**Spark Streaming**

1. **Internet of Things (IoT)**

<http://en.wikipedia.org/wiki/Internet_of_Things>

**2.** **Definitions**

**speed** refers to the ability of the platform to process data in real-time

**throughput** is the amount of data that system is capable of handling and processing

simultaneously.

**Real-time processing** of a system is its ability to process the data and produce the results strictly within certain time constraints. Real-time responses are often delivered in the order of milliseconds and sometimes microseconds depending on the application and the user requirements.

**Fault tolerance** is the characteristic of a system to continue operating properly in the event of a failure of one or more components.

**Data I/O** performance refers to the rate at which the data is transferred to/from a peripheral device. In the context of big data analytics, this can be viewed as the rate at which the data is read and written to the memory (or disk) or the data transfer rate between the nodes in a cluster.

**Scalability** is defined as the ability of the system to handle growing amount of work load in a capable manner or its ability to be enlarged to accommodate that growth. In our case, scalability is considered to be the ability to add more hardware (scale up or scale out) to improve the capacity and performance of a system.

**Vertical Scaling**: Vertical Scaling involves installing more processors, more memory and faster hardware, typically, within a single server.

**High Performance Computing Clusters** (HPC)

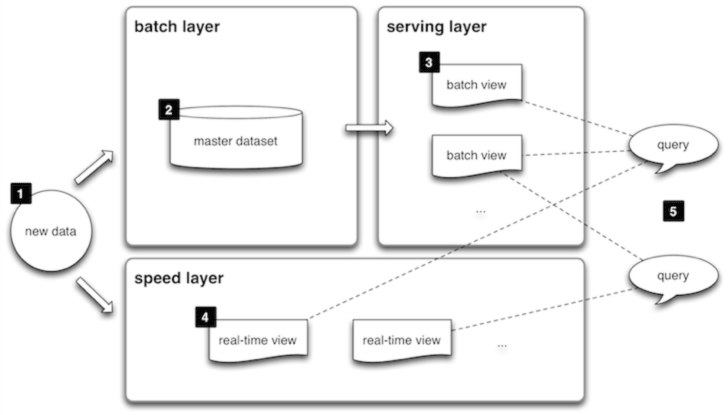
HPC clusters, also called as blades or supercomputers, are machines with thousands of cores.

**Horizontal Scaling:** Horizontal scaling involves distributing the workload across many servers

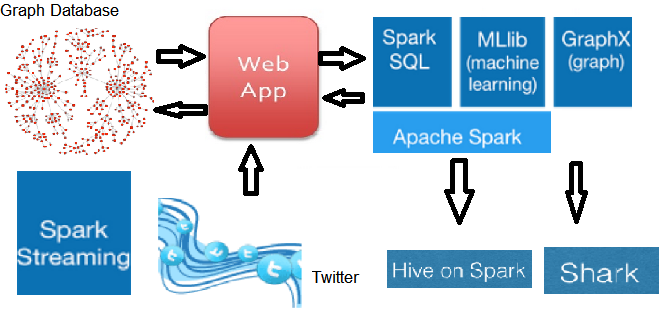
MapReduce Google GFS, Apache HDFS

Apache Spark

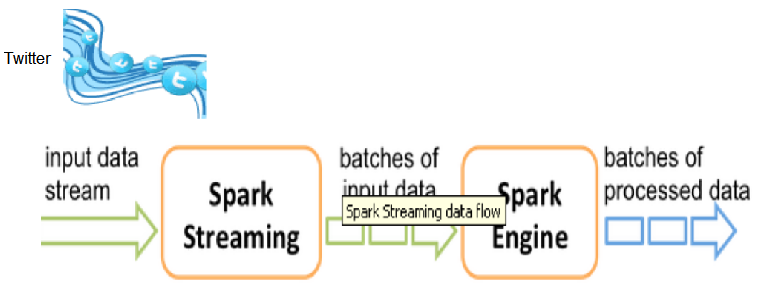
**Lambda Architecture**



Batch Layer Serving Layer



speed layer



**Summary of key points for Spark Streaming as it applies to the Lambda Architecture**

The key idea is to treat streaming computations as a series of batch computations on small time intervals

The input data received during each interval is stored reliably across the cluster to form an input dataset for that interval.

Once the time interval completes, this dataset is processed in parallel with scala operations, such as map and filter, to produce new datasets representing program outputs or intermediate state

A discretized stream or D-Stream groups together a series of RDDs and lets the user manipulate them to through various operators. D-Streams provide both stateless operators, such as map, which act independently on each time interval, and stateful operators, such as aggregation, which operate on multiple intervals and may produce intermediate RDDs as state

// D-Stream pageViews

pageViews = readStream("http://...", "1s") // groups it into 1-second intervals.

ones = pageViews.map(event => (event.url, 1)) //transforms the event stream to get a

//D-Stream of (URL, 1) pairs called ones

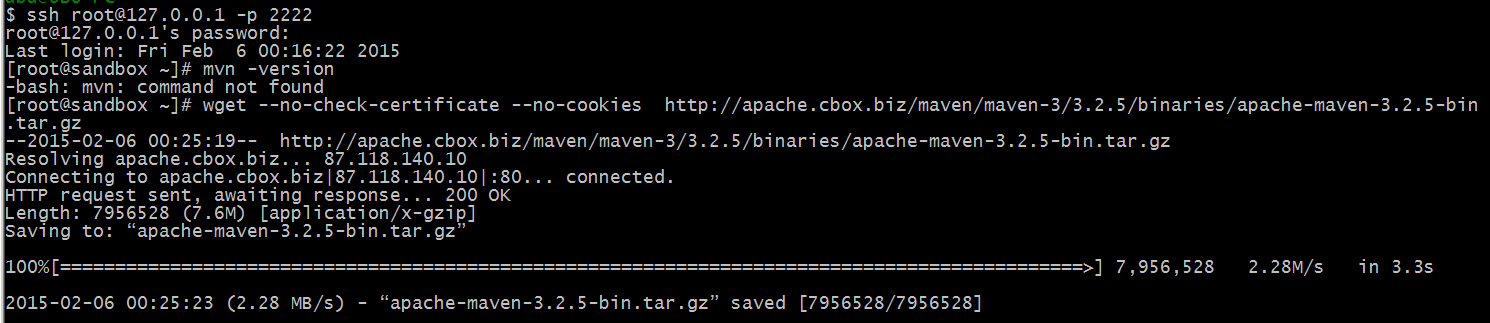
counts = ones.runningReduce((a, b) => a + b) //running count using the runningReduce operator

Set up development on hortonworks vm

1. maven

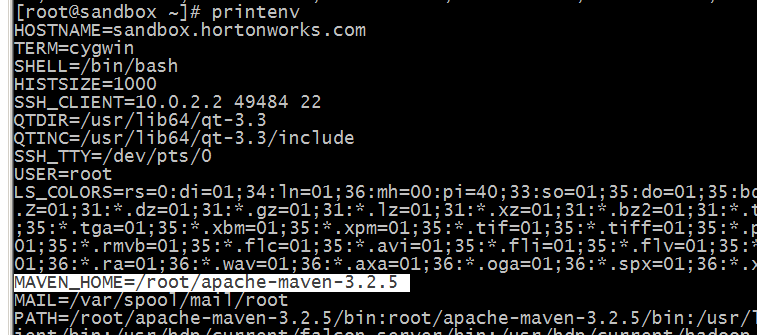
wget --no-check-certificate --no-cookies http://apache.cbox.biz/maven/maven-3/3.2.5/binaries/apache-maven-3.2.5-bin.tar.gz

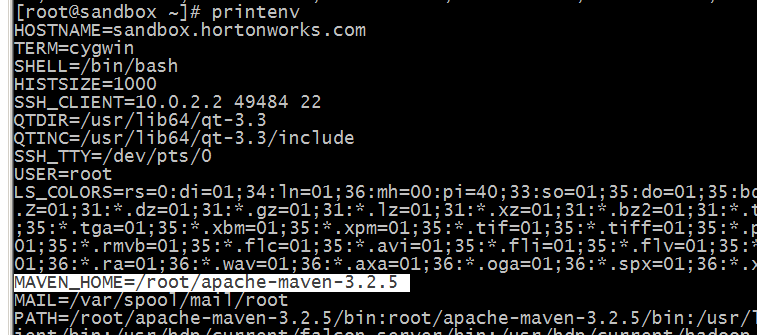
tar -zxf apache-maven-3.2.5-bin.tar.gz



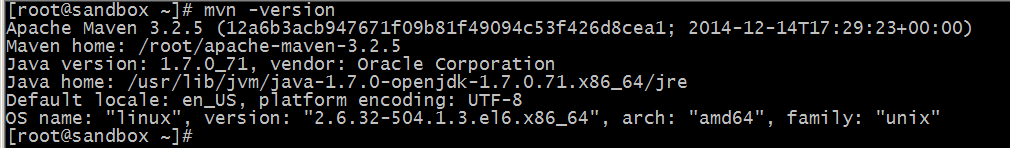
export MAVEN\_HOME="/root/apache-maven-3.2.5"

export PATH=$MAVEN\_HOME/bin:$PATH

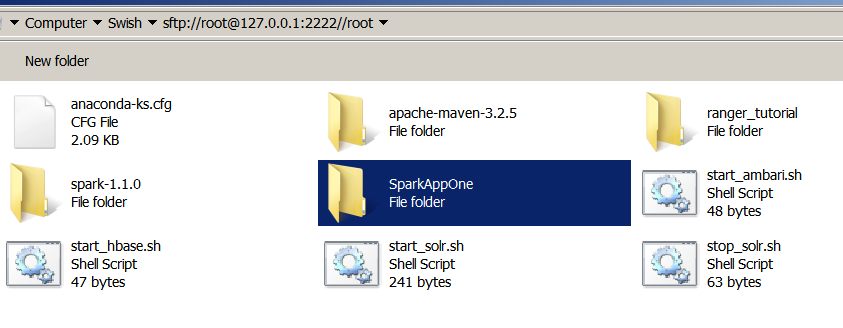




mvn -version



copy source code to vm root



build source

ssh root@127.0.0.1 -p 2222