Hadoop Interview Set-1

Part-A- HDFS

1. How to check block size in HDFS?

Answer - The property to set the block size is present in hdfs-site.xml. The propery name is dfs.blocksize (dfs.block.size was the old property name, this is deprecated).
For checking default block size

>> hdfs getconf -confKey dfs.blocksize

For checking block size of a specific file

>> hdfs fsck /largedeck.txt -files -blocks

2. What are the uses of “fsck”?

Answer - “fsck” or File System check can be used for checking corrupted blocks of a file, block size and overall healthy status of a file in HDFS.

```
[root@localhost prave]# hdfs fsck /largedeck.txt -files -blocks
19/04/10 04:23:07 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java
 classes where applicable
Connecting to namenode via http://localhost:50070/fsck?ugi=root&files=1&blocks=1&path=%2Flargedeck.txt
FSCK started by root (auth:SIMPLE) from /127.0.0.1 for path /largedeck.txt at Wed Apr 10 04:23:09 IST 2019
/largedeck.txt 726663168 bytes, 6 block(s):  Under replicated BP-635524556-127.0.0.1-1489092816187:blk_1073741916_1092. Ta
rget Replicas is 3 but found 1 replica(s).
 Under replicated BP-635524556-127.0.0.1-1489092816187:blk_1073741917_1093. Target Replicas is 3 but found 1 replica(s).
 Under replicated BP-635524556-127.0.0.1-1489092816187:blk_1073741918_1094. Target Replicas is 3 but found 1 replica(s).
 Under replicated BP-635524556-127.0.0.1-1489092816187:blk_1073741919_1095. Target Replicas is 3 but found 1 replica(s).
 Under replicated BP-635524556-127.0.0.1-1489092816187:blk_1073741920_1096. Target Replicas is 3 but found 1 replica(s).
 Under replicated BP-635524556-127.0.0.1-1489092816187:blk_1073741921_1097. Target Replicas is 3 but found 1 replica(s).
0. BP-635524556-127.0.0.1-1489092816187:blk_1073741916_1092 len=134217728 repl=1
1. BP-635524556-127.0.0.1-1489092816187:blk_1073741917_1093 len=134217728 repl=1
2. BP-635524556-127.0.0.1-1489092816187:blk_1073741918_1094 len=134217728 repl=1
3. BP-635524556-127.0.0.1-1489092816187:blk_1073741919_1095 len=134217728 repl=1
4. BP-635524556-127.0.0.1-1489092816187:blk_1073741920_1096 len=134217728 repl=1
5. BP-635524556-127.0.0.1-1489092816187:blk_1073741921_1097 len=55574528 repl=1

Status: HEALTHY
 Total size:    726663168 B
 Total dirs:    0
 Total files:   1
 Total symlinks:                0
 Total blocks (validated):      6 (avg. block size 121110528 B)
 Minimally replicated blocks:   6 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       6 (100.0 %)
 Mis-replicated blocks:         0 (0.0 %)
 Default replication factor:    3
```

3. Can we change block size of a specific file in HDFS?

Answer -
a) Changing block size, while putting a file into HDFS
By setting **“fs.blocksize”** configuration setting appropriately when you use the command line. For example below command puts file into HDFS with block size 64MB while the default size is set to 128 MB.

>> hdfs dfs -D dfs.block.size=67108864 -put /home/prave/sample_input/deckofcards.txt /demo

Other methods of setting configuration in program or hdfs-site.xml block size config property but this changes default block size.

```
Configuration conf = new Configuration() ;
conf.set( "dfs.block.size", 128*1024*1024) ;
```

4. How can files be copied parallelly in HDFS clusters or what is use of distcp in HDFS?
Answer - “distcp” for copying data to and from Hadoop filesystems in parallel.

>>hadoop distcp /demo /demo1

Copy dir1 to dir2

>> hadoop distcp /dir1 /dir2

create if does not exist or just copy unnder /dir2/dir1 tree if exist. Can also be updated, only the files that have changed using the -update option.

>> hadoop distcp -update /dir1 /dir2

Taking backup of the first cluster’s /foo directory on the second: -delete option for deleting unmatched file in destination

>> hadoop distcp -update -delete -p hdfs://namenode1/foo hdfs://namenode2/foo

Use webhdfs protocol if incompatible HDFS

>> hadoop distcp webhdfs://namenode1:50070/foo webhdfs://namenode2:50070/foo

5. What is difference between submitting hadoop job using “hadoop jar” or “yarn jar”?
Answer- The **/usr/bin/yarn** script sets up the execution environment so that all of the yarn commands can be run. The **/usr/bin/hadoop** script isn't quite as concerned about yarn specific functionality. However, if you have your cluster set up to use yarn as the default implementation of mapreduce (MRv2), then **hadoop jar** will probably act the same as **yarn jar** for a mapreduce job.

PART- B HIVE

6. What's the best way to find out duplicate rows in a hive?
Answer-

```
SELECT
    [every column], count(*)
FROM (
    SELECT [every column], *
    FROM table
    DISTRIBUTE BY [every column]
    HAVING count(*) > 1
) t;
```

ORDER BY is an expensive process that uses only one reducer at a time, so DISTRIBUTE BY is used. Also, the regular aggregation SELECT…HAVING statement would also be relegated to utilizing one reducer, which is also computationally-expensive so it had to be an inner HQL/SQL statement in order to maximize multiple reducers. Any column having two or more of the same row will be considered a duplicate.

7. what is the use of Distribute By clause in Hive?

Answer- Hive uses the columns in Distribute by to distribute the rows among reducers. All Distribute BY columns will go to the same reducer.

- It ensures each of N reducers gets non-overlapping ranges of column
- It doesn't sort the output of each reducer.

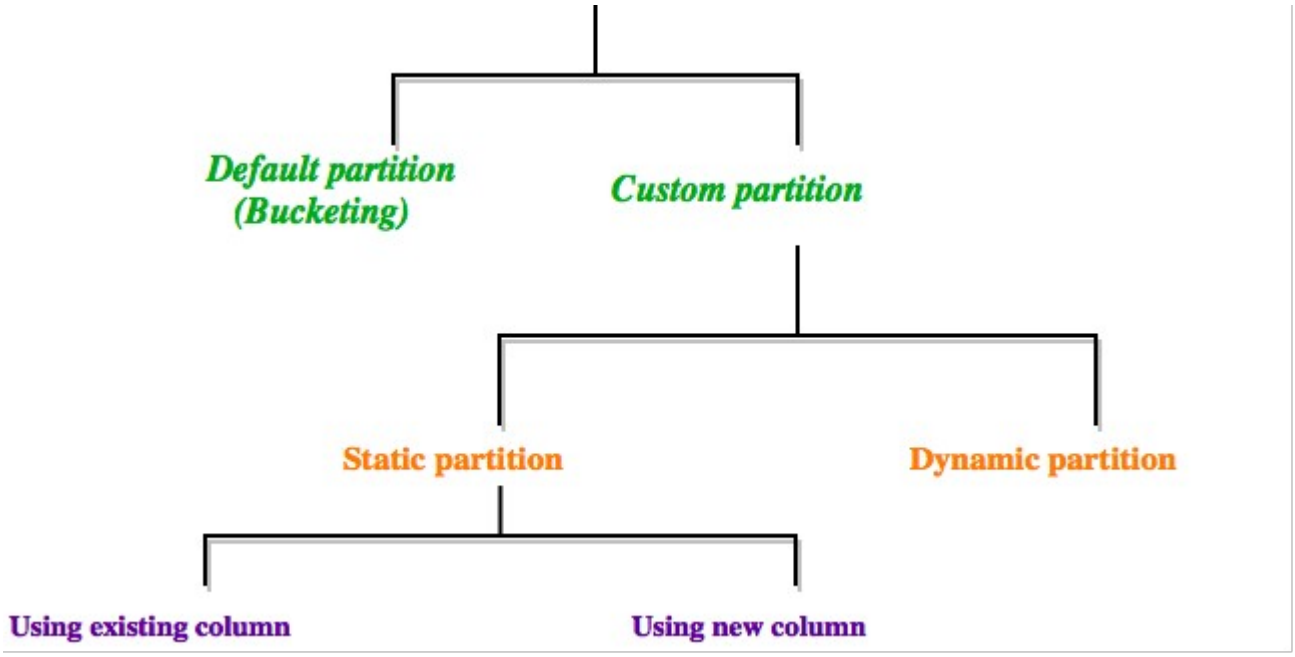
8. How to delete duplicate records in Hive?
Answer- We can use insert overwrite statement to update data
insert overwrite table xyz select distinct * from xyz;

9. Is Bucketing possible without partitioning in Hive?
Answer- Bucketing can also be done even without partitioning on Hive tables.

```
CREATE TABLE bucketed_table(
    firstname VARCHAR(64),
    lastname VARCHAR(64),
    address STRING,
    city VARCHAR(64),
    state VARCHAR(64),
    web STRING
)
CLUSTERED BY (state)
SORTED BY (city)
INTO 32 BUCKETS STORED AS SEQUENCEFILE;
```

10. Can we do First bucketing then partitioning in Hive?
Answer- No

HIVE Partitioning



Hive partition divides the table into a number of partitions and these partitions can be further subdivided into more manageable parts known as Buckets or Clusters. The Bucketing concept is based on Hash function, which depends on the type of the bucketing column. Records which are bucketed by the same column will always be saved in the same bucket.

The Bucketing concept is based on Hash function, which depends on the type of the bucketing column. Records which are bucketed by the same column will always be saved in the same bucket. Here, CLUSTERED BY clause is used to divide the table into buckets. each partition will be created as a directory. But in Hive Buckets, each bucket will be created as a file. Bucketing can also be done even without partitioning on Hive tables.

Bucketed tables allow much more efficient sampling than the non-bucketed tables. Allowing queries on a section of data for testing and debugging purpose when the original data sets are very huge. Here, the user can fix the size of buckets according to the need. This concept also provides the flexibility to keep the records in each bucket to be sorted by one or more columns. Since the data files are equal sized parts, map-side joins will be faster on the bucketed tables.

11. What are the different tables type supported in Hive?
Answer- Hive fundamentally knows two different types of tables:

1. Managed (Internal) table
2. External Table

Managed tables
A managed table is stored under the hive.metastore.warehouse.dir path property, by default in a folder path similar to /user/hive/warehouse/databasename.db/tablename/. The default location can be overridden by the location property during table creation. If a managed table or partition is dropped, the data and metadata associated with that table or partition are deleted. If the PURGE option is not specified, the data is moved to a trash folder for a defined duration.
Use managed tables when Hive should manage the lifecycle of the table, or when generating temporary tables.

External tables
An external table describes the metadata / schema on external files. External table files can be accessed and managed by processes outside of Hive. External tables can access data stored in sources such as Azure Storage Volumes (ASV) or remote HDFS locations. If the structure or partitioning of an external table is changed, an MSCK REPAIR TABLE table_name statement can be used to refresh metadata information.
Use external tables when files are already present or in remote locations, and the files should remain even if the table is dropped.

Comparision

- ARCHIVE/UNARCHIVE/TRUNCATE/MERGE/CONCATENATE only work for managed tables
- DROP deletes data for managed tables while it only deletes metadata for external one
- ACID/Transactional only works for managed tables
- Query Results Caching only works for managed tables
- Only the RELY constraint is allowed on external tables
- Some Materialized View features only work on managed tables

12. What are dynamic partition and static partition in Hive?
Answer-
Partitions are created when data is inserted into table. Depending on how you load data you would need partitions. Usually when loading files (big files) into Hive tables static partitions are preferred. That saves your time in loading data compared to dynamic partition. You "statically" add a partition in table and move the file into the partition of the table. Since the files are big they are usually generated in HDFS. You can get the partition column value form the filename, day of date etc without reading the whole big file.

Incase of dynamic partition whole big file i.e. every row of the data is read and data is partitioned through a MR job into the destination tables depending on certain field in file. So usually dynamic partition are useful when you are doing sort of a ETL flow in your data pipeline. e.g. you load a huge file through a move command into a Table X. then you run a inert query into a Table Y and partition data based on field in table X say day , country. You may want to further run a ETL step to partition the data in country partition in Table Y into a Table Z where data is partitioned based on cities for a particular country only. etc.

Thus depending on your end table or requirements for data and in what form data is produced at source you may choose static or dynamic partition.

PART- C Sqoop

13. Incremental Import in Sqoop?
Answer- Sqoop provides an incremental import mode which can be used to retrieve only rows newer than some previously-imported set of rows.

The following arguments control incremental imports:

Table 4. Incremental import arguments:

Argument	Description
--check-column (col)	Specifies the column to be examined when determining which rows to import.
--incremental (mode)	Specifies how Sqoop determines which rows are new. Legal values for mode include append and lastmodified.
--last-value (value)	Specifies the maximum value of the check column from the previous import.

Sqoop supports two types of incremental imports: append and lastmodified. You can use the --incremental argument to specify the type of incremental import to perform.

You should specify append mode when importing a table where new rows are continually being added with increasing row id values. You specify the column containing the row’s id with --check-column. Sqoop imports rows where the check column has a value greater than the one specified with --last-value.

An alternate table update strategy supported by Sqoop is called lastmodified mode. You should use this when rows of the source table may be updated, and each such update will set the value of a last-modified column to the current timestamp. Rows where the check column holds a timestamp more

recent than the timestamp specified with --last-value are imported.

At the end of an incremental import, the value which should be specified as --last-value for a subsequent import is printed to the screen. When running a subsequent import, you should specify --last-value in this way to ensure you import only the new or updated data. This is handled automatically by creating an incremental import as a saved job, which is the preferred mechanism for performing a recurring incremental import. See the section on saved jobs later in this document for more information.