Mapreduce
==========

Mapreduce is used to process huge amount of data.

Map Phase - this gives us parallelism

Reduce Phase - aggregation

try doing more work at mapper end and try doing minimum work at reducer end..

Try to shift your computation from reducer to mapper..

Linkedin profile views
=======================

ID,From-Member,To-Member
1,Manasa,Sumit
2,Deepa,Sumit
3,Sumit,Manasa
4,Manasa,Deepa
5,Deepa,Manasa
6,Shilpy,Manasa

I want to check how many people viewed Sumit's profile

Manasa's profile

Deepa's profile

(0,[1,Manasa,Sumit])
(1,[2,Deepa,Sumit])
(2,[3,Sumit,Manasa])

The above is given to the mapper..

The mapper output will be
(Sumit,1)
(Sumit,1)
(Manasa,1)
(Deepa,1)
(Manasa,1)
(Manasa,1)

So the above key value pairs are shuffled and given to the reducer machine.

(Deepa,1)

(Manasa,1)
(Manasa,1)
(Manasa,1)
(Sumit,1)
(Sumit,1)

(Deepa,{1})
(Manasa,{1,1,1})
(Sumit,{1,1})

our reducer will be working on the above

(Deepa,1)
(Manasa,3)
(Sumit,2)

How to calculate number of views for each profile on linkedIN.

How many mappers are launced & how many reducers?

for example if you have a 1 GB file..

128mb..

8 blocks..

you just have a 3 node cluster

8 mappers will be running..

How many reducers are launched?

By default we have only 1 reducer..

But as a developer we can even change the number of reducers..

we can even make it 0 or we can make it more than 1.

Changing the Number of Reducers
==========

Default is 1

When should we consider to increase the number of reducers.

5 mappers are completed in 2 mins... and the 1 reducer is taking 10 mins..

12 mins..

5 mappers are completed in 3 mins... and the reducer is taking 5 mins..

8 mins..

5 mappers and 2 reducers..

mappers will take 3 mins.. and 2 reducers 2.5 mins each..

5.5 mins..

when to make the number of reducers to 0

There can be few jobs which do not require aggregation..

Filter..

Shuffle and sort only happens when we have 1 or more reducer..

so when the number of reducers in 0 then your mapper output will be the final output.

1 GB file..

as per block size of 128 mb we have 8 blocks

and eventually we have 8 mappers..

8 output files will be there..

Concept of partition comes into play when we have more than one reducer.

This is to tell which key value pair goes to which reducer..

By default there is a hash function..

mod

3 reducers - 0,1,2

1%3 = 1
2%3 = 2
3%3 = 0
4%3 = 1

5%3 = 2
6
7
8
9

That this hash function is consistent..

(Hello,1) - Reducer1
(hi,1) - Reducer2
(hello,1) - Reducer1

Reducer1 - 30 (hello,30)

if you do not want to use the system defined hash function..

all the words whose length is less than 4 chars should go to reducer 1

and all the words whose length is greater than or equal to 4 chars should go to reducer 2

In such cases then we can go with our own custom partitioning logic..

hi

hello

how

is

sumit

converting dollars to rs..

Map -> Partition, Shuffle, Sort -> Reducer

What is shuffle?

Sending the data from mapper machine to reducer machine.

So that reducer machine gets the data to work on.

What is Sort?

Operation done on reducer machine to get the same keys together in a group.

Values with same keys appear as collection to the reducer..

(hi,1)
(hello,1)
(hi,1)
(hi,1)
(hello,1)

(hello,1)
(hello,1)
(hi,1)
(hi,1)
(hi,1)

(hello,{1,1})
(hi,{1,1,1})

One more example
==========

Date,Time,Temperature
12/12/2015,00:00,50
12/12/2015,01:00,52
12/12/2015,02:00,49
12/13/2015,00:00,48
12/13/2015,01:00,54
12/13/2015,02:00,55

Result

12/12/2015,52
12/13/2015,55

300 mb data...

3 blocks - 128 mb

DN1 - b1
12/12/2015,00:00,50
12/12/2015,01:00,52

output from mapper.
(12/12/2015,50)
(12/12/2015,52)

DN2 - b2
12/12/2015,02:00,49
12/13/2015,00:00,48

(12/12/2015,49)
(12/13/2015,48)

DN3 - b3
12/13/2015,01:00,54
12/13/2015,02:00,55

(12/13/2015,54)
(12/13/2015,55)

on reducer machine
==========

(12/12/2015,50)
(12/12/2015,52)
(12/12/2015,49)
(12/13/2015,48)
(12/13/2015,54)
(12/13/2015,55)

(12/12/2015,{50,52,49})
(12/13/2015,{48,54,55})

(12/12/2015,52)
(12/13/2015,55)

Ideally we want mappers to do more work..

In this above scenario mapper does the bare minimum.

It could have done more..

Combiner

1. Improve the Parallelism

2. Reduce the data transfer

Use combiners with Caution

Max, Min, Sum

using combiner or without using combiner the result
will be the same.

Average

M1 - 1, 3, 5 = 3.0 (9,3)

M2 - 3, 7, 9 = 19/3 = 6.33 (19,3)

M3 - 4, 7, 8, 7 = 26/4 = 6.5 (26,4)

Reducer

(54,10) = 5.4

(3.0,6.33,6.5) = 15.83/3 = 5.2

correct answer should be 54/10 = 5.4

Mapreduce session 5
==========

google actually used mapreduce for their existing web search.

A lot of crawlers are working day and night to crawl the web.

key	value
flipkart.com	clothes handbag laptop
amazon.com	clothes mobile purse
myntra.com	purse clothes tv

search: clothes
flipkart.com
myntra.com
amazon.com

search purse
amazon.com
myntra.com

key	value
clothes	amazon.com, myntra.com, flipkart.com
purse	amazon.com, myntra.com

key	value
flipkart.com	clothes handbag laptop
amazon.com	clothes mobile purse
myntra.com	purse clothes tv

output from mapper

key	value
clothes	flipkart.com
handbag	flipkart.com
laptop	flipkart.com
clothes	amazon.com
mobile	amazon.com

purse amazon.com

(clothes,{flipkart.com,amazon.com,myntra.com})
(purse,{amazon.com,myntra.com})

Lets Execute
==========

Map

combiner

shuffle

sort

Reducer

count the frequency of each word

mapreduce_prog.jar

gateway node

in gateway node
/home/itv005857/data/input/inputfile.txt

in hdfs
/user/itv005857/data/input

hadoop fs -put /home/itv005857/data/input/inputfile.txt /user/itv005857/data/input

jar (java archive)

mapreduce_prog.jar

hadoop jar <jarname> <input_file_path> <output_directory>

hadoop jar mapreduce_prog.jar /user/itv005857/data/input/inputfile.txt
/user/itv005857/data/output

a new dir with the name output is created

it contains 2 files

_SUCCESS
part-r-00000

gateway node
/data/trendytech/bigLog.txt

/user/itv005857/data/input

=====

0 reducer - mapreduce_prog_0_reducer.jar
==========

the mapper output will be the final output.. there will be no shuffle and sort..

==
big data is interesting
big data is trending technology
===

hadoop jar mapreduce_prog_0_reducer.jar /user/itv005857/data/input/inputfile.txt
/user/itv005857/data/output

2 reducers
==========

==
big data is interesting
big data is trending technology
===

hash function is consistent

hadoop jar mapreduce_prog_2_reducer.jar /user/itv005857/data/input/inputfile.txt
/user/itv005857/data/output

custom partitioner
==========

2 reducers..

if the word length is less than or equal to 3 than it should go to reducer0 and all the other words should go to the second reducer reducer1

mapreduce_prog_cpartitioner.jar

hadoop jar mapreduce_prog_cpartitioner.jar /user/itv005857/data/input/inputfile.txt
/user/itv005857/data/output

combiner
=========

hadoop jar mapreduce_prog_combiner.jar /user/itv005857/data/input/inputfile.txt /user/itv005857/data/output

hadoop jar mapreduce_prog_combiner.jar /user/itv005857/data/input/bigLog.txt /user/itv005857/data/output

Apache Spark
==========

Hadoop - HDFS / MR / YARN

Challenges with Mapreduce

1. Performance - lot of Disk IO's
2. Its really hard to write the code
3. Mapreduce supports only Batch processing
    - Batch
    - Realtime Processing/Streaming
4. Hive, Pig, Sqoop... Have to learn a lot of things...
5. Map Reduce - Filter, map will work and reduce won't..
6. Interactive mode

Apache Spark
==========

Distributed Processing / Compute Engine

Spark does not come with storage and it does not come with resource manager..

Spark + Datalake + Resource Manager

Spark says I am a plug and play compute engine ...

you plug me with any storage of your choice -
HDFS / S3 / ADLS Gen2 / GCS / Local

you plug me with any resouce manager of your choice -
YARN / Mesos / Kubernetes

Apache spark is a multi language engine for executing data engineering, data science and machine learning on single node or cluster.

Scala, python, Java, R

Spark with Python - Pyspark

General Purpose
Inmemory -
Compute Engine -

MR-Job-1 MR-Job-2 MR-Job-3 MR-Job-4 MR-Job-5

==========

HDFS

chain of 5 mapreduce jobs - 10 disk IO's
but in spark it could be as less as 2 disk IO's

Spark Abstracts away the notion that we are writing code to run across the cluster

=> The challenges of Mapreduce
=> what is apache spark
=> its just a compute engine

Apache Spark vs Databricks
==========

Apache Spark is Open Source.

Databricks is a company and they have a product also called as databricks.

Extra features

- Spark on Cloud - AWS / Azure / GCP
- optimized spark
- Cluster Management
- Delta Lake
- Collaborate Notebooks
- Implemented Security

Spark Core API's - you work at the RDD level

RDD stands for Resilient Distributed Dataset

We can write the code based on RDD's - the hardest way to work with Apache Spark

Higher Level
- Spark SQL / Data Frames
- Structured Streaming
- Mllib
- GraphX

RDD - toughest Most Flexible
DataFrame - Medium medium Flexibility
Spark SQL - easiest Less flexible

3 things
=========

1. Load the data from the source - HDFS / Cloud Datalakes

2. Do various transformations - filter, join, groupby

3. Write the final results to the target location

RDD is the basic unit which holds the data in apache spark

Resilient Distributed Dataset

=> Apache Spark vs Databricks
=> Spark core API's
-RDD
High Level API's
-Spark SQL/DataFrame
-Structured Streaming
-Mllib
-GraphX

=> Load -> Transform -> Write

=> RDD

RDD

512 mb - 4 blocks

I have a file in hdfs names file1 (512 mb file)

rdd1 = load file1 from datalake (hdfs / Cloud datalake)

rdd2 = rdd1.map

rdd3 = rdd2.filter

rdd3.collect()

There are 2 kind of operations in apache spark

1. Transformation
2. Action

Tranformations are Lazy
but actions are not

DAG - Directed Acyclic Graph

A Driver and multiple workers...

RDD - Resilient Distributed Dataset

RDD is resilient to failures

RDD's are immutable

that means you cannot make the changes to the existing RDD

you end up creating a new RDD always for a transformation...

RDD1 = RDD1.map

RDD1 = RDD1.filter

why tranformations are Lazy?

materializing the rdd

rdd1 = load file1 from datalake (hdfs / Cloud datalake)

rdd2 = rdd1.map

rdd3 = rdd2.filter

rdd3.collect()

you have a file in hdfs which is having 1 billion records..

rdd1 = load file1 from hdfs
print the first line from this rdd1..

rdd1 = load file1 from hdfs (1 billion)
rdd2 = rdd1.map (1 billion)
rdd3 = rdd2.filter (5 records)
rdd3.collect()

filter - 5 records
map - 5 records

```
from pyspark.sql import SparkSession
import getpass
username = getpass.getuser()
spark = SparkSession. \
    builder. \
    config('spark.ui.port', '0'). \
    config("spark.sql.warehouse.dir", f"/user/{username}/warehouse"). \
    enableHiveSupport(). \
    master('yarn'). \
    getOrCreate()
```

SparkSession acts an entry point to the spark cluster..

spark context
hive context
sql context

spark session

3 things in spark

1. Load a file
2. Do bunch of transformations
3. save the results

RDD - sparkcontext
DataFrame - sparksession
Spark SQL - sparksession

```
rdd1 = spark.sparkContext.textFile("/user/itv005857/data/inputfile.txt")


rdd2 = rdd1.flatMap(lambda x : x.split(" "))

rdd3 = rdd2.map(lambda word : (word,1))

rdd4 = rdd3.reduceByKey(lambda x,y : x+y)
```

(big,2)
(data,4)

```
rdd1 = spark.sparkContext.textFile("/user/itv005857/data/inputfile.txt")

rdd2 = rdd1.flatMap(lambda x : x.split(" "))

rdd3 = rdd2.map(lambda word : (word,1))
```

rdd4 = rdd3.reduceByKey(lambda x,y : x+y)

rdd4.collect()

inputfile.txt

=> load

rdd1

=> flatMap

rdd2

=> map

rdd3

=> reduceByKey

rdd4

rdd4.saveAsTextFile("/user/itv005857/data/newoutput")

http://m02.itversity.com:18080/