Sample Questions
Hadoop Developer Interview Guide

1. How do you debug a performance issue or a long running job?

This is an open ended question and the interviewer is trying to see the level of hands-on experience you have in solving production issues. Use your day to day work experience to answer this question. Here are some of the scenarios and responses to help you construct your answer. On a very high level you will follow the below steps.

- Understand the symptom
- Analyze the situation
- Identify the problem areas
- Propose solution

Scenario 1 - Job with 100 mappers and 1 reducer takes a long time for the reducer to start after all the mappers are complete. One of the reasons could be that reduce is spending a lot of time copying the map outputs. So in this case we can try couple of things.

1. If possible add a combiner to reduce the amount of output from the mapper to be sent to the reducer
2. Enable map output compression - this will further reduce the size of the outputs to be transferred to the reducer.

Scenario 2 - A particular task is using a lot of memory which is causing the slowness or failure, I will look for ways to reduce the memory usage.

1. Make sure the joins are made in an optimal way with memory usage in mind. For e.g. in Pig joins, the LEFT hand side tables are sent to the reducer first and held in memory and the RIGHT most table in streamed to the reducer. So make sure the RIGHT most table is largest of the datasets in the join.
2. We can also increase the memory requirements needed by the map and reduce tasks by setting - *mapreduce.map.memory.mb* and *mapreduce.reduce.memory.mb*

Scenario 3 - Understanding the data helps a lot in optimizing the way we use the datasets in PIG and HIVE scripts.

1. If you have smaller tables in join, they can be sent to distributed cache and loaded in memory on the Map side and the entire join can be done on the Map side thereby avoiding the shuffle and reduce phase altogether. This will tremendously improve performance. Look up *USING REPLICATED* in Pig and *MAPJOIN* or *hive.auto.convert.join* in Hive
2. If the data is already sorted you can use *USING MERGE* which will do a Map Only join
3. If the data is bucketted in hive, you may use *hive.optimize.bucketmapjoin* or *hive.optimize.bucketmapjoin.sortedmerge* depending on the characteristics of the data

Scenario 4 - The Shuffle process is the heart of a MapReduce program and it can be tweaked for performance improvement.

1. If you see lots of records are being spilled to the disk (check for Spilled Records in the counters in your MapReduce output) you can increase the memory available for Map to perform the Shuffle by increasing

the value in *io.sort.mb*. This will reduce the amount of Map Outputs written to the disk so the sorting of the keys can be performed in memory.

2. On the reduce side the merge operation (merging the output from several mappers) can be done in disk by setting the *mapred.inmem.merge.threshold* to 0

2. Assume you have Research, Marketing and Finance teams funding 60%, 30% and 10% respectively of your Hadoop Cluster. How will you assign only 60% of cluster resources to Research, 30% to Marketing and 10% to Finance during peak load?

Capacity scheduler in Hadoop is designed to support this use case. Capacity scheduler supports hierarchical queues and capacity can be defined for each queue.

For this use case, you would have to define 3 queues under the root queue and give appropriate capacity in % for each queue.

Illustration

Below properties will be defined in *capacity-scheduler.xml*

```
<property>
  <name>yarn.scheduler.capacity.root.queues</name>
  <value>research,marketing,finance</value>
</property>

<property>
  <name>yarn.scheduler.capacity.research.capacity</name>
  <value>60</value>
</property>

<property>
  <name>yarn.scheduler.capacity.research.capacity</name>
  <value>30</value>
</property>

<property>
  <name>yarn.scheduler.capacity.research.capacity</name>
  <value>10</value>
</property>
```


3. How do you benchmark your Hadoop cluster with tools that come with Hadoop?

- TestDFSIO

TestDFSIO gives you an understanding of the I/O performance of your cluster. It is a read and write test for HDFS and helpful in identifying performance bottlenecks in your network, hardware and set up of your NameNode and DataNodes.

- NNBench

NNBench simulate requests for creating, reading, renaming and deleting files on HDFS and is useful for load testing NameNode hardware configuration

- MRBench

MRBench is a test for the MapReduce layer. It loops a small MapReduce job for a specific number of times and checks the responsiveness and efficiency of the cluster.

Illustration

TestDFSIO write test with 100 files and file size of 100 MB each.

$ hadoop jar /dirlocation/hadoop-test.jar TestDFSIO -write -nrFiles 100 -fileSize 100

TestDFSIO read test with 100 files and file size of 100 MB each.

$ hadoop jar /dirlocation/hadoop-test.jar TestDFSIO -read -nrFiles 100 -fileSize 100

MRBench test to run a lob of 50 small test jobs

$ hadoop jar /dirlocation/hadoop-test.jar mrbench -numRuns 50

NNBench test that creates 1000 files using 12 maps and 6 reducers.

*$ hadoop jar /dirlocation/hadoop-test.jar nnbench -operation create_write *
*-maps 12 -reduces 6 -blockSize 1 -bytesToWrite 0 -numberOfFiles 1000 *
-replicationFactorPerFile 3

4. Assume you are doing a join and you notice that all but one reducer is running for a long time how do you address the problem in Pig?

Pig collects all of the records for a given key together on a single reducer. In many data sets, there are a few keys that have three or more orders of magnitude more records than other keys. This results in one or two reducers that will take much longer than the rest. To deal with this, Pig provides skew join.

- In the first MapReduce job pig scans the second input and identifies keys that have so many records.
- In the second MapReduce job, it does the actual join.
- For all except the records with the key(s) identified from the first job, pig would do a standard join.
- For the records with keys identified by the second job, bases on how many records were seen for a given key, those records will be split across appropriate number of reducers.
- The other input to the join that is not split, only the keys in question are then then split and then replicated to each reducer that contains that key

Illustration

jnd = join cinfo by city, users by city using 'skewed';

5. What is the difference between SORT BY and ORDER BY in Hive?

ORDER BY performs a total ordering of the query result set. This means that all the data is passed through a single reducer, which may take an unacceptably long time to execute for larger data sets.

SORT BY orders the data only within each reducer, thereby performing a local ordering, where each reducer's output will be sorted. You will not achieve a total ordering on the dataset. Better performance is traded for total ordering.

6. Assume you have a sales table in a company and it has sales entries from salesman around the globe. How do you rank each salesperson by country based on their sales volume in Hive?

Hive support several analytic functions and one of the functions is *RANK*() and it is designed to do this operation.

Lookup details on other window and analytic functions -
https://cwiki.apache.org/confluence/display/Hive/LanguageManual+WindowingAndAnalytics

Illustration

```
Hive>SELECT
        rep_name, rep_country, sales_volume,
        rank() over (PARTITION BY rep_country ORDER BY sales_volume DESC) as rank
    FROM
      salesrep;
```