Week 3 Big data integration and processing

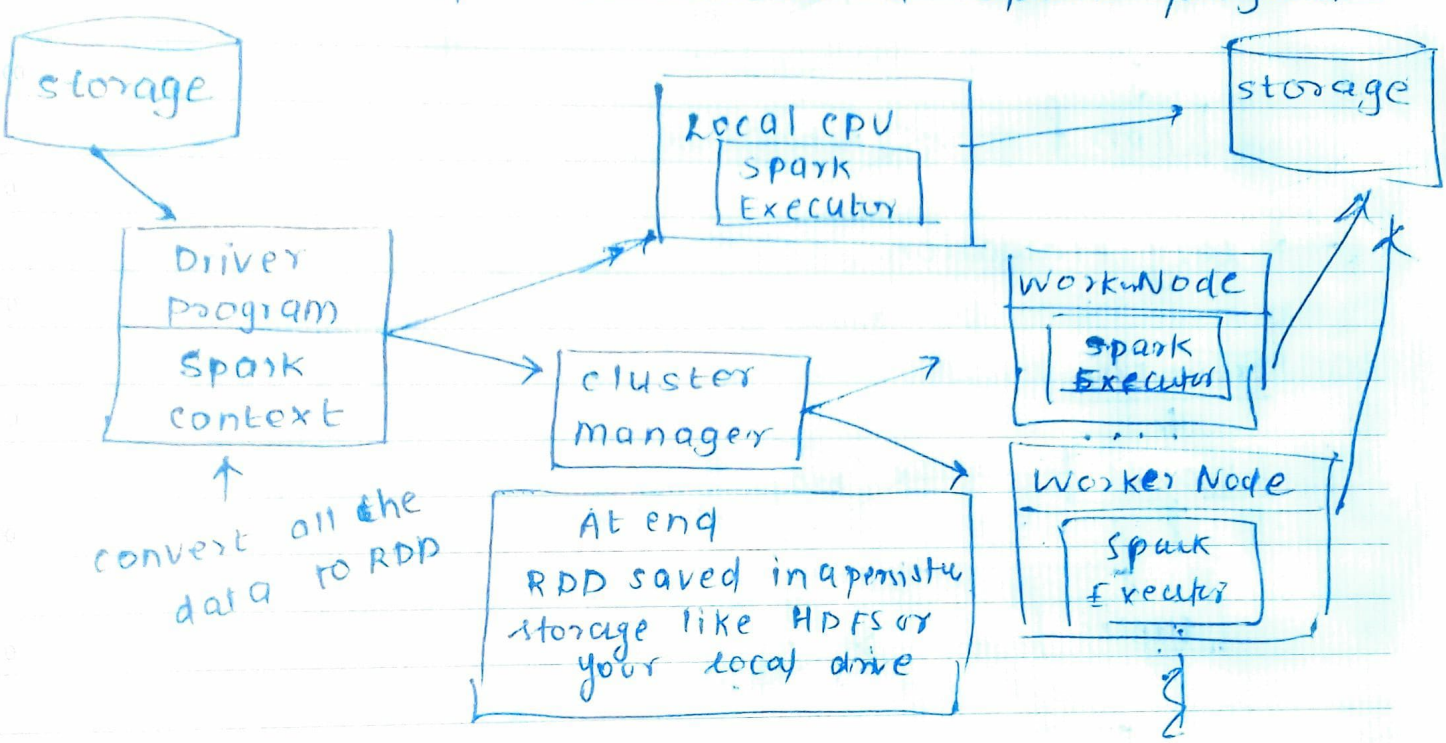
Video1:-

*Components of Spark

Spark core is the underlying general execution engine for Spark platform that all other functionality is built upon.

Spark Core:- Programming in Spark

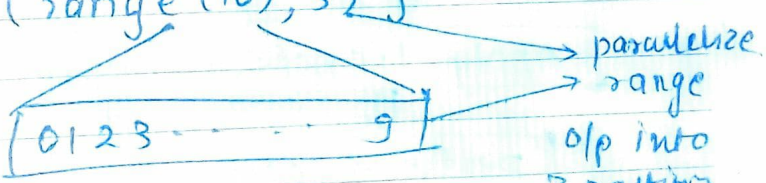
→ two methods to create RDDs in Spark
→ Explain with immutable means
→ Interpret a spark program as a pipeline of transformations and actions
→ Lists the steps to create a spark program



⇒ lines = sc.textFile ["hdfs:/user/cloudera/words.txt"]
lines = sc.parallelize (["big", "data"])

numbers = sc.parallelize(range(10), 3) } using parallelize

numbers.collect()



[0, 1, 2] , [3, 4, 5] , [6, 7, 8, 9]

Apache spark core provides ... and referencing datasets in external storage systems.

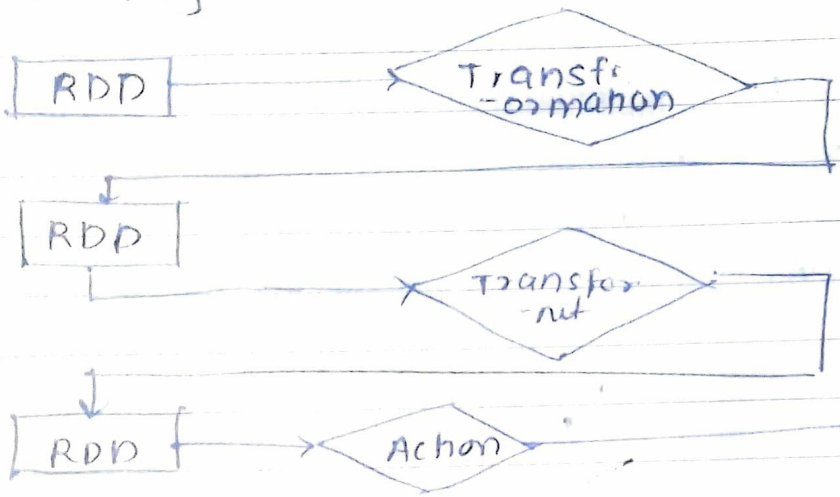
18 | 2016 Thursday August | 231-135 • WK 34

07 | July 2016 | 08

wk	M	T	W	T	F	S	S
27					1	2	3
28	4	5	6	7	8	9	10
29	11	12	13	14	15	16	17
30	18	19	20	21	22	23	24
31	25	26	27	28	29	30	31

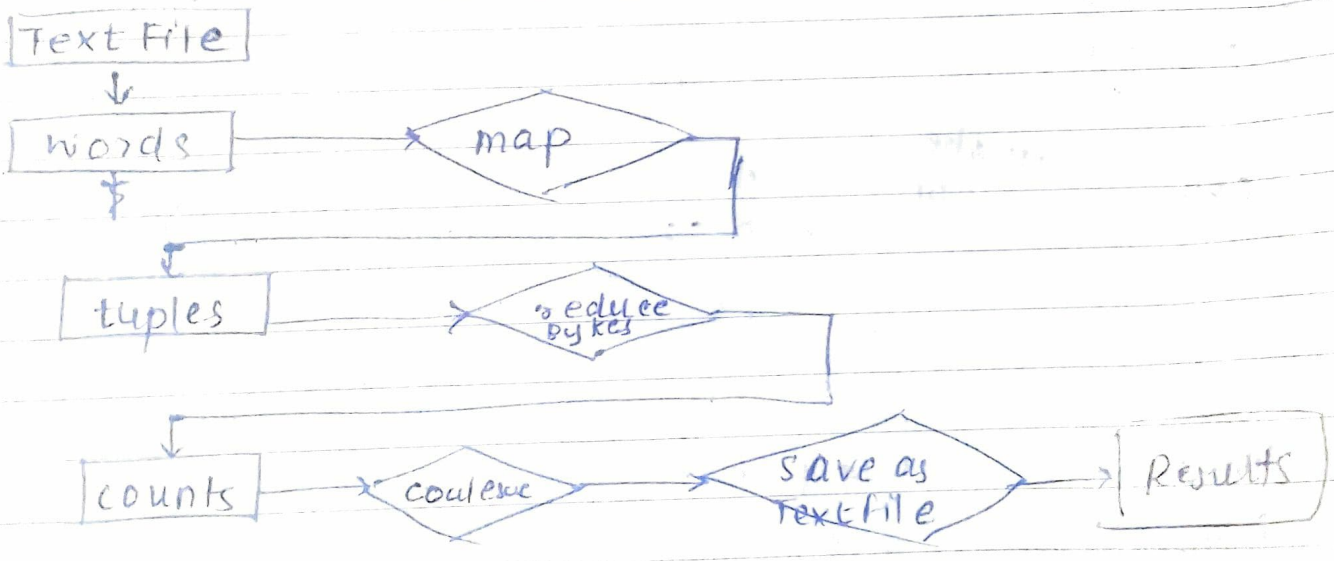
wk	M	T	W	T	F	S	S
32	1	2	3	4	5	6	7
33	8	9	10	11	12	13	14
34	15	16	17	18	19	20	21
35	22	23	24	25	26	27	28
36	29	30	31				

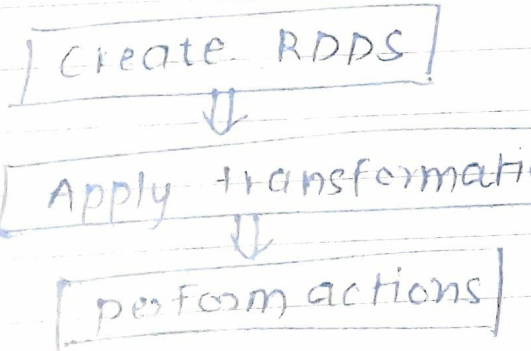
Processing RDDs



→ lazy evaluation

Word Count





NOTES

Spark Core :- Transformations

→ RDD gets generated from external datasets and get partitioned.

RDD = immutable ⇒ they can't be changed in place even partially.

RDD → transformation → New RDD.

They need a ~~transformation~~ operation applied to them and get

* This is essential for keeping track of all processing that has been applied to our dataset providing the ability to keep a linear chain of RDDs.

→ In pipeline :- several transformation steps, many other RDDs as intermediate products gets executed until we got to our final result.

Spark → transformation are Lazy.

It will not execute immediately

→ Narrow Transmission vs Wide Transmission

map, flatmap, filter and coalesce
⇒ Narrow Transmission.

NOTES

Home network ⇒ It applies the function to each partition or element of an RDD
⇒ one to one transformation
⇒ element wise transformation

* In some cases, even if you started with even partitions within the worker nodes, the RDD size ca significantly vary across the workers after a filte operations, then this happens is a good idea to join some of partitions to increase performance and even out processing across clusters. This transformation is called

coalesce

→ It helps with balancing the data partition numbers and sizes

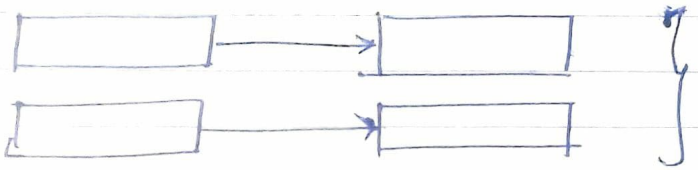
Transformations :- RDD = immutable

→ Difference bt^n a narrow transformation and wide transformation

→ Define map, flatmap, filter and coalesce as narrow transformations.

→ List two wide transformations.

① map : Apply function to each element of RDD



one to one transformation

```
def lower(line):
    return line.lower()
lower_text_RDD = text_RDD.map(lower)
```

lower case letters

② flatmap : map then flatten o/p

NOTES

1. map (func) :- Returns a new distributed dataset, formed by passing each element of source through a function func.

2. filter (func) :- Returns a new dataset formed by selecting those elements of the source on which func returns true.

3. flatMap (func) :- Similar to map, but each i/p item can be mapped to 0 or more o/p items (so func should return a seq. rather than a single item)

4. mapPartitions (func) :- similar to map, but runs separately on each partition (block) of the RDD, so func must be of type Iterator<T> => Iterator<U> when running on an RDD of type T.

5. mapPartitionsWithIndex (func)
Similar to mapPartitions, but also provides func with an Integer value representing the index of the partition, so func must be of type (Int, Iterator<T>) => Iterator<U> when running

NOTES

T/W/T/F/S/SU/ Date

Actions :- Following table gives a list of Actions which returns value.

1. reduce (func)
→ Aggregate the elements of the dataset using a function func (which takes two arguments and returns one). The function should be commutative and associative so that it can be computed in parallel.

2. collect ()
→ Returns all the elements of the dataset as an array at the driver program. This is usually useful after a filter or other operation that returns a sufficiently small subset of the data.

3. count () :- Returns the no. of elements in the dataset.

4. first () :- Returns the first element of the dataset (similar to take (1))

5. take (n) :- Returns an array with the first n elements of the dataset

6. takeSample (withReplacement, num, [seed])
→ Returns an array with a random sample of num elements of the dataset, with or without replacement, optionally pre-specifying a random number generator seed.

Priority

• Can you paint the outer wall neglecting the inner wall •

7. takeOrdered (n, [ordering])

Returns the first n elements of the RDD using either their natural order or a custom comparator.

8. saveAsTextFile (path)

Writes the elements of the dataset as a text file (or set of text files) in a given directory in the local filesystem, HDFS or any other Hadoop-supported file system. Spark calls toString on each element to convert it to a line of text in the file.

9. saveAsSequenceFile (path) (Java & Scala)

→ Writes the elements of the dataset as a Hadoop SequenceFile in a given path in the local filesystem, HDFS or any other Hadoop-supported file system.

This is available on RDDs of key-value pairs that implements Hadoop's writeable interface.

10. saveAsObjectFile (path) (Java & Scala)

Writes the elements of the dataset in a simple format using Java serialization, which can then be loaded using SparkContext.objectFile()

11. CountByKey() - Only available on RDD of type (K, V). Returns a hashmap of (K, Int) pairs with the count of each key.

T / W / T / F / S / SU /

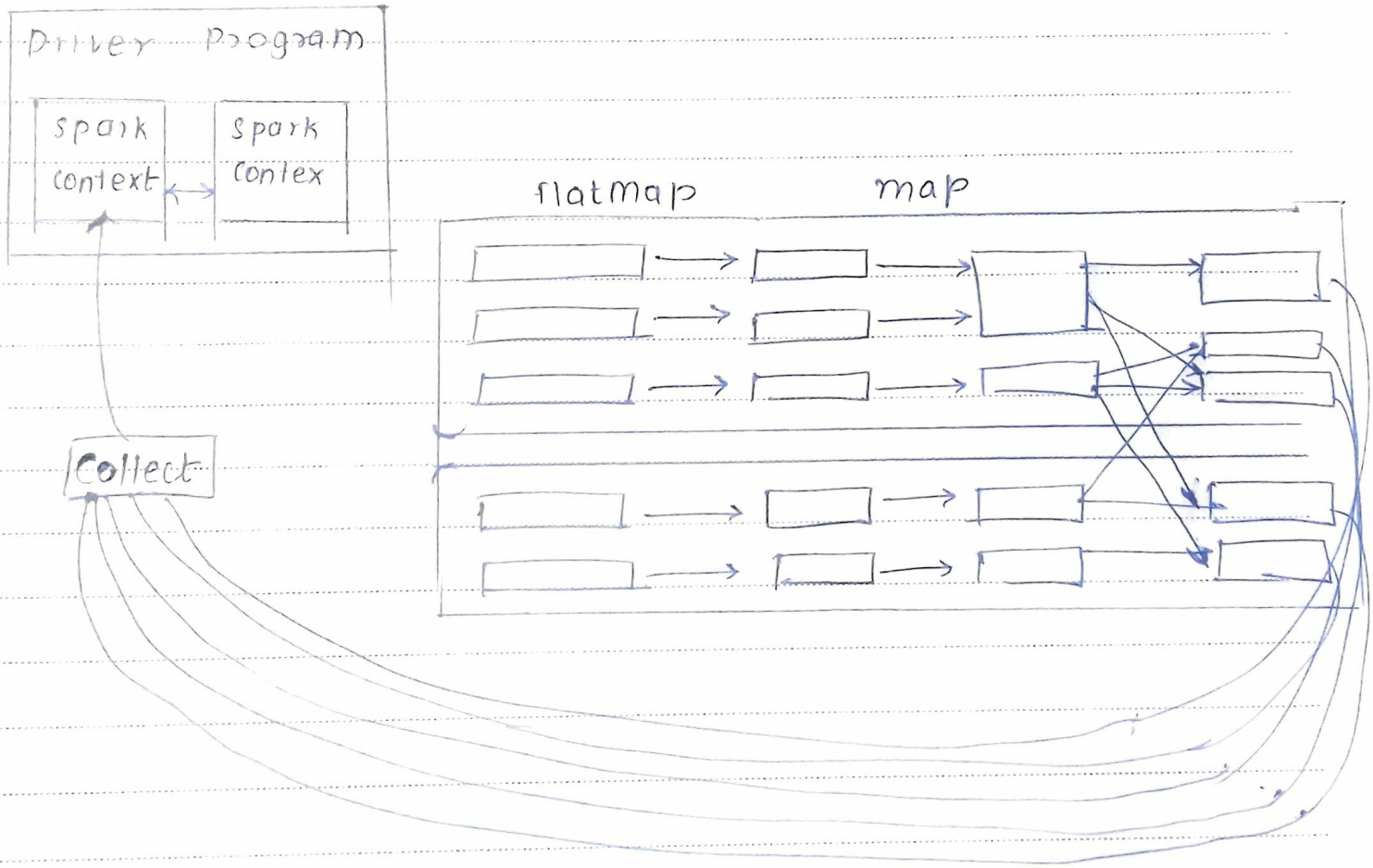
Date

2. foreach (func) :- Runs a function func on each element of the dataset. This is usually, done for side effects such as updating an Accumulator or interacting with external storage systems.

Coursera :- Spark Core - Actions

→ Explain the steps of spark pipeline ending with a collect actions.

→ List four common actions operations in spark.



Some common Actions

1. collect
2. take(n)
3. reduce (func)
4. ~~cf~~ ? → save to local file or ~~HDFS~~

Priority

provides a common query language · It is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.

video 4

Spark SQL :- Spark SQL is the component of SQL that enables querying structured and unstructured data through a common language.

→ It can connect to many data sources and provide APIs to convert the query results to RDDs in python, Scala, Java programs.

→ Spark SQL gives a mechanisms for SQL users to deploy SQL queries on Spark.

→ Business intelligence tools to connect to spark using protocols like JDBC and ODBC.

→ Spark SQL also provides API to convert query data into DataFrames to hold (distributed data named columns) ↓ look like relational databases.

S1 : To run any SQL spark is to create a SQLContext. From pyspark.sql import SQLContext
sqlContext = SQLContext(sc)

S2 : Once you have an SQL context, you want to leverage it to create a DataFrame so you can deploy complex operations on the data set.

DataFrames ← existing RDD
← Hive Tables
← any other data source

Relational Operations :- Sunday 13

→ perform Relational processing such as declarative Queries

NOTES

→ Embed SQL queries inside Spark progra.

leverages spark core's fast scheduling capability to perform streaming analytics. It ingests

14 | 2016 data in mini-batches
Monday and performs RDD
November (Resilient Distributed Datasets) transformations

319-047 • WK 47

October 2016

wk	M	T	W	T	F	S	S
40/45	31					1	2
41	3	4	5	6	7	8	9
42	10	11	12	13	14	15	16
43	17	18	19	20	21	22	23
44	24	25	26	27	28	29	30

wk	M	T	W	T	F	S	S
45		1	2	3	4		
46	7	8	9	10	11	12	
47	14	15	16	17	18	19	
48	21	22	23	24	25	26	27
49	28	29	30				

RDDs can be converted to dataframe of data on those mini-batches

9:00 (1) Convert each line into a row.

(2) Data is in dataframe, ⇒ perform all sort of transformation

10:00 (show, printschema, select, filter, groupby)

(1) How spark reads streaming data
(2) List several sources of streaming data
(3) Spark's sliding windows

11:00 deos

Spark Sheamng :-

12:00 (1) Spark streaming provides scalable processing for

eal-time data and runs on top of spark core.

1:00 continuous data streams ⇒ grouped / discrete RDD ⇒ parallel

2:00

Spark Streaming provides APIs for Scala, Java at

3:00 Python, like other spark products.

Spark streaming can read data from many different types of resources including Kafka and Flume.

5:00 (1) Kafka :- High throughput published subscribed messaging system.

6:00 (2) Flume :- collects and aggregates log data.

Spark streaming can also read from batch i/p data source, such as HDFS, S3, at many other non data sources.

(3) (4) (5) Twitter (6) socket

NOTES Spark streaming reads streaming data and converts in into micro batches which we call DStreams which is short for discretized stream.

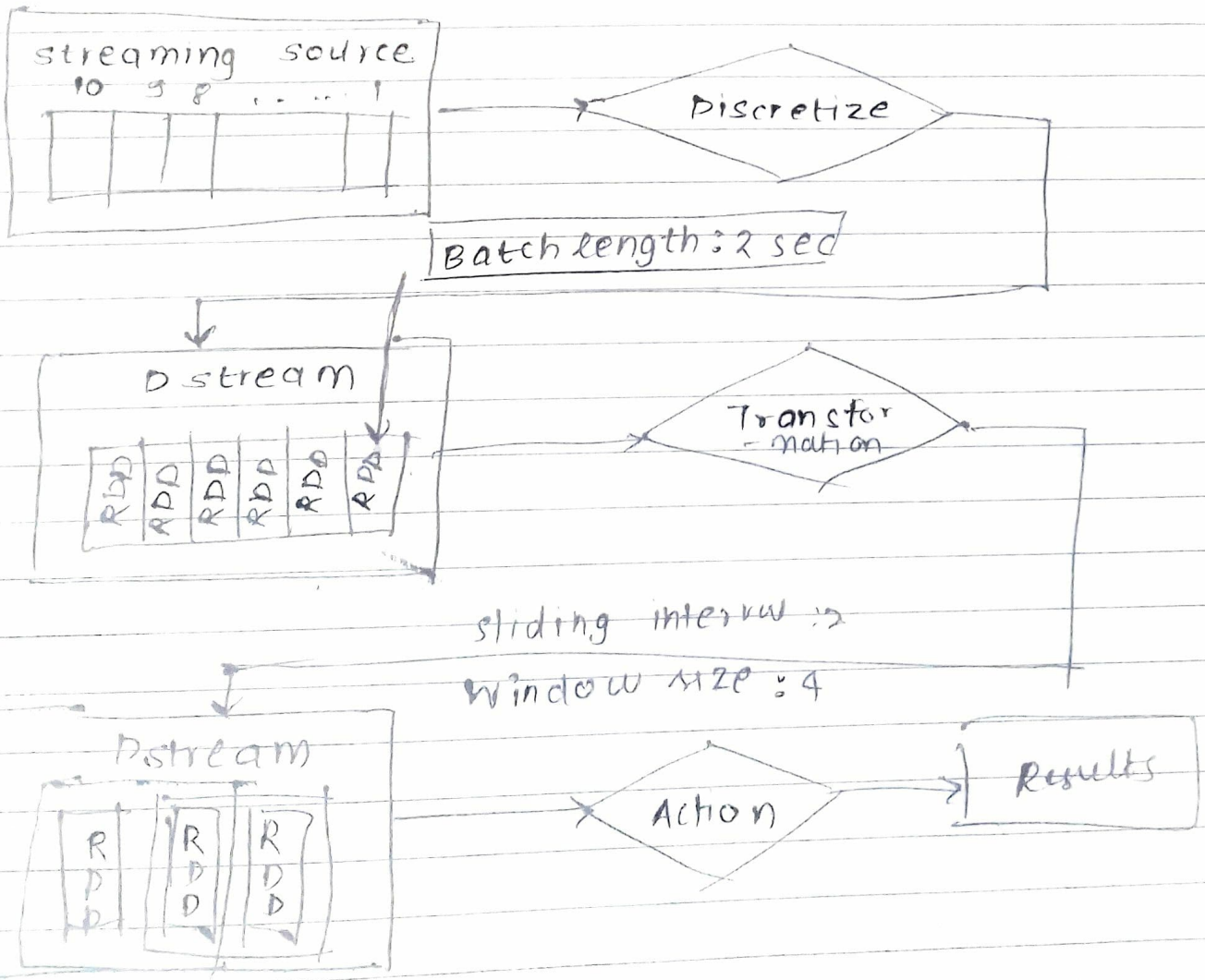
Transformations ⟹ Dstreams
(map, reduce, filter)

→ Spark streaming is sparks library to work with streaming data in near real time.

→ Dstreams can be used just like any other RDD and can go through the same transformation as batch datasets

→ Dstreams can create a sliding window to perform calculations on a window of time.

Creating and Processing Dstreams



NOTES

leverages spark core's fast scheduling ab

MLlib is a distributed ~~mla~~ ml framework because of the distributed memory-based spark Architecture. It is according to benchmarks done by the MLlib developers against the alternating least squares (ALS) implementations.

16 | 2016 | Wednesday | November | 321-045 • WK 47

October 2016

wk	M	T	W	T	F	S	S
10						1	2
40/45	31						
	3	4	5	6	7	8	9
42	10	11	12	13	14	15	16
43	17	18	19	20	21	22	23
	24	25	26	27	28	29	30
11							
45							
46	7	8	9	10	11	12	
47	14	15	16	17	18	19	
48	21	22	23	24	25	26	
49	28	29	30				

Nove

Main Take-Aways

→ Spark uses Dstreams to make discreate RDDs from streaming data.
- same transfromations and calculations applied to batch RDDs can be applied.

→ DStreams can create a sliding window to perform calculations on a window of time.

ideo

Spark MLlib :- What MLlib is

→ List main categories of techniques available in MLlib

→ Code segments containing MLlib algo.

Spark MLlib :- scalable ml library

→ Provides distributed implementations of common ml algo and utilities.

→ Has APIs for Scala, Java, Python, and R.

ML Algo & techniques:-

- ML learning
  - → classification
  - → regression
  - → clustering
- statistics
  - → summary statics
  - → sampling
- Utilities
  - → Dimensionality reduction
  - → transformation

→ Evaluation metrics

Spark MLlib is nine times as fast as the Hadoop disk-based version of Apache Mahout.

NOTES

MLib Ex. → Summary Statistics

- Compute column summary statistics

```
from pyspark.mllib.stat import statistics
    # Data as RDD of Vectors.
dataMatrix = sc.parallelize([[1,2,3],[4,5,6],[7,8,9],[10,11,12]])
    # compute column summary statistics
summary = statistics.colStats(dataMatrix)
print(summary.mean())
print(summary.variance())
print(summary.numNonzeros())
```

MLib Exa. - Classification

* Build decision tree model for classification

```
>> from pyspark.mllib.tree import DecisionTree,
                DecisionTreeModel
>> from pyspark.mllib.util import MLUtils
    # Read and parse data.
data = sc.textFile("data.txt")
# Decision tree for classification
model = DecisionTree.trainClassifier(parsedData, numClasses=2)
print(model.toDebugString())
model.save(sc, "decisionTreeModel")
```

July 2015

23 | 2016 Wednesday November

328-038 • WK 48

10	October 2016							
wk	M	T	W	T	F	S	S
40/45	31					1	2
41	3	4	5	6	7	8	9
42	10	11	12	13	14	15	16
43	17	18	19	20	21	22	23
44	24	25	26	27	28	29	30

11	November 20						
wk	M	T	W	T	F	S	S
45		1	2	3	4	5	6
46	7	8	9	10	11	12	13
47	14	15	16	17	18	19	20
48	21	22	23	24	25	26	27
49	28	29	30				

MLib Example - clustering

Build k-means model for clustering

```
>> from pyspark.mllib.clustering import kmeans,
                kMeansModel
>> from numpy import array
   # Read and passe data
>>  data = sc.textFile ("data.txt")
>> parsedData = data.map (lamba line:
      array([float(x) for x in the line.split(' ')]))

   # k-means model for clustering
>> clusters = Kmeans.train (parsedData, K=3)
>> print (clusters.centers)
```

MLlib is spark's machine learning library

- Distributed implementations

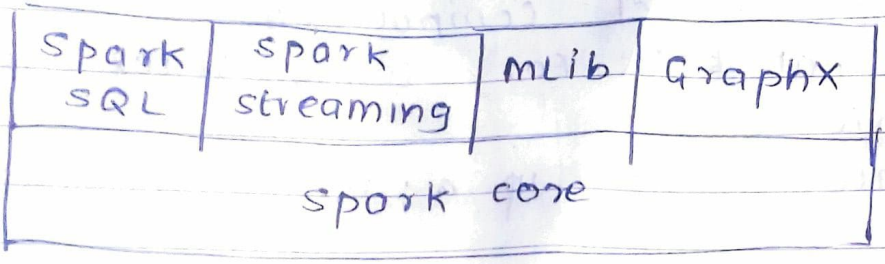
Main categories :-

NOTES

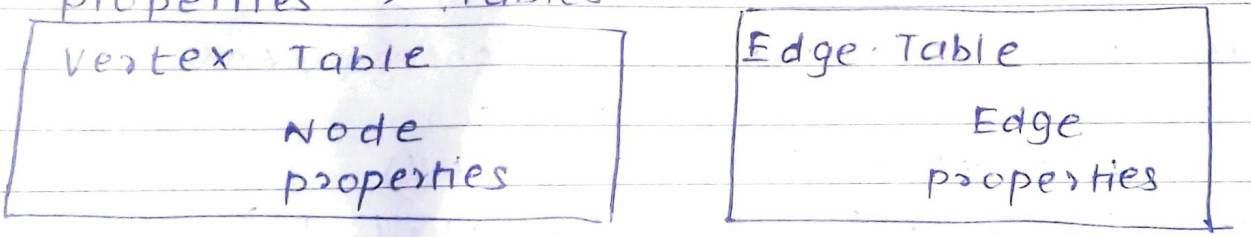
GraphX is a distributed graph-processing on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

Video :-

Spork GraphX :- Describe what GraphX is
→ Explain how Vertices and Edges are stored
→ Describe how pregel works at a high level.



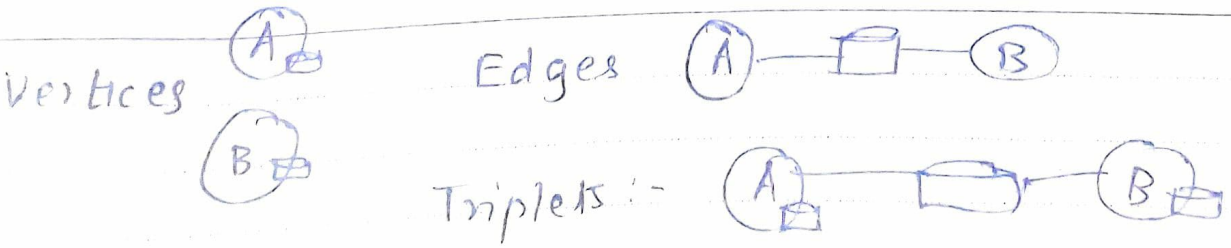
GraphX is Apache Spark's API for graphs and graph-parallel computation.
uses a property graph model.
⇒ Both Nodes and Edges can have attributes and values.
properties → Tables



GraphX uses special RDDs

Vertex RDD[A] extends RDD[(VertexID, A)]
Edge RDD[ED, VD] extends RDD[Edge[ED]]

Triplets :- The triplet view logically joins the vertex and edge properties.



Pregel API :- Bulk-synchronous parallel messaging mechanism.

→ Constrained to the topology of the graph.

GraphX :- → Graph parallel computations
→ Special RDDs for storing Vertex and Edge info.

→ Pregel operator works in a series of super steps.

JSON → DataFrame

```
# Read
df = sqlContext.read.json("/filename.json")
# Display
df.show()
```

RDD of Row objects → Dataframes

```
From pyspark.sql import SQLContext, ROW
sqlContext = SQLContext(sc)
lines = sc.textFile("filename.txt")
cols = lines.map(lamba l: l.split(","))
data = cols.map(lambda l: Row(name=p[0],
                              zip=int(p[1])))
df = sqlContext.createDataFrame(data)
df.registerTempTable("table")
Output = sqlContext.sql("SELECT * FROM table
                         WHERE ...")
```

Notes

Which part of spark is in charge
→ Driver program.

How does lazy evaluation work in Spark?
→ Transformations are not executed until the action stag

What are the consequences of lazy evaluation as mentioned in 2e?
→ Errors sometimes don't show up until the action stage

Wide Transformation:- A transformation that requires data shuffling across node partitions.

Where does the data for each worker node get sent to after a collect function is called?
→ spark context.

Dataframes:- A column like data format that can be read by Spark SQL.

Can RDD's be converted into Dataframes directly without manipulation?
→ No: lines have to converted into row.

NOTES Functions of Spark SQL:-
1. Enables relational queries on spark.
2. Deploy business intelligence tools over spark.
3. Connect to variety of databases.

Apache spark

18 | 2016 Friday November

323-043 • WK 47

9. What is a triplet in GraphX?

→ A type of data to contain the information on connections b/t vertices and edges.

1. How may tweets have location not null?

→ 6937

2. How many people have more followers than friends?

→ 5809