Arquitecturas Big Data

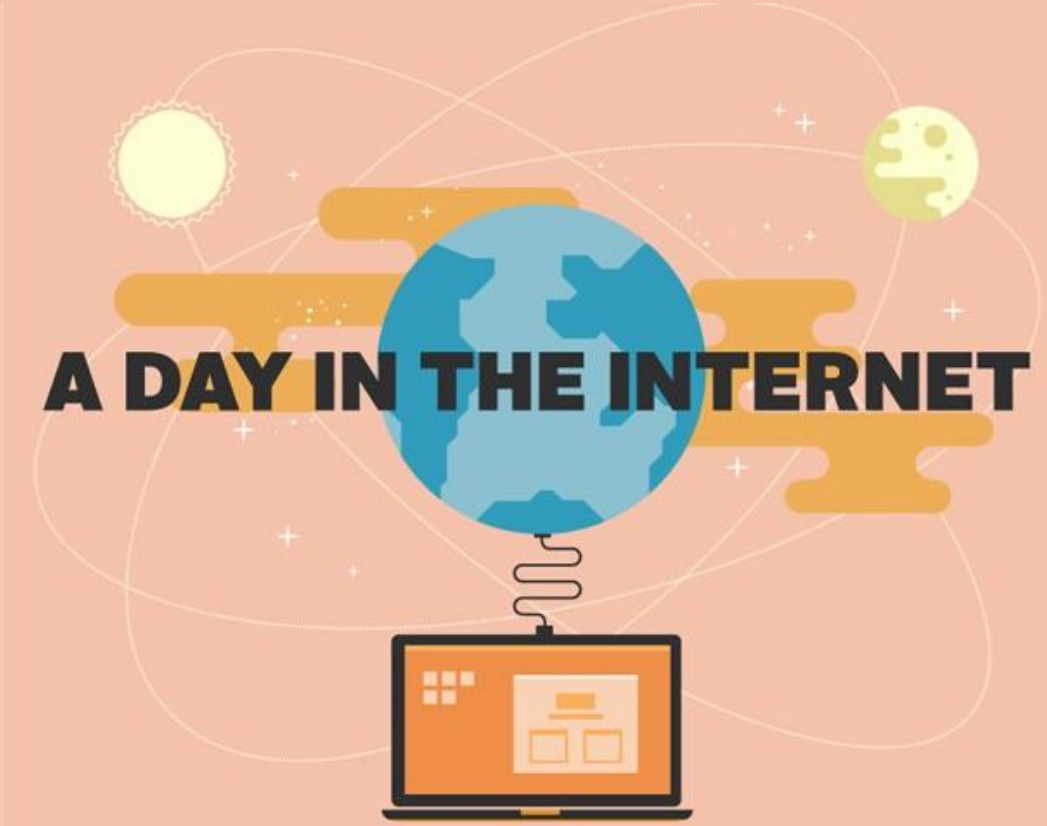
Mario Pérez Esteso

@_Mario_Perez

mario@geekytheory.com

Índice


BIG DATA



In one day, enough information is consumed by internet traffic to fill

168 MILLION DVDS.

294 BILLION

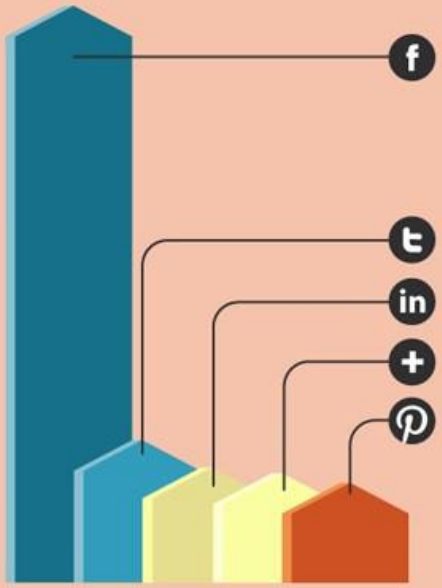
emails are sent.

It would take
2 years to process
that many pieces
of mail in the US.

2 MILLION BLOG POSTS

are written.

Enough posts to fill
Time Magazine for 770 years.



172 MILLION

different people
visit Facebook.

Twitter: **40 MILLION**

LinkedIn: **22 MILLION**

Google+: **20 MILLION**

Pinterest: **17 MILLION**

864,000 HOURS OF VIDEO

are uploaded to YouTube.

That's 98 years of
non-stop cat videos.

Internet users spend

14.6 MINUTES

viewing porn online.

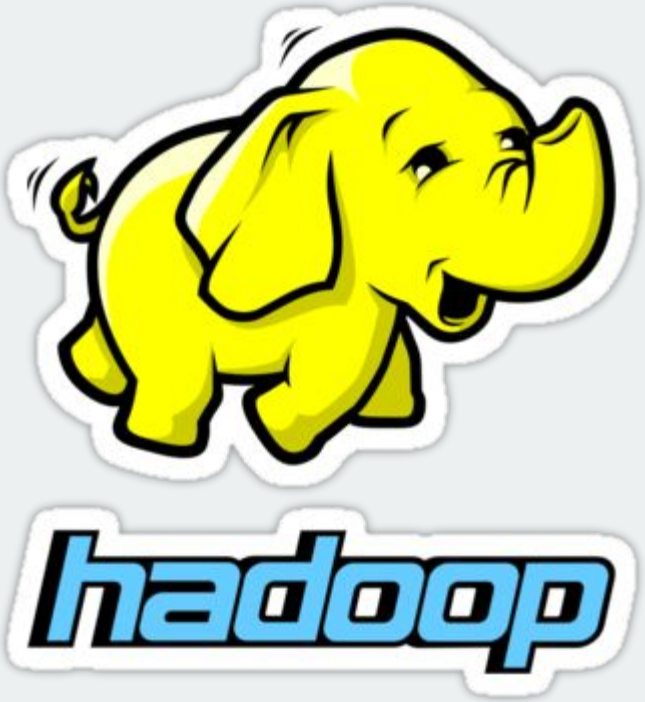
The average fap session
is 12 minutes.

378,000

Number of iPhones Sold

371,000

Number of babies born


hadoop


Spark

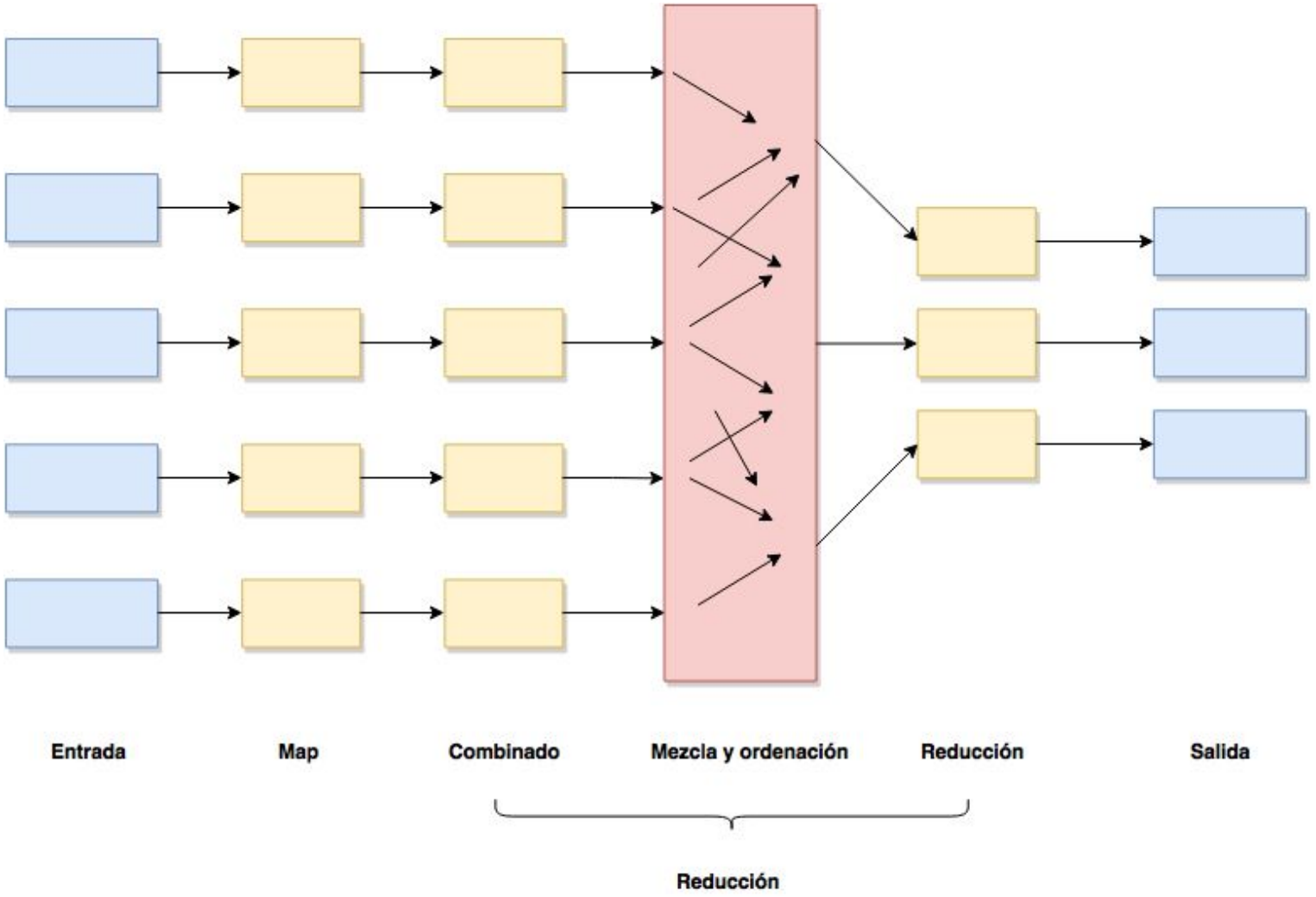
GeekyTheory

Apache Hadoop

Hadoop está formado por los siguientes módulos:

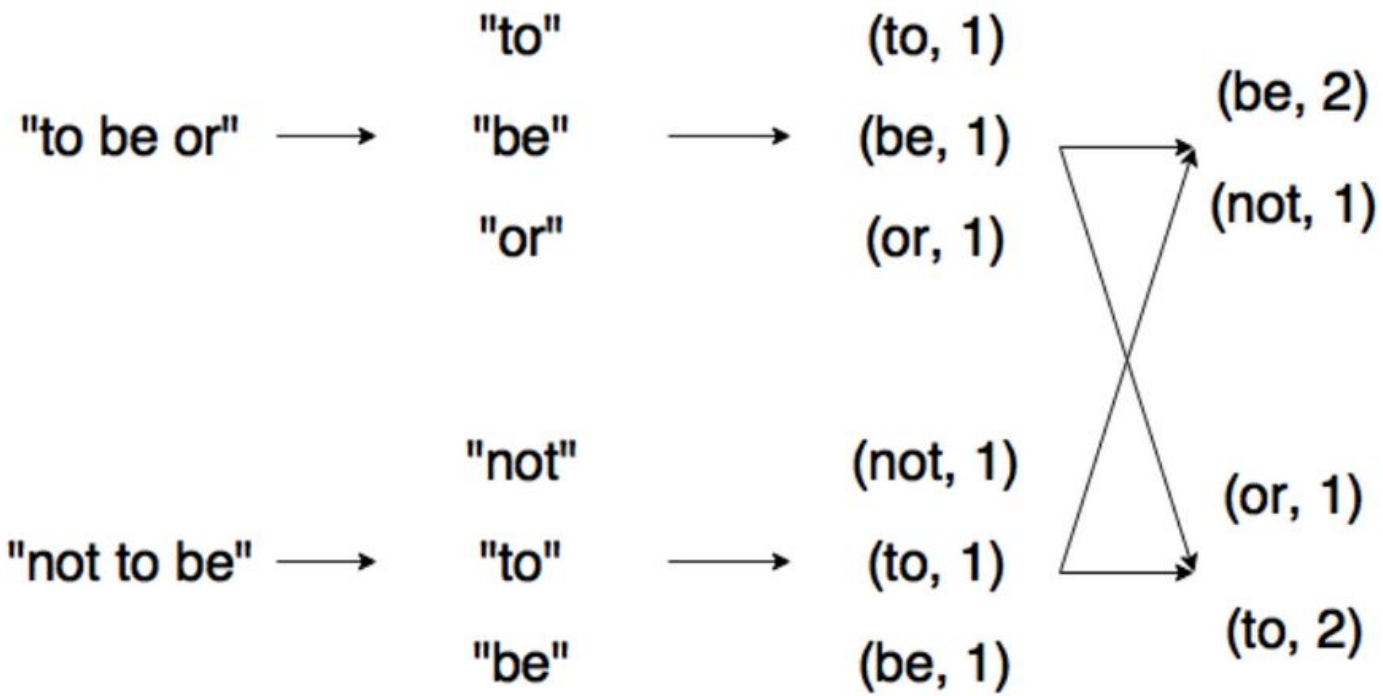
- Hadoop Common.
- Hadoop Distributed Filesystem (HDFS).
- Hadoop YARN (Yet Another Resource Negociator).
- MapReduce.

Apache Hadoop MapReduce



GeekyTheory

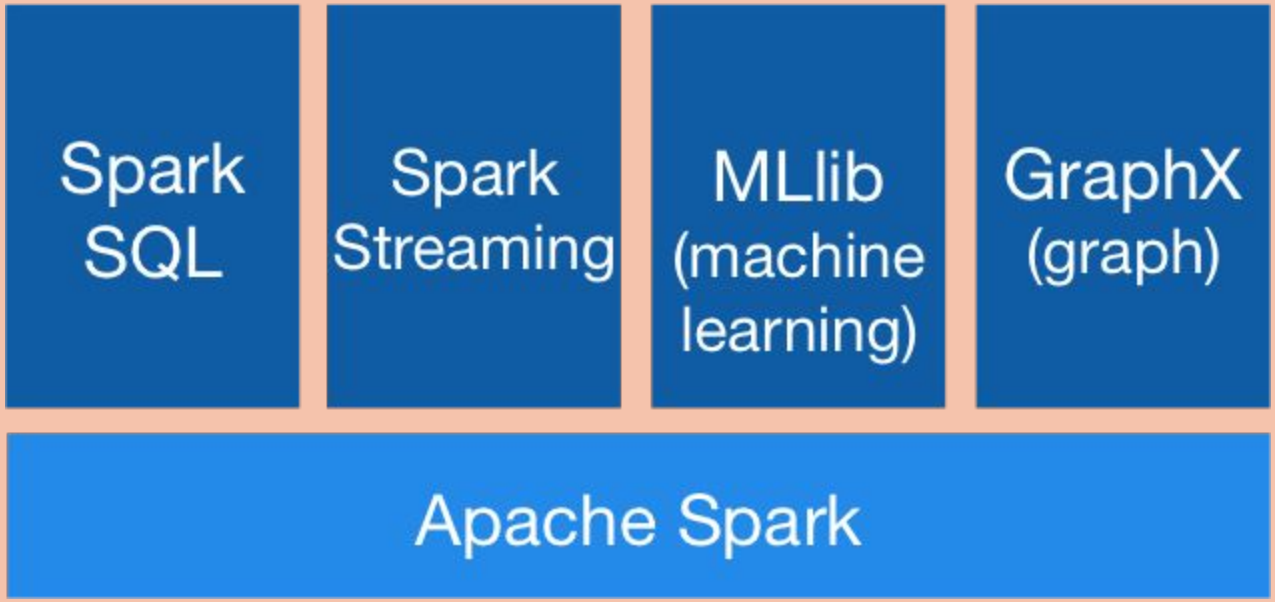
Apache Hadoop MapReduce



Apache Spark

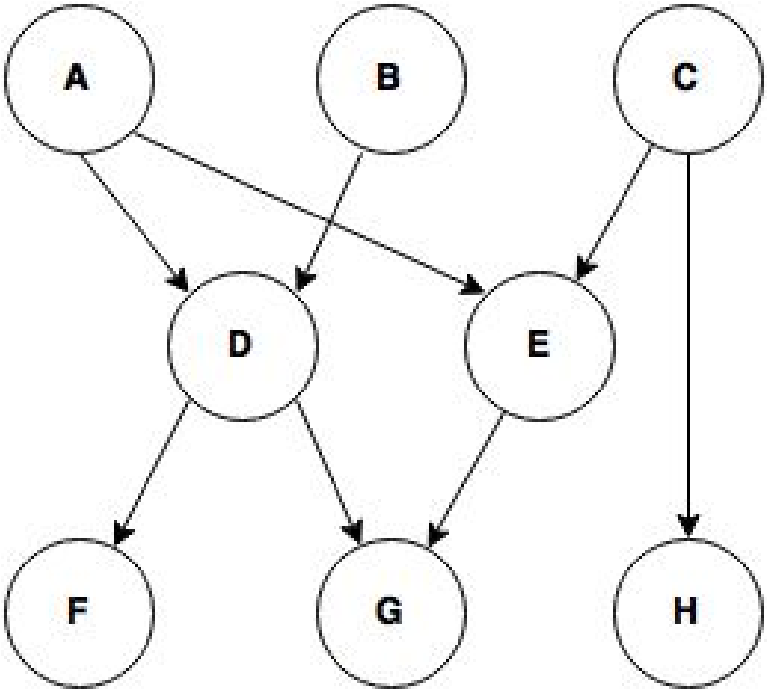
Es considerado el primer software de código abierto que hace la programación distribuida realmente accesible a los científicos de datos.

Spark mantiene la escalabilidad lineal y la tolerancia a fallos de MapReduce, pero amplía sus bondades gracias a varias funcionalidades: **DAG** y **RDD**.


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

GeekyTheory

Apache Spark - DAG (Directed Acyclic Graph)



GeekyTheory

Apache Spark - RDD (Resilient Distributed Dataset)

Un objeto RDD permite a los programadores realizar operaciones sobre grandes cantidades de datos en clusters de una manera rápida y tolerante a fallos.

Mantener los datos en memoria puede mejorar el rendimiento de una aplicación considerablemente.

Sistemas de streaming

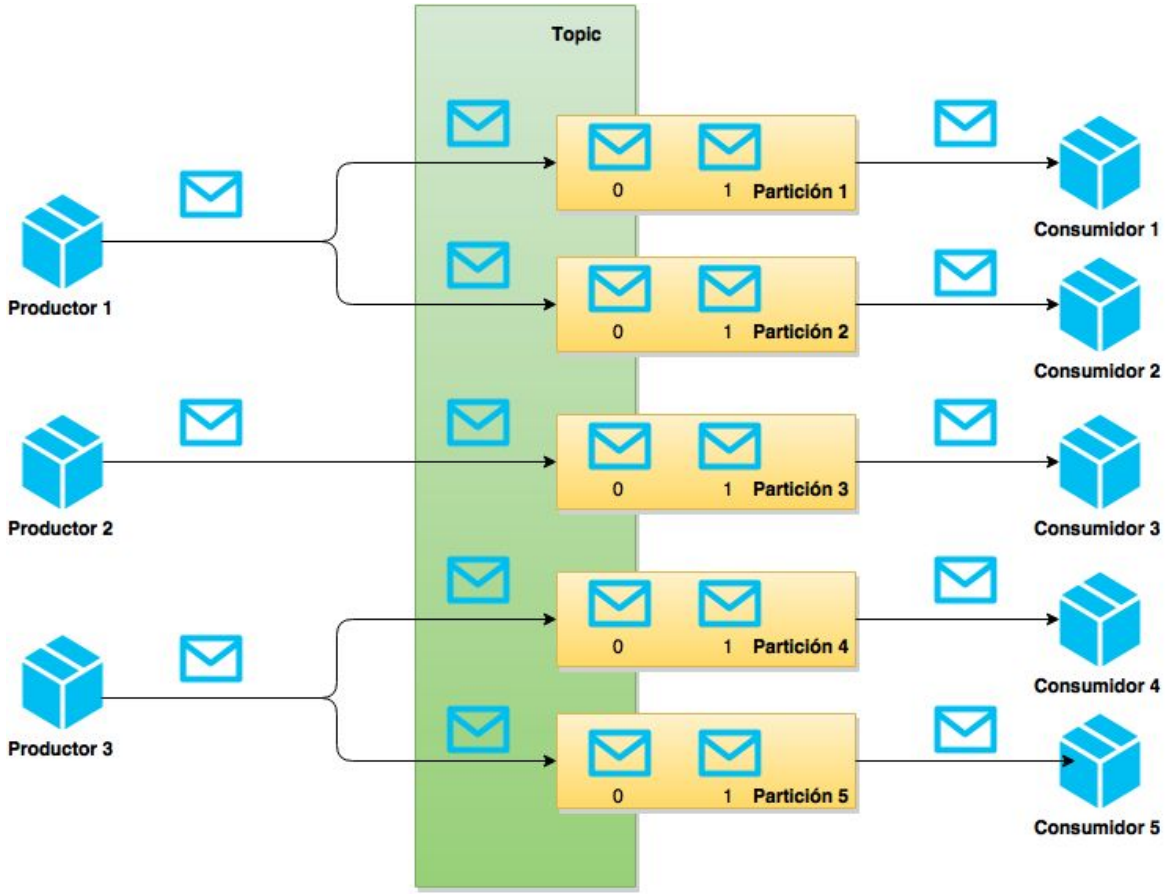
GeekyTheory

Apache Kafka

Sistema de mensajería distribuido basado en publicación-suscripción.

GeekyTheory

Apache Kafka



GeekyTheory

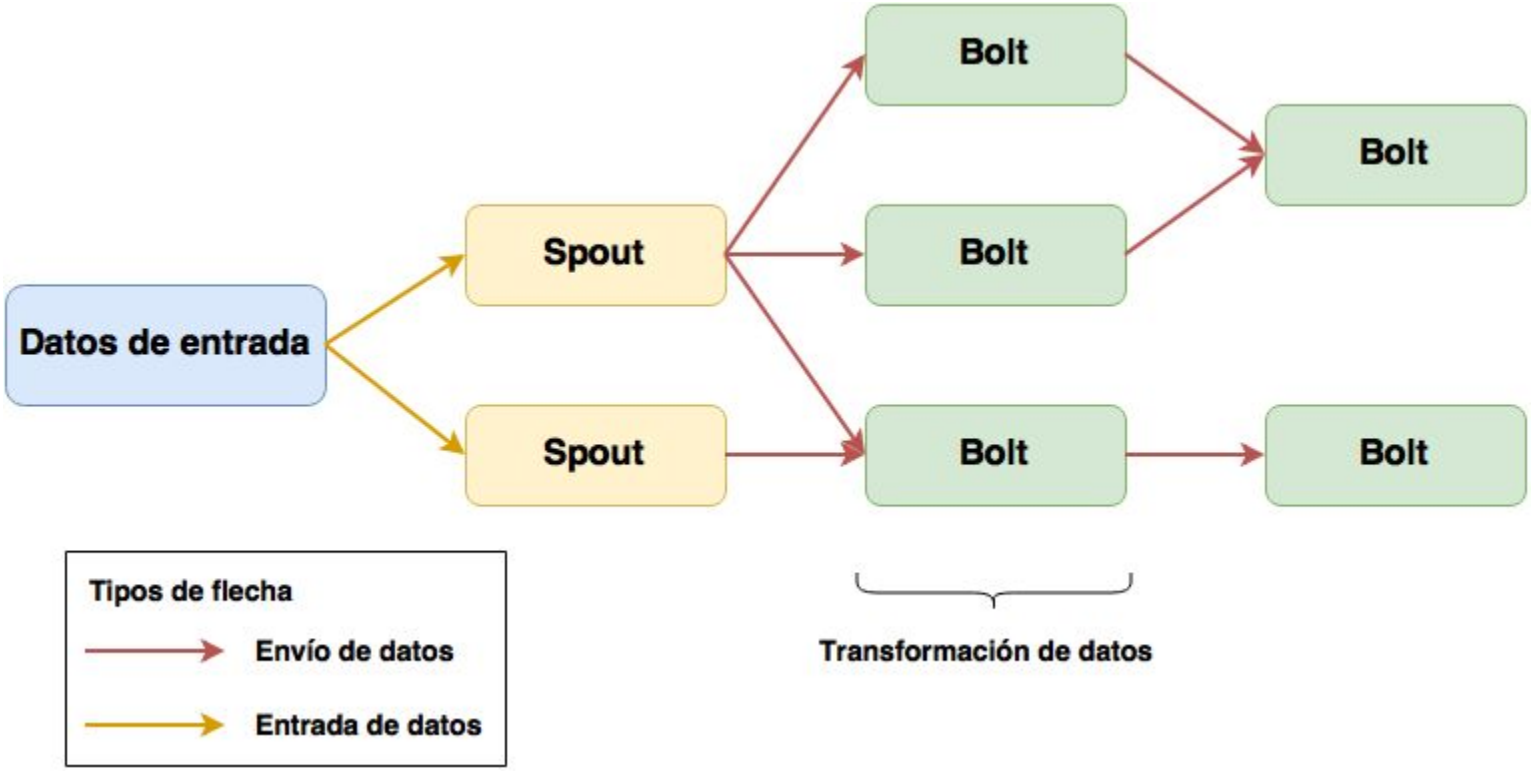
Apache Storm

Se basa en una arquitectura maestro-esclavo y su objetivo es procesar datos en tiempo real.

Se compone de dos partes principales:

- Spout
- Bold

Apache Storm

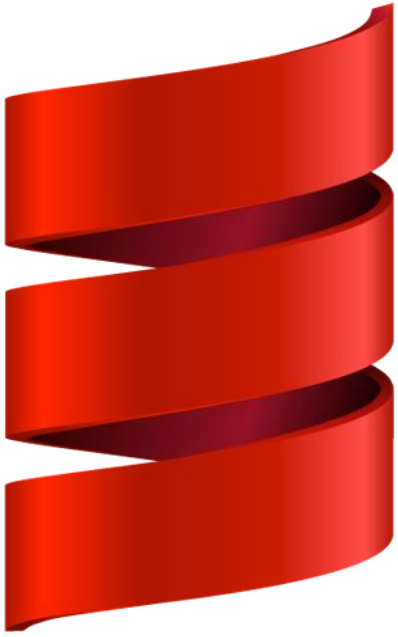


Spark Streaming

GeekyTheory

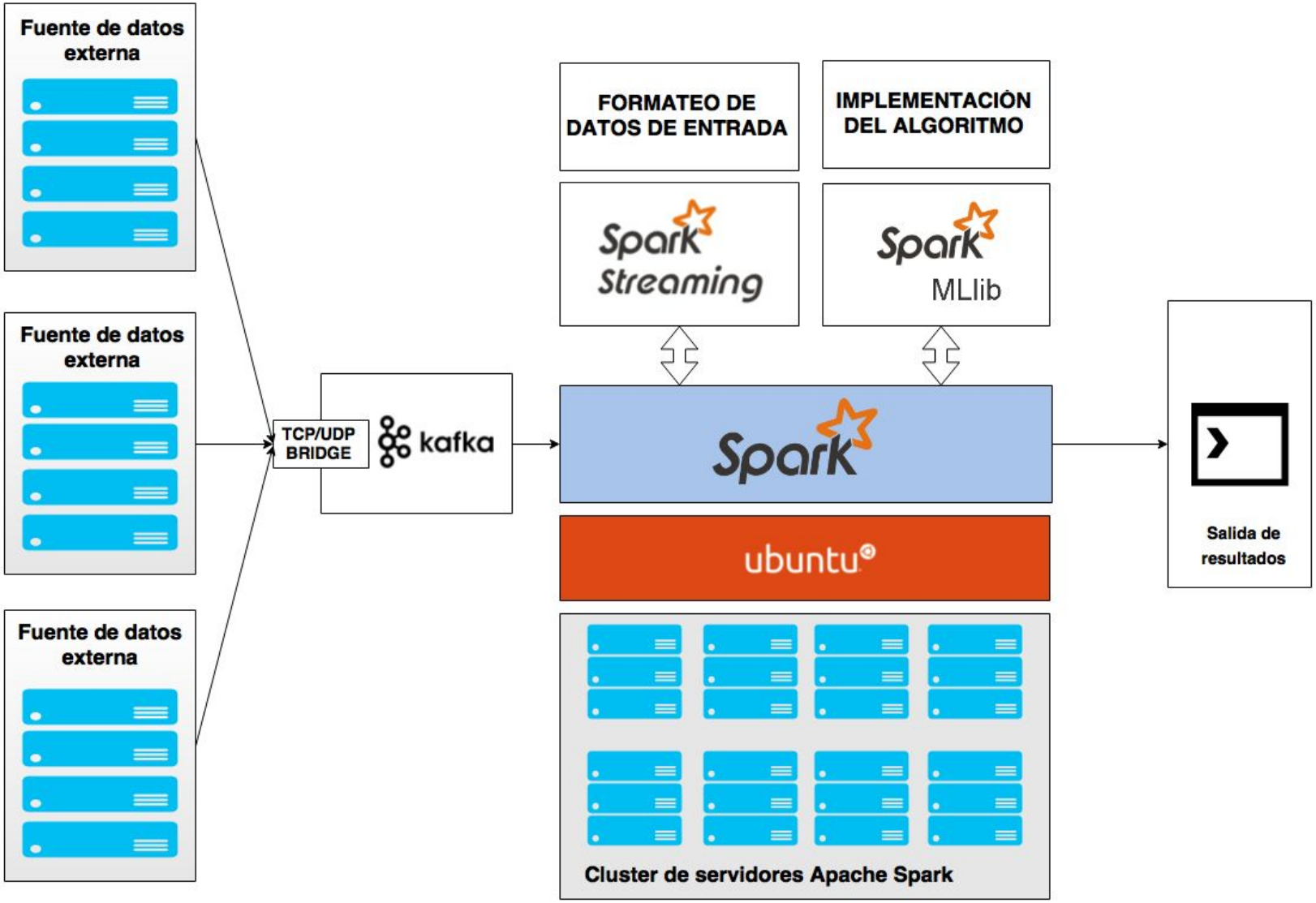
Spark Streaming


GeekyTheory




Java™

Arquitectura

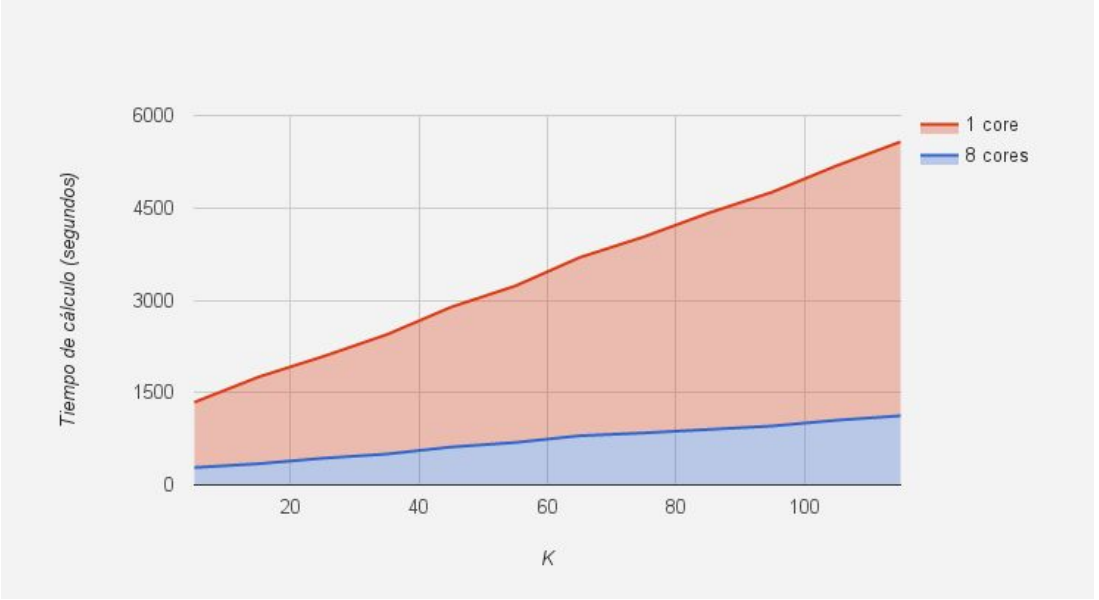


Detección de anomalías en redes

Caso de estudio - Detección de anomalías

Conjunto de datos:
4.9 millones de registros.

Algoritmo:
KMeans Clustering.




```
from pyspark.mllib.clustering import KMeans
model = KMeans.train(data,
                     k,
                     maxIterations=10,
                     runs=5,
                     initializationMode="random")
```

Predicción de fallos online

Caso de estudio - Predicción de fallos online

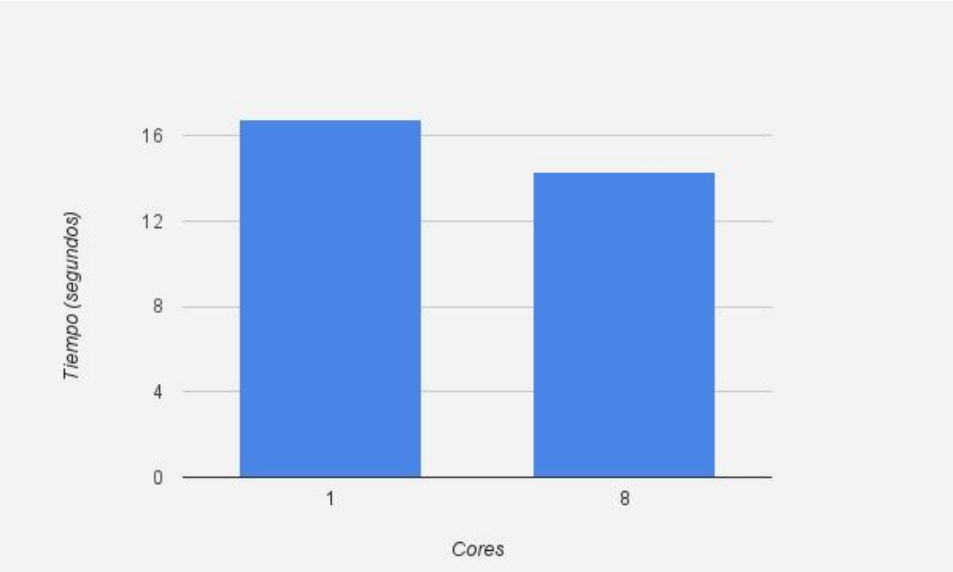
Conjunto de datos:

10661 líneas

88 columnas

Algoritmo:

Random Forests.




```
model = RandomForest.trainClassifier(
                        trainingData,
                        numClasses=10,
                        categoricalFeaturesInfo={},
                        numTrees=5,
                        featureSubsetStrategy="auto",
                        impurity='gini',
                        maxDepth=5,
                        maxBins=32)
```

Análisis de sentimientos en Twitter

Caso de estudio - Análisis de sentimientos

Conjunto de datos:
Mensajes en Twitter.

```
oauth.consumerKey = XXXXXXXXXXXXXXX
oauth.consumerSecret = XXXXXXXXXXX
oauth.accessToken = XXXXXXXXXXXXXXX
oauth.accessTokenSecret = XXXXXXXXX
```

Algoritmo:
Recuento de palabras positivas y negativas.

```
val sc = new SparkContext(conf)
val ssc = new StreamingContext(sc, Seconds(5))
val filters = new Array[String](1)
filters(0) = "greece"
val twitterStream = TwitterUtils.createStream(ssc,
                                              None,
                                              filters)
```

Análisis de sentimientos en Twitter

Puntuación de mensajes

```
Time: 1436353630000 ms
---------------------
(6187380632,RT @IvanaKottasova: Tried to buy a train ticket in
Athens. Ït's free", I am told. "The banks are closed."#Greece)
(6187380678,Greece has the opportunity to repudiate a lot of
inconvenient foreign debt.)
```

```
1 twitterStream.
2    map{tweet => (tweet.getId(), tweet.getText())}.
3    map{case (id, text) => (id, clean(text))}.
4    map{case (id, words) => (id, rateWordList(words))}.
5    print()
```

```
Time: 1436361275000 ms
----------------------
(618770130890133505,1)
(618770131334918144,0)
(618770132014362624,0)
(618770132144377856,0)
(618770133599825920,1)
(618770135097192448,0)
(618770136187727873,-1)
(618770136149790720,0)
(618770137676517376,1)
(618770138267914240,-2)
```

Conclusiones

Apache Spark es entre 10 y 100 veces más rápido que Hadoop MapReduce.

La programación es muy similar en Scala, Java y Python.

La arquitectura propuesta es válida para cualquier caso de estudio.

@_Mario_Perez

mario@geekytheory.com