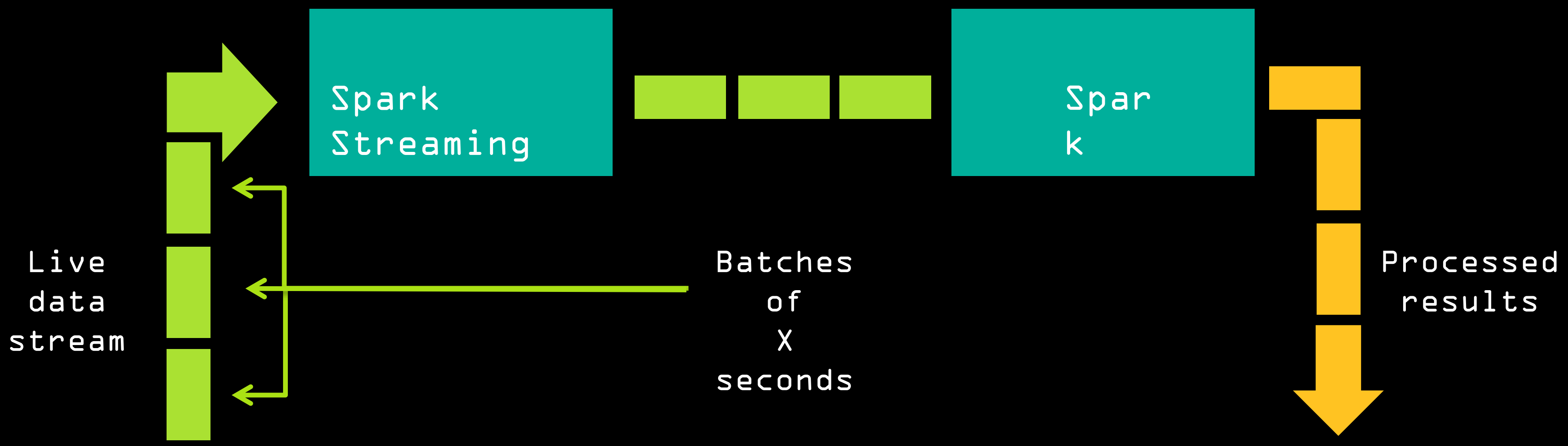
- Chop up the live stream into batches of X seconds
- Spark treats each batch of data as RDDs and processes them using RDD operations
- Finally, the processed results of the RDD operations are returned in batches



DSTREAM PROCESSING

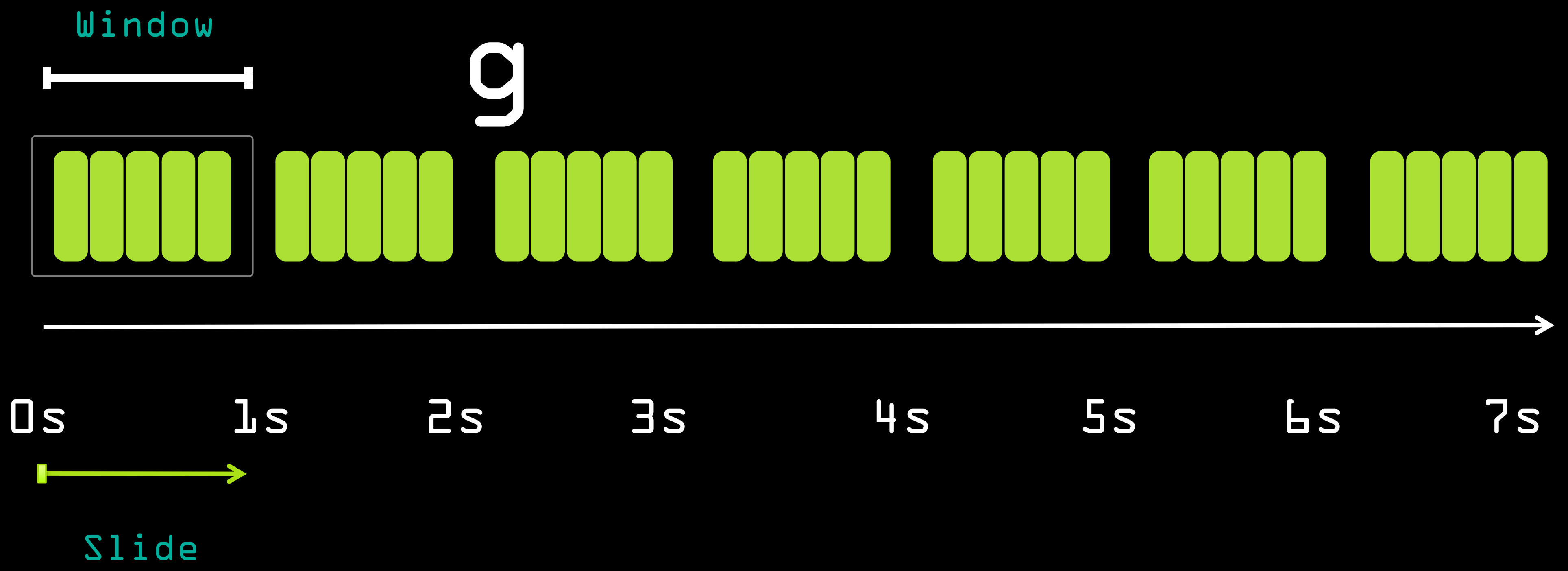
Run a streaming computation as a series of very small, deterministic batch jobs

- Batch sizes as low as ½ second, latency ~ 1 second
- Potential for combining batch processing and streaming processing in the same system



EXAMPLE: WINDOWED DSTREAM

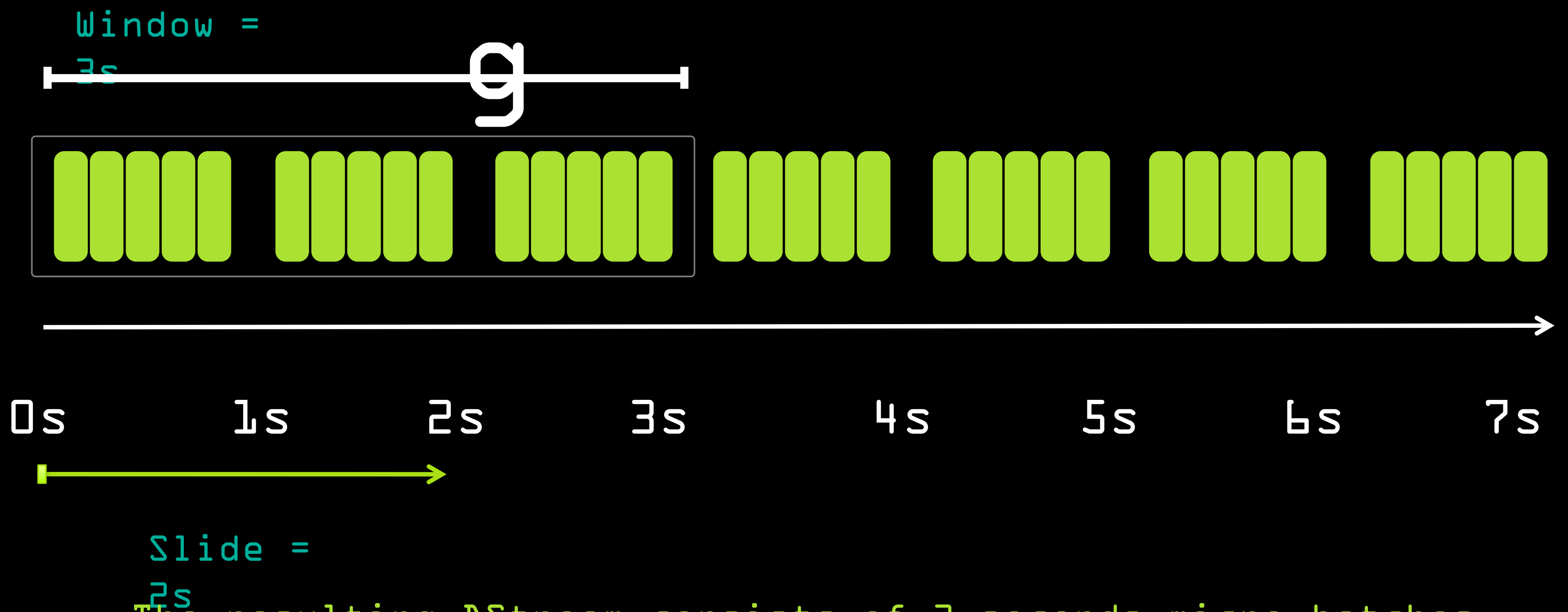
Windowing



By default:
Window = slide = batch duration

EXAMPLE: WINDOWED DSTREAM

Windowing



The resulting DStream consists of 3 seconds micro-batches

Each resulting micro-batch overlaps the preceding one by 1 seconds

EXAMPLE: MAPPED DSTREAM

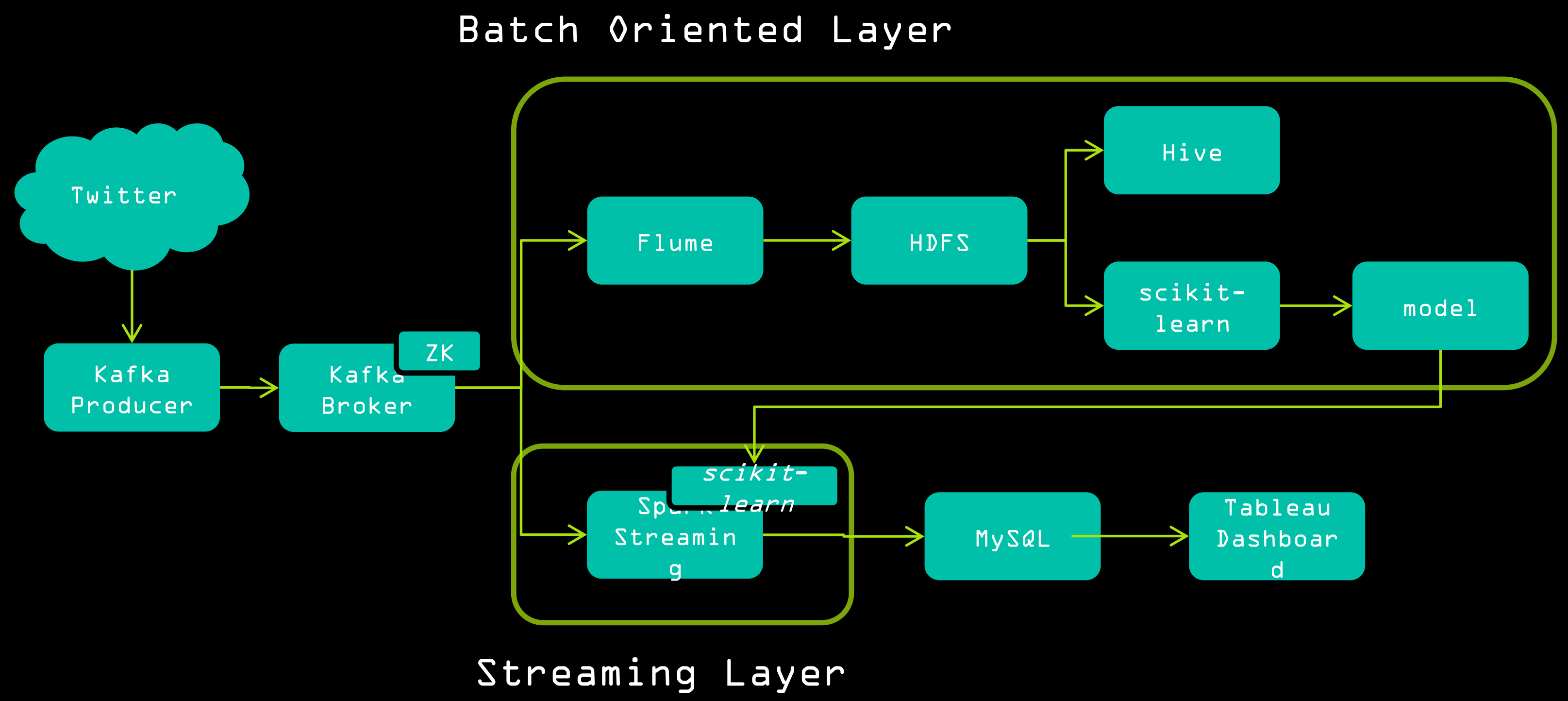
- *Dependencies:* Single parent DStream
- *Slide Interval:* Same as the parent DStream
- *Compute function for time t:* Create new RDD by applying map function on
  parent DStream's RDD of time t

```
override def compute(time: Time): Option[RDD[U]] = {
    parent.getOrCompute(time).map(_.map[U](mapFunc))
}
```

Gets RDD of time t if already computed once, or generates it

Map function applied to generate new RDD

ARCHITECTURE FOR THE REAL TIME ANALYTICS WITH SPARK STREAMING & KAFKA



-The above architecture uses a combination of batch oriented and streaming layer as in the case of Lambda architecture.

-For the batch oriented architecture Hive which is an abstraction on top of MapReduce is used.

-Scikit-learn is used to create the model from the tweets in HDFS after the labelling of the tweets has been done..

-For processing the streaming tweets, Spark framework is used.

-The sentiment of the tweet is figured out in Spark using Skikit Python Library and the same is stored in

TECHNOLOGY STACK USED FOR THE DEMO

Operating System

- Ubuntu 14.04 64-bit Server on Microsoft Azure

Languages

- Java 1.8
- Scala 2.10
- Python 2.7.6

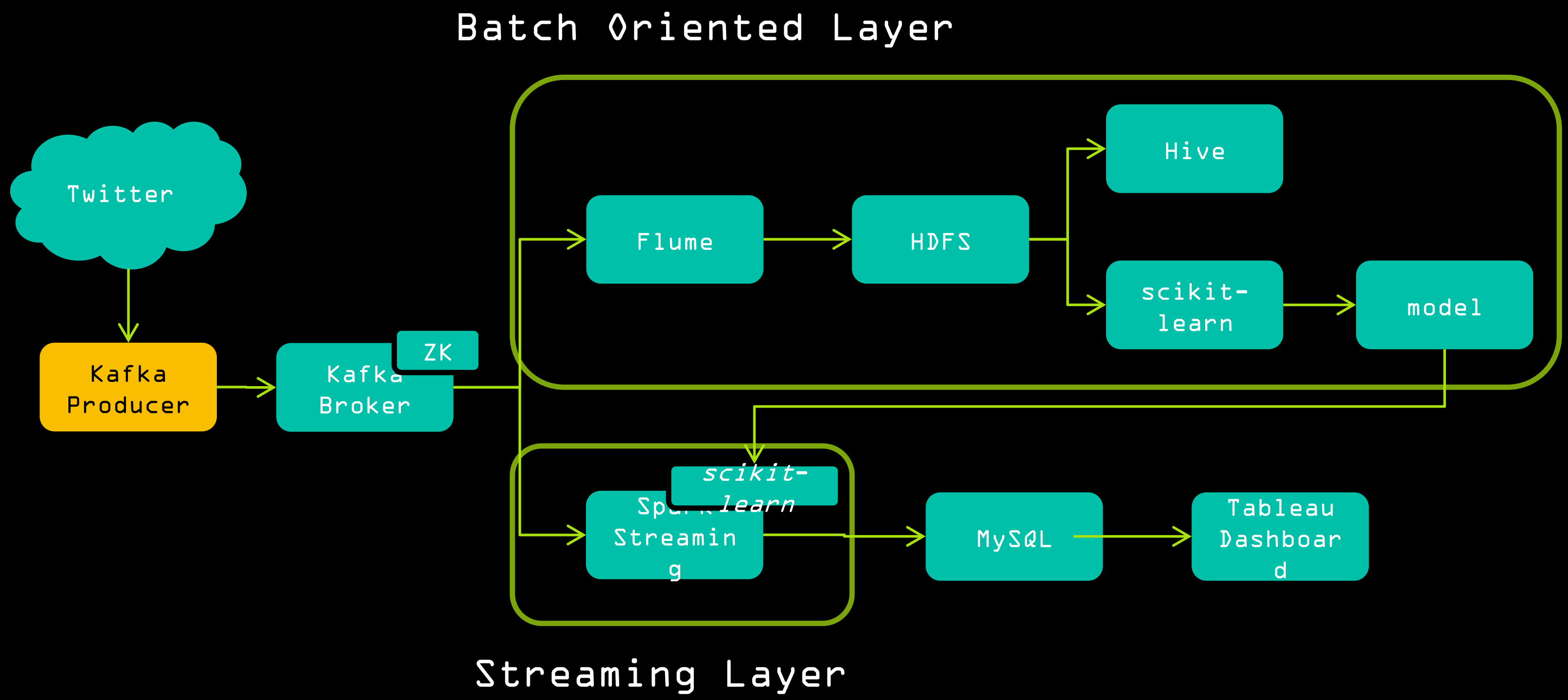
Database

- MySQL 5.5.44

Big Data Software

- Apache Flume 1.6.0
- Apache Hive 1.2.1
- Apache Hadoop 1.2.1
- Kafka 2.10 0.8.2.1
- Apache Spark 1.4.1

KAFKA PRODUCER



KAFKA PRODUCER

The Kafka producer uses the Hosebird Client (hbc - https://github.com/twitter/hbc) developed by Twitter to pull the data for the topics of interest like the presidential candidates of 2016 elections. Hbc is Java HTTP client for consuming the Twitter's Streaming API.

Once the tweets are pulled by Hbc, they are published to the Kafka topic where different subscribers can pull the tweets for further analysis.

The tweets are pulled and published in a real time. As soon as someone tweets on the relevant topics, the same is pulled by Hbc and published to the Kafka topic.

GETTING THE RELEVANT TWEET FOR 2016 PRESIDENTIAL ELECTIONS

Users tweet about politics, movies, sports and other interesting topics. On the StatusesFilterEndpoint the appropriate hashtags, keywords and Twitter User Id have to be specified as below to get the relevant tweets.

The StatusesFilterEndpoint is part of the Hbc API provided by Twitter. More details about the StatusesFilterEndpoint API can be found at https://goo.gl/hV89Sd.

The below are the hashtags and the keywords for which the Tweets are being pulled.

```
endpoint.trackTerms(Lists.newArrayList("#election2016", "#2016", "#FeelTheBern",
"#bernie2016", "#berniesanders", "#Walker16", "#scottwalker", "#gop",
"#hillaryclinton", "#politics", "#donaldtrump", "#trump", "#teaparty", "#obama",
"#libertarians", "#democrats", "#randpaul", "#tedcruz", "#republicans",
"#jebbush", "#hillary2016", "#gopdebate", "#bencarson",
"#usapresidentialelection", "Abortion", "Budget", "Economy", "Civil Rights",
"Corporations", "Crime", "Defence spending", "Drugs", "Education", "Energy and
Oil", "Environment", "Families and Children", "Foreign Policy", "Free Trade",
"Government Reform", "Gun Control", "Health Care", "Homeland Security",
"Immigration", "Infrastructure and Technology", "Jobs", "Social Security", "Tax
Reform", "State welfare"));
```

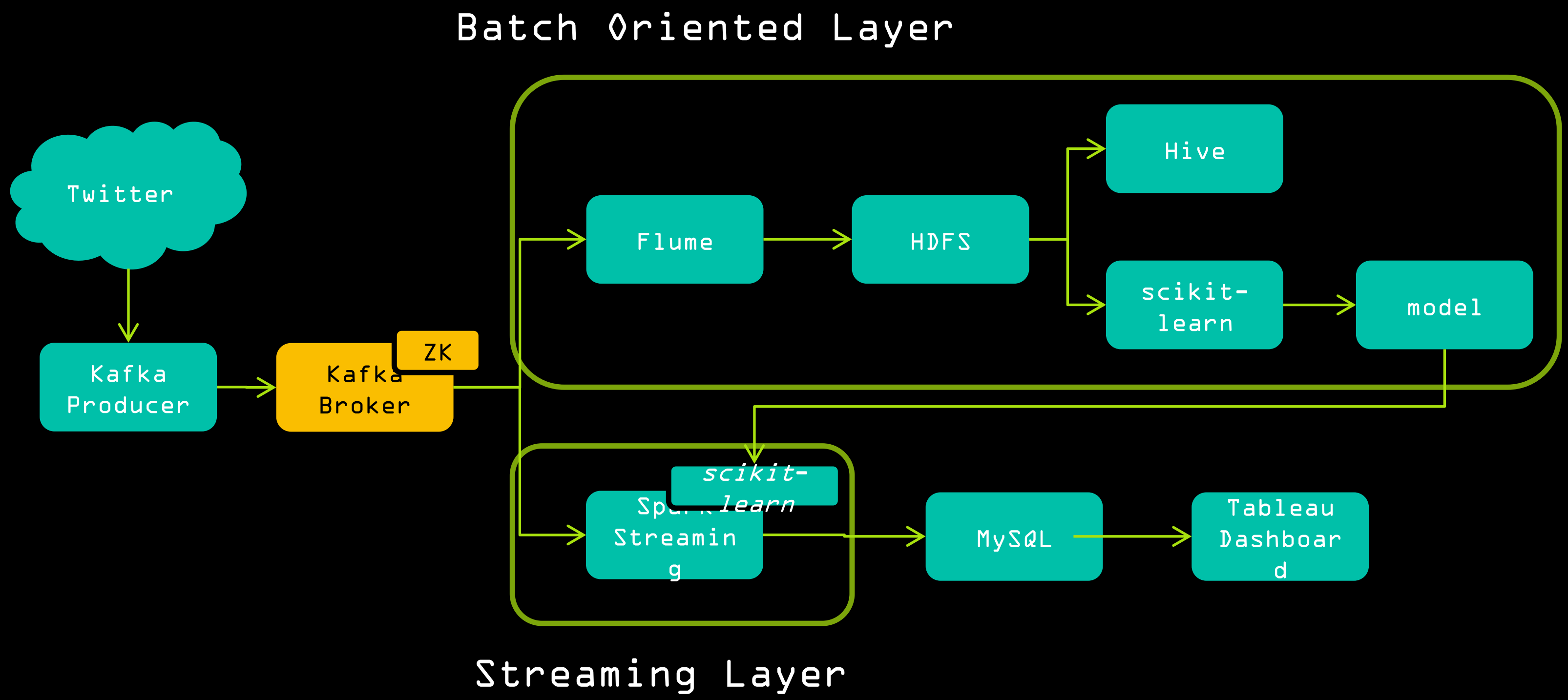
The below are the Twitter User ID for the presidential candidates got from http://gettwitterid.com

```
endpoint.followings(Lists.newArrayList(new Long(473383173), new Long(74662993),
new Long(11134252), new Long(939091), new Long(89781370), new Long(1339835893),
new Long(3343532685L), new Long(29442313), new Long(2746932876L), new
Long(113047940), new Long(1180379185), new Long(90484508), new Long(23022687),
new Long(2998799499L), new Long(65691824), new Long(3352706206L), new
Long(432895323), new Long(15416505), new Long(17078632), new Long(18020081), new
Long(2865560724L), new Long(216881337), new Long(18906561), new Long(15745368),
```

CODE FOR PUSHING THE MESSAGES FROM THE KAFKA PRODUCER TO THE BROKER

```
//Create an instance of the Kafka Producer

Producer<String, String> producer = new Producer<String,
   String>(producerConfig);

//Create an instance of  Kafka KeyedMessage which is passed to the Kafka
   Broker later

KeyedMessage<String, String> message = null;

try {

        //Populate the KeyedMessage with the topic, key and the message

        message = new KeyedMessage<String, String>(topic, "key",
   queue.take().trim());

} catch (InterruptedException e) {

        e.printStackTrace();

}

//Send the message from the producer to the broker
```

KAFKA BROKER

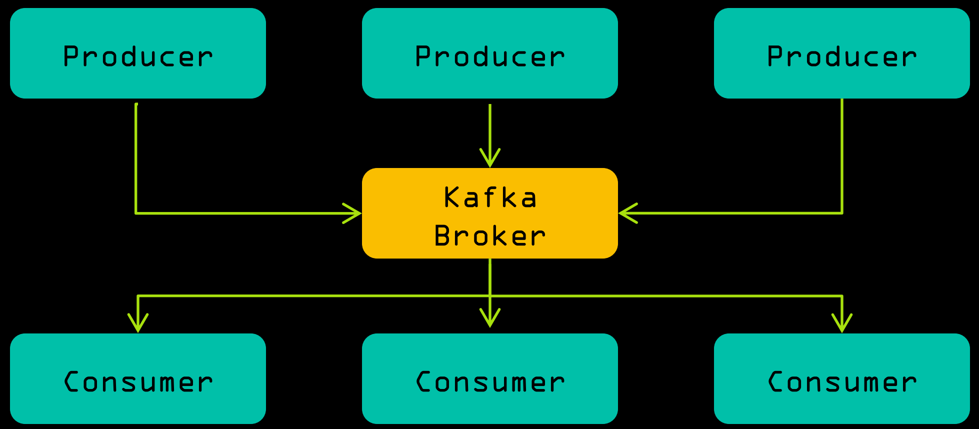


brillio

KAFKA BROKER

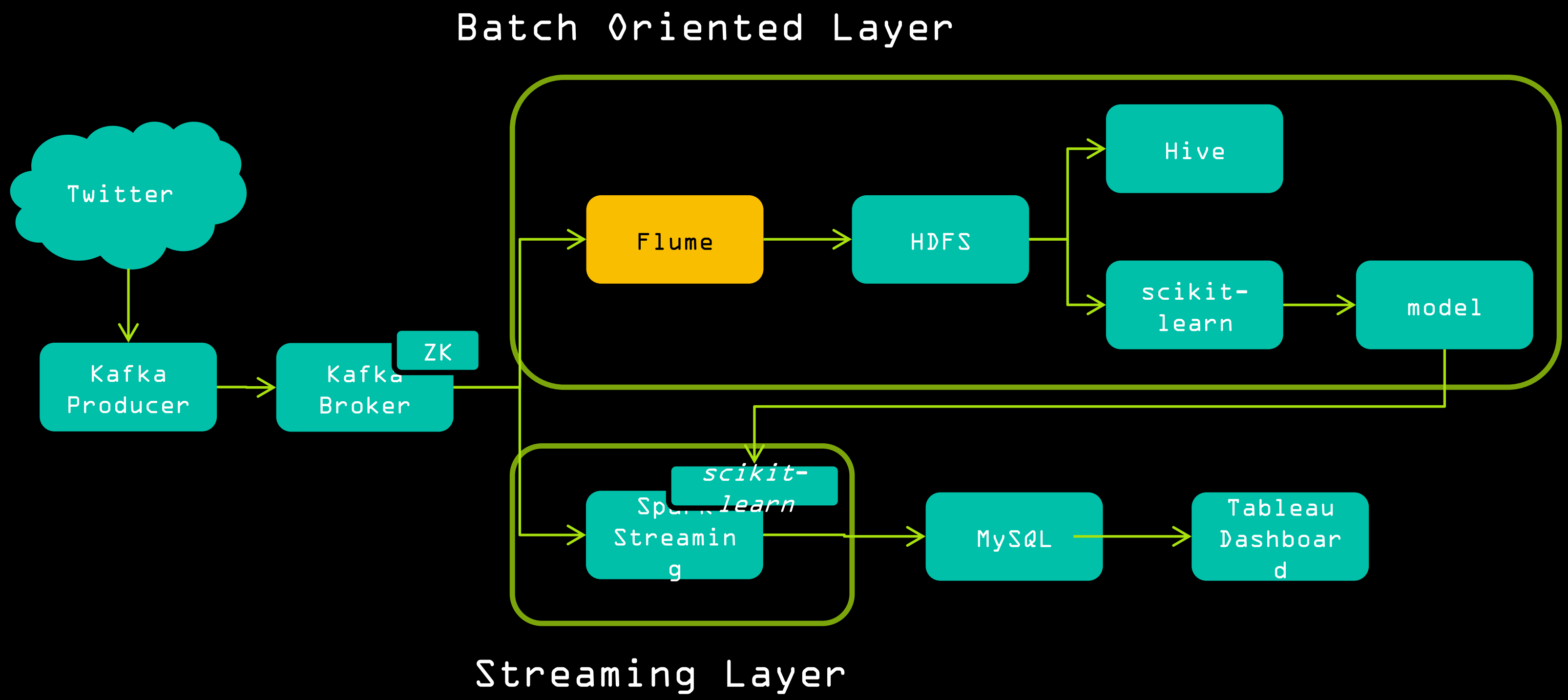
Apache Kafka
A high-throughput distributed messaging system.

- The Kafka Broker (http://kafka.apache.org/) provides a unified, high-throughput, low-latency platform for handling real-time data feeds. The design is heavily influenced by transaction logs. Kafka was developed by LinkedIn and is now a Top Level Project of the Apache Software Foundation.



- The previous slide mentions the Kafka producer. The Kafka consumers are the Flume and Spark Streaming in this scenario.

FLUME


Batch Oriented Layer
Twitter
Kafka Producer
Kafka Broker
ZK
Flume
HDFS
Hive
scikit-learn
model
scikit-learn
Spark Streaming
MySQL
Tableau Dashboard
Streaming Layer

FLUME

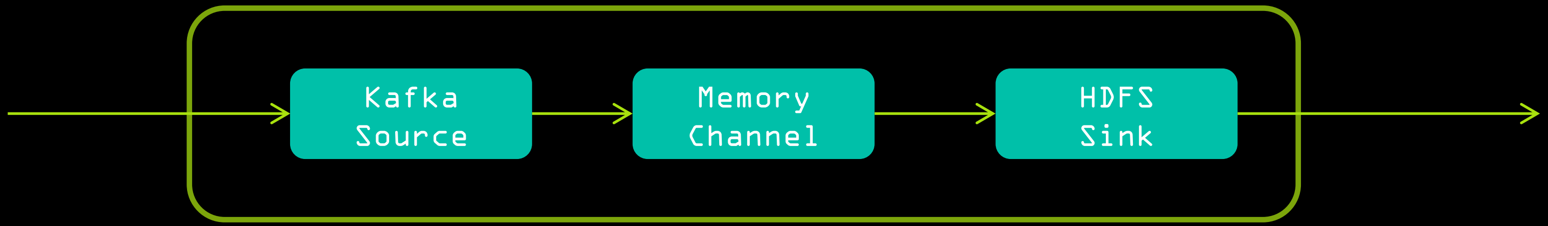
Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data.

Flume has better connectivity to the Big Data ecosystem like HDFS and HBase when compared to Kafka, so it makes it easy to develop applications in Flume.

But, Flume doesn't provide a

Flume is being configured to pull the messages from Kafka (http://flume.apache.org/FlumeUserGuide.html#kafka-source), pass them to the memory channel and the channel send it to the HDFS.

The next slide has the Flume configuration on how the different Flume components (Kafka Source, Memory Channel, HDFS Sink) are defined and chained together to the flow shown above.



FLUME CONFIGURATION FILE

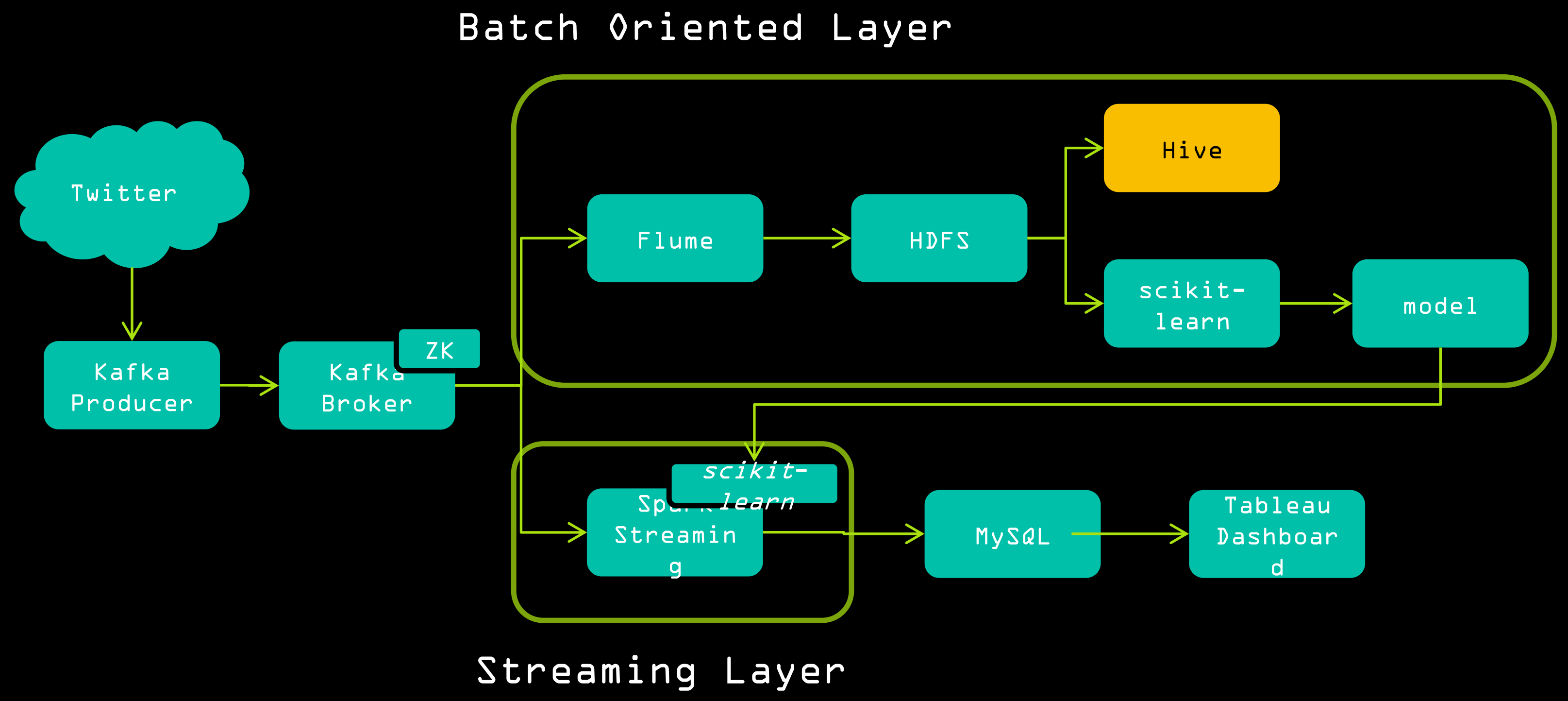
```
# Sources, channels, and sinks are
defined per
# agent name, in this case flume1.

flume1.sources  = kafka-source-1
flume1.channels = memory-channel-1
flume1.sinks    = hdfs-sink-1
```

```
# For each source, channel, and sink, set standard
properties.

flume1.sources.kafka-source-1.type =
org.apache.flume.source.kafka.KafkaSource
flume1.sources.kafka-source-1.zookeeperConnect =
localhost:2181
flume1.sources.kafka-source-1.topic = twitter-topic
flume1.sources.kafka-source-1.batchSize = 100
```

```
flume1.sources.kafka-source-1.channels = memory-
# Other properties are specific to each type
of
# source, channel, or sink. In this case, we
# specify the capacity of the memory channel.

flume1.channels.memory-channel-1.type   =
memory
flume1.channels.memory-channel-1.capacity =
```

```
#Specify the HDFS Sink properties

flume1.sinks.hdfs-sink-1.channel = memory-channel-1
flume1.sinks.hdfs-sink-1.type = hdfs
flume1.sinks.hdfs-sink-1.hdfs.writeFormat = Text
flume1.sinks.hdfs-sink-1.hdfs.fileType = DataStream
flume1.sinks.hdfs-sink-1.hdfs.filePrefix = tweets
flume1.sinks.hdfs-sink-1.hdfs.useLocalTimeStamp = true
flume1.sinks.hdfs-sink-1.hdfs.path =
hdfs://localhost:9000/user/analysis/tweets
flume1.sinks.hdfs-sink-1.hdfs.rollCount=10
flume1.sinks.hdfs-sink-1.hdfs.rollSize=0
flume1.sinks.hdfs-sink-1.hdfs.rollInterval=0
```

HIVE



brillio

HIVE

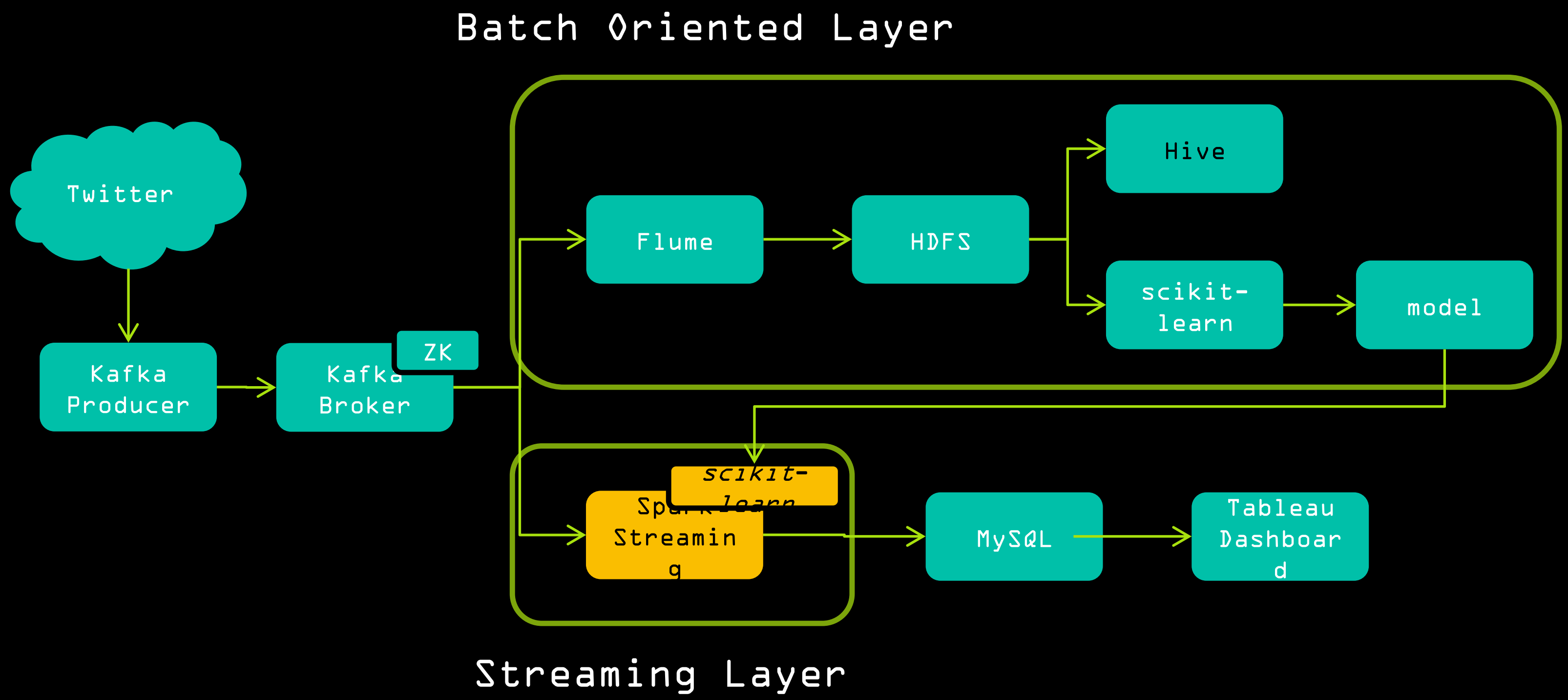
The tweets in HDFS are in JSON format. Hive which provides a SQL layer of abstraction is used to for analysis of the JSON tweets in a batch oriented fashion.

Hive out of the box can only understand csv, tsv and other types of data, but not the JSON data. So, a custom JSON (SerDe - Serializer Deserializer) has to be used for the same. A SerDe allows Hive to read the data from a table, and write it back to HDFS in any custom format.

https://cwiki.apache.org/confluence/display/Hive/SerDe

The queries through Hive are batch oriented in nature, so Spark Streaming has been used as discussed in the coming slides.

SPARK STREAMING



SPARK STREAMING

- Spark streaming makes it easy to build scalable fault-tolerant streaming applications.
- The main advantage of Spark streaming is that it lets reuse of the same code for batch processing as well as processing the streaming data.
- Spark had been configured to receive data from Kafka. More about the integration at http://spark.apache.org/docs/latest/streaming-kafka-integration.html.
- The Spark Streaming program has been developed in Python and uses Scikit learn to figure out the sentiment of the tweet in a real time fashion and populate the same in the

SPARK STREAMING CODE SNIPPET

```
#Create the Spark Context and Spark Streaming Context

    sc = SparkContext(master = "spark://Demo-Ubuntu:7077",
  appName="PythonStreamingKafkaWordCount")

    ssc = StreamingContext(sc, 1)



    #Get the tweets from Kafka, Spark Streaming is a consumer to the Kafka
  broker

    zkQuorum, topic = sys.argv[1:]

    kvs = KafkaUtils.createStream(ssc, zkQuorum, "spark-streaming-consumer",
  {topic: 1})



    #Figure out the sentiment of the tweet and write the same to MySQL

    kvs.map(lambda x: x[1]).map(classify_tweet).pprint()
```

brillio

EXAMPLE – GET HASHTAGS FROM TWITTER

```
kvs = KafkaUtils.createStream(ssc, zkQuorum, "spark-streaming-consumer", {topic: 1})
```

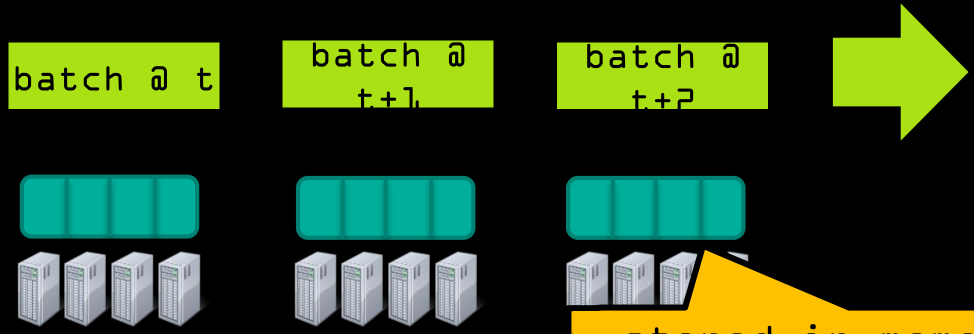
new DStream – a sequence of distributed datasets (RDDs) representing a distributed stream of data

ZooKeeper

Kafka Topic

Twitter Streaming API

tweets DStream

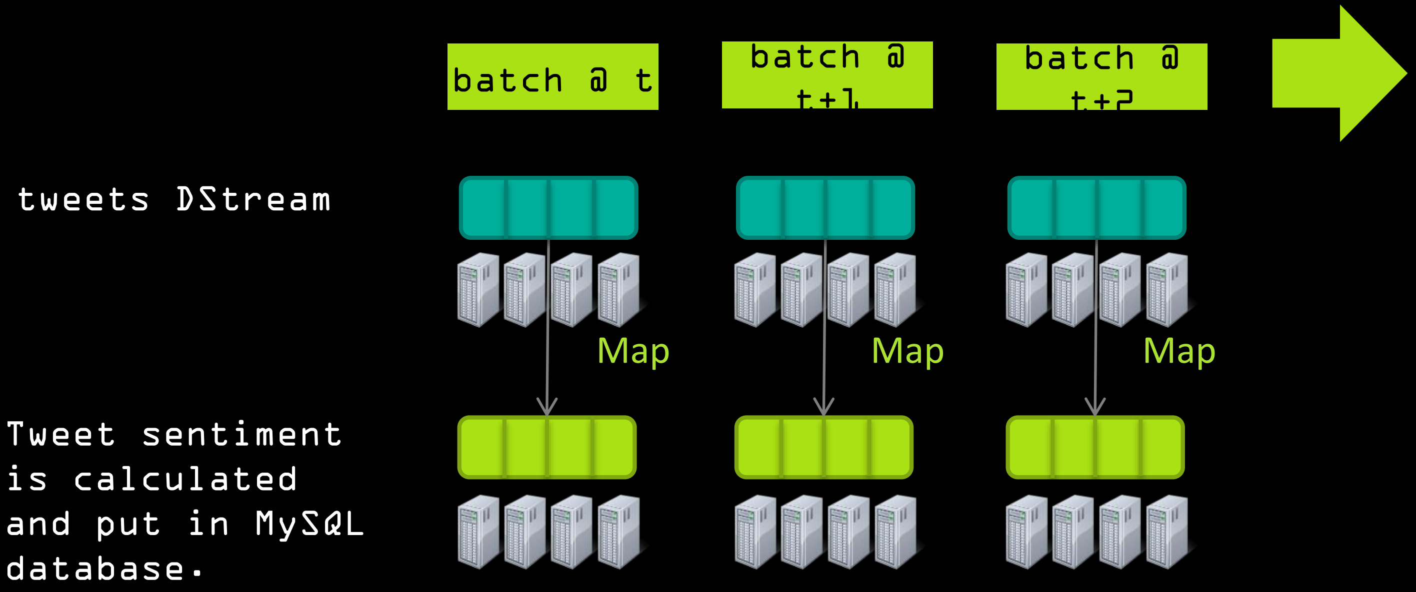


stored in memory as an RDD (immutable, distributed dataset)

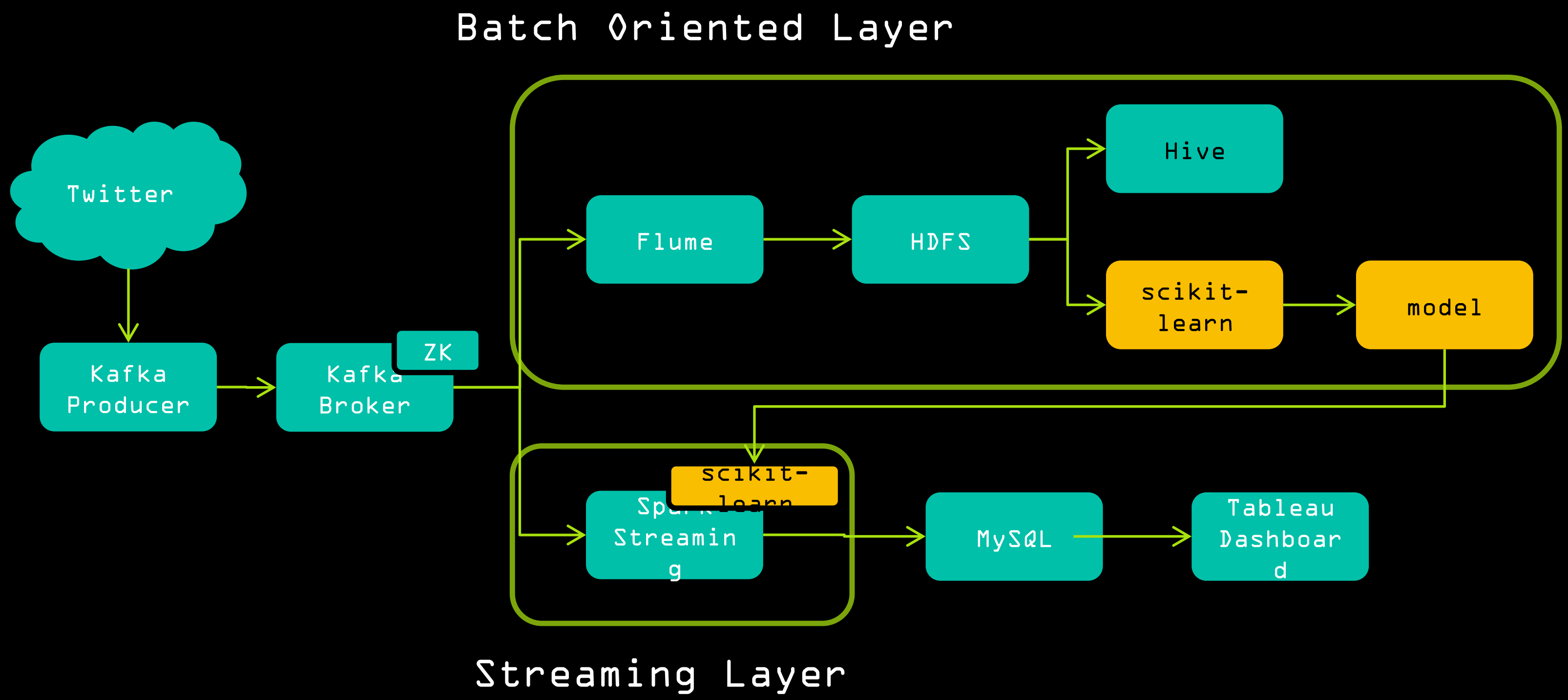
EXAMPLE - GET HASHTAGS FROM TWITTER

```
kvs.map(lambda x: x[1]).map(classify_tweet).pprint()
```

Transformation: the classify_tweet python function calculates the sentiment of the tweet and writes to MySQL



SCIKIT-LEARN



ANALYTICS MODEL

- In order to classify tweets as positive or negative, we built a model using the Random Forest classifier in Python's scikit-learn package
- Used a publically available dataset of pre-classified tweets as our training data set
- Extracted n-grams of the text (uni-grams, bi-grams and tri-grams), and created dummy variables from them, where 1 indicates that an n-gram is present in a tweet
- Examples of some n-grams are on the right. Here, the count indicates the number of times a string appears
- Using this, we can compute the TF-IDF to select important words

TF(t) = (Number of times term t appears) / (Total number of terms in the document)

IDF(t) = log_e(Total number of documents / Number of documents with term t in it)

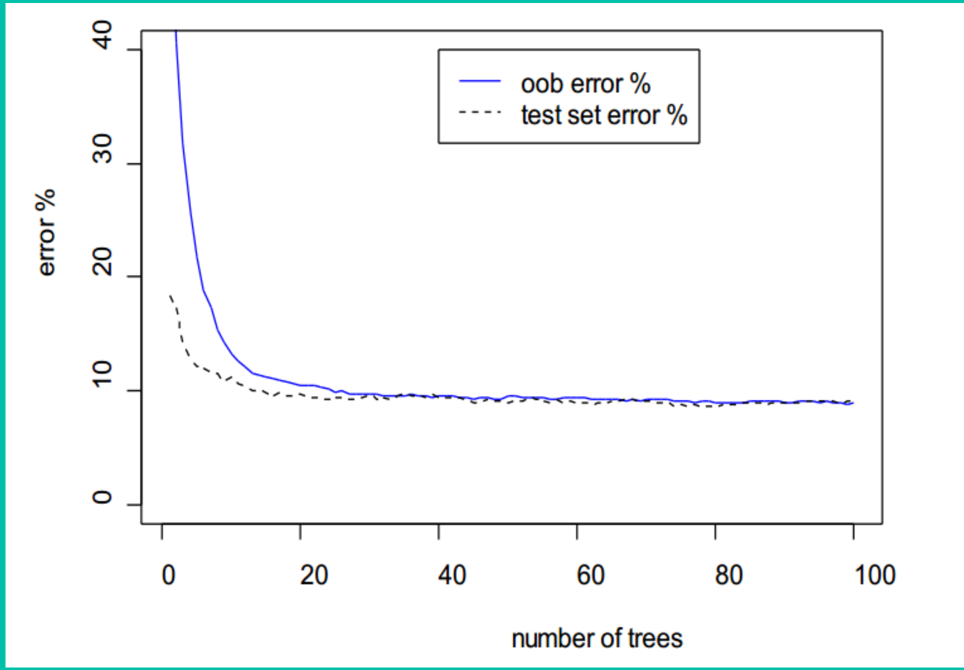
STRING	COUNT
not	3541
just	3406
was	3400
good	2771
like	2706
this	2651
no	2628
up	2557
now	2552
get	2535
all	2526
i have	2376
love	2342
lol	2323
do	2276
i can	2233
but i	2226
out	2150
what	2127
know	2100
and i	2019
i am	1977
i love	1925
go	1896
day	1867
don	1767
have a	1710
to be	1698
going to	1698
have to	1669
thanks	1661

ANALYTICS MODEL

- The data set contained around 1500 variables, out of which the Random Forest selected a maximum of 100 in each tree. There were 30 trees in total (this was the point at which the error rate converged)

- Split the data set in an 80:20 ratio for training and testing, and then trained the RF model on the 80% dataset. The 20% dataset was used for testing the results (sample is shown below)

- Using K-fold cross validation (with K=5), we



Sentiment	Negative	Positive	Count	Prediction Accuracy
Negative	3866	770	4636	83.4%
Positive	1261	4101	5362	76.5%
Count	5127	4871	9998	
Result Accuracy	75.4%	84.2%		79.7%

brillio

RANKING CANDIDATES ON THEIR SIMILARITIES

1. A similarity matrix is a transposed n x n matrix of n candidates, where each candidate has a score with respect to another
2. A candidate has the maximum score with himself, and a gradually decreasing score with candidates less like him/her
3. The scoring used some of the following metrics :
   - Overall popularity of the candidate
   - The positive sentiment towards the candidate

Chris Christie has a max score with himself

NodeName	Affiliate_Party	Candidate1	Candidate2	Relationship	LineX	LineY	CircleY	Similarity_Score
Mark Everson	Republican	Mark Everson	Jim Gilmore	Mark Everson --> Jim Gilmore	4600	6100	6100	3.14285714286
Ted Cruz	Republican	Ted Cruz	Donald Trump	Ted Cruz --> Donald Trump	4100	6400	6400	4.0
Ted Cruz	Republican	Ted Cruz	Marco Rubio	Ted Cruz --> Marco Rubio	4100	6400	6400	5.0
Chris Christie	Republican	Chris Christie	Chris Christie	Chris Christie --> Chris Christie	3600	6600	6600	6.0
Rick Perry	Republican	Rick Perry	Mark Everson	Rick Perry --> Mark Everson	3000	2300	2300	4.01851851852
Rick Perry	Republican	Rick Perry	Martin O'Malley	Rick Perry --> Martin O'Malley	3000	2300	2300	3.11111111111
Rick Perry	Republican	Rick Perry	Jim Gilmore	Rick Perry --> Jim Gilmore	3000	2300	2300	3.00255102041
Mike Huckabee	Republican	Mike Huckabee	Mark Everson	Mike Huckabee --> Mark Everson	5300	3800	3800	4.01851851852
Mike Huckabee	Republican	Mike Huckabee	Jim Gilmore	Mike Huckabee --> Jim Gilmore	5300	3800	3800	3.00257069409
Mike Huckabee	Republican	Mike Huckabee	George Pataki	Mike Huckabee --> George Pataki	5300	3800	3800	4.5
Mike Huckabee	Republican	Mike Huckabee	Lincoln Chafee	Mike Huckabee --> Lincoln Chafee	5300	3800	3800	3.5
Ted Cruz	Republican	Ted Cruz	Mark Everson	Ted Cruz --> Mark Everson	4100	6400	6400	4.02222222222
Ted Cruz	Republican	Ted Cruz	Martin O'Malley	Ted Cruz --> Martin O'Malley	4100	6400	6400	3.14285714286

MYSQL DATABASE

1. The output of the analytic model, and the classification of tweets into positive and negative sentiments, are stored in a table on the database
2. The sentiment matrix is also stored as another table on the database
3. The two tables are linked by the Candidate's name
4. Whenever a new tweet is streamed in real-time, both tables are updated

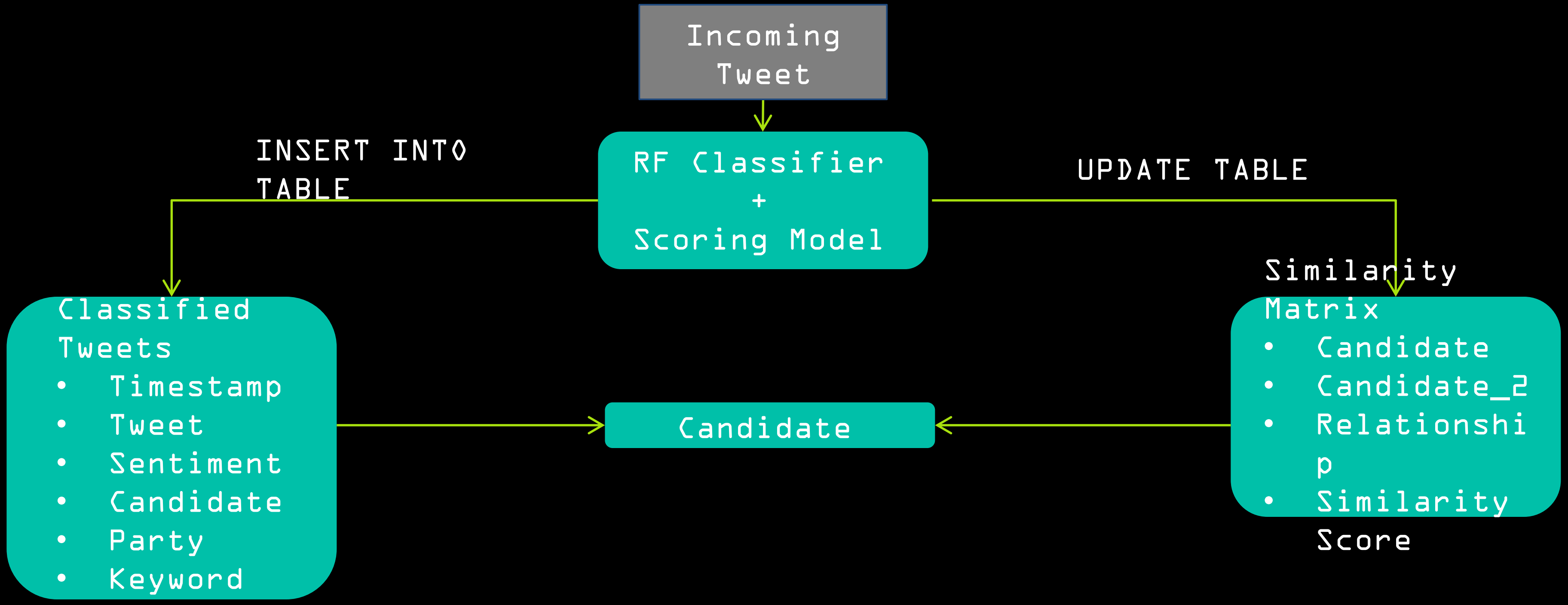


TABLEAU DASHBOARD

1. A connection is created between the database and Tableau, which gets refreshed in real-time when the database gets updated

2. The front page gives an overview of tweet counts, tracks the daily number of tweets, most popular issues being discussed, and the number of tweets for each candidate

The Tweet Counter shows the number of tweets each party receives, and the break-up of

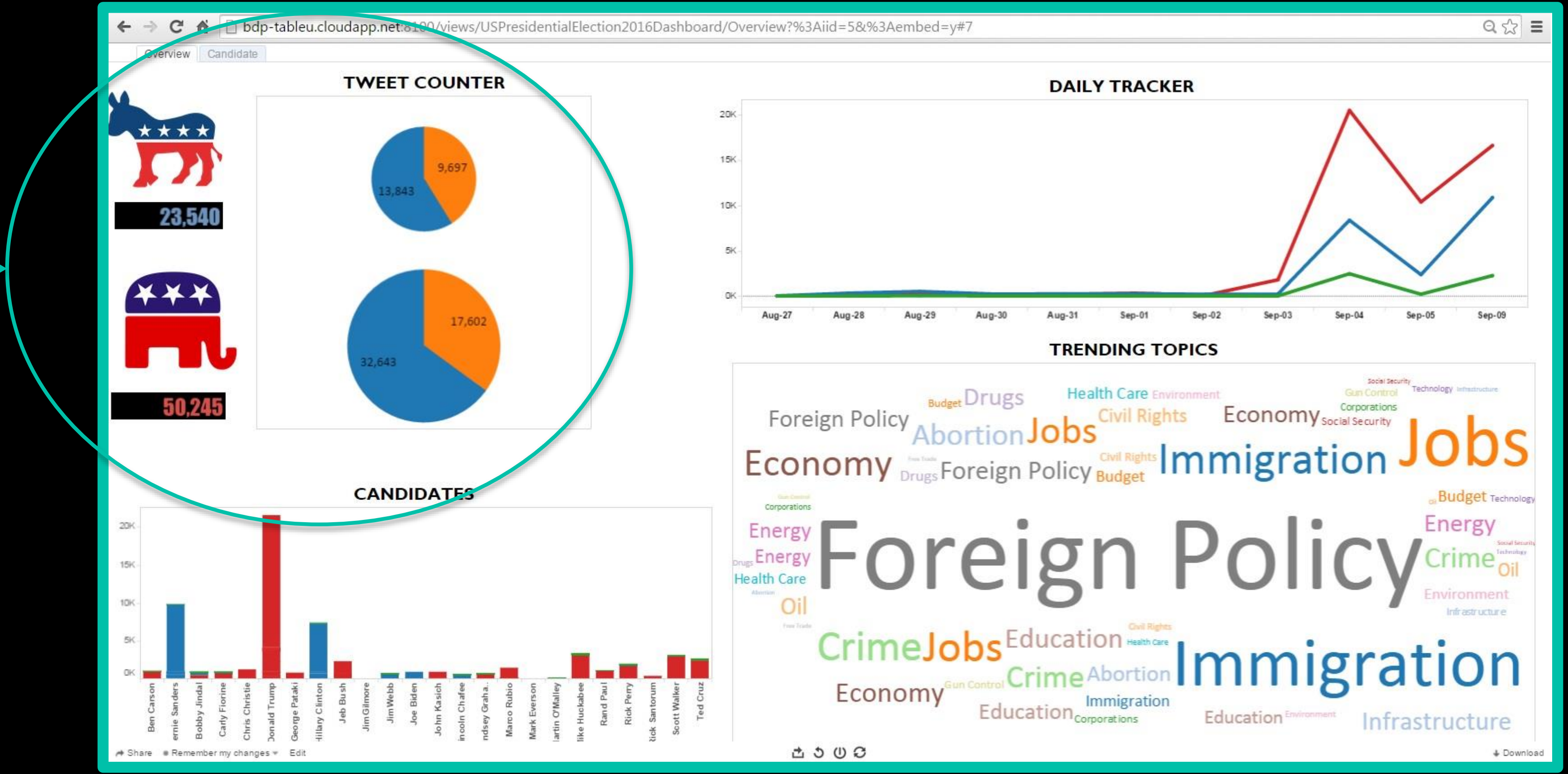


TABLEAU DASHBOARD

- Clicking on the positive or negative portion of the party's pie chart, will drill down each of the graphs.
- Here, we select the Positives of the Democrats, and can see their daily progress, their most important issues, and how their candidates are faring

Selecting Democrats filters the word cloud and candidate plots

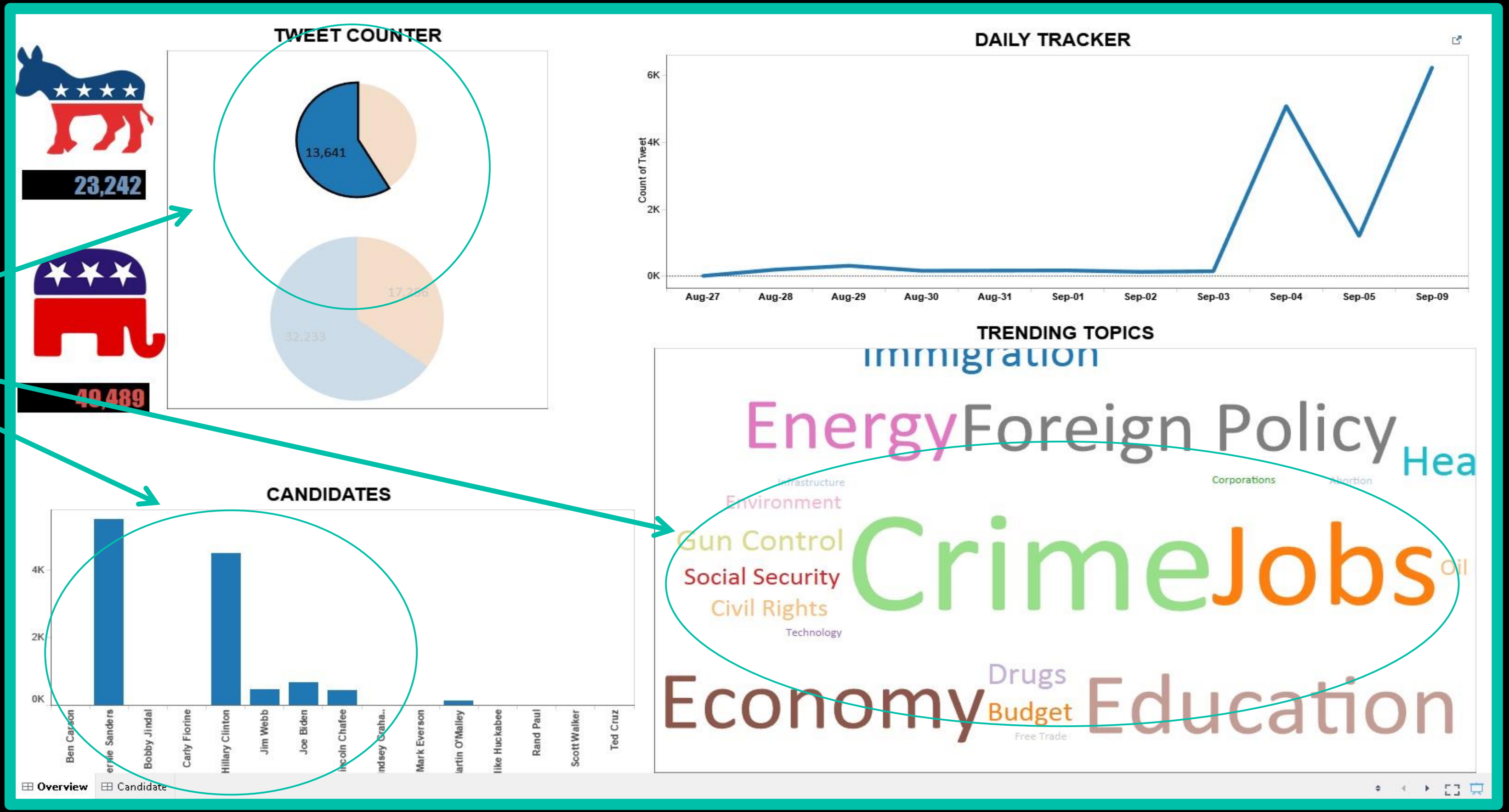


TABLEAU DASHBOARD

- If we click on a word in the word cloud, we are able to view all candidates who are talking about that issue
- We can even see the positive and negative reception of the candidate towards that issue

Clicking on Jobs displays all candidates talking about Jobs

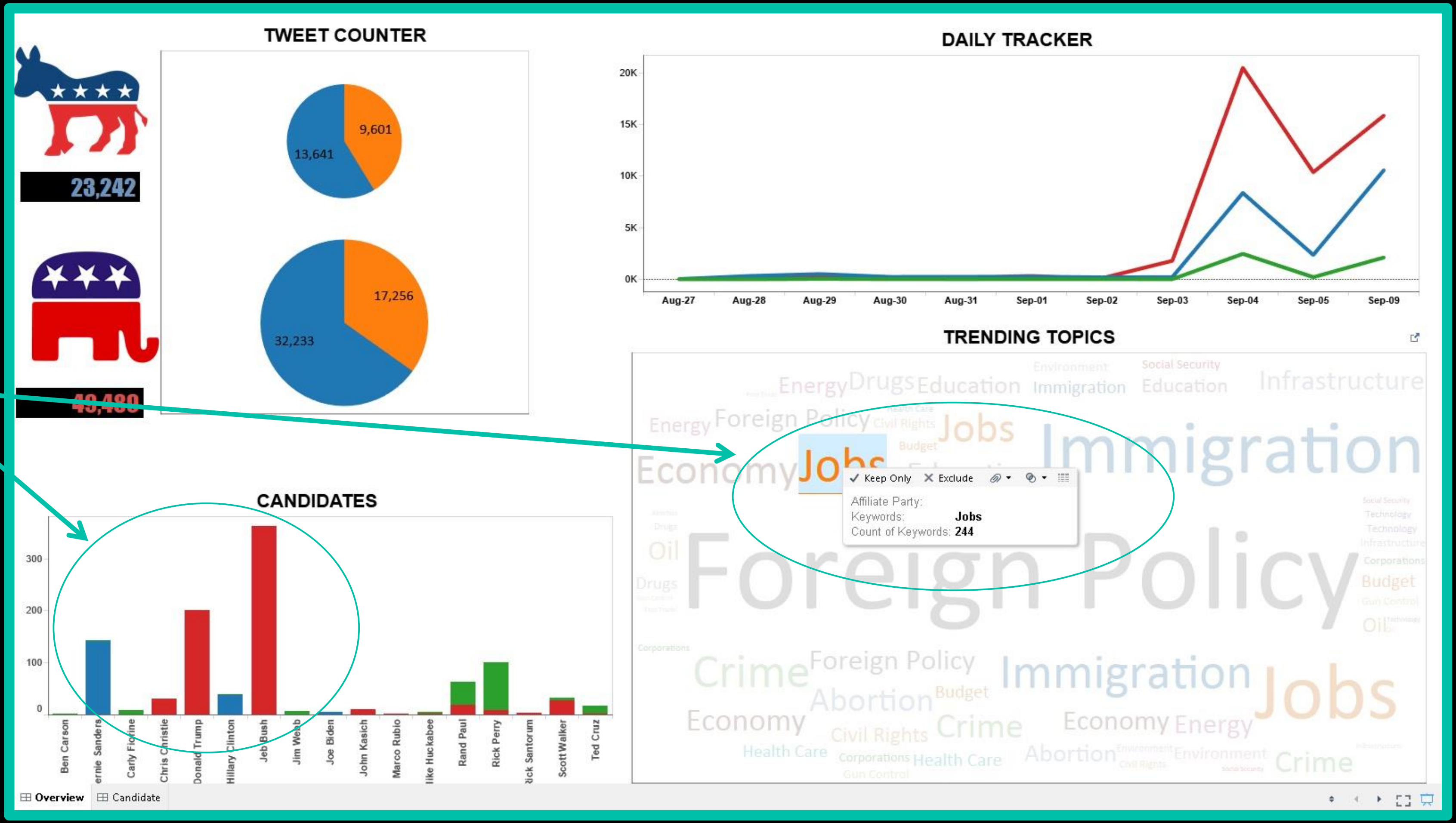


TABLEAU DASHBOARD

- The second page is network graph of candidates, where each one is linked to the other based on their similarity score
- The score ranges from 0 to 6, with the links varying from red to green
- Using the buttons on the top right, we can see the candidates that are most alike

Displays the similar candidate links. Here, Joe Biden is similar to Lincoln Chafee and Jim Webb, but dissimilar to Hillary Clinton

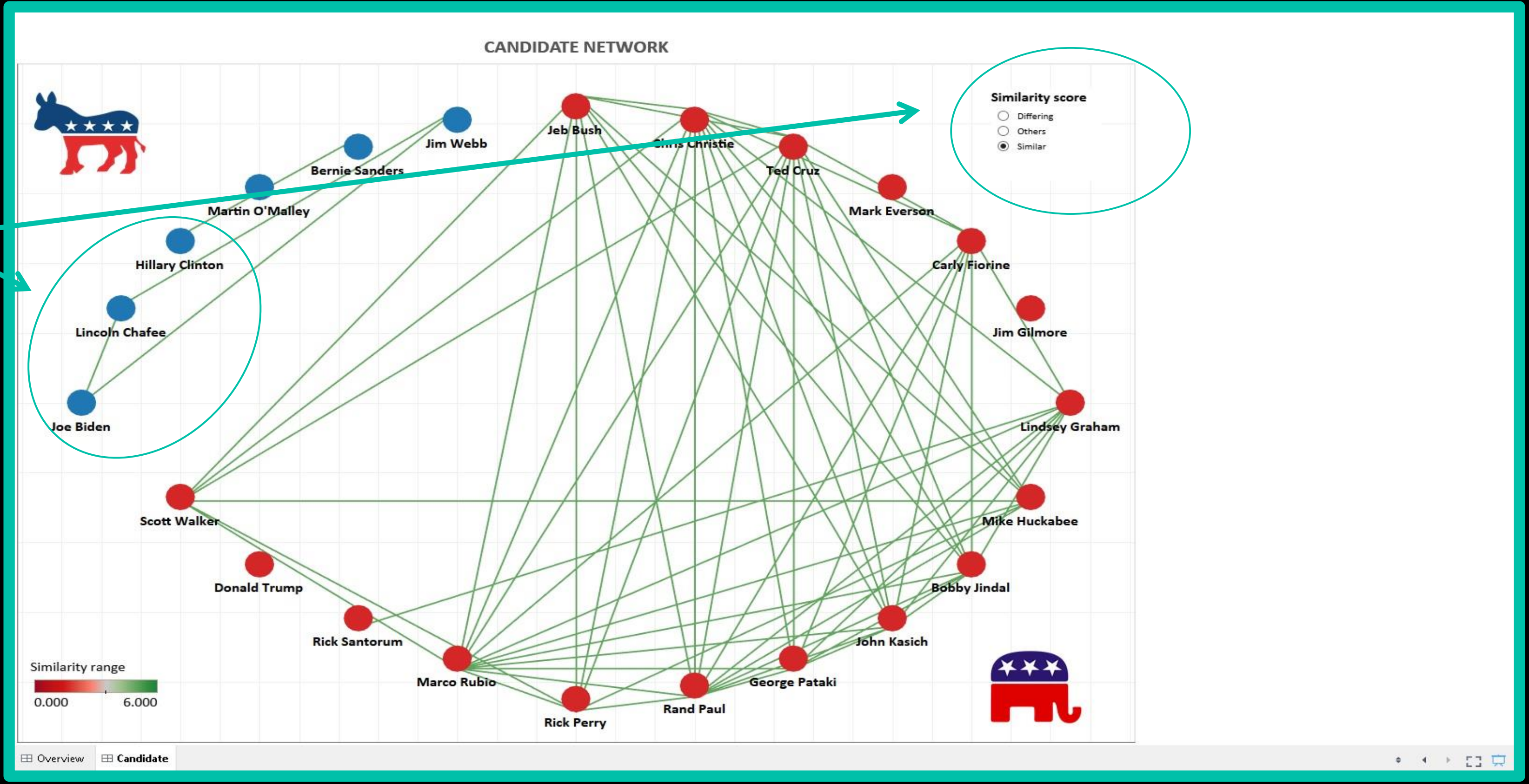


TABLEAU DASHBOARD

- Clicking on a candidates bubble, displays all the other candidates that are similar to him

Clicking on Joe Biden displays his information, and all the candidates similar to him

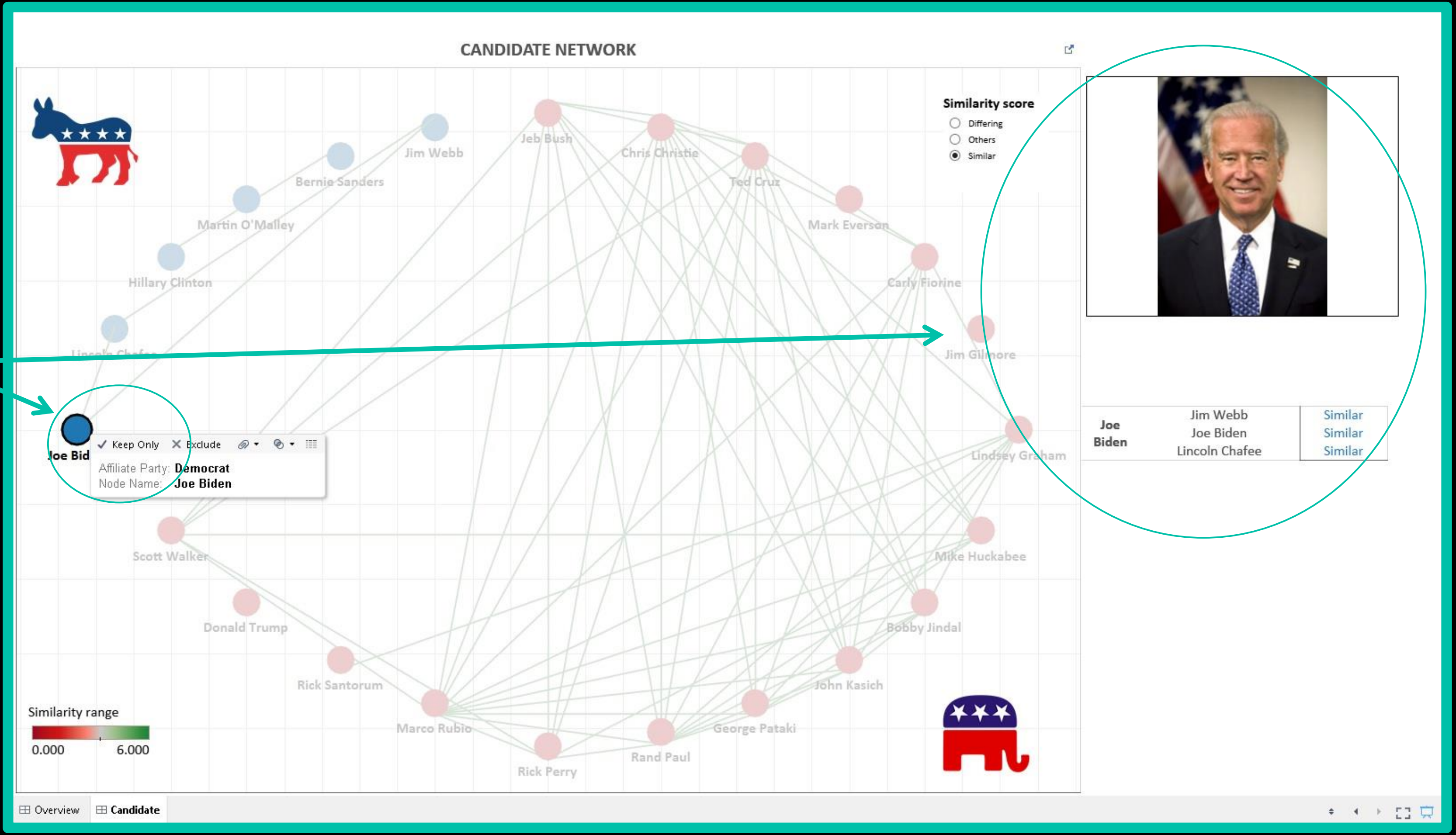
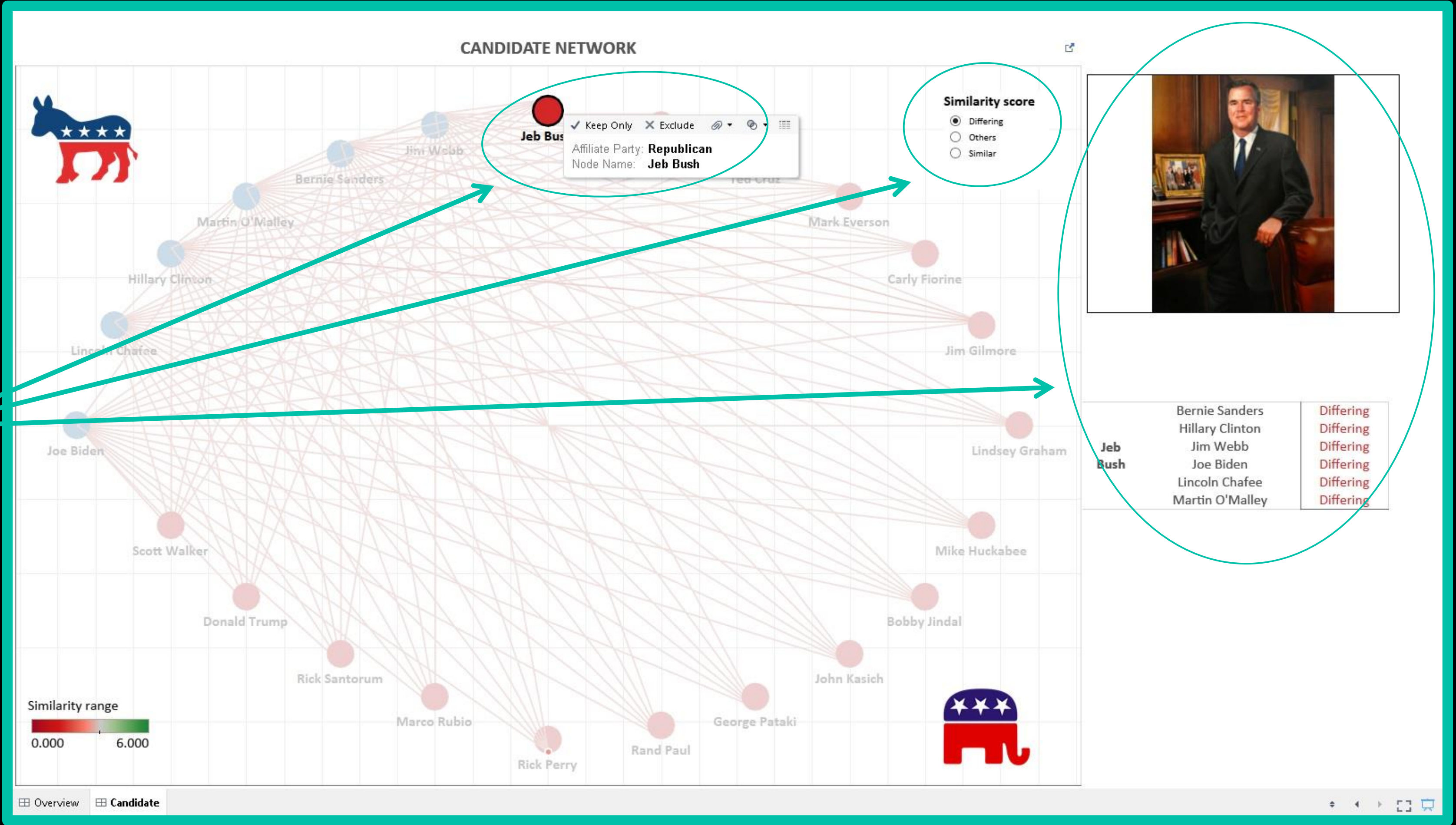


TABLEAU DASHBOARD

- Similarly, we can change the view and see the candidates most unlike a particular candidate, by toggling the button on the top rightZ

Dissimilar candidates, and select Jeb Bush, we see his information displayed, and all the candidates most



THANK YOU

Predictive Analytics & Business Insights Team
All Attendees
Everyone on the Next slide…..

REFERENCES AND CITATION FOR THIS PRESENTATION

My personal thanks to Praveen Sripati, Jim D'Souza, Sandeep Devagiri, Deepak Vinoth, Divya, Sanisha, Mayank Pant, Srinivas Guntur and rest of team at Brillio who made this presentation and demo possible.

1. Spark Streaming. Apache Software Foundation. http://spark.apache.org/streaming
2. http://www.wiziq.com/blog/hype-around-apache-spark/
3. databricks.com/blog/2014/11/05/spark-officially-sets-a-new-record-in-large-scale-sorting.html
4. http://preview.tinyurl.com/o983fmw
5. http://preview.tinyurl.com/nugnz6t Big Data: Principles and best practices of scalable real-time data systems
6. Hausenblas, Michael, and Nathan Bijnens. "Lambda Architecture". N.p., 2015. http://lambda-architecture.net
7. "The Log" from Jay Kreps http://preview.tinyurl.com/qc43s5j
8. jjmalina/pygotham-2015
9. http://spark.apache.org/
10. http://kafka.apache.org/
11. http://www.slideshare.net/spark-project/deep-divewithsparkstreaming-tathagatadassparkmeetup20130617
12. http://www.michael-noll.com/blog/2014/10/01/kafka-spark-streaming-integration-example-tutorial/#what-is-spark-streaming
13. http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-259.html Discretized Streams: A Fault-Tolerant Model for Scalable Stream