Map Reduce - Week 2

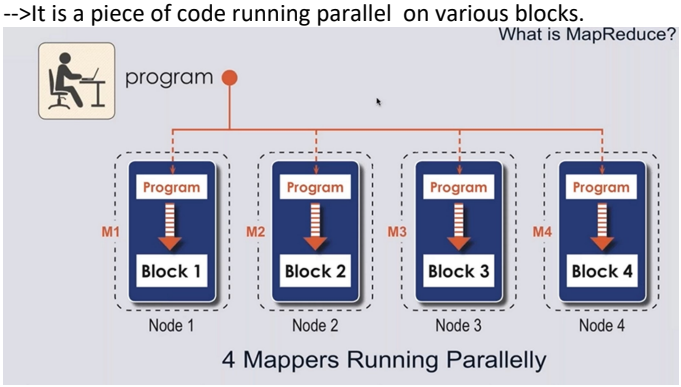
Tuesday, June 15, 2021 10:52 PM

Map reduce

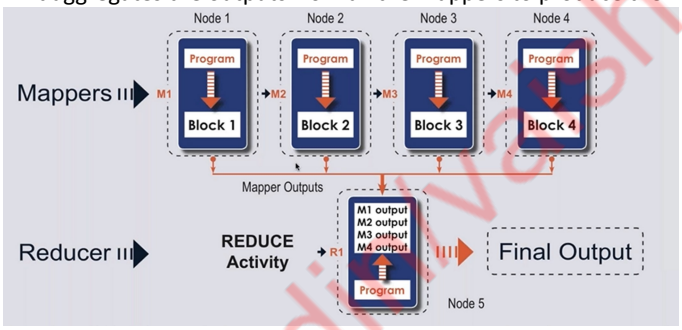
--> is a computing paradigm in distributed systems for processing data.
-->It works on the concept of key-value pair (k,v)
-->It has mainly two phases namely Map and Reduce
-->Input and output in both map and reduce will be (key,value) only
-->It is a programming paradigm(ie a special way of programming is adopted to solve the problem) because in traditional programming models data is kept only on a single machine.

-->Hadoop works on the principle of **data locality** ie the code moves to the place where data is located and not vice versa. As we have huge data to be processed its easy to move code rather than data.

Map:
-->It is a piece of code running parallel on various blocks.

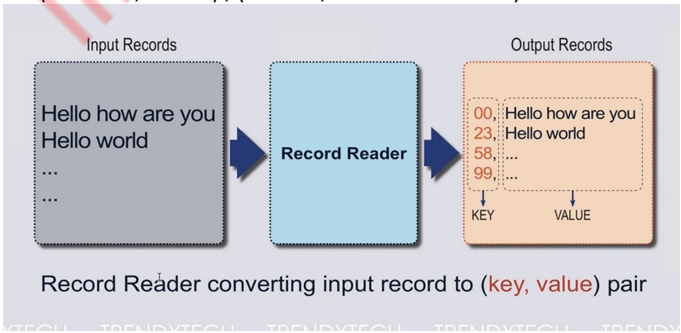


Reduce:
-->It aggregates the outputs from all the mappers to produce the final result



-->**How map-reduce works?**

1. After we have got the data and its divided into blocks.
2. Now we have to make our mapper code run on the blocks. For this we need to have (key, value) pairs as mapper can work only on (key, value)
3. Record reader comes into our rescue. It converts the data into (key, value) pairs
   Where key=random addr value=data
   Ex. (4938920, vaishu) , (483729, You are awesome)

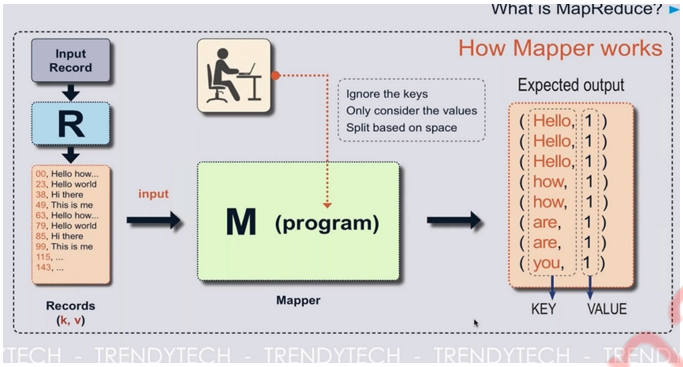


-->**How mapper function is applied?**

1. Mapper code/program ignores the address as it has no significance in computing the records
2. Mapper code has to be written by the developer
3. It counts the frequency of each words and gives it a value
   Ex. Vaishu you are awesome vaishu you are extraordinary vaishu
   Mapper code is applied and gives output as

   (vaishu , 1)
   (are, 1)
   (awesome, 1)
   (vaishu, 1)
   (you, 1)
   (are, 1)
   (extraordinary, 1)
   (vaishu, 1)

4. For each word it sets a frequency
5. On each block this mapper code runs and produces frequency for each word



-->How reducer works?

1. Shuffling: Process of moving mapper output data to reducer node
2. Sort: Sorts the result in ascending order
3. Suffle and sort is taken care by the framework
4. After shuffle & sort data reaches the reducer node
5. Reducer code has to be written by developer
   In mapper:

   (vaishu , 1)
   (are, 1)
   (awesome, 1)
   (vaishu, 1)
   (you, 1)
   (are, 1)
   (extraordinary, 1)
   (vaishu, 1)

   Data aggregated:

   (are, 1)
   (are, 1)
   (awesome, 1)
   (extraordinary, 1)
   (vaishu, 1)
   (vaishu, 1)
   (vaishu, 1)

   After shuffle and sort, data is in below format:
   Input Data in reducer:

   (are, {1,1})

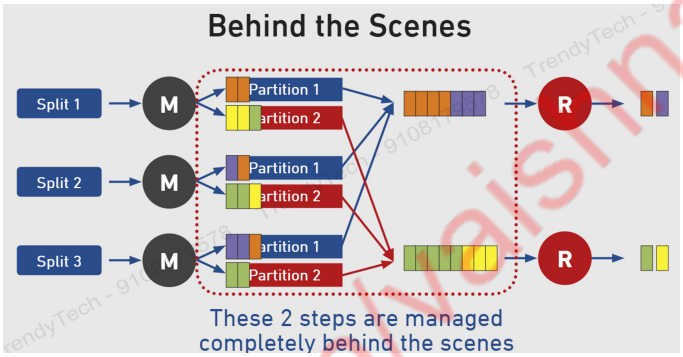
(awesome, {1})
(extraordinary, {1})
(vaishu, {1,1,1})

Output from reducer:

(are, 2)
(awesome, 1)
(extraordinary, 1)
(vaishu, 3)

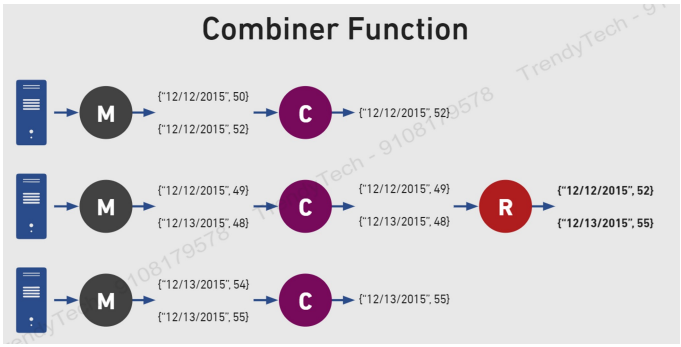
****Note:**
-->No. of reducers(default is 1) can be set by developer based on data requirements to achieve parallelism
-->Number of blocks = no. of mappers //Based on no. of blocks mappers will be generated by hadoop
-->Single data node can have multiple blocks which in turn have multiple mappers to process them.
Parallelism within a data node depends on no. of cores(CPU) it has ie if there is a data node which is 2 CPU machine and there are 4 blocks then at first 2 blocks will be processed after which the other 2 blocks will be processed.
-->Data in a single block having a single mapper can be processed only serially(one after the after)
-->Mapper and reducer code has to be written by the developer and the rest is taken care by hadoop
[MAPPER CLASS] [REDUCER CLASS] [MAIN CLASS]

How shuffling & sorting occurs when we have more than 1 reducer?
-->Output from mapper is fed into partitions
-->Each key in mapper output is assigned to a partition
-->By default hadoop uses hash function to map the keys to a partition
-->shuffle and sort takes place in partitions which are then fed into reducer as input.
MAP --> [PARTITIONING SHUFFLE SORT]--> REDUCE



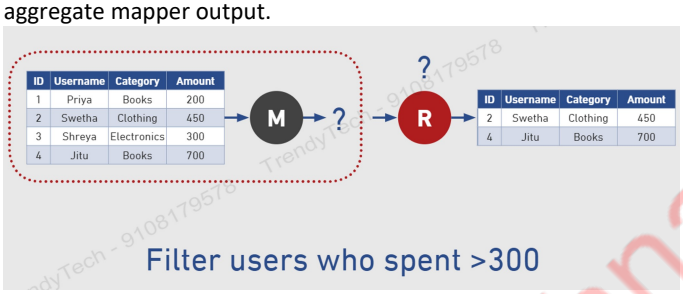
How we can reduce data transfer between nodes and improve parallelism?
How to reduce load on reducer?
-->Combiner is used to perform part of reduce functions on mapper output before it is sent to reduce node.
-->we always need to make sure that max of work is done at mapper end as we have parallelism implemented here. Due to this we use combiner
-->Make sure combiner doesn't change the logic of computing and produce results different from that of mapper output which may result in wrong output.
-->Reducers can be same as combiners for functions like MIN, MAX,SUM as they produce same results.
-->For functions like AVG,MEAN they can't be same

MAP REDUCE --> COMBINER --> REDUCE

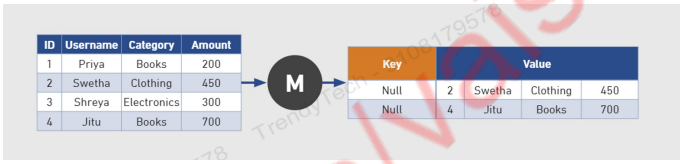


-->Improves parallelism (because more processing is done at mapper side due to which we need more mappers as data blocks increases and parallelism improves)
-->Reduce data transfer (because combiner performs part of reduce functions due to which data transferred to reduce node is lessened)

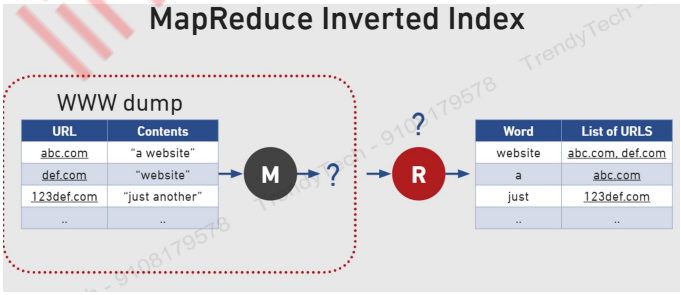
When can we avoid reducers?
-->At instances when only the mapper output can be final result as there is no need for any aggregations to be performed by reducers.
-->Shuffling and sorting also doesn't happen as we don't need reducers
-->In this case we need records >300 ie each mapper can be output records which will be final output, no need for reducers to pitch in and aggregate mapper output.



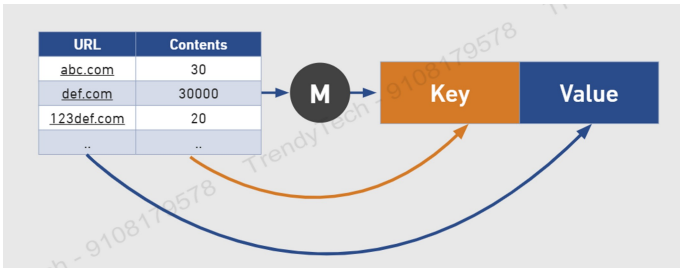
-->Key will be a random address
Value will be considered as the whole record itself

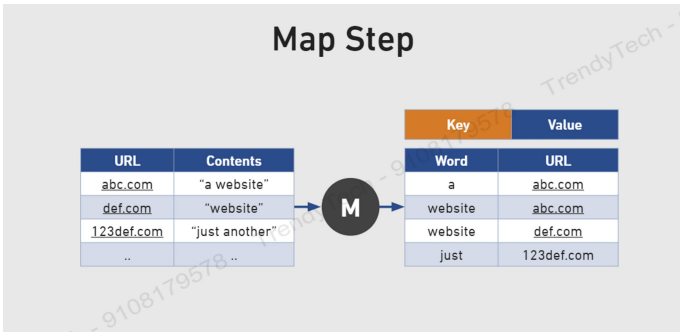


How Map-reduce is used in search engines?
-->Google has a very powerful indexing due to which it became a very popular search engine.
-->Based on the search made by the user the related web pages will be shown to user.
-->How indexing is done using Map-Reduce?
-->Ex. Websites will have a list of huge no. of words in as below which will be converted into key (words) and value(websites having the words) with the help of a web crawler

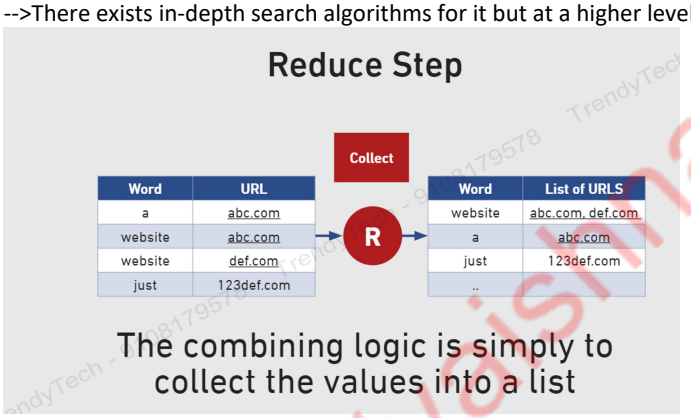


-->At mapper phase the correct key-value pairs are generated
Key - Contents(searches given by user)
Value - URL (result web pages)





-->Reducer aggregates all the keys to a list of values (url web pages) due to which we get a list of web pages in our result page
Ex. (clothes,{amazon.com,flipkart.com , myntra.com})
-->There exists in-depth search algorithms for it but at a higher level this is what is happening

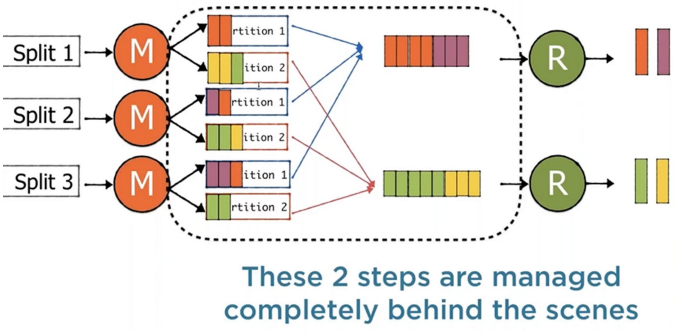


-->How many mappers and reducers do we need?

1. Number of mappers = number of data blocks
2. By default number of reducers is set to 1

When we have more than 1 reducer?
-->We introduce partitions to make sure we deal better with multiple reducers.
-->Partitions send data of only a particular value to a particular reducer ie all key, value pairs having the word "hello" are sent to only 1 reducer so that it can aggregate and give the final count.
-->Hadoop uses a hash function on mapper output to decide which (key, value) pair has to go to which partition
-->The hash function has to be consistent ie for the given set of input the same output has to come always, it should not be changing frequently. hashFun(vaish)=1 ie send to partition 1. Always input "vaishu" has to be sent to 1.

[MAPPER OUTPUT]-->[PARTITIONING]-->[SHUFFLE & SORT]-->[REDUCER INPUT]-->[REDUCER OUTPUT]



Run Map-Reduce Job:

-->while configuring it to run, we mention 2 paths
Input path : where we have the files to be processed
Output path: where our output has to be stored
We give a output folder path which doesn't exist as the program creates a new folder and puts the output file out there.
-->The reason for doing so is that the results are not over written each time the program runs.
-->When the input folder has multiple files, all of them will be executed and put in a single output file
-->No. of reducers = No. of part-r files in the output
-->If the no. of reducers is set to 1 then we have part-m files
-->_SUCCESS indicates program is successful

-->If u includes 2 spaces between words in a line it is considered as a separate word
-->output is in ascending order of words (dictionary order)

Jar-Creation:

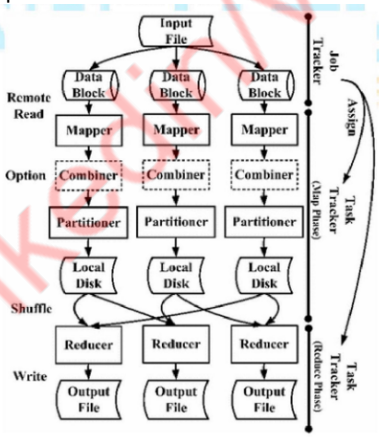
-->In real world scenarios developers create jar files and make the jars run on hdfs rather than local systems
-->Jar : Java Archive which is a bundle of entire code needed to run the program (main, map, reduce)

****Note:**
-->localhost:8088 //to view all the previous map-reduce jobs we ran
-->localhost:50070 //to view hdfs file system in UI
-->If you want to have a specific no. of mappers in your program ie 10 u can divide ur data accordingly before ingesting it into MR program. Default size is 128MB.
-->You can set the no. of reducers in MR program in main class.

20. Consider you have a 500 MB file in HDFS. Number of reducers is set to 0. then how many part files will be created in output folder.
    a. 0
    b. 1
    c. 2
    d. *4

    Explanation: Here there is no reducer so the output from mapper is final. No of mapper = file size/block size =500/128=4

-->No. of mappers = No. of combiners



e.

MAP REDUCE KEYWORDS:

1. Map reduce (computing paradigm works on key-value pairs)
2. Record reader (converts data into key-value pairs)

3. Mapper – where the mapper code is applied on KV pairs and gives results
4. Reducer – Aggregates the output from mapper and gives the final result
5. Reducer side process – shuffling, sorting,
6. Shuffling – moving the data from mapper to reducer side
7. Sorting – sort the output data of mapper at reducer side before sending it to reducer
8. Partitioning – It shuffles and sorts data before sending it to reducer (use case when we have more than 1 reducer)
   [INPUT DATA]-->[MAPPER INPUT]-->[MAPPER OUTPUT]-->[PARTITIONING]-->[SHUFFLE & SORT]-->[REDUCER INPUT]-->[REDUCER OUTPUT]

9. Combiner - perform part of reducer functions at mapper side. It decreases load on reducer as max work done at mapper