INTRODUCTION TO BIG DATA & LAMBDA ARCHITECTURE

Author:

CHAITANYA PRASHAR

https://github.com/chaitanya-prashar/Lambda-Spark

Table of content

1 Introduction to BIG DATA

2 Lambda Architecture

3 Demo

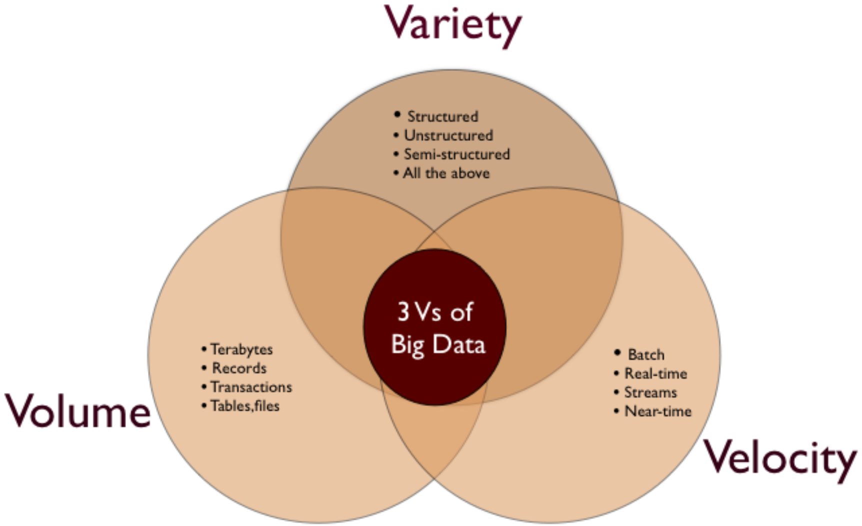
Introduction to BIG DATA

- What is BIG DATA?
- Need for Distributed Computing
- Architecture
- Example MapReduce
- Java or Scala
- Challenges with big data

Introduction to BIG DATA

What is BIG DATA

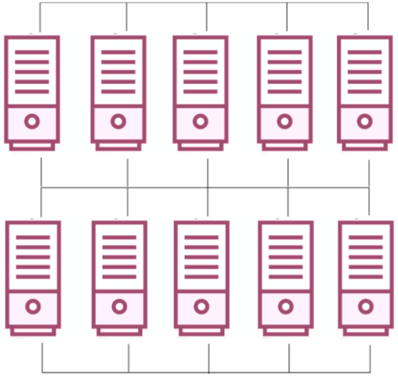
- Google
  - Storage – 15 Exabytes
  - Process per day – 100 Petabytes
  - Searched per second – 2.3 million
- Facebook
  - Storage is 300 Petabytes and
  - 600 Terabytes processed per day.
- NSA
  - Current Storage - 5 Exabytes
  - Processed per day - 30 Petabytes



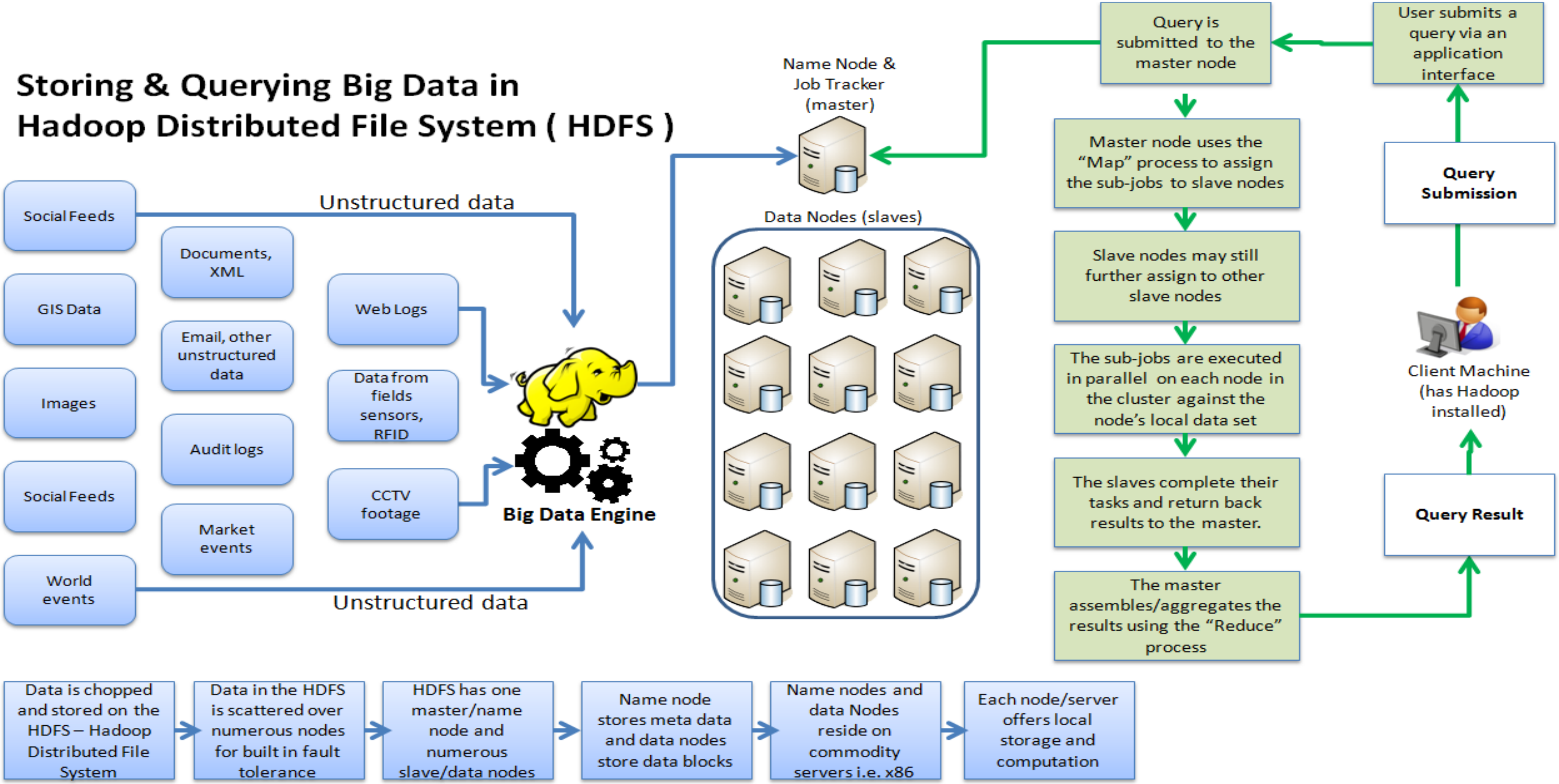
Introduction to BIG DATA

Need for Distributed Computing

- We need a system which..
  - can handle massive amounts of data.
  - can process it in a timely manner.
  - can scale easily when it grows.
  - Traditional databases can not do it.
  - Distributed systems like Hadoop were developed for exactly this.
- Google Introduced 2003:
  - Google File System: To solve distributed storage
  - MapReduce: To solve distributed computing
- Apache Introduced open sources of these technologies and named:
  - HDFS: Hadoop distributed file system, A file system to manage the storage of data.
  - MapReduce: A framework to process data accross multiple servers.



Introduction to BIG DATA



Storing & Querying Big Data in Hadoop Distributed File System (HDFS)

Social Feeds

GIS Data

Images

Social Feeds

World events

Documents, XML

Email, other unstructured data

Audit logs

Market events

Web Logs

Data from fields sensors, RFID

CCTV footage

Unstructured data

Unstructured data

Big Data Engine

Name Node & Job Tracker (master)

Data Nodes (slaves)

User submits a query via an application interface

Query is submitted to the master node

Master node uses the "Map" process to assign the sub-jobs to slave nodes

Slave nodes may still further assign to other slave nodes

The sub-jobs are executed in parallel on each node in the cluster against the node's local data set

The slaves complete their tasks and return back results to the master.

The master assembles/aggregates the results using the "Reduce" process

Query Result

Client Machine (has Hadoop installed)

Query Submission

Data is chopped and stored on the HDFS – Hadoop Distributed File System

Data in the HDFS is scattered over numerous nodes for built in fault tolerance

HDFS has one master/name node and numerous slave/data nodes

Name node stores meta data and data nodes store data blocks

Name nodes and data Nodes reside on commodity servers i.e. x86

Each node/server offers local storage and computation

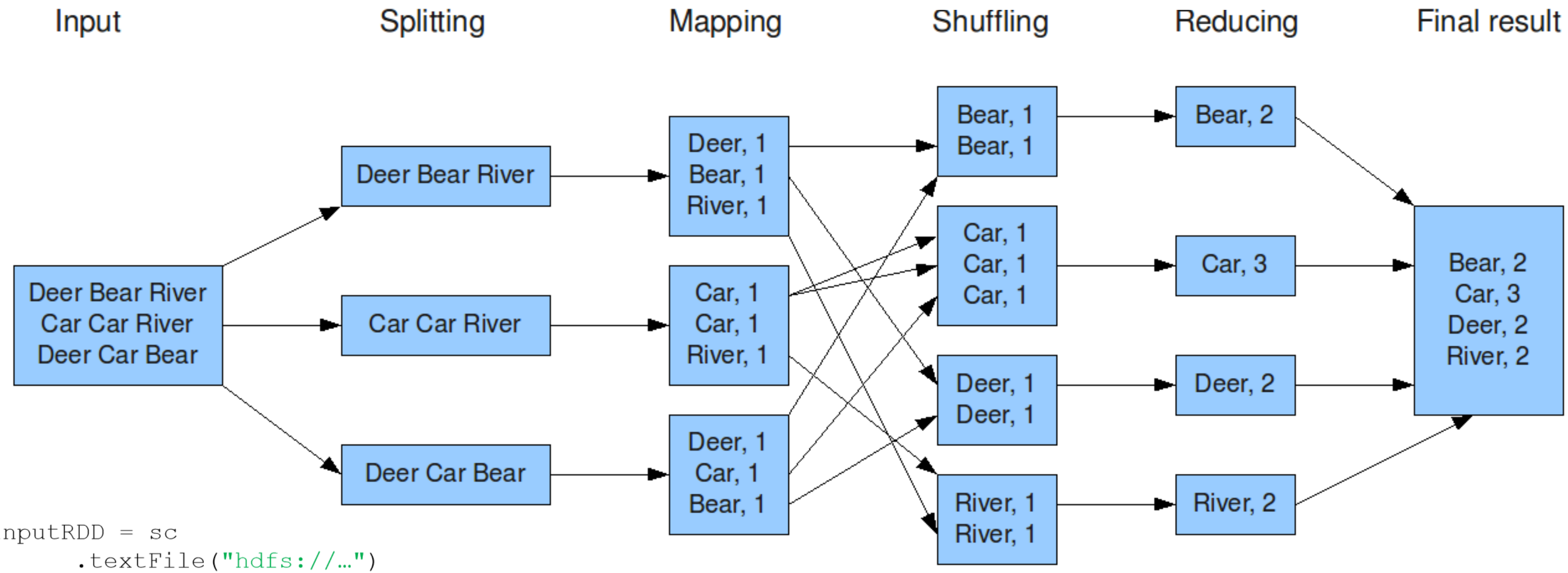
Introduction to BIG DATA

MapReduce

```
saveAsTextFile("hdfs:// … ")
```

```
val wordsRDD = inputRDD
  .flatMap(_.split(" "))
  .map(word => (word, 1))
```

```
val wordCountsRDD = wordsRDD
                      .reduceByKey(_ + _)
```



Input

Splitting

Mapping

Shuffling

Reducing

Final result

Deer Bear River
Car Car River
Deer Car Bear

Deer Bear River

Car Car River

Deer Car Bear

Deer, 1
Bear, 1
River, 1

Car, 1
Car, 1
River, 1

Deer, 1
Car, 1
Bear, 1

Bear, 1
Bear, 1

Car, 1
Car, 1
Car, 1

Deer, 1
Deer, 1

River, 1
River, 1

Bear, 2

Car, 3

Deer, 2

River, 2

Bear, 2
Car, 3
Deer, 2
River, 2

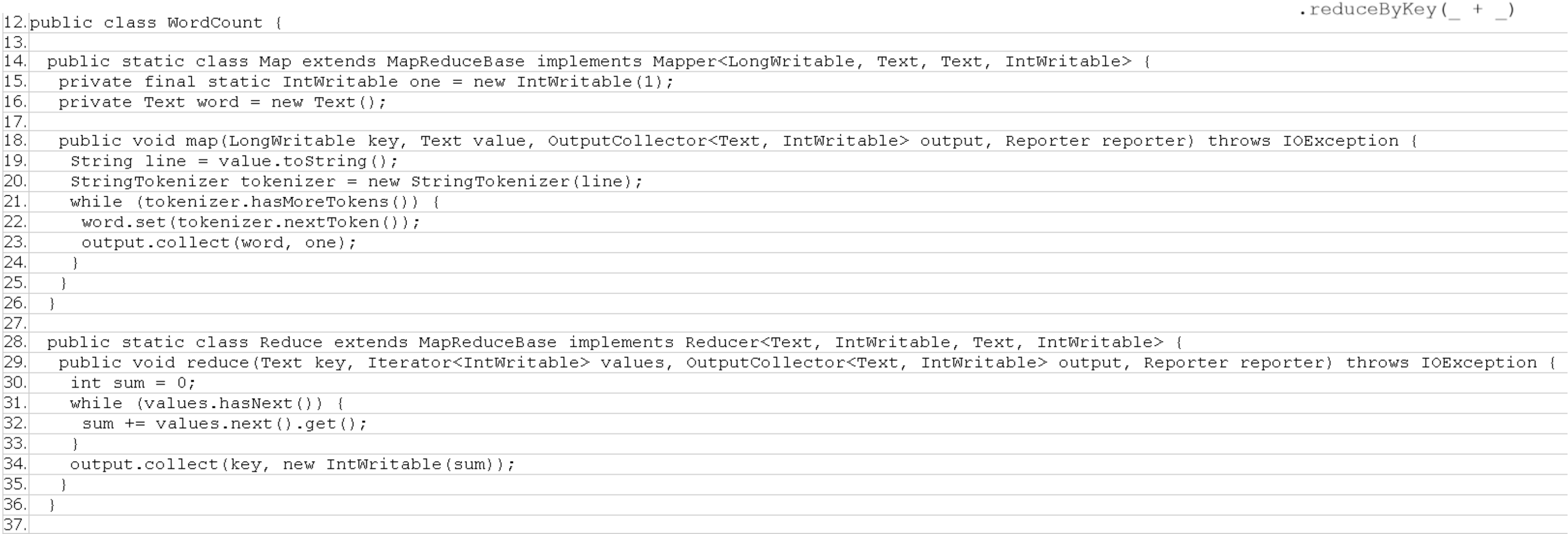
```
val inputRDD = sc
          .textFile("hdfs://…")
```

Introduction to BIG DATA

Java Code

Scala

```scala
val inputRDD = sc
          .textFile("hdfs://…")

val wordsRDD = inputRDD
  .flatMap(_.split(" "))
  .map(word => (word, 1))

val wordCountsRDD = wordsRDD
                      .reduceByKey(_ + _)
```



```java
public class WordCount {

 public static class Map extends MapReduceBase implements Mapper<LongWritable, Text, Text, IntWritable> {
  private final static IntWritable one = new IntWritable(1);
  private Text word = new Text();

  public void map(LongWritable key, Text value, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
   String line = value.toString();
   StringTokenizer tokenizer = new StringTokenizer(line);
   while (tokenizer.hasMoreTokens()) {
    word.set(tokenizer.nextToken());
    output.collect(word, one);
   }
  }
 }

 public static class Reduce extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {
  public void reduce(Text key, Iterator<IntWritable> values, OutputCollector<Text, IntWritable> output, Reporter reporter) throws IOException {
   int sum = 0;
   while (values.hasNext()) {
    sum += values.next().get();
   }
   output.collect(key, new IntWritable(sum));
  }
 }
```

Introduction to Big Data

Challenges with big data….

- Big Data Batch Processing – is not as quick as a real-time , which eventually leads to business users or customers asking to get immediate or near real-time insight, such as the most recent data updates to react faster to market changes.
- Big Data Streaming – No track of records. No recovery from the old data.

Table of content

1 Introduction to BIG DATA

2 Lambda Architecture

3 Demo

Lambda Architecture

- Why we need Lambda Architecture?
- Lambda Architecture
- Batch Layer
- Speed Layer
- Serving Layer
- Criticism

Lambda Architecture

Why we need Lambda Architecture ?

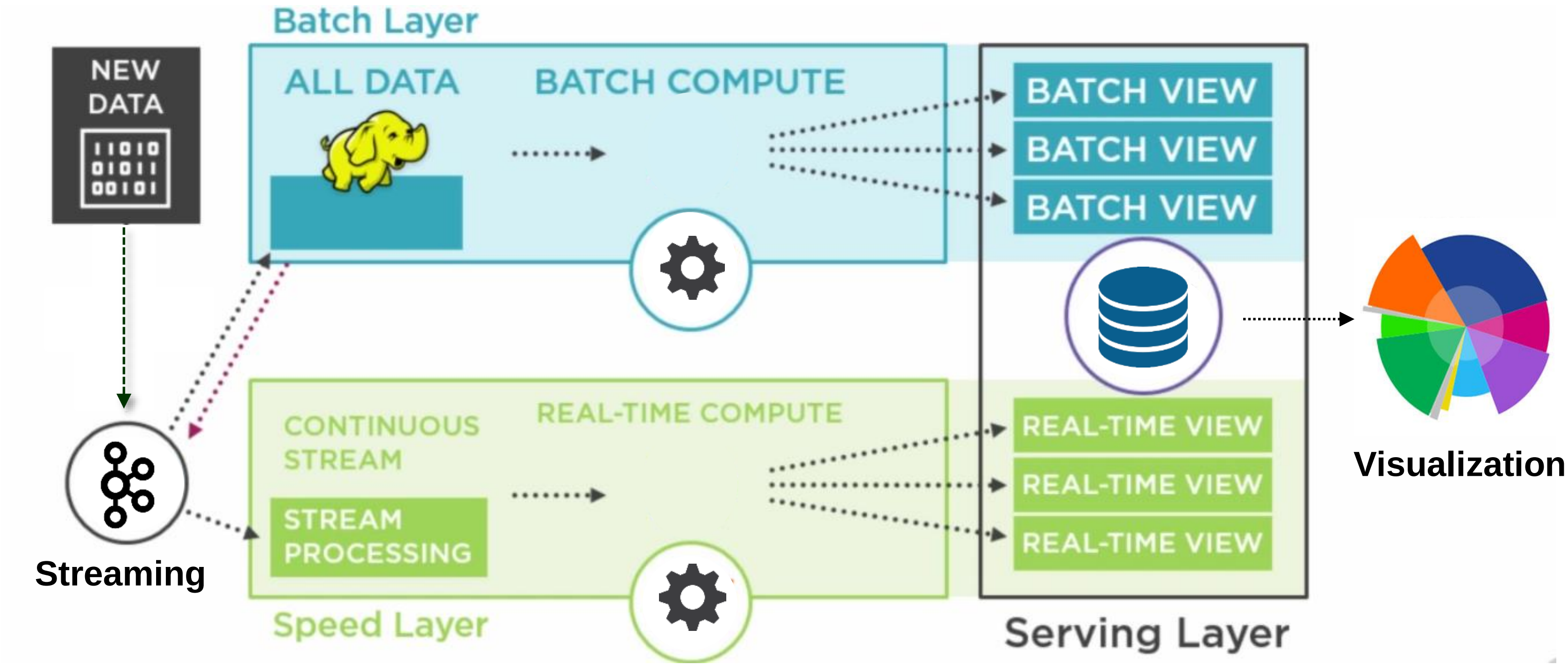
- We need a System which is robust to scalability as well as fault-tolerant, be it a human or machine fault-tolerant.
- The human fault-tolerance of the batch system is as good as you can get. There are only two mistakes a human can make in a system like this: deploy a buggy implementation of a query or write bad data.
- If you deploy a buggy implementation of a query, all you have to do to fix things is fix the bug, deploy the fixed version, and recompute everything from the master dataset. This works because queries are pure functions.
- Likewise, writing bad data has a clear path to recovery: delete the bad data and precompute the queries again. Since data is immutable and the master dataset is append-only, writing bad data does not override or otherwise destroy good data.
- We need a System which can have Low latency as well as a High accuracy.

YAHOO!

NETFLIX

CISCO

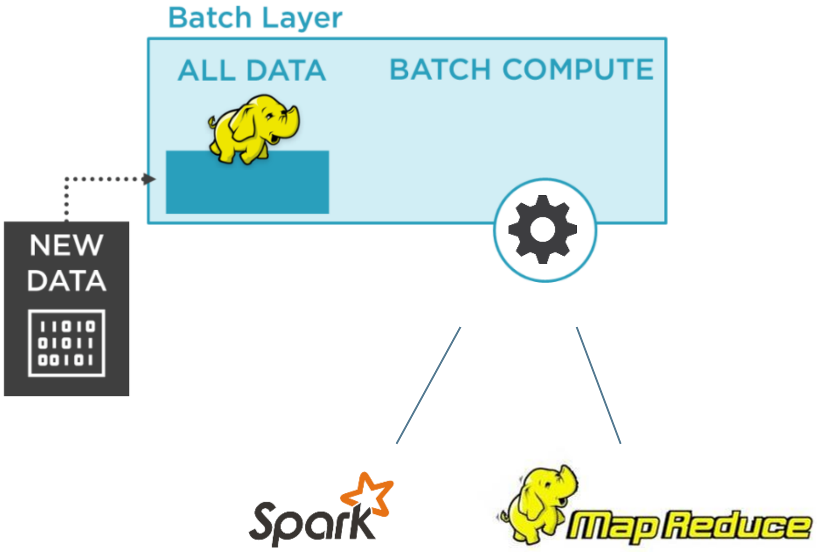
Lambda Architecture



Lambda Architecture

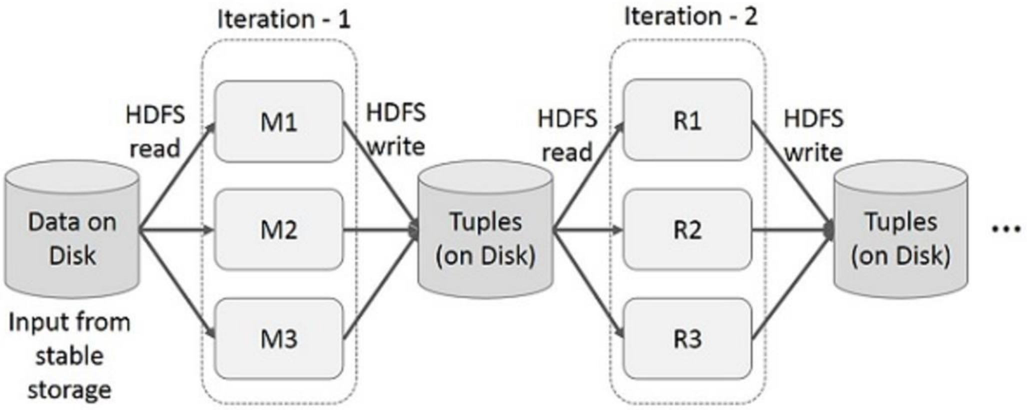
Batch Layer

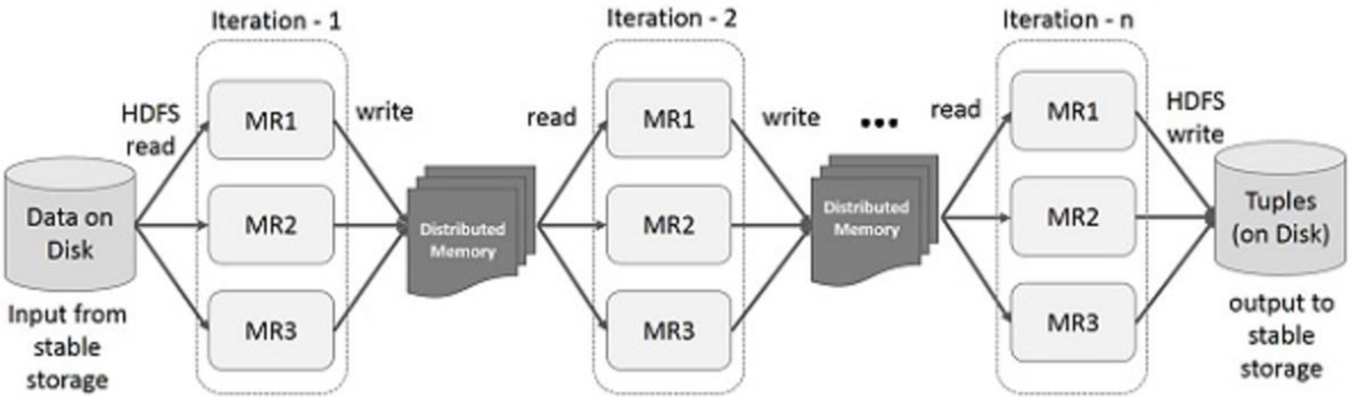
- Takes input, stores it in distributed system, process the data and sends to the serving layer.
- The foremost characteristic of this layer is that it holds the master data.
- So whatever the source of your data, it lands here, untouched, unscathed, in an immutable append-only fashion.
- This is your record of truth for your entire dataset.



Lambda Architecture

Hadoop vs Spark

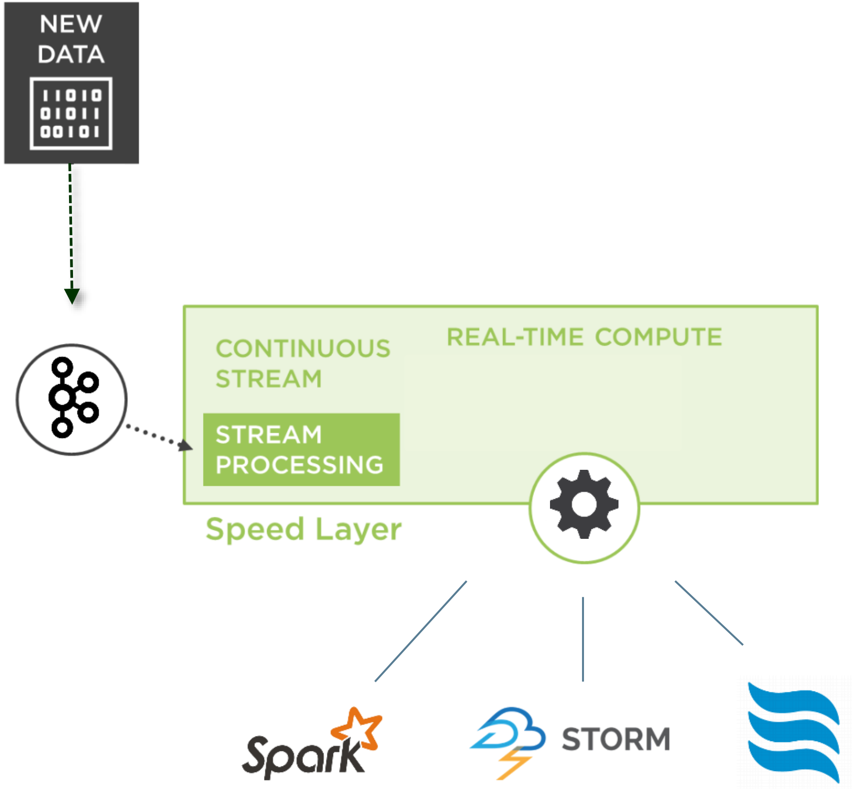




Lambda Architecture

Speed Layer

- Takes the input from a streaming source, does the transformations and sends it to the serving layer.
- The speed layer processes data streams in real time and without the requirements of fix-ups or completeness.
- This layer sacrifices throughput as it aims to minimize latency by providing real-time views into the most recent data.
- Essentially, the speed layer is responsible for filling the "gap" caused by the batch layer's lag in providing views based on the most recent data.



Lambda Architecture

Apache Spark vs Apache Storm…

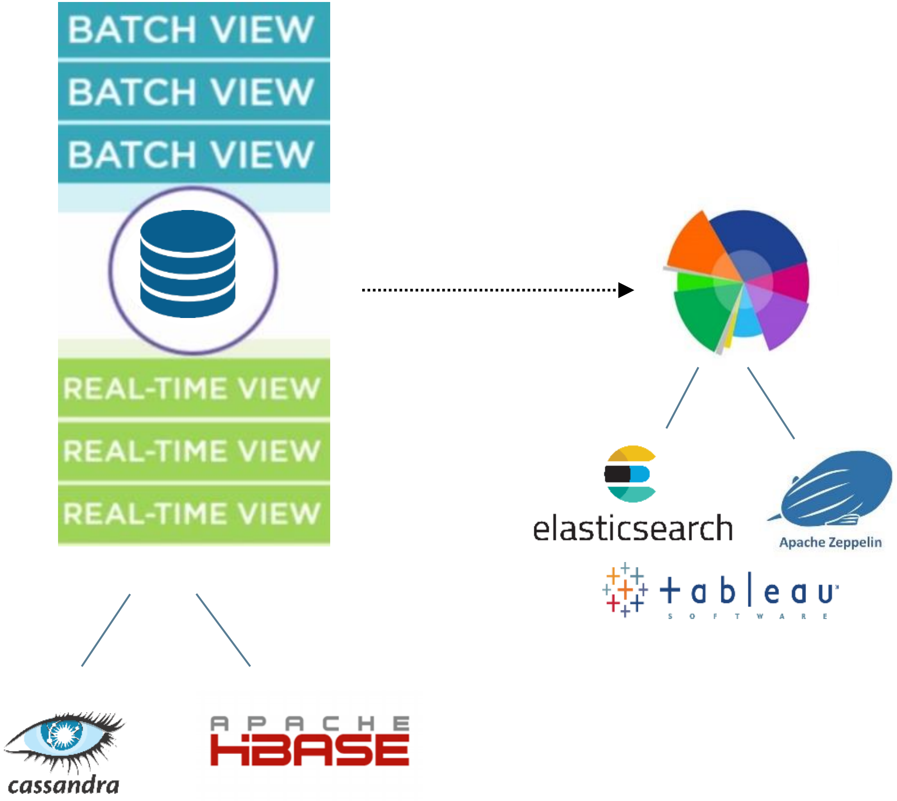
Spark Streaming	Others (Apache Storm etc.)
Moderate Latency	Single Record at a Time; Very Low Latency
Relies on RDDs (Delivery Guarantees)	Different Systems
Higher Throughput	Continuous Operator Model
Same Core as Batch	Different Systems for Batch and Streaming
Excellent for Lambda Architectures	Higher Total Cost of Ownership

*RDD:

Lambda Architecture

Serving Layer

- Output from the batch and speed layers are stored in the serving layer, which responds to ad-hoc queries by returning precomputed views or building views from the processed data.
- Batch transformations and speed layer transformations are stored in the database and further can be joined to present to the end-user.



Lambda Architecture

Criticism of lambda architecture

- Complexity
  - The batch and streaming sides each require a different code base that must be maintained and kept in sync so that processed data produces the same result from both paths.

Table of content

1 Introduction to BIG DATA

2 Lambda Architecture

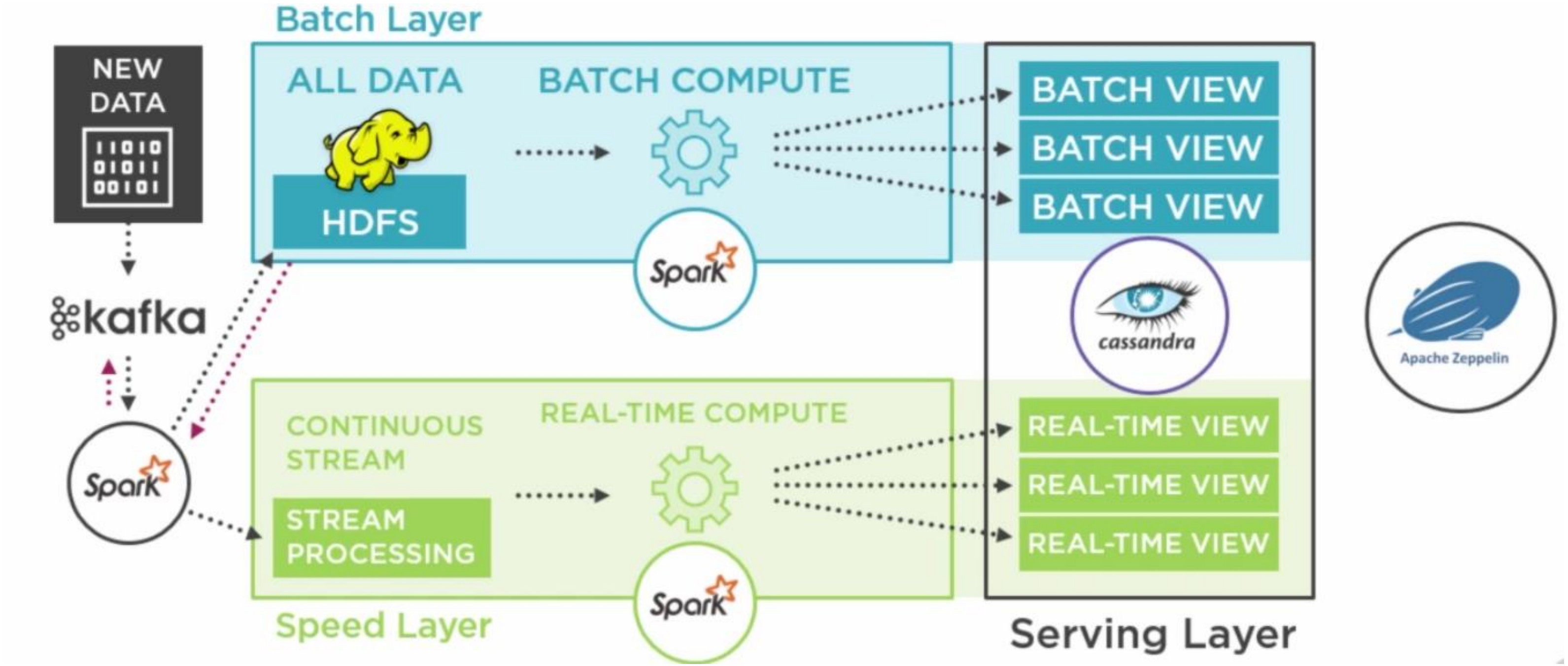
3 Demo

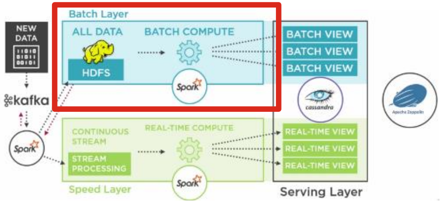
Demo

- Ideal Architecture
- Technologies
- Scenario - Database
- Project

Demo

Ideal Architecture



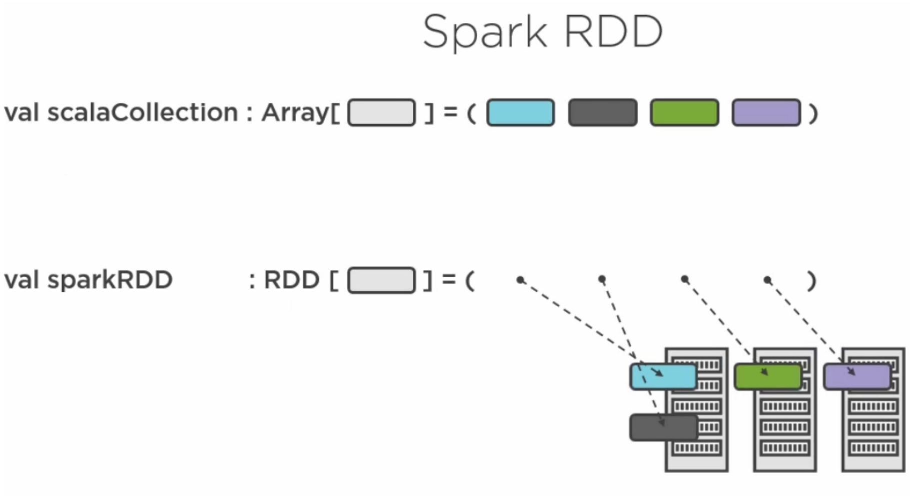


Demo

Technologies

- **Apache Spark** is a fast and general engine for big data processing.
- Spark powers a stack of libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming.
- It can access diverse data sources including HDFS, Cassandra, HBase, and S3.

Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark.


Spark RDD
val scalaCollection : Array[ ] = ( )
val sparkRDD : RDD [ ] = ( )

Demo

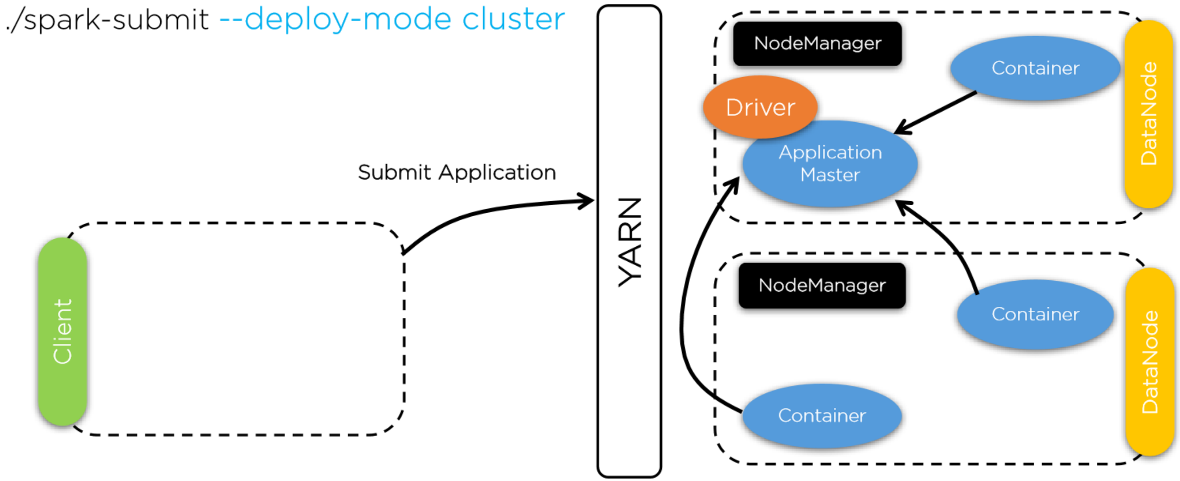
Technologies – How it works inside view

Cluster mode

Driver managed by cluster manager

Better availability guarantees

Non-interactive



```
./spark-submit --deploy-mode cluster
```

Demo

Technologies – How it works inside view

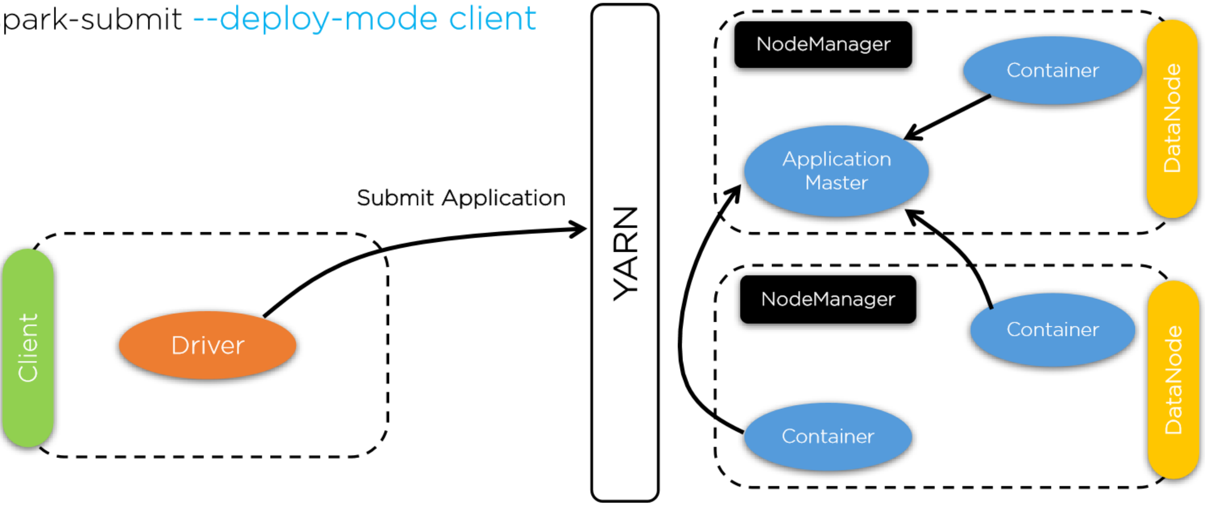
Client mode

Driver managed by client host

Availability dependent on client

Interactive (Spark Shell/REPL)

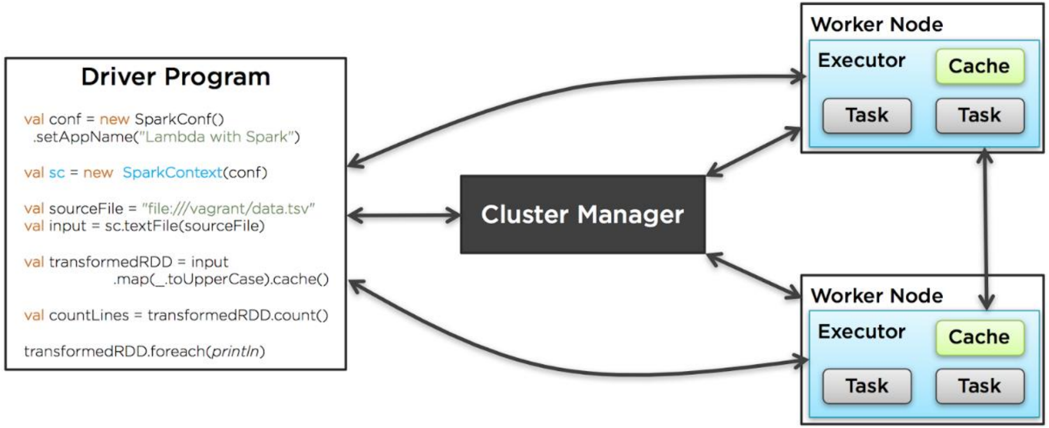
```
./spark-submit --deploy-mode client
```



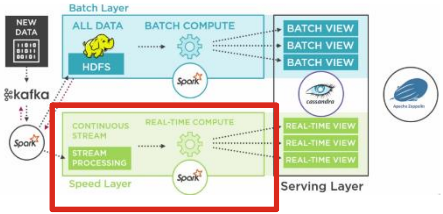
Demo

Technologies

Spark Execution Components



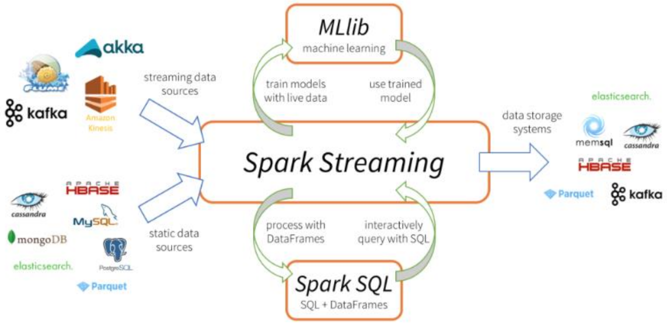
```
val conf = new SparkConf()
  .setAppName("Lambda with Spark")

val sc = new  SparkContext(conf)

val sourceFile = "file:///vagrant/data.tsv"
val input = sc.textFile(sourceFile)

val transformedRDD = input
               .map(_.toUpperCase).cache()

val countLines = transformedRDD.count()

transformedRDD.foreach(println)
```



Demo

Technologies

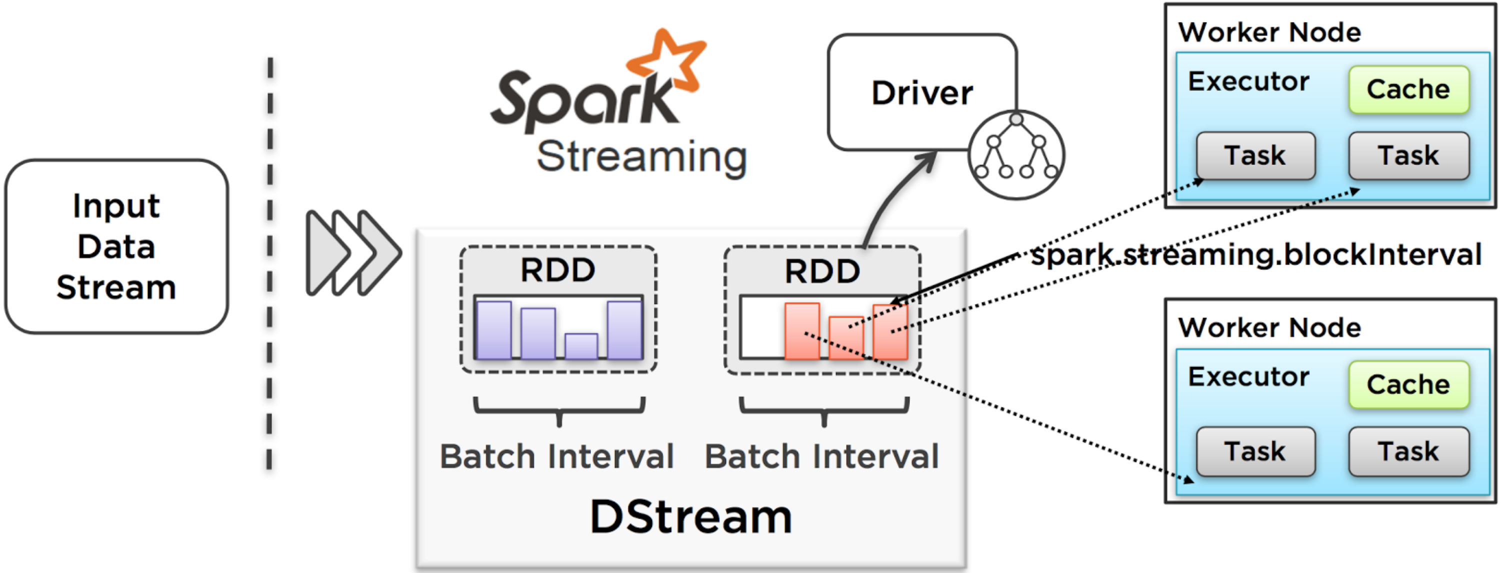
- Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics.
- DStream – Discretized Streaming, is a continuous stream of RDD's
- This design enables the same set of application code written for batch analytics to be used in streaming analytics, thus facilitating easy implementation of lambda architecture



Extension of the core Spark API that enables building scalable, high-throughput and fault-tolerant streaming applications

Demo

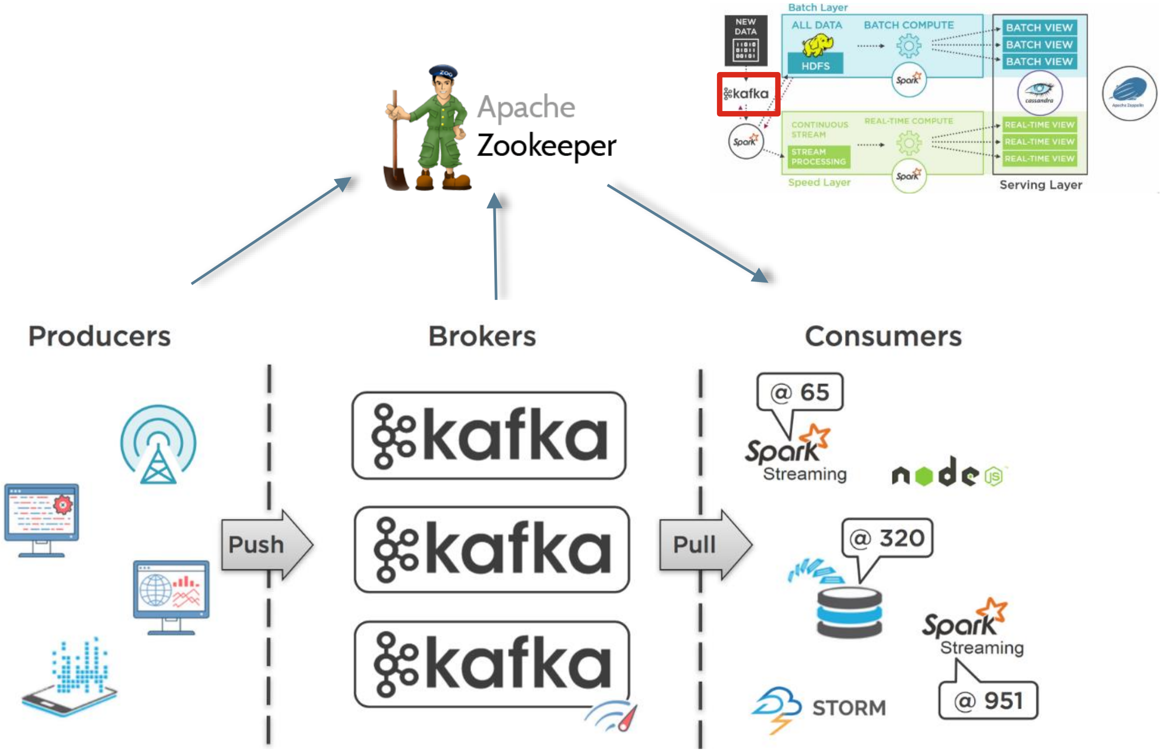
Technologies – How it works inside view



Demo

Technologies – Apache Kafka

- Kafka supports low latency message delivery and gives guarantee for fault tolerance in the presence of machine failures.
- Ability to handle a large number of diverse consumers.
- Distributed - is built on top of the ZooKeeper synchronization service

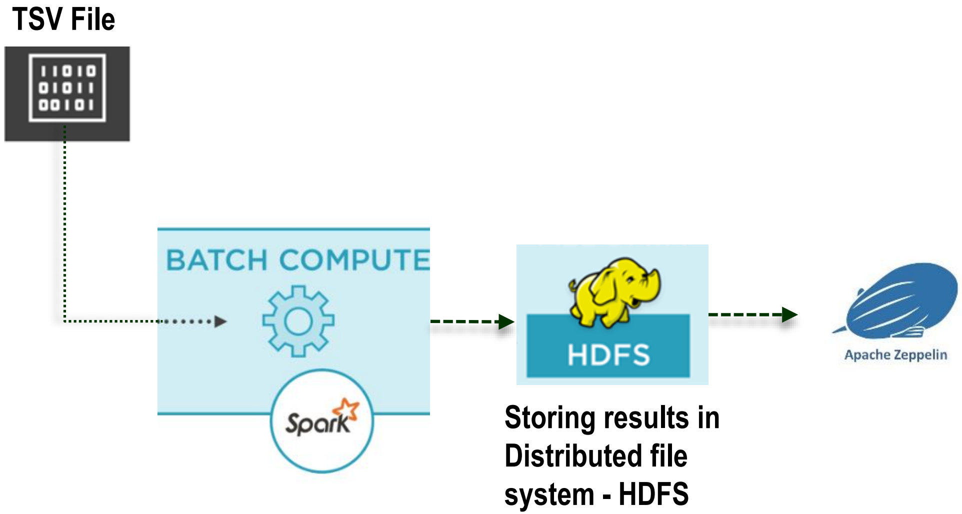


Distributed publish-subscribe messaging system

Demo

Project

- Batch Layer
  - Reads data from the TSV file
  - Batch process in Apache Spark
  - Writes the results in HDFS
  - Query data and visualization in Apache Zeppelin

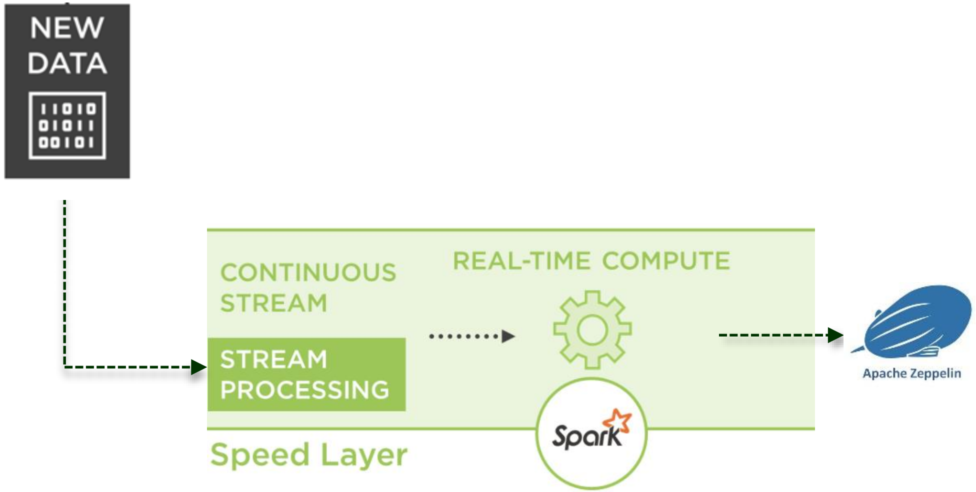


Storing results in Distributed file system - HDFS

Demo

Project

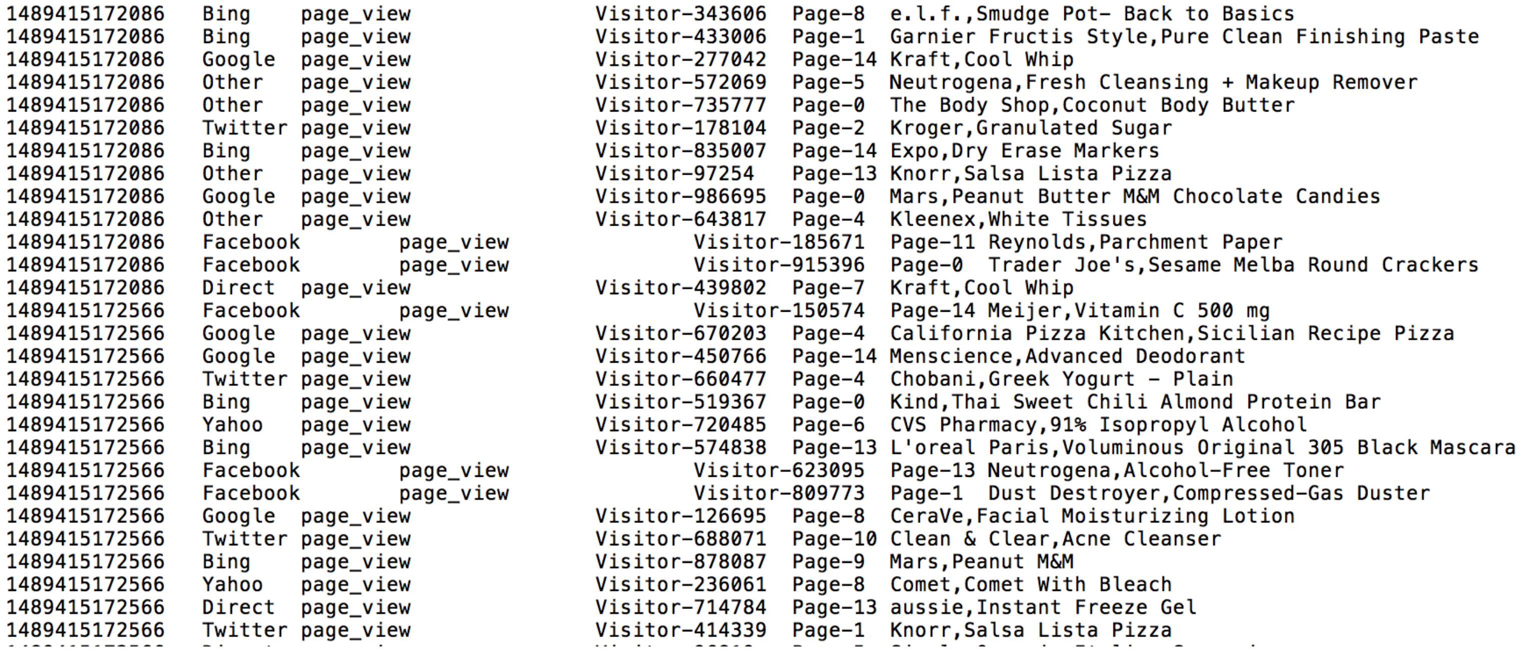
- Speed Layer
  - Reads the stream of data produced by a Program written in scala “LogProducer”
  - Processing and Visualization
    - Processing the input data and Printing aggregated results for visualization in IDE.
    - Processing and visualization in Apache Zeppelin.



Demo

ScreenShot of the Database

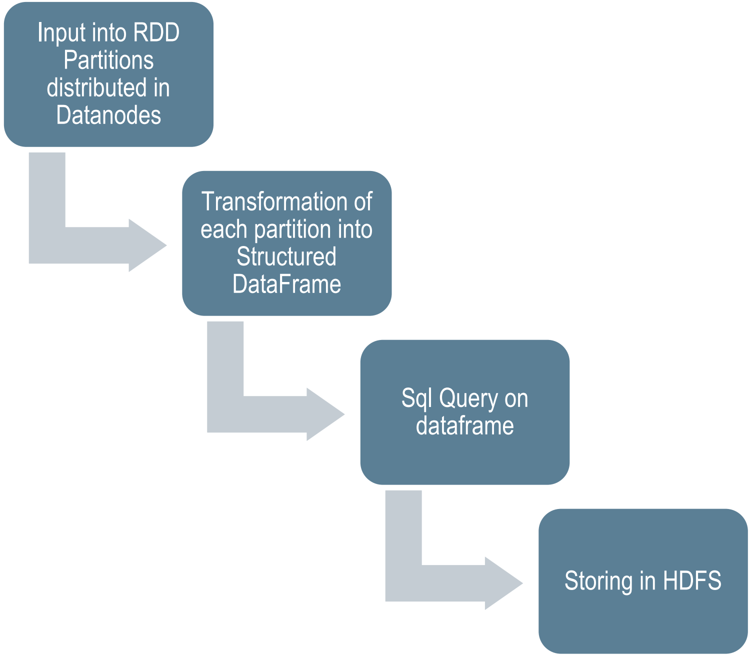
TimeStamp Referrer Action Visitor Page Product



```
1489415172086   Bing    page_view        Visitor-343606  Page-8  e.l.f.,Smudge Pot- Back to Basics
1489415172086   Bing    page_view        Visitor-433006  Page-1  Garnier Fructis Style,Pure Clean Finishing Paste
1489415172086   Google  page_view        Visitor-277042  Page-14 Kraft,Cool Whip
1489415172086   Other   page_view        Visitor-572069  Page-5  Neutrogena,Fresh Cleansing + Makeup Remover
1489415172086   Other   page_view        Visitor-735777  Page-0  The Body Shop,Coconut Body Butter
1489415172086   Twitter page_view        Visitor-178104  Page-2  Kroger,Granulated Sugar
1489415172086   Bing    page_view        Visitor-835007  Page-14 Expo,Dry Erase Markers
1489415172086   Other   page_view        Visitor-97254   Page-13 Knorr,Salsa Lista Pizza
1489415172086   Google  page_view        Visitor-986695  Page-0  Mars,Peanut Butter M&M Chocolate Candies
1489415172086   Other   page_view        Visitor-643817  Page-4  Kleenex,White Tissues
1489415172086   Facebook        page_view        Visitor-185671  Page-11 Reynolds,Parchment Paper
1489415172086   Facebook        page_view        Visitor-915396  Page-0  Trader Joe's,Sesame Melba Round Crackers
1489415172086   Direct  page_view        Visitor-439802  Page-7  Kraft,Cool Whip
1489415172566   Facebook        page_view        Visitor-150574  Page-14 Meijer,Vitamin C 500 mg
1489415172566   Google  page_view        Visitor-670203  Page-4  California Pizza Kitchen,Sicilian Recipe Pizza
1489415172566   Google  page_view        Visitor-450766  Page-14 Menscience,Advanced Deodorant
1489415172566   Twitter page_view        Visitor-660477  Page-4  Chobani,Greek Yogurt - Plain
1489415172566   Bing    page_view        Visitor-519367  Page-0  Kind,Thai Sweet Chili Almond Protein Bar
1489415172566   Yahoo   page_view        Visitor-720485  Page-6  CVS Pharmacy,91% Isopropyl Alcohol
1489415172566   Bing    page_view        Visitor-574838  Page-13 L'oreal Paris,Voluminous Original 305 Black Mascara
1489415172566   Facebook        page_view        Visitor-623095  Page-13 Neutrogena,Alcohol-Free Toner
1489415172566   Facebook        page_view        Visitor-809773  Page-1  Dust Destroyer,Compressed-Gas Duster
1489415172566   Google  page_view        Visitor-126695  Page-8  CeraVe,Facial Moisturizing Lotion
1489415172566   Twitter page_view        Visitor-688071  Page-10 Clean & Clear,Acne Cleanser
1489415172566   Bing    page_view        Visitor-878087  Page-9  Mars,Peanut M&M
1489415172566   Yahoo   page_view        Visitor-236061  Page-8  Comet,Comet With Bleach
1489415172566   Direct  page_view        Visitor-714784  Page-13 aussie,Instant Freeze Gel
1489415172566   Twitter page_view        Visitor-414339  Page-1  Knorr,Salsa Lista Pizza
```

Demo

Process



Input into RDD Partitions distributed in Datanodes

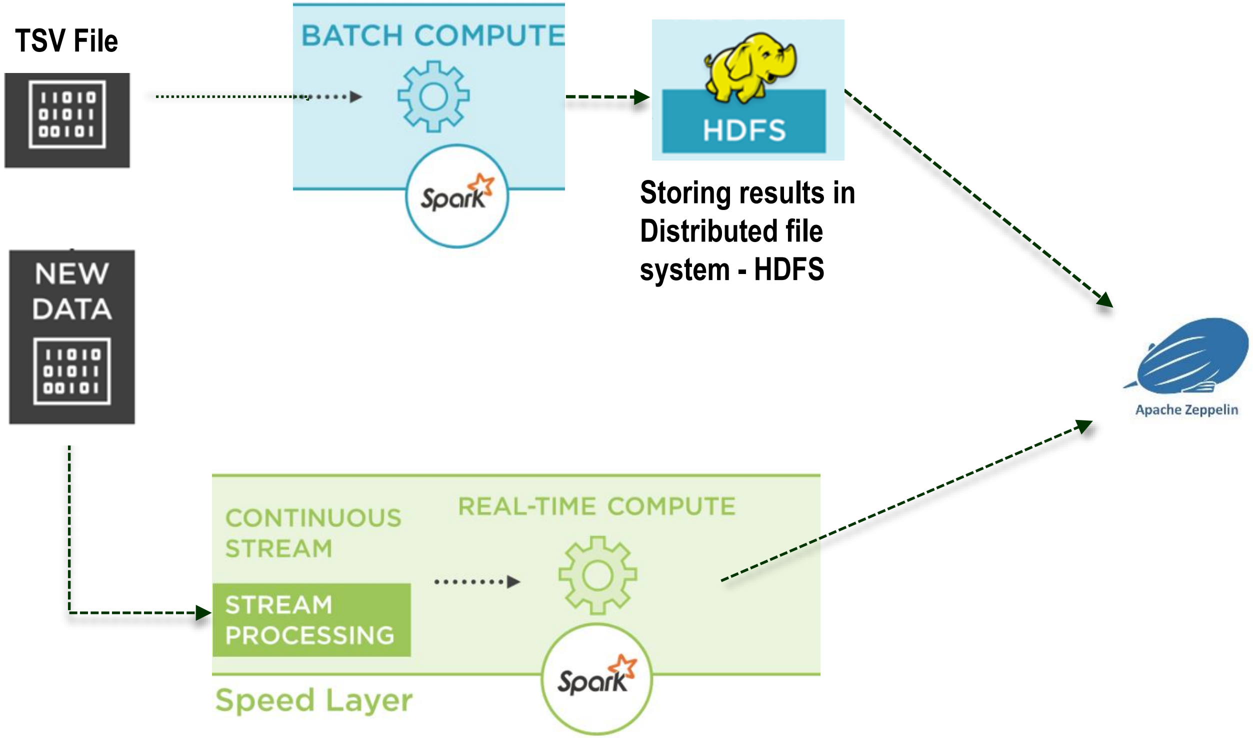
Transformation of each partition into Structured DataFrame

Sql Query on dataframe

Storing in HDFS

Let’s go!

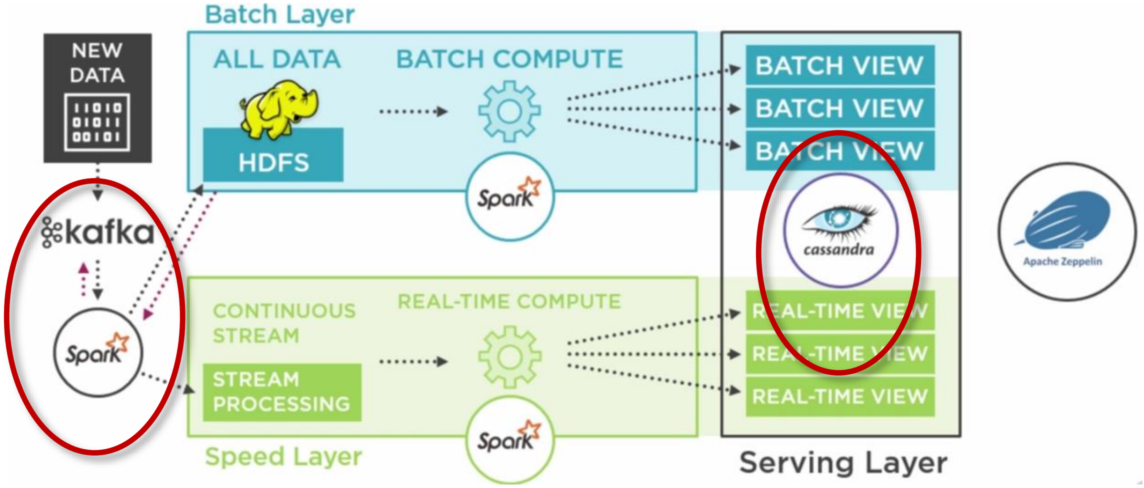
Demonstration



Demo

What Next?

- Integration of Apache Kafka as a streaming source.
- Sync data into Batch and Stream Layer
- Storing the both Batch view and Real-time view data in a Database, most probably Cassandra.
- Merging the data by querying and Visualization in the Apache Zeppelin.
- Cloud ???



3DS.COM © Dassault Systèmes | Confidential Information | 3/21/2017 | ref.: 3DS_Document_2012

IF WE
TM