**What is Spark**

A fast and general engine for large-scale data processing

Unlike MapReduce, it’s a lot easier to use spark

Write simple driver scripts to control how these big data sets are transformed and output into different systems.

Driver program: how to transform it, how to process it, how to output it. – All live within the driver program on your master node.

Driver program: create spark context – that create all the different components that you need to manipulate the data

Spark-context will communicate with your cluster manager if you area using a cluster(Spark, YARN, Mesos)

Spark maybe a next big thing that replace MapReduce in Hadoop

We can run spark on top of hadoop cluster manager(YARN)

Spark also have its own cluster manager built-in.

Cluster manager is responsible for controlling the processing of your data across entire cluster

Many Node that are running their own executors and there could potentially be multiple executors in 1 machine => Horizontal scalable – Add more and more server and process more and more data as a result, never have to bound by the memory of a single machine.

It’s Fast, 100x faster than MR in memory and 10x time on disk.

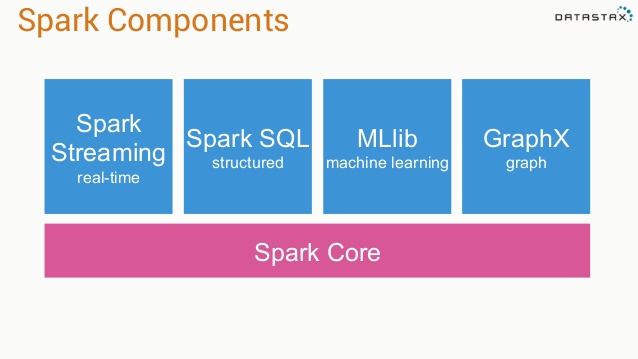
USE DAG ENGINE( directed acyclic graph ) to optimizes workflows – Nothing Actually happen until you tell spark to go and do it. => spark go off and said what’s the optimal way to approach this problem as a whole.

Many Company use it. => It’s hot

Code in JAVA, SCALA, PYTHON

Spark is built around RDD

Components of Spark



Every thing is built on top of spark core which contains all the guts of actually creating, transforming performing action on RDD.

SPARK Streaming – let us deal with real time constant stream of data that come in forever, we can transform them as they come in.

Spark SQL – create SQL view of our data out of an RDD

Mllib – machine learning

GraphX – graph, chart

**RDD** – resilient distributed dataset

Resilient – Can be distributed and run on a cluster that is actually resilient to failure, if 1 node goes down, RDD can still survive

Distributed – distribute the processing of RDD. RDD is basically a giant key value datastorage.

Dataset – It’s a dataset

**Spark Context**

Created by your driver program.

Responsible for making RDD resilient and distributed

Whenever we start with any spark program, we’ve got to create a spark context first.

Spark context object is what create our rdd objects. It’s like the mastermind behind the whole script

In SparkStreaming, you create a streaming context – Just a special spark context that can deal with real time data

val ssc = new StreamingContext(“local[\*]”, “LogAlarmer”, Seconds(1))

Local = RUN ON THIS PC

The star \*: spin up a process for every core that I have available on my PC

LogAlarmer: name

Process 1 second as a time.

We can also create RDD from data that’s in memory

Nums = parallelize([1,2,3,4])

Also create RDD from textfile

Sc.textFile(“PATH”)

We can also create from Distributed file System – real world

Can Also create from:

JDBC

Cassandra

Hbase

…

Json

…

**Transforming RDD**

Once we have RDD that represent some big dataset, what do we do with it.

* We Transform it or we do some action with it.
* Map 1-1
* Flatmap – Take an RDD contains X number of lines and create RDD that has Y lines(Nothing or 1-N or 1-N)
* Filter
* Distinct
* Sample
* Union, intersection, subtract, cartesian

**Map() example**

Val input = sc.parallelize(List(1,2,3,4))

Val result = input.map(x=>x\*x)

This result in (1, 4, 9, 16)

**RDD Actions**

* Collect (Take a snapshot of your RDD and dump it to a conventional data stucture that you can manipulate within your driver’s script) – (good usecase: take the final result that you can actually manage on 1 computer and doing some further output on them)
* Count (rows)
* Count by value (Count Distinct)
* Take (take top few result)
* Reduce (combine element of RDD base on their keys)

Ex: Lines of text from a book that we want to count how many times each word appear in RDD => Unique Key | SUM (Also can do with county by value)

* …
* Just Like MR, we have Map in the Transforming Stage, We have the Reduce in the Action Stage

**Lazy evaluation**

Nothing actually happens in your driver program until an action is called

**Why Spark Streaming**

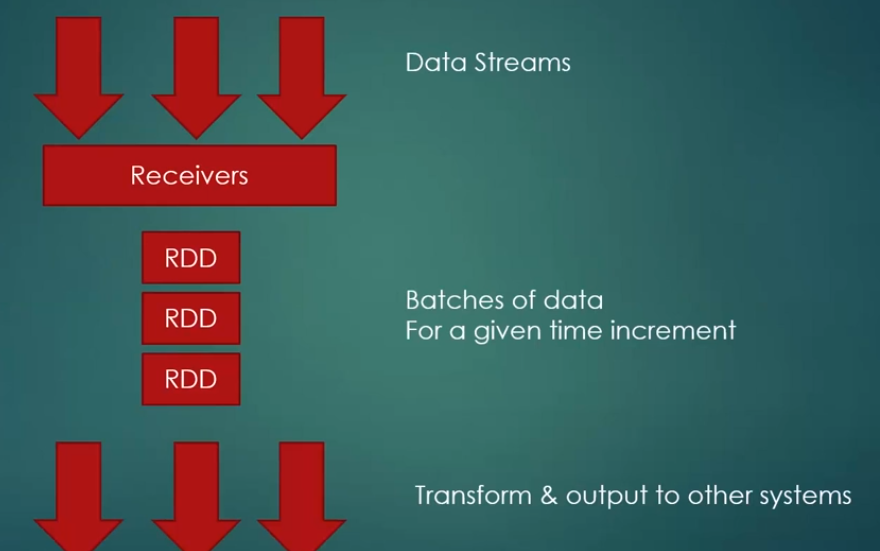
Big data never stop

Analyze dt stream in real time, instead of huge batch jobs daily

Analyzing web log to react to user behavior

Analyzing stream of real-time sensor data for IOT stuff

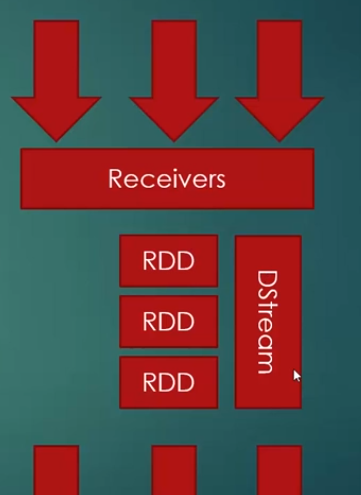
**Spark Streaming High level**



Receiver listen for that data, its break that data stream up to small rdds, we keep transforming little rdd over and over again => external db,…

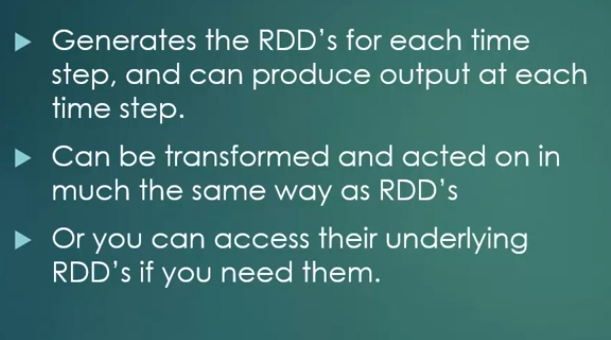
* Process of little rdd can happen in parallel on different worker nodes.

**Dstream(Discretized Stream)**

* Logical stream of information that go on and on forever is called Dstream in Spark Streaming
* 

That Stream is broken up into small RDD

* Apply transform to Dstream as a whole instead of small RDD



**Stateless transformation on Dstream**

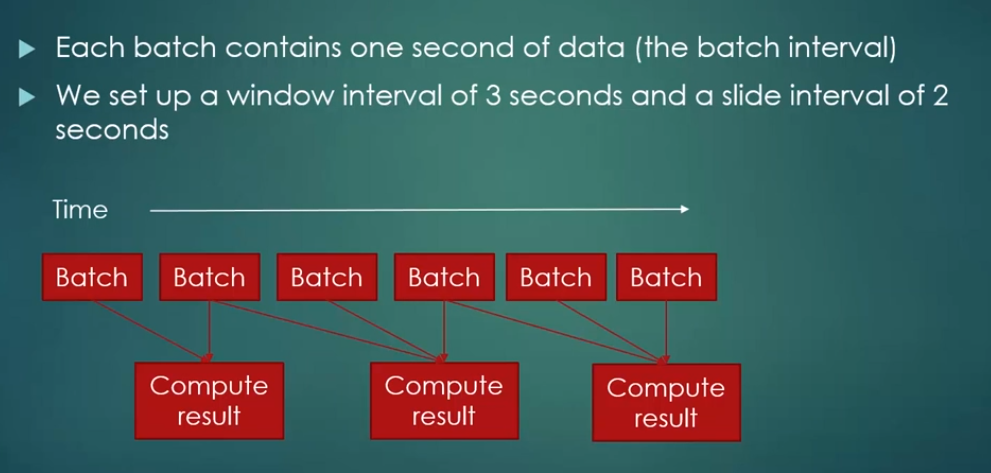
* Map
* Flatmap
* Filter
* Reduce by key

**Windowing**

Allow you to compute results across a longer time period than your batch interval

* Batch interval: how often data is suck into Dstream
* Slide interval: how often a window transformation is computed
* Window interval: how far back in time the windowed transformation goes

Ex:



Batch interval setup:

Val SSC = new StreamingContext (“Local[\*]”, “<NAME>”, <SECOND>)

Window interval and Slide interval setup:

Val <NAME> = <OtherValName>.reduceByKeyAndWindow(<FUNCTION>, Seconds(X), Seconds(Y))

Second(X) = Window interval

Second(Y) = Slide interval

**Spark Streaming Fault-Tolerent**

Incoming data is replicated to at least 2 worker nodes

A checkpoint directory can be use to store state in case we need to restart the stream

* What happen if receiver fail?

Some receivers are better than others.

Lost data while down

Receiver based on replicated, reliable data source is more resilient:

+ HDFS

+ Directly-consumed Kafka

+ Pull-based Flume

* Driver Script failure

Driver script run on master node is actually a single point failure

Use check point directory and streaming context GetorCreate (Input is a checkpoint directory)