Spark Streaming Session - 1

Real Time Processing
==========

whatever we have done till now is nothing but batch processing.

Batch Processing
==========

doing analysis on top of files.

your processing might take few minutes, few hours or few days.

Real time Processing
==========

we will have continuously flowing data and we have to calculate the results instantly.

credit card fraud detection

finding trending hashtag

track website failures using server logs..

when we talk about hadoop

HDFS + MR + YARN

MR (mapreduce) Processing.

Mapreduce can only handle batch jobs.

mapreduce cannot handle streaming or real time data.

How do we process real time streaming data?

Apache Spark.

Spark is a general purpose compute engine which can handle.

Batch + Streaming Data

IN spark we have a module called as Spark Streaming.

Spark Streaming Session - 2
==========

batch vs real time stream processing

mapreduce can handle only batch processing.

spark can handle streaming data as well.

Spark Streaming.

how is the data stored in spark?

RDD (resilient distributed dataset)

when we load the file

val baseRdd = sc.textFile("file path in hdfs")

if the file is of 500 mb & if the default block size in hdfs is 128 mb.

then our baseRDD will have 4 partitions.

when there is no concept of a static file then how do you visualize your rdd.

consider a water tap which is continuosly flowing.

tap is running for 1 hour

size of my stream is 1 hour

every 2 minutes you are filling one water balloon.

30 water balloons.

a new rdd is framed every 2 minutes.

we will have 30 rdds which are created.

some balloons might have more data than other balloons based on flow of water.

batch interval - 5 seconds

consider whatever 30 balloons that we have filled we are putting them in a tub. (Dstream)

so basically

Dstream -> Rdd's -> Messages (entities)
1 -> 30 rdd's -> each rdd can have lot of messages.

we need to operate at Dstream level

underneath spark still works in batch style.

the batch size is small like 1 second.

spark's compute engine is a batch engine and not a streaming engine.

whenever batch size is very small example 1 second.

then we get a feeling that it's real time streaming.

Spark Streaming Session - 3
==========

In batch processing in Spark

1. Lower level Constructs (RDD)

2. Higher Level Constructs (Structured API's Dataframes, dataset and spark sql)

Stream Processing in Spark

we have both the lower level as well as higher level constructs.

1. Spark Streaming (RDD's) traditional way
2. Structured Streaming - newer thing - Dataframes

Lets talk about example of normal spark streaming

Producer will be an application which is giving you continuous data.

consumer will be our spark streaming application.

twitter will be producer.

spark streaming will be the consumer.

we will try to simulate a dummy producer.

there will be a terminal and whatever you type will be produced.

producer will write to a socket

and consumer will read from the same socket.

socket
=======

IP Address + Port number (localhost + 9998)

step 1: start a producer
we want to create a dummy producer where we continously type things.

nc -lk 9998

step 2: we have to start a consumer
spark streaming code which reads the data from the socket

spark-shell --master local[2]

sc is already available
we have spark context which is available

when we talk about spark streaming applications we require a spark streaming context

```
//creating spark streaming context
val ssc = new StreamingContext(sc, Seconds(5))

//lines is a dstream
val lines = ssc.socketTextStream("localhost",9998)

//words is a transformed dstream
val words = lines.flatMap(x => x.split(" "))
```

```
val pairs = words.map(x => (x,1))

val wordCounts = pairs.reduceByKey((x,y) => x+y)

wordCounts.print()

ssc.start()
```

Spark Streaming Session - 4
==========

what is real time processing.

Dstream -> RDD's -> messages

word count streaming example

when calculating word count it was forgetting the state of previous rdds.

if I write hello 5 times

then (hello,5)

hello 2 times again

(hello,2)

it was giving output for each rdd individually.

when we talk about spark streaming. There are 2 kind of transformations.

1. stateless transformation

is the one which forgets the previous state. we perform operation on a single rdd always.

2. stateful transformation

the one in which we do aggregation over more than one rdd.

when we talk about batch processing. we load the entire file as one single rdd.

batch processing is always stateless. there is no point of talking about stateful transformation in case of batch processing.

when we talk about stateful transformations we have 2 choices:

lets consider you have a streaming application which runs for 6 hours.

batch interval size to be 5 minutes - a new rdd will be created every 5 minutes.

during the course of entire streaming application how many rdds will be created?

72 rdd's will be created in 6 hours.

1. consider all of the rdd's within a stream - consider all 72

2. you want to do operations over the last 30 minutes..

we will consider last 6 rdd's always.

Sum() is stateless and work for each rdd individually.

what if we want to get a running total?

in this case we should be using a stateful transformation.

we need to convert his normal rdd's into pair rdd's.

and add a dummy key to all these elements.

1 5 7

(k,1)

(k,5)

(k,7)

updateStateByKey() is a stateful transformation.

when we talk about stateless we just talk about 1 single rdd. - stateless

considering the entire stream we talked about including all rdds. - stateful

considering a few rdds (window) - stateful

lets say the batch interval is 10 seconds..

that means a new rdd will be created every 10 seconds.

3 things
=========

1. batch interval - 10 seconds

2. window size - 30 seconds

3. sliding interval - 20 seconds

countByWindow() - stateful transformation

Spark Streaming Session - 5
============================

```
//creating spark streaming context
val ssc = new StreamingContext(sc, Seconds(5))

//lines is a dstream
val lines = ssc.socketTextStream("localhost",9998)

//words is a transformed dstream
val words = lines.flatMap(x => x.split(" "))


val pairs = words.map(x => (x,1))

val wordCounts = pairs.reduceByKey((x,y) => x+y)

wordCounts.print()

ssc.start()
```

Spark Streaming Session - 6
============================

I want to calculate the frequency of each word across the entire stream...

stateful transformation..

updateStateByKey is a stateful transformation we can think of using.

this requires 2 steps:

1. Define a state to start with.

2. a function to update the state

big data is interesting big data is fun

(big,1)
(data,1)
(is,1)
(interesting,1)
(big,1)
(data,1)
(is,1)
(fun,1)

(big,1)
(big,1)
(data,1)
(data,1)
(is,1)
(is,1)
(fun,1)

rdd1

(big,{1,1}) newValues = {1,1} 2 previousState = 0 ,  (big,2)
(data,{1,1}) newValues = {1,1} previousState = 0 , (data,2)
(is,{1,1}) (is,2)
(fun,{1}) (fun,1)

rdd2

big data is vast

(big,1) newValues = {1} previousState = 2 , (big,3)
(data,1) (data,3)
(is,1) (is,3)
(vast,1) (vast,1)

when we talk about stateful transformations then we have to do checkpointing.

==========

Spark Streaming Session - 7
==========

reduceByKey - stateless transformation - one rdd

updateStateByKey - stateful transformation , it considers the entire dstream from the beginning to the end. - all rdd's

sliding window -

1. batch interval - the time in which each rdd is created.

if we have the batch interval as 2 seconds. that means after every 2 seconds a new rdd will created.

2. window size - 10 seconds.. we are interested in the last 5 rdd's always.

3. Sliding interval. - 2 seconds..

after every 2 seconds one oldest rdd will go away and one new rdd will come in.

if this sliding interval is 4 seconds.

after every 4 seconds.. 2 oldest rdd's will go away and 2 new rdd will come in.

sliding interval has to be a integral multiple of batch interval.

window size should also be an integral multiple of batch size.

reduceByKeyAndWindow

hello,1
how,1
are,1
you,1
hello,1

reduceByKeyAndWindow transformation takes 4 parameters.

1. the summary function. (x,y) => x+y

2. the inverse function. (x,y) => x-y

3. the window size. Seconds(10)

4. the sliding interval. Seconds(2)

My batch interval is 2 seconds.

our problem statement is find the frequency of each word in the 10 seconds sliding window.

when will we see the very first result - 10 seconds.

when do you see the second result. - after 12th second

Spark Streaming Session - 8
==========

reduceByKey - stateless

updateStateByKey - stateful (considers entire stream)

reduceByKeyAndWindow - stateful (sliding window) - pair rdd is required.

reduceByWindow - here pair rdd is not required

countByWindow - it will count the number of lines in the window.

Traditional spark streaming - lower level constructs

Dealing at the level of rdd's.

Structured API's are trending these days.

Spark Structured Streaming