BDA UNIT-2 SHORT ANSWERS

1. What is Map Reduce?

Ans. MapReduce is the data processing layer of Hadoop. It is a software framework for easily writing applications that process the vast amount of structured and unstructured data stored in the Hadoop Distributed Filesystem (HDFS). It processes the huge amount of data in parallel by dividing the job (submitted job) into a set of independent tasks (sub-job). By this parallel processing speed and reliability of cluster is improved.

2. How does MapReduce work?

Ans. In Hadoop, MapReduce works by breaking the data processing into two phases: Map phase and Reduce phase. The map is the first phase of processing, where we specify all the complex logic/business rules/costly code. Reduce is the second phase of processing, where we specify light-weight processing like aggregation/summation.

3. What is scaling?

Ans. Generally, we execute the application in a single node. When we distribute our map reduce application on various nodes, we call it scaling out. Running map reduce program in a distributed environment is called scaling.

4. How is key-value pair generated in Hadoop?

Ans. Input Split – It is the logical representation of data generated by the Input Format. In MapReduce program, it describes a unit of work that contains a single map task.

Record Reader- It communicates with the input Split. And then converts the data into key-value pairs suitable for reading by the Mapper. Record Reader by default uses Text Input Format to convert data into the key-value pair.

5. What is a reducer?

Ans. Reducer in Hadoop MapReduce reduces a set of intermediate values which share a key to a smaller set of values. In MapReduce job execution flow, Reducer takes a set of an intermediate key-value pair produced by the mapper as the input. Then, Reducer aggregate, filter and combine key-value pairs and this requires a wide range of processing.

6. What are the different phases of reducer?

Ans. Three phases of Reducer are as follows:

a. Shuffle Phase

This is the phase in which sorted output from the mapper is the input to the reducer. The framework with the help of HTTP fetches the relevant partition of the output of all the mappers in this phase.

b. Sort Phase

This is the phase in which the input from different mappers is again sorted based on the similar keys in different Mappers. Both Shuffle and Sort occur concurrently.

c. Reduce Phase

This phase occurs after shuffle and sort. Reduce task aggregates the key-value pairs. With the OutputCollector.collect() property, the output of the reduce task is written to the FileSystem. Reducer output is not sorted.

7. What are the consequences if there are more number of reducers?

Ans. With the increase of the number of reducers:

- Framework overhead increases.

- Load balancing increases.

- Cost of failures decreases.

8. What is a key, value pair in Hadoop?

Ans. Key– It is field/ text/ object on which the data groups and aggregates on the reducer.

Value– It is the field/ text/ object which each individual reduces method handles.

9. What is an input split?

Ans. InputSplit is the logical representation of data in Hadoop MapReduce. It represents the data which individual mapper processes. Thus the number of map tasks is equal to the number of InputSplits. Framework divides split into records, which mapper processes.

10. How InputSplits are created in Hadoop MapReduce?

Ans. InputSplit is user defined. The user can also control split size based on the size of data in MapReduce program. Hence, In a MapReduce job execution number of map tasks is equal to the number of InputSplits.

11. What is Hadoop InputFormat?

Ans. InputFormat describes how to split up and read input files. In MapReduce job execution, InputFormat is the first step. It is also responsible for creating the input splits and dividing them into records.

12. How we get the data from Mapper?

Ans. Methods to get the data from mapper are: getsplits() and createRecordReader().

13. Types of InputFormat in MapReduce

Ans.

- FileInputFormat

It is the base class for all file-based InputFormats. FileInputFormat also specifies input directory which has data files location. When we start a MapReduce job execution, FileInputFormat provides a path containing files to read. This InpuFormat will read all files. Then it divides these files into one or more InputSplits.

- TextInputFormat

It is the default InputFormat. This InputFormat treats each line of each input file as a separate record. It performs no parsing. TextInputFormat is useful for unformatted data or line-based records like log files. Hence,

Key – It is the byte offset of the beginning of the line within the file (not whole file one split). So it will be unique if combined with the file name.

Value – It is the contents of the line. It excludes line terminators.

-KeyValueTextInputFormat

It is similar to TextInputFormat. This InputFormat also treats each line of input as a separate record. While the difference is that TextInputFormat treats entire line as the value, but the KeyValueTextInputFormat breaks the line itself into key and value by a tab character ('/t'). Hence,

Key – Everything up to the tab character.

Value – It is the remaining part of the line after tab character.

-SequenceFileInputFormat

It is an InputFormat which reads sequence files. Sequence files are binary files. These files also store sequences of binary key-value pairs. These are block-compressed and provide direct serialization and deserialization of several arbitrary data. Hence, Key & Value both are user-defined.

- SequenceFileAsTextInputFormat

It is the variant of SequenceFileInputFormat. This format converts the sequence file key values to Text objects. So, it performs conversion by calling 'tostring()' on the keys and values. Hence, SequenceFileAsTextInputFormat makes sequence files suitable input for streaming.

-SequenceFileAsBinaryInputFormat

By using SequenceFileInputFormat we can extract the sequence file's keys and values as an opaque binary object.

-NlineInputFormat

It is another form of TextInputFormat where the keys are byte offset of the line. And values are contents of the line. So, each mapper receives a variable number of lines of input with TextInputFormat and KeyValueTextInputFormat. The number depends on the size of the split. Also, depends on the length of the lines. So, if want our mapper to receive a fixed number of lines of input, then we use NLineInputFormat.

N- It is the number of lines of input that each mapper receives.

By default (N=1), each mapper receives exactly one line of input.

Suppose N=2, then each split contains two lines. So, one mapper receives the first two Key-Value pairs. Another mapper receives the second two key-value pairs.

DBInputFormat

This InputFormat reads data from a relational database, using JDBC. It also loads small datasets, perhaps for joining with large datasets from HDFS using MultipleInputs. Hence,

Key – LongWritables

Value – DBWritables.

14. What is RecordReader in MapReduce?

Ans. A RecordReader converts the byte-oriented view of the input to a record-oriented view for the Mapper and Reducer tasks for processing. Hadoop RecordReader in MapReduce job execution uses the data within the boundaries that are being created by the inputsplit. And it then creates Key-value pairs for the mapper.

15. Types of Hadoop RecordReader

Ans. InputFormat defines the RecordReader instance, in Hadoop. By default, by using TextInputFormat ReordReader converts data into key-value pairs. TextInputFormat also provides 2 types of RecordReaders which as follows:

(a) LineRecordReader

It is the default RecordReader. TextInputFormat provides this RecordReader. It also treats each line of the input file as the new value. Then the associated key is byte offset.

(b) SequenceFileRecordReader

This Hadoop RecorReader reads data specified by the header of a sequence file.

16. What is a combiner?

Ans. Combiner is also known as "Mini-Reducer" that summarizes the Mapper output record with the same Key before passing to the Reducer. The primary job of Combiner a "Mini-Reducer is to process the output data from the Mapper, before passing it to Reducer. It runs after the mapper and before the Reducer. Its usage is optional.

17. Advantages of Combiner in MapReduce

- Use of combiner reduces the time taken for data transfer between mapper and reducer.
- Combiner improves the overall performance of the reducer.
- It decreases the amount of data that reducer has to process.

18. Disadvantages of Combiner in MapReduce

- In the local filesystem, when Hadoop stores the key-value pairs and run the combiner later this will cause expensive disk IO.
- MapReduce jobs can't depend on the combiner execution as there is no guarantee in its execution.

19. What is a partitioner?

Ans. Partitioner in MapReduce job execution controls the partitioning of the keys of the intermediate map-outputs. With the help of hash function, key (or a subset of the key) derives the partition. The total number of partitions is equal to the number of reduce tasks.

20. Need of MapReduce Partitioner in Hadoop

Ans. Hadoop Partitioning specifies that all the values for each key are grouped together. It also makes sure that all the values of a single key go to the same reducer. This allows even distribution of the map output over the reducer. Partitioner in a MapReduce job redirects the mapper output to the reducer by determining which reducer handles the particular key.

21. What is the default partitioner used in Hadoop?

Ans. Hash Partitioner is the default Partitioner. It computes a hash value for the key. It also assigns the partition based on this result.

22. How many Partitioner in Hadoop?

Ans. The total number of Partitioner depends on the number of reducers. Hadoop Partitioner divides the data according to the number of reducers. It is set by JobConf.setNumReduceTasks() method. Thus the single reducer processes the data from single partitioner. The important thing to notice is that the framework creates partitioner only when there are many reducers.

23. What is MapReduce Shuffling and Sorting?

Ans. Shuffling is the process by which it transfers mappers intermediate output to the reducer. Reducer gets 1 or more keys and associated values on the basis of reducers. The intermediated key – value generated by mapper is sorted automatically by key. In Sort phase merging and sorting of map output takes place.
Shuffling and Sorting in Hadoop occurs simultaneously.

24. Difference Between InputSplit vs Blocks in Hadoop

Ans. (a) Data Representation

- **Block –** HDFS Block is the physical representation of data in Hadoop.
- **InputSplit –** MapReduce InputSplit is the logical representation of data present in the block in Hadoop. It is basically used during data processing in MapReduce program or other processing techniques. The main thing to focus is that InputSplit doesn't contain actual data; it is just a reference to the data.

(b) Size

- **Block –** By default, the HDFS block size is **128MB** which you can change as per your requirement. All HDFS blocks are the same size except the last block, which can be either the same size or smaller. Hadoop framework break files into 128 MB blocks and then stores into the Hadoop file system.
- **InputSplit –** InputSplit size by default is approximately equal to block size. It is user defined. In MapReduce program the user can control split size based on the size of data.

(c) Example of Block and InputSplit in Hadoop

Suppose we need to store the file in HDFS. Hadoop HDFS stores files as blocks. Block is the smallest unit of data that can be stored or retrieved from the disk. The default size of the block is 128MB. Hadoop HDFS breaks files into blocks. Then it stores these blocks on different nodes in the cluster.

25. Steps of MapReduce Job Execution flow

MapReduce processess the data in various phases with the help of different components.

--Input Files

In input files data for MapReduce job is stored. In HDFS, input files reside. Input files format is arbitrary. Line-based log files and binary format can also be used.

--InputFormat

After that InputFormat defines how to split and read these input files. It selects the files or other objects for input. InputFormat creates InputSplit.

-- InputSplits

It represents the data which will be processed by an individual Mapper. For each split, one map task is created. Thus the number of map tasks is equal to the number of InputSplits. Framework divide split into records, which mapper process.

-- RecordReader

it communicates with the inputSplit. And then converts the data into key-value pairs suitable for reading by the Mapper. RecordReader by default uses TextInputFormat to convert data into a key-value pair. It communicates to the InputSplit until the completion of file reading. It assigns byte offset to each line present in the file. Then, these key-value pairs are further sent to the mapper for further processing.

-- Mapper

It processes input record produced by the RecordReader and generates intermediate key-value pairs. The intermediate output is completely different from the input pair. The output of the mapper is the full collection of key-value pairs. Hadoop framework doesn't store the output of mapper on HDFS. It doesn't store, as data is temporary and writing on HDFS will create unnecessary multiple copies. Then Mapper passes the output to the combiner for further processing.

--Combiner

Combiner is Mini-reducer which performs local aggregation on the mapper's output. It minimizes the data transfer between mapper and reducer. So, when the combiner functionality completes, framework passes the output to the partitioner for further processing.

--Partitioner

Partitioner comes into the existence if we are working with more than one reducer. It takes the output of the combiner and performs partitioning. Partitioning of output takes place on the basis of the key in MapReduce. By hash function, key (or a subset of the key) derives the partition.

On the basis of key value in MapReduce, partitioning of each combiner output takes place. And then the record having the same key value goes into the same partition. After that, each partition is sent to a reducer.

Partitioning in MapReduce execution allows even distribution of the map output over the reducer.

--Shuffling and Sorting

After partitioning, the output is shuffled to the reduce node. The shuffling is the physical movement of the data which is done over the network. As all the mappers finish and shuffle the output on the reducer nodes. Then framework merges this intermediate output and sort. This is then provided as input to reduce phase.

--Reducer

Reducer then takes set of intermediate key-value pairs produced by the mappers as the input. After that runs a reducer function on each of them to generate the output. The output of the reducer is the final output. Then framework stores the output on HDFS.

--RecordWriter

It writes these output key-value pair from the Reducer phase to the output files.

-- OutputFormat

OutputFormat defines the way how RecordReader writes these output key-value pairs in output files. So, its instances provided by the Hadoop write files in HDFS. Thus OutputFormat instances write the final output of reducer on HDFS.