31 FREQUENTLY ASKED INTERVIEW QUESTIONS & ANSWERS FOR HADOOP DEVELOPERS

Q1. Name the most common Input Formats defined in Hadoop? Which one is default?

The two most common Input Formats defined in Hadoop are:

– TextInputFormat

- KeyValueInputF2ormat

- SequenceFileInputFormat

TextInputFormat is the Hadoop default.

Q2. What is the difference between TextInputFormat and KeyValueInputFormat class?

TextInputFormat: It reads lines of text files and provides the offset of the line as key to the Mapper and actual line as Value to the mapper.

KeyValueInputFormat: Reads text file and parses lines into key, Val pairs. Everything up to the first tab character is sent as key to the Mapper and the remainder of the line is sent as value to the mapper.

Q3. What is InputSplit in Hadoop?

When a Hadoop job is run, it splits input files into chunks and assign each split to a mapper to process. This is called InputSplit.

Q4. How is the splitting of file invoked in Hadoop framework?

It is invoked by the Hadoop framework by running getInputSplit()method of the Input format class (like FileInputFormat) defined by the user.

Q5. Consider case scenario: In M/R system, - HDFS block size is 64 MB1

- Input format is FileInputFormat

– We have 3 files of size 64K, 65Mb and 127Mb

How many input splits will be made by Hadoop framework?

Hadoop will make 5 splits as follows:

- 1 split for 64K files

- 2 splits for 65MB files

- 2 splits for 127MB files

Q6. What is the purpose of RecordReader in Hadoop?

The InputSplit has defined a slice of work, but does not describe how to access it. The RecordReader class actually loads the data from its source and converts it into (key, value) pairs suitable for reading by the Mapper. The RecordReader instance is defined by the Input Format.

Q7. After the Map phase finishes, the Hadoop framework does "Partitioning, Shuffle and sort". Explain what happens in this phase?

Partitioning: It is the process of determining which reducer instance will receive which intermediate keys and values. Each mapper must determine for all of its output (key, value) pairs which reducer will receive1 them. It is necessary that for any key, regardless of which mapper instance generated it, the destination partition is the same.

Shuffle: After the first map tasks have completed, the nodes may still be performing several more map tasks each. But they also begin exchanging the intermediate outputs from the map tasks to where they are required by the reducers. This process of moving map outputs to the reducers is known as shuffling.

Sort: Each reduce task is responsible for reducing the values associated with several intermediate keys. The set of intermediate keys on a single node is automatically sorted by Hadoop before they are presented to the Reducer.

Q8. If no custom partitioner is defined in Hadoop then how is data partitioned before it is sent to the reducer?

The default partitioner computes a hash value for the key and assigns the partition based on this result.

Q9. What is a Combiner?

The Combiner is a 'mini-reduce' process which operates only on data generated by a mapper. The Combiner will receive as input all data emitted by the Mapper instances on a given node. The output from the Combiner is then sent to the Reducers, instead of the output from the Mappers.

Q10. What is JobTracker?

JobTracker is the service within Hadoop that runs MapReduce jobs on the cluster.

Q11. What are some typical functions of Job Tracker?

The following are some typical tasks of JobTracker:-

- Accepts jobs from clients

- It talks to the NameNode to determine the location of the data.

- It locates TaskTracker nodes with available slots at or near the data.

- It submits the work to the chosen TaskTracker nodes and monitors progress of each task by receiving heartbeat signals from Task tracker.

Q12. What is TaskTracker?

TaskTracker is a node in the cluster that accepts tasks like MapReduce and Shuffle operations – from a JobTracker.

Q13. What is the relationship between Jobs and Tasks in Hadoop?

One job is broken down into one or many tasks in Hadoop.

Q14. Suppose Hadoop spawned 100 tasks for a job and one of the task failed. What will Hadoop do?

It will restart the task again on some other TaskTracker and only if the task fails more than four (default setting and can be changed) times will it kill the job.

Q15. Hadoop achieves parallelism by dividing the tasks across many nodes, it is possible for a few slow nodes to rate-limit the rest of the program and slow down the program. What mechanism Hadoop provides to combat this?

Speculative Execution.

Q16. How does speculative execution work in Hadoop?

JobTracker makes different TaskTrackers pr2ocess same input. When tasks complete, they announce this fact to the JobTracker. Whichever copy of a task finishes first becomes the definitive copy. If other copies were executing speculatively, Hadoop tells the TaskTrackers to abandon the tasks and discard their outputs. The Reducers then receive their inputs from whichever Mapper completed successfully, first.

Q17. Using command line in Linux, how will you

- **See all jobs running in the Hadoop cluster**

- Kill a job?

Hadoop job – list

Hadoop job – kill jobID

Q18. What is Hadoop Streaming?

Streaming is a generic API that allows programs written in virtually any language to be used as Hadoop Mapper and Reducer implementations.

Q19. What is the characteristic of streaming API that makes it flexible run MapReduce jobs in languages like Perl, Ruby, Awk etc.?

Hadoop Streaming allows to use arbitrary programs for the Mapper and Reducer phases of a MapReduce job by having both Mappers and Reducers receive their input on stdin and emit output (key, value) pairs on stdout.

Q20. What is Distributed Cache in Hadoop?

Distributed Cache is a facility provided by the MapReduce framework to cache files (text, archives, jars and so on) needed by applications during execution of the job. The framework will copy the necessary files to the slave node before any tasks for the job are executed on that node.

Q21. What is the benefit of Distributed cache? Why can we just have the file in HDFS and have the application read it?

This is because distributed cache is much faster. It copies the file to all trackers at the start of the job. Now if the task tracker runs 10 or 100 Mappers or Reducer, it will use the same copy of distributed cache. On the other hand, if you put code in file to read it from HDFS in the MR Job then every Mapper will try to access it from HDFS hence if a TaskTracker run 100 map jobs then it will try to read this file 100 times from HDFS. Also HDFS is not very efficient when used like this.

Q.22 What mechanism does Hadoop framework provide to synchronise changes made in Distribution Cache during runtime of the application?

This is a tricky question. There is no such mechanism. Distributed Cache by design is read only during the time of Job execution.

Q23. Have you ever used Counters in Hadoop. Give us an example scenario?

Anybody who claims to have worked on a Hadoop project is expected to use counters.

Q24. Is it possible to provide multiple input to Hadoop? If yes then how can you give multiple directories as input to the Hadoop job?

Yes, the input format class provides methods to add multiple directories as input to a Hadoop job.

Q25. Is it possible to have Hadoop job output in multiple directories? If yes, how?

Yes, by using Multiple Outputs class.

Q26. What will a Hadoop job do if you try to run it with an output directory that is already present? Will it

- Overwrite it

- Warn you and continue

- Throw an exception and exit

The Hadoop job will throw an exception and exit.

Q27. How can you set an arbitrary number of mappers to be created for a job in Hadoop?

You cannot set it.

Q28. How can you set an arbitrary number of Reducers to be created for a job in Hadoop?

You can either do it programmatically by using method setNumReduceTasks in the Jobconf Class or set it up as a configuration setting.

Q29. How will you write a custom partitioner for a Hadoop job?

To have Hadoop use a custom partitioner you will have to do minimum the following three:

- Create a new class that extends Partitioner Class

- Override method getPartition

- In the wrapper that runs the Mapreduce, either

- Add the custom partitioner to the job programmatically using method set Partitioner Class or – add the custom partitioner to the job as a config file (if your wrapper reads from config file or oozie)

Q30. How did you debug your Hadoop code?

There can be several ways of doing this but most common ways are:-

- By using counters.

- The web interface provided by Hadoop framework.

Q31. Did you ever built a production process in Hadoop? If yes, what was the process when your Hadoop job fails due to any reason?

It is an open-ended question but most candidates if they have written a production job, should talk about some type of alert mechanism like email is sent or there monitoring system sends an alert. Since Hadoop works on unstructured data, it is very important to have a good alerting system for errors since unexpected data can very easily break the job.