BEGINNER
****

1. Difference between Linux file system and Hadoop distributed file system
ANS:
Linux file system
****
a. you can store the Linux file under a single disk
b. you can store the small file in a disk , if the file size is more than the disk size then you cannot store.
C. each block size is 4 kb
d. If the machine is down, we cannot able to get the data and failover chances are more
Hadoop Distributed file system
****
a. Distributed file system can store in a logical layer which is creating on one or more disk.
b. you can store large file as larger as you can, you need to add more disk to the logical layer
c. Each block size is 64mb/128mb/25mb as per the Hadoop version and you can customize the size too.
d. Data is replicated in different nodes. Clients able to read the data, if any node is failed. Failover is less

2. As per the configuration HDFS is in High availability mode with automatic failover. Brief about the daemon which will take care of the failover.

ANS:
High Availability of cluster was introduced in Hadoop 2 to solve the single point of Name node failure problem in Hadoop 1.
The High availability Name node architecture is providing an opportunity to have two name nodes as Active name node and Passive/Standby name node. So, both are running Name Nodes at the same time in a High Availability cluster.
Whenever Active Name Node goes down due to crashes of server or graceful failover during the maintenance period at the same time control will go to passive/Standby Name Node automatically and it reduce the cluster down time. There are two problems in maintaining consistency in the HDFS High Availability cluster:
a. Active and Passive/Standby Name Node should be in sync always because they are referring to the same metadata, to doing the same group of daemons called journal nodes will help .This will allow to restore the Hadoop cluster to the same namespace state whenever it got crashed or failed and it will provide us to have fast failover.
b. One name node should be active at a time because two active Name Node will cause to loss or corruption of the data. This kind of scenario is known as a split-brain scenario where a cluster gets divided into smaller cluster and each one believing that it is the only active cluster. Fencing is help to avoid such scenarios. Fencing is a process where it ensures that only one Name Node remains active at a particular time it means whenever two Name Node will be in Active state fencing will kill one of the Name node which is in active state.
As discussed above, there are two types of failover:

A. Graceful Failover: In this case, we manually initiate the failover for routine maintenance.
B. Automatic Failover: In this case, the failover is initiated automatically in case of Name Node failure or Name node crashed.

In either case of Name Node failure, Passive or Standby Name Node can take a control of exclusive lock in Zookeeper and showing as it wants to become the next Active Name Node.
In HDFS High availability cluster, Apache Zookeeper is a service which provides the automatic failover. When Name Node is active at that time Zookeeper maintains a session with the active Name Node. In any scenario when active Name Node get failed at that time the session will expire and the Zookeeper will inform to Passive or Stand by Name Node to initiate the failover process.
The Zookeeper Failover Controller (ZKFC) is a Zookeeper client that also monitors and manages the Name Node status. Each of the Name Node runs a ZKFC also. ZKFC is responsible for monitoring the health of the Name Nodes periodically.
When zookeeper is installed in your cluster you should make sure that below are the process or daemons running in Active Name Node, Standby Name Node and Data node.
When you do JPS (Java Virtual Machine Process Status Tool) in Active Name Node you should get below Daemons:

- Zookeeper
- Zookeeper Failover controller
- Journal Node
- Name Node

When you do JPS (Java Virtual Machine Process Status Tool) in Standby Name Node you should get below Daemons:

- Zookeeper
- Zookeeper Failover controller
- Journal Node
- Name Node

When you do JPS (Java Virtual Machine Process Status Tool) in Data Node you should get below Daemons:

- Zookeeper
- Journal Node
- Data Node

3. What is distributed cache in Hadoop?
ANS:
It is a facility provided by Hadoop map reduce framework to access small file needed by application during it execution. These files are small as it is in KB's and MB's in size. Type of files mainly text, archive or jar files, these files are small that is why it will keep in the cache memory which is one of the fast memories. Application which needs to use distributed cache to distribute a file should make sure that the file is available and can be accessed via URLS. URLs can be either hdfs:// or http://
Once the file is present on the mentioned URL, the Map-Reduce framework will copy the necessary files on all the nodes before initiation of the tasks on those nodes. In case the files provided are archives, these will be automatically un-archived on the nodes after transfer.
Example: In Hadoop cluster we have three data nodes there are 30 tasks we run in the cluster. So each node will get 10 tasks each .our nature of task is such kind of task where it needs some information or particular jar to be adopted before its execution, to fulfill this we can cache this files which contains the info or jar files. Before execution of the job the cache files will copy to each slave node application master. Application master than reads the files and start the tasks. The task can be mapper or reducer and these are read only files. By default Hadoop distributed cache is 10gb ,if you want to change the same you have to modify the size in mapred-site.xml. Here it's

coming to our mind that why cache memory is required to perform the tasks. why can't we keep the file in HDFS on each data node already present and have the application read it .they are total 30 tasks and in real time it should be more than 100 or 1000 tasks .If we put the files in HDFS than to perform 30 tasks the application has to access the HDFS location 30 times and then read it but HDFS is not very efficient to access small files for this many times. this is the reason why we are using cache memory and it reduces the number of reads from HDFS locations .

4. Explain how the Name node get to know all the available data node in the hadoop cluster.
ANS:

In Hadoop cluster when we are talking about Data node, Data node is where the actual data we are keeping. Data nodes are sending a heartbeat message to the name node in every 3 seconds to confirm that they are active. If the Name Node does not receive a heart beat from a particular data node for 10 minutes, then it considers that data node to be dead then Name Node initiate the replication of Dead data node blocks to some other data nodes which are active. Data nodes can talk to each other to rebalance the data, move and copy the data around and keep the replication active in the cluster. You can get the block report using below HDFS commands.

Example:

hadoop fsck / ==> File system check on HDFS

```
# hadoop fsck /hadoop/container/pbibhu
 Total size:     16666944775310 B     <=== see here
 Total dirs:     3922
 Total files:    418464
 Total blocks (validated):       202705 (avg. block size 32953610 B)
 Minimally replicated blocks:    202705 (100.0 %)
 Over-replicated blocks:         0 (0.0 %)
 Under-replicated blocks:        0 (0.0 %)
 Mis-replicated blocks:          0 (0.0 %)
 Default replication factor:     3
 Average block replication:      3.0
 Corrupt blocks:                 0
 Missing replicas:               0 (0.0 %)
 Number of data-nodes:           18
 Number of racks:                1
FSCK ended at Thu Oct 20 20:49:59 CET 2011 in 7516 milliseconds

The file system under path '/hadoop/container/pbibhu 'is HEALTHY
```

Name node is the node which stores the file system metadata, when we are talking about metadata, it is having information like List of file names, Owner, Permissions, Timestamps, Size, Replication Factor, List of Blocks for each file etc. Metadata, which files maps to what block location and which blocks are stored in which data node. When data nodes are storing a block of information, it maintains a checksum for each block as well. When any data has been written to HDFS, checksum value has been written simultaneously and when it reads by default verifies the same checksum value. The data nodes update the name node with the block information periodically and before updating verify the checksum value. when the checksum value is not correct for a particular block then we will consider as disk level corruption for that particular block, it skips that block information while reporting to the name node, in this way name node will

get to know that disk level corruption on that data node and takes necessary steps like it can be replicated from its alternate locations to other active data nodes to bring the replication factor back to the normal level. Data nodes can be listed in DFS.HOSTS file, It contains a list of hosts that are permitted to connect to the Name Node.
Example:
Add this property to hdfs-site.xml:

```
<property>
  <name>dfs.hosts</name>
  <value>/home/hadoop/includes</value>
</property>
```

includes:
host name1
hostname2
hostname3
If include file is empty then all hosts are permitted but it is not a definitive list of active data nodes. Name node will consider those data nodes from which Name Node will receive the heart beats.

5. Why we cannot use LVM in a Hadoop cluster.

ANS:
LVM stands for Logical Volume Management. It is a system of managing logical volumes, or file systems, that is much more advanced and flexible than the traditional method of partitioning a disk into one or more segments and formatting that partition with a file system. Today the disks are huge (> 1TB) and LVM is the right tools to dynamically allocate and resize portions of these huge disks.
If you are using Linux to deploy Hadoop nodes, master or slaves, it is strongly recommended that you should not use LVM in Linux because of below points

A. Attempts to reduce the size of a logical volume or reallocate storage space used by it to another volume commonly result in corruption and data loss.
B. The loss or removal of a disk from LVM will leave the entire storage together with all the files inaccessible.
C. Other operating systems do not recognize LVM, therefore, LVM partitions cannot be accessed from Windows or Mac OS.

6. What are the steps when you run the YARN job by calling submit Application () method
ANS:

a. If any job submitted by client it will come to resource manager and resource manager is having scheduler .It's a duty of resource manager to see and calculate the require resources to run the job.
b. Once it is calculated the amount of resources whatever required, after that the resource manager launch an application specific application master.
c. Application master daemon will be available till the job got completed.
d. Application master responsible is to negotiate the resources from resource manager it means application master will ask for more or less resources based on the requirement to resource manager.
e. Application master launches the container in different node manager, where data will be available

f. Container will be working under surveillance of node manager, so node manager is responsible for all the container available in that node. Container will give periodically updates to the application master about the job.
g. Once the job was completed and the container/resources are freed up, than application master update to the resource manager about the completion of job and client will get the update from resource manager.

7. Why should we run the HDFS balancer periodically? Brief about the same.

ANS:

HDFS data might not always be distributed uniformly across Data Nodes for different reasons like if some Data Nodes have less disk space available for use by HDFS or During the normal usage/ when usage is more, the disk utilization on the Data Node machines may become uneven or when a new Data Nodes are added to an existing cluster at that time also data nodes utilizations are uneven. To mitigate this problem balancer is required.

The balancer is a tool that balances disk space usage on an HDFS cluster and it analyzes block placement and balances data across the Data Nodes. The balancer moves blocks until the cluster is deemed to be balanced, which means that the utilization of every Data Node more or less equally distributed. The balancer does not balance between individual volumes on a single Data Node.

hdfs balancer [-policy <policy>]

The two supported policies are blockpool and datanode. Setting the policy to blockpool means that the cluster is balanced if each pool in each node is balanced while Data node means that a cluster is balanced if each Data Node is balanced. The default policy is Data node.

hdfs balancer [-threshold <threshold>] specifies a number in [1.0, 100.0] representing the acceptable threshold of the percentage of storage capacity so that storage utilization outside the average +/- the threshold is considered as over/under utilized. The default threshold is 10.0.

8. Brief about Hadoop rack topology.

ANS:

When we are talking about Rack, It is the collection of multiple server based on your requirement. All these servers are connected using the same network switch and if that network goes down then all machines in that rack will be out of service and we can say rack is down state.

To mitigate the same, Rack Awareness was introduced for Hadoop by Apache. In Rack Awareness, Name Node chooses the Data Node which is closer to the rack where Name Node will be available or nearby that rack. Name Node maintains all the Rack ids of each Data Node to get the rack information and based on Rack ID Name Node can communicate with Data Node. In Hadoop when we are maintaining a Rack we have to follow certain rules as mentioned below.

- All the replicas should not stored on the same rack or in a single rack due to which Rack Awareness Algorithm can reduces the latency as well as Fault Tolerance.
- By Default replication factor is 3 so according to Rack Awareness Algorithm below are the points to be followed:
  a. The first replica of the block will be stored on a local rack.

b. The next replica will be stored another Data Node within the same rack.
c. The third replica stored on the different rack other than earlier Rack.

Below are some points due to which we are following Rack Awareness in Hadoop. Please find the details as mentioned below.

- To improve the data high availability and consistency as same block will be available in different Racks.

- The performance of the cluster will be improved as reading and writing in the cluster will be quick because two of data nodes will be available in the same rack and third data node will be available near to earlier rack.
- Network bandwidth will be improved for sure because of rack awareness rule. Especially with rack awareness, YARN is able to optimize the Map reduce job performance because YARN will assign the task to data nodes that are closer to each other based on Rack policy and where replica will be available to do the process.
- As per the Rack policy, the Name Node assigns 2nd & 3rd replicas of a block to Data Nodes in a rack different from Data Node where the first replica is available. It will provides data protection even against Rack failure; It is possible only if Hadoop was configured with rack awareness.

9. What are the Table type available in Hive?

ANS:

There are two types of tables which HIVE supports.

- Managed Table
- External Table

Hive Managed Tables:

Hive Managed Table is also known as an internal table. When we will create a table in Hive, by default Managed table will create and it manages the data as well. It means that Hive is storing the data into its warehouse directory. A managed table is stored under the hive.metastore.warehouse.dir path property and default location of table will be /apps/hive/warehouse/<db_name>.db/<table_name>. This path will be modifiable. If a managed table or partition is dropped, then data and corresponding metadata of the table or partition are deleted. If you do not specify the PURGE option then the data is moved to a trash folder for certain period, it will be deleted permanently after that.

Example:

a. Create Table

```
hive> create table univercity_db.school_table(name string, roll no int) row format delimited fields terminated by ',';
OK
Time taken: 0.202 seconds
```

b. Describe table

```
hive> describe formatted univercity_db.school_table;
OK
```

you will get extra information like whether the table is managed or external table. when the table is created, what kind of file format, Location of the data path in HDFS, whether the object is a table or view.

c. Load the data to table from the local path

```
hive>load data local in path '/home/pbibhu/Desktop/blog/school' into table univercity_db.school_table;
```

After loading from local path you can further use hive commands to select/count/describe etc

Hive External Tables:

while creating an External table the location of the data path is not the usual warehouse path, you have to provide the HDFS path outside of the warehouse directory, While Creating an external

table location is mandatory in the create syntax. By any chance structure or partitioning of an external table is changed then an MSCK REPAIR TABLE table_name statement can be used to refresh metadata information. MSCK stands for Metastore consistency check. Basically In External Table we cannot load the table from local path. You have to load data from HDFS mentioning the path.
Use external tables when files are present in the remote locations, and the files should remain even if the external table is dropped.
Example:
a. Create Table
CREATE EXTERNAL TABLE IF NOT EXISTS univercity_db.school_table(student_ID INT, FirstName STRING, Last Name STRING) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE/ORC LOCATION 'hdfs/pbibhu/school';
b. Create partition Table
CREATE EXTERNAL TABLE IF NOT EXISTS univercity_db.school_table (student_ID INT, FirstName STRING, LastName STRING) partitioned by (student_ID int) STORED AS ORC LOCATION 'hdfs/pbibhu/school';
C. insert the data to internal table from external table, data structure should be same for both the tables.

```
hive> CREATE TABLE IF NOT EXISTS office(EmployeeID INT,FirstName STRING, Title STRING,
    State STRING, Laptop STRING) STORED AS ORC;
OK
hive> CREATE EXTERNAL TABLE IF NOT EXISTS Office_text(
   EmployeeID INT,FirstName STRING, Title STRING,
   State STRING, Laptop STRING)
   ROW FORMAT DELIMITED
   FIELDS TERMINATED BY ','
   STORED AS TEXTFILE
   LOCATION '/user/pbibhu/office';
OK
hive> INSERT OVERWRITE TABLE office SELECT * FROM office_text;
```

10. What is the problem in having lots of small files in HDFS? What is the remediation plan?
ANS:
Basically we are storing files under some folders in HDFS, most of the time the folder name we are giving based on Application Name .when we are talking about small files it should be lesser then the block size, for example if the block size is 64mb or 128mb then smaller files are considered lesser then the block size. When files are smaller than the block size at that time we are facing problem in HDFS level as well as Map Reduce Level.
In HDFS when we are storing files/Directories, corresponding metadata will be stored in Name Node, each file, directory, block metadata information will approximately occupy 150 bytes. Suppose if you have 1 million files and each file are using approximately a block size or lesser then the block size then metadata size of the corresponding files/directories are approximately 300MB of memory, In such case lot of memory is occupied in the name node and after some time threshold will be reached and further it will be a problem with the current hardware. Certainly performance will be a downgrade.

During the execution of Map reduce, when the file size is lesser than or equivalent to the block size, for each block size or equivalent split size one Mapper will launch so approximately large number of Mapper will launch for large number of small files in this case processing time will be more for each file having small chunk of data .when we are reading and writing large number of small files seek time will be more which will impact performance and seeks are generally expensive operation. Since Hadoop is designed in such a way to run over your entire dataset, it is best to minimize seeks by using large files.
Remediation plan:
We can merge all the small files using HDFS getmerge command into a big file. Getmerge command can copy all the files available in HDFS folder to a single concatenated file in the local system. After concatenated in the local system you can place the same file from local to HDFS using HDFS PUT command. Please find the example as mentioned below.
hadoop fs -getmerge /hdfs_path/pbibhu/school_info_* /local_path/pbibhu/school_inf.txt
hadoop fs -put school_inf.txt /hdfs_path/pbibhu/school_inf.txt

11. What are the file format supports HADOOP. Brief about the same.
ANS:
Below are the file format which support Hadoop.

a. Text (Ex: CSV (Comma separated file) and TSV (Tab separated file))
b. JSON (JavaScript object notation)
c. AVRO
d. RC (Record Columnar)
e. ORC (Optimized Record Columnar)
f. Parquet.

a. Text file format (Ex: CSV (Comma separated values) and TSV (Tab separated values))
Usually text format was very common prior to Hadoop and even it is very common in Hadoop environment as well. Data's are presenting as lines and each line terminated by a new line character as /n or Tab separated as /t.
CSV stands for comma separated values, so data fields are separated or delimited by comma. For example we have below value in excel sheet

Name	class	section	subject
Bibhu	7	A	English

The above data will be present in a CSV formatted file as follows.
Bibhu,7,A,English
b. JSON
JSON stands for JavaScript object Notion. It is a readable format for structuring data, basically it is used to transfer the data from server to web Application. We can use as an alternative to XML .In JSON data are presenting as key and value pairs. Key is always a string data type which is enclosed with quotation marks. Value can be a string, Number, Boolean, Array or object.
The basic syntax is Key followed by a colon followed by a value.
Example: "Name": "Bibhu"
c. AVRO
AVRO stores the data in JSON format which is easy to read and understand. The Data itself stored in Binary format which is making it compressed and Efficient, Each value is stored without having any meta data other than a small schema identifier having size of 1 to 4 bytes. It is having

capability to split the large data set into subsets which are very much suitable for Map Reduce processing.
In Hive following command is used to use AVRO.
Create table avro_school
(column_address)
stored as avro;

d. RC
RC stands for Record Columnar which is one type of Binary file format, it will provide high compression on the top of rows or on multiple rows at a time for which we want to do some operation. RC Files consisting of Binary Key/Value pairs. RC File format first partition the rows horizontally into Row split and after that all the row split presented vertically in a columnar way. Please find the example as mentioned below

Step 1
First partition the rows horizontally into Row split

501	502	503	504
505	506	507	508
509	510	511	512
513	514	515	516

Step 2
All the row split presented vertically in a columnar way

501	505	509	513
502	506	510	514
503	507	511	515
504	508	512	516

RC file combines multiple functions such as data storage formatting, data compression and data access optimization. It is able to meet all the four below requirement of data storage.
1. Fast data storing 2. Improved query processing 3. Optimized storage space utilization 4. Dynamic data access patterns.
e. ORC (Optimized Record Columnar)
The ORC File provide more efficient way to store the Relational Data then RC file. It is basically reducing the data storage format by up to 75% of the original. As compared to RC file ORC file takes less time to access the data and take less space to store the data as well, it internally divides the data again with a default size of 250M.
In Hive following command is used to use ORC file.
CREATE TABLE ...STORED AS ORC
ALTER TABLE ... SET FILEFORMAT ORC
SET hive.default.fileformat=ORC
F. Parquet
It's another column oriented storage like RC format and ORC format but especially it’s very good at handling nested data as well as good at query scan for a particular column in a table. In Parquet New column can be added at the end of the structure. It is handling the compression using Snappy, ggip currently snappy is default. Parquet is supported by cloudera and optimized for cloudera impala.

Hive Parquet File Format Example:
Create table parquet_school_table
(column_specs)
stored as parquet;

12. Can we use both Fair scheduler and Capacity Scheduler in the same Hadoop cluster, Brief about the same?

ANS:

Both the scheduler cannot be used in the same cluster .Both the scheduling algorithms have come up due to specific use-cases and cluster wise you have to set up the configuration file for either Fair scheduler or Capacity scheduler. You cannot set up both the scheduler for one cluster.

You can choose the Fair Scheduler using below scheduler class in yarn-site.xml as mentioned below:

```
<property>
  <name>yarn.resourcemanager.scheduler.class</name>
  <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler</value>
</property>
```

To use the Capacity Scheduler you have to configure the Resource Manager in the conf/yarn-site.xml as mentioned below:

yarn.resourcemanager.scheduler.class-
org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.CapacityScheduler while setting up the queues in Capacity Scheduler you need to make some changes in etc/hadoop/capacity-scheduler.xml configuration file.

The Capacity Scheduler has a predefined queue called root. Whatever queues we will create in the system are children of the root queue.

Setting up further queues==> Configure property yarn.scheduler.capacity.root.queues with a list of comma-separated child queues.

Setting up sub-queues with in a queue==> configure property yarn.scheduler.capacity.<queue-path>.queues

queue-path can mention the full path of the queue's hierarchy and it is starting at root with. (Dot) as the delimiter.

Queue capacity is provided in percentage (%). The sum of capacities for all queues, at each queue level, must be equal to 100. If there are free resources in the queue then Applications in the queue may consume the required resources.

Capacity scheduler queue configuration example:

If there are two child queues starting from root XYZ and ABC. XYZ further divides the queue into technology and development. XYZ is given 60% of the cluster capacity and ABC is given 40% in this scenario please find the details as mentioned below to set up your yarn-site.xml.

```
<property>
  <name>yarn.scheduler.capacity.root.queues</name>
  <value>XYZ, ABC</value>
</property>
<property>
  <name>yarn.scheduler.capacity.root.XYZ.queues</name>
  <value>technology,marketing</value>
</property>
```

```
<property>
  <name>yarn.scheduler.capacity.root.XYZ.capacity</name>
  <value>60</value>
</property>
<property>
  <name>yarn.scheduler.capacity.root.ABC.capacity</name>
  <value>40</value>
</property>
```

13. What is Kafka, what are the components available in Kafka? What is the role of Zookeeper in Kafka and what is Sequence of starting the Kafka services.

ANS:

Basically It is massaging system which is exchanging the large volume of Streaming/log data in between processes, Application and servers. Distributed messaging is based on the queue which can handle a high volume of data and allow you to pass the messages from one end to another. Kafka is appropriate for both offline and online massage consumption.

Prior to talk about Kafka further, need to know about the components belongs to Kafka and below are the details.

1. Kafka Broker 2. Kafka Topics 3.Kafka Topic Partition 4. Kafka producers 5. Kafka consumer 6. Consumer Group

Kafka Broker: Kafka cluster consists of one or more server that is called Kafka broker in which Kafka is running. Producers are nothing but processes that distribute data into Kafka topics within the brokers, then consumer of topics drag the massages off from the Kafka topics.

Few basic points related to Kafka Broker:

a. Each broker has an identification number I mean to say it's an integer number.

b. Each broker contains some topic partition and multiple partition of same topics too.

c. Producer or consumer can connect to any broker

Kafka Topics: A Topic is nothing but category or feed name to which messages are stored and distributed. All Kafka massages are prepared into topics. So whenever you want to send a message you can send it to specific Topic and whenever you want to read the messages you can read it from a specific topic.

Kafka Topic Partition: Kafka topics are divided into number of partitions and it contains the messages in a sequence, sequence is only applicable within a partition. Each massage in partition is recognized by its offset value. Here offset is represented as an incremental ID which is maintained by Zookeeper. The offsets are meaningful for that partition, it does not have any value across the partition. A topic may contain any number of partition. Basically there is no such rule and regulation for write the available messages to which partition. However, there is an option available to adding a key to a massage. If a producer distribute the massages with a Key then all the messages with the same key will go to same partition.

Kafka producers: Basically producers are writing data to a topic, while writing data, producers need to specify the Topic name and one broker name to connect to. Kafka is having own mechanism to send the data to the right partition of the right broker automatically.

Producers having the Mechanism where producer can receive an acknowledgement of data it writes. Below are the acknowledgement which producer receives.

acks = 0 ==> It means successful, in this case producer does not wait for any acknowledgements.

acks = 1 ==> In this case producer will wait for leader acknowledgement and it will make sure at least one broker has got the message. There is no surety whether data has made it to replica or not.
acks= all ==> In this case the leader as well as replica has to acknowledge back . Performance will be impact.

Kafka Consumer: Basically consumer reads data from topics. As we know Topics are divided into multiple partitions so consumer reads data from each partition of topic. Consumer need to mention the topic name as well as broker. Consumer read data from a partition in sequence. When consumer connects a broker Kafka will make sure that it connected to the entire cluster.

Kafka Consumer Group: Consumer group consist of multiple consumer process. One consumer group having one unique group Id. One consumer instance in one consumer group will read data from one partition. If the number of consumer exceeds the number of partition then in this case extra number of consumer will be inactive. For example, there are 6 partitions in total and there are 8 consumers in a single consumer groups. In this case, there will be 2 inactive consumers.

Here in Kafka, there are two types of massaging patterns are available as below:

1. Point to point messaging system
2. Publish subscribe messaging system.

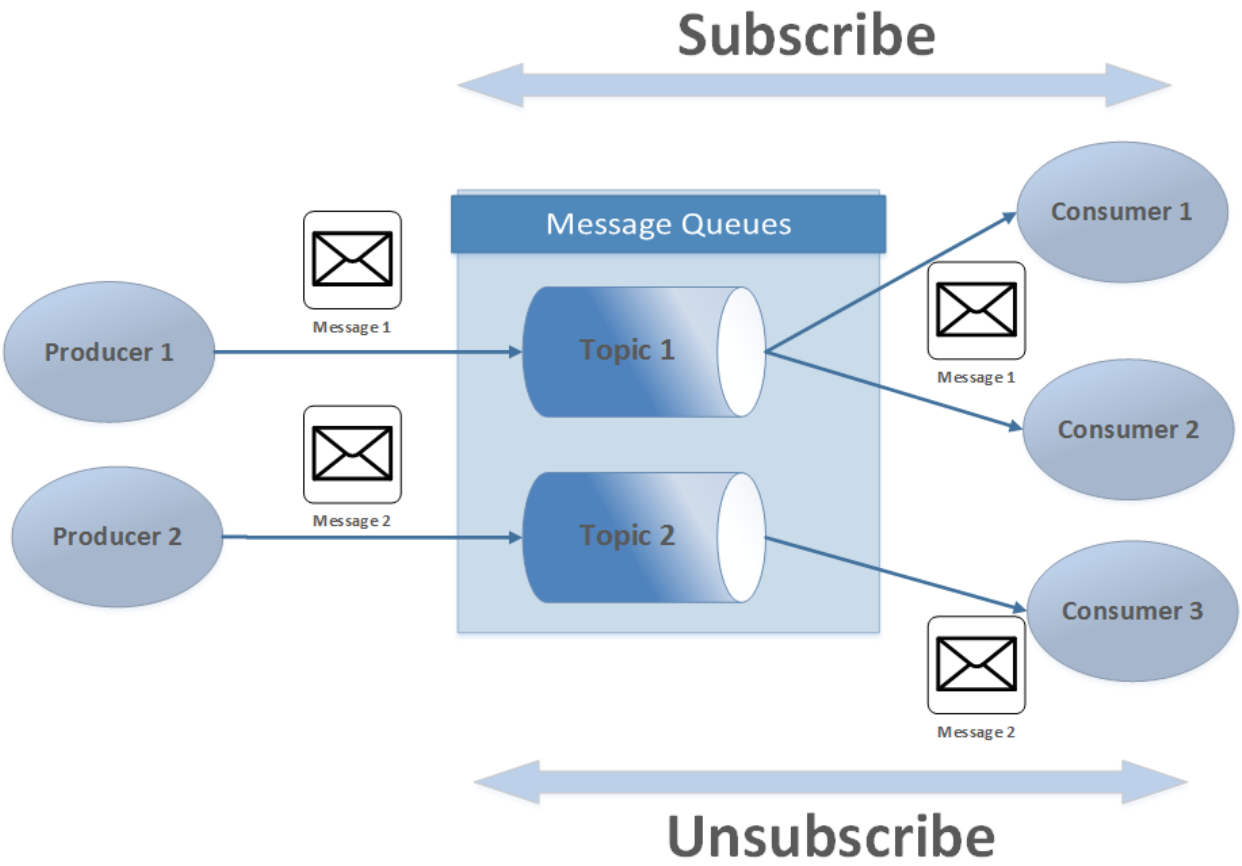
1. **Point to point messaging system**:

In point to point messaging system, Messages are keeping on the queue. One or more consumer read the message in the queue but a particular message can be read by one consumer at a time.
Basically Point-to-point messaging is used when a single message will be received by only one message consumer. There may be multiple consumers reading on the queue for the same message but only one of the consumers will receive it. There can be multiple producers as well. They will be sending messages to the queue but it will be received by only one receiver.

2. **Publish subscribe messaging system:**

Here in Publish subscribe messaging system, message producers are called publishers and message consumers are called subscribers. Here in this scenario Topic can have multiple receiver

and each and every receiver receives a copy of each message.



Based on above picture below are a few points which I am trying to brief about the publish subscribe messaging system.
Massages are shared through channel and it is called as Topic, Topics are placed in a centralized place where producer can distribute and consumer can read the messages.
Each message is delivered to one or more than one consumer and it is called subscribers. The publisher or producer is not aware of which massage or topic is receiving by which consumer or subscriber. A single message created by one publisher, It may be copied and distributed to hundreds or thousands of subscribers.

Role of Zookeeper in Kafka:

Zookeeper is a mandatory component in Kafka ecosystem, it helps in managing Kafka brokers and helps in leader election of partitions. It helps in maintaining the cluster membership. For example, when a new broker is added or a broker is removed and a new topic is added or a topic is deleted, when a broker goes down or comes up etc, Zookeeper manages such situations informing Kafka. It also handles the topic configurations like number of partitions a topic has and leader of the partitions for a topic.

Sequence of starting the Kafka services:

a. zookeeper configuration file provided by Kafka, This is default zookeeper configuration file available in Kafka, for which below are the properties
dataDir=/tmp/zookeeper

Client Port= 2183

[root@xxxx]# /bin/zookeeper-server-start.sh /config/zookeeper.properties

b. After Zookeeper next will be Kafka broker. You can start the Kafka broker with the default configuration file. Below are the configuration properties
broker.id=0
log.dir=/tmp/Kafka-logs
zookeeper. connect=localhost:2183

Here one broker whose ID is 0 and its connecting the zookeeper using port as 2183.
[root@xxxx]# /bin/kafka-server-start.sh /config/server. Properties

c. Both zookeeper and Broker has been started now it's time to create the Topic, Below is an example to create a topic with a single partition and replica
[root@xxxx]#/bin/kafka-create-topic.sh -zookeeper localhost:2183 -replica 1 -partition 1 -topic examtopic

Here in the above example we created a topic as examtopic.

d.Topic has been created now we need to start a producer to send messages to the Kafka cluster. To start the producer below are the commands.
[root@xxxx]#/bin/kafka-console-producer.sh -broker-list localhost:9090 -topic examtopic

broker-list ==> this is the server and port information for the brokers ,here in the above example we have provided server as localhost and port as 9090
Topic==> Name of the Topic here in the above example we have provided as examtopic

in command line we created the producer client that accepts your massages and distribute it to cluster as massages then consumer can consume or read the messages .
Producer client is running you can type something on terminal where producer is running
Hi Bibhu, How are you

E. we have to start the consumer to read or consume the data sending by producer
[root@xxxx]#/bin/kafka-console-consumer.sh -zookeeper localhost:2183 -topic examtopic -from-beginning
Consumer runs with the default configuration properties as mentioned below , these information will be there in consumer. Properties file.
groupid=test-consumer-group
zookeeper.connect=localhost:2183

14. In what scenario Sqoop can use and what are the features. How Sqoop works to Move Data into Hive and HDFS.

ANS:
Basically SQOOP can use to get the Data from Relational database that is DB2,MYSQL,Oracle etc and load into Hadoop that is HDFS, Hive, Hbase etc. or vice versa this process is called ETL for Extract, Transform and Load. Alternatively SQOOP can import and export data from Relational database to Hadoop.
Below are some of the important features which are Sqoop having.
a. ***Full Load***: Sqoop can load the single table or all the tables in a database using Sqoop command.
b. ***Incremental Load***: Sqoop can do incremental load, it means it will retrieve only rows newer than some previously-imported set of rows.
c. ***Parallel import/export***: Sqoop is using YARN framework to import and export the data, yarn framework provides parallelism as it is read and write multiple nodes parallel and fault tolerance is very much possible because by default replication is happening.
d. ***Import results of SQL query***: It is having facility to import the result of the query in HDFS.
e. ***Compression***: Sqoop having facility to do the compression the data, what it imports from a database. Sqoop having various option to compress the data. Simply if you specify -compress while importing data, Sqoop compress the output file with gzip format by default and it will create an extension as .gz, if you provide -compression-codec instead of compress then sqoop compress the output with bzip2 format.
f. ***Connectors for all major RDBMS Databases***: Sqoop having almost all the connectors to connect the relational databases.
g. ***Kerberos Security Integration***: Sqoop supports Kerberos Authentication, Kerberos Authentication is a protocol which works on the basis of Ticket or key tab which will help you to authenticate users as well as services prior to connect the services like HDFS/HIVE etc.

How Sqoop works:
Sqoop creating SQL query for each mapper internally which is ingesting data from source table to HDFS, basically 4 mapper will be generated by default but you can modify the number of mapper based on your logic and requirement. Number of mapper influence the split by column. Split by column work based on where condition and each mapper have a logical partition of the Target table or directory. For example if we used three mappers and a split-by column. Suppose 1,000,000 records are there. sqoop can segregate using min and max call to the DB on the split-by column . Sqoop first mapper would try to get values from 0 to 333333 records, the second mapper would pull from 333334 to 666666 records and the last would grab from 666667 to 1000000 records. Scoop is running a Map-only job, as we know Reduce phase is required in case of aggregations. But here in Apache Sqoop we just imports and exports the data. It does not perform any aggregations. Map job launch multiple mappers depending on the number defined by the user in the above example we are considering as 3. For Sqoop import, each mapper task will be assigned with a part of data to be imported. Sqoop distributes the input data among the Mappers equally to get high performance. Then each mapper creates a connection with the database using JDBC and fetches the part of data assigned by Sqoop and writes it into HDFS or Hive or HBase based on the arguments provided in the CLI so alternatively Mappers drop the data in the Target-dir with a file named as part-m-00000, part-m-00001, part-m-00002

How Sqoop works to move data into Hive/HDFS:
Here in this scenario we will discuss how sqoop will import data from Relational database to Hive. Sqoop can only import the data as a text file or sequence file into hive database. If you want to use ORC file format then you must follow two stage approach, in first stage sqoop can get the data into HDFS as a text file format or sequence file format, then in second stage hive can convert the data into ORC file format.
Please find the steps as mentioned below.
a. you need to get all the source connection details to get connect with relational database.
Database URL: db.bib.com
Database name: sales
Connection protocol: jdbc:mysql
Source database username and password
Specify the file where password is stored
Example:
sqoop import --connect jdbc:mysql://db.bib.com/sales --table EMPLOYEES --username <username> --password-file ${user.home}/.password
b. Below are the few considerations when you are using parallelism while doing import.
You can mention entire source table for import
You can mention columns from the table
You can mention only latest records by specifying them with a WHERE clause
You can mention number of map tasks specifying write parallelism, in this case Sqoop evenly splits the primary key range of the source table
Example:
sqoop import --connect jdbc:mysql://db.bib.com/sales --table EMPLOYEES --columns "employee_id,first_name,last_name,job_title" --where "start_date > '2010-01-01'" --num-mappers 8
c. Below are the few scenarios where you can mention split-by column
We can use split key using --split-by
We can use split by with condition
Example:
sqoop import --query 'SELECT a.*, b.* FROM a JOIN b on (a.id == b.id) WHERE $CONDITIONS' --split-by a.id --target-dir /user/bib/sales
d. Destination you can mention as HDFS directory or Hive table
Example:
a. HDFS as target directory
sqoop import --query 'SELECT a.*, b.* FROM a JOIN b on (a.id == b.id) WHERE $CONDITIONS' --split-by a.id --target-dir /user/bib/sales
b. Hive as Target table
sqoop import --connect jdbc:mysql://db.foo.com/corp --table EMPLOYEES --hive-import

15. Brief about the Job or Application ID. how job history server is handling the Job details and brief about logging and log files.

ANS:

After jobs submissions Job IDs are generated by job tracker in Hadoop 1 and in Hadoop 2/3 Application IDs are generated. Application ID or Job ID is represents as a global unique identifier for an Application or Job.

Example: job_1410450250506_002 / application_1410450250506_002

_1410450250506 ==> this is the start time of Resource manager which is achieved by using "cluster time stamp"

_002 ==> basically counter is used to keep track of occurrences of job

Task IDs are formed by replacing the job or Application with task prefix within the job

Example: task_1410450250506_0002_m_000002

Here in the above example _000002 is the third map task of the job "job_1410450250506_002"

Tasks may be executed more than once due to task failure so to identify different instances of a task execution, Task attempts are given unique IDs.

Example: attempt_1410450250506_0002_m_0000002_0

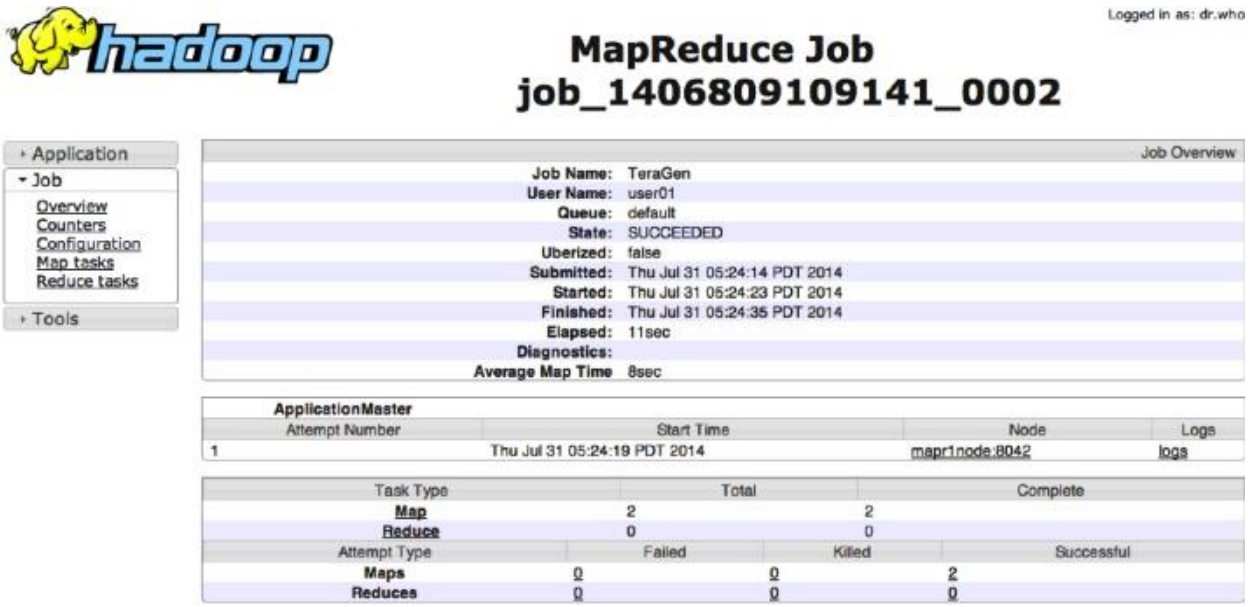
_0 is the first attempt of the task task_1410450250506_002_m_0000002

When you will open the Job history WEB UI, you will get the image as below. Here in the image you can able to see the Job state where the Job is succeeded or failed. How many Mappers and Reducers are launched whether all the Mappers and Reducers are completed or not you can find all these details.

JOB HISTORY Server:

When you click the Job id from the Job history server, you will get below image and more or less similar information you will get as above .

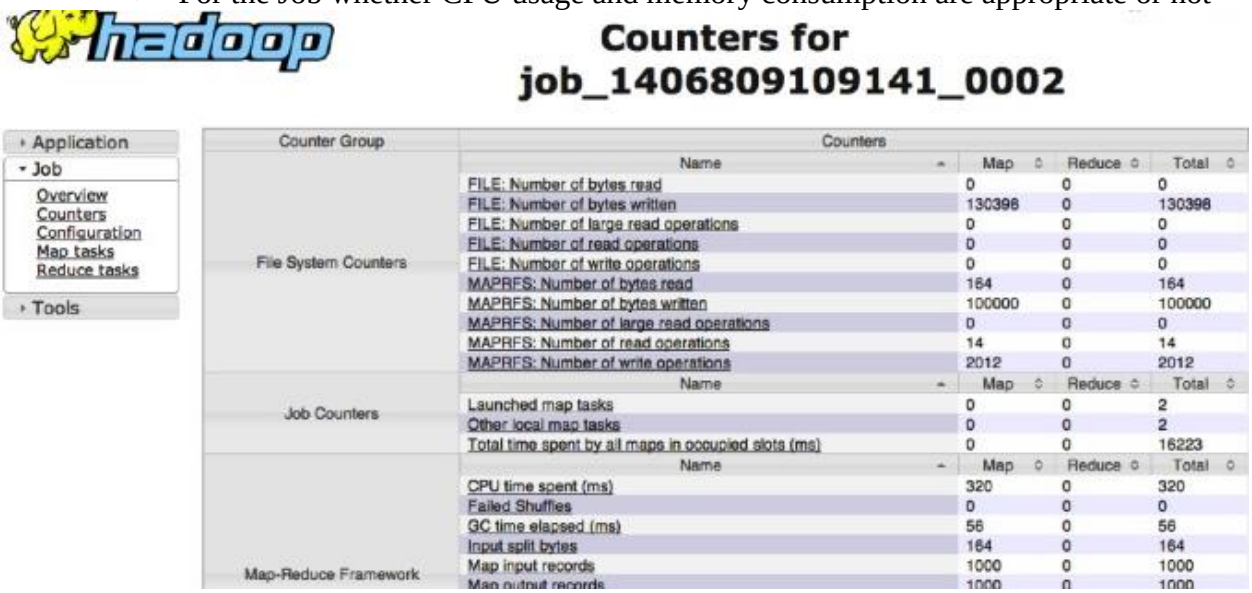
Overview:



Hadoop Counters:

This is the most useful option to examine the job performance. Hadoop provides several built in counters as well as you can customize counters as per your requirements. Counters help you to get the below kind of information.

- Whether correct number of Mappers and Reducers were launched and completed or not
- Whether the correct number of input bytes were read and the expected number of output bytes were written or not.
- Whether correct number of records were read and written in the local file as well as HDFS files or not.
- For the Job whether CPU usage and memory consumption are appropriate or not



Hadoop counters provides three type of Built in counters such as 1. File system counters 2. Job Counters 3. Map reduce Framework counters. Addition to this Hadoop provide another 3 counters from other groups by default, such as 1. Shuffle error counters 2. File input format counters 3. File output format counters.

File system counters:

Under File system counter you can get the information regarding read and write operation in both the local file system and HDFS as well. Total number of bytes read and written depend upon compression algorithms. Here are the few key counters.

File_Bytes_Read: Total number of Bytes read from the local file system by the map reduce Tasks.File_Bytes_Write: Total number of bytes written to the local file system. During the Map phase,mapper task writes the intermediate results to the local file system and During shuffle phase of the Reducer task also write to the local file system when they spill intermediate results to the local file system during sorting.

HDFS_Bytes_Read: Total bytes read from HDFS

HDFS_Bytes_Written: Total bytes written to HDFS.

JOB Counters:

You will get Job information related to Mapper and reducer under JOB Counters. Following are the key job counters.

DATA_LOCAL_MAPS: It indicates how many map tasks executed on local file system alternatively Number of map tasks are running on the same node where the Tasks related data are also available in the same node.

TOTAL_LAUNCHED_MAPS: It shows the Total number of Launched map tasks including failed tasks too. Basically it is the same as the number of input splits for the job.

TOTAL_LAUNCHED_REDUCES: It shows the total number of reducer task launched for the job

NUM_KILLED_MAPS: Number of killed map tasks

NUM_KILLED_REDUCES: Number of killed reduce tasks.

MILLIS_MAPS: This is the total time (In milli sec) spent by all map tasks which are running for the job.

MILLIS_REDUCES: This is the total time spent by all reduce tasks which are running for the job.

Map Reduce Framework counters:

You will get all the statistic of Map Reduce job under Map Reduce framework counter. It will help you to do the performance tuning of the job.

MAP_INPUT_RECORDS: Total number of input records reads for the job during the Map phase.

MAP_OUTPUT_RECORDS: Total number of records written for the job during the Map phase.

CPU_MILLISECONDS: CPU time spent for all the tasks

GC_TIME_MILLIS: Total time spent during the garbage collection of the JVMs. Garbage collection is the process of get back the run time unused memory automatically.

PHYSICAL_MEMORY_BYTES: Total physical memory used by all tasks.

REDUCE_SHUFFLE_BYTES: Total number of output bytes copied from Map tasks to reduce tasks during the shuffle phase.

SPILLED_RECORDS: Total number of records spilled to the disk for all the Map and reducer tasks.

Other counters are as follows:

Shuffle error counters: Error details during the shuffle phase such as BAD_ID,IO_ERROR,WRONG_MAP and WRONG_REDUCE

File input format counters: It includes the bytes read by each task.

File output format counters: Bytes written by each map and reduce task using output format

ADVANCED

****

1. For each YARN job, the HADOOP framework generates a task log file .where are Hadoop task log files stored.

ANS:

Hadoop task log files are stored on the local disk of the slave node running in the disk. In general log related configuration properties are yarn.nodemanager.log-dirs and yarn.log-aggregation-enable. yarn.nodemanager.log-dirs property determines where the container logs are stored on the node when the containers are running. its default value is ${yarn.log.dir}/userlogs. An application localized log directory will be found in /{yarn.nodemanager.log-dirs}/application_${application_id}.individual containers log directories will be shown in subdirectories named container_{$containerid}.

For Map reduce application each container directory will contain the files STDERR,STDOUT and SYSLOG generated by the container.

The yarn.log-aggregation-enable property specifies whether to enable or disable log aggregation, if this function is disabled than node manager will keep the logs locally and not aggregate them. Following properties are in force when log aggregation is enabled.

yarn.nodemanager.remote-app-log-dir: This location is found on the default file system (usually HDFS) and indicates where the node manager should aggregate logs. It should not be the local file system otherwise serving daemon such as the history server will not be able to serve the aggregated logs. The default value is /tmp/logs.

yarn.nodemanager.remote-app-log-dir-suffix: the remote log directory will be created at {yarn.nodemanager.remote-app-log-dir}/${user}/{suffix}. The default suffix value is "logs".

yarn.log-aggregation.retain.seconds: this property defines how long to wait before deleting aggregated logs.

-1 or any other negative value disables the deletion of aggregated logs.

yarn.log-aggregation.retain-check-interval-seconds: this property determines how long tom wait between aggregated log retention checks. If its value is set to 0 or a negative value then the value is computed as one-tenth of the aggregated log retention time.default value is -1.

yarn.log.server.url: once an application is done ,Nodemanagers redirect the web UI users to this URL, where aggregated logs are served ,it points to Mapreduce-Specific jobhistory.

The following properties are used when log aggregation is disabled:

yarn.nodemanager.log.retain-seconds: The time in seconds to retain user logs on the individual nodes if log aggregation is disabled. the default is 10800.

yarn.nodemanager.log.deletion-threads-count: The number of threads used by the node managers to clean up logs once the log retention time is hit for local log files when aggregation is disabled.

2. What is YARN, What are the sequence of services when you are starting YARN?

ANS:

a. YARN stands for Yet Another Resource Negotiator, YARN is taking care of Job tracker's work like resource management and a part of that YARN is working as a scheduler as well. YARN Supports variety of processing engine and Application. When we are saying different data processing engine it means it supports Graph processing, Interactive processing, Stream

processing and batch processing to run and process the data which is stored in HDFS. Basically Resource manager receive the Job request from client and accordingly it will Launch Application master JVM having default memory as 1 core and 2gb.Application Master will contact Name Node and get the location of block, based on availability of block in Node Manager It will check whether sufficient resources are available or not, Accordingly it will inform to Resource manager and Resource manager will provide resources to Node Manager to Launch the JVM for the JOB.
Yarn is working as a scheduler it means the Scheduler is responsible for allocating the resources to running the Application. It will not monitor the Application as well as it will not track the Application. It will not restart the failed task whether it is failed due to Application failure or Hardware Failure.
YARN Scheduler supports three type of scheduler 1. FIFO scheduler 2. FAIR scheduler 3. Capacity scheduler. Based on the Application requirement Hadoop Admin will select either FIFO, FAIR or Capacity scheduler.
FIFO scheduling is First in and First out, in our current environment this is rarely used. Fair scheduling is a method where resources are distributed in such a way that it is more or less equally divided to each job. Capacity scheduler where you can make sure that some percentage of resources you can assign to cluster based on your demand or computing need.
Prior to start the YARN services, start the Resource manager and node manager services. In between Resource manager and Node manager make sure resource manager should start before starting node manager services. Please start your YARN services in sequence mentioned as below.
a. on the resource manager system
#service hadoop-yarn-resourcemanager start
b. on each Node manager(where data node services runs) system
#service hadoop-yarn-nodemanager start
c. To start the Mapreduce job history server
#service hadoop-mapreduce-historyserver start

3. How Hadoop uses HDFS staging directory as well as local directory during a job run
ANS: YARN requires a staging directory for temporary files created by running jobs. Local directories for storing various scripts that are generated to start up the job's containers (which will run the map reduce task).
Staging directory:
a. when user execute a mapreduce job they usually invoke a job client to configure the job and lunch the same.
b. As part of the job execution job client first checks to see if there is a staging directory under the user's name in HDFS, If not then staging directory created under /user/<username>/.staging
C. in addition to job related files a file from the hadoop JAR file named hadoop-mapreduce-client-jobclient.jar also placed in the .staging directory after renaming it to job.jar
d. once the staging directory is set up the job client submits the job to resource manager.
e. Job client also sends back to the console the status of the job progression (ex map 5%, reduce 0%).

Local directory:
a. The resource manager service selects a node manager on one of the cluster's nodes to launch the application master process, which is always the very fast container to be created.

in yarn job.
b. The resource manager choose the node manager depend on the available resources at the time of launching the job. You can not specify the node on which to start the job.
c. The node manager service starts up and generates various scripts in the local cache directory to execute the application Master container/directory.
d. The Application Masters directories are stored in the location that you have specified for the node manager's local directories with the yarn.nodemanager.local-dirs
Configuration property in the yarn-site.xml.
e. The yarn.nodemanager can store its localized file directory with the following directory structure.
[yarn.nodemanager.local-dirs]/usercache/$user/appcache/application_${app_is}

4. in what scenario the container being killed by node manager
ANS:
Basically Container is killed due to high memory usage which is exceeding your container Memory size. When client launches an application, corresponding application master container is launched with ID 000001.the default memory is 1 core and 2gb for each application master container but some time data size will be more, in that case application master reaches the limits of its memory due to which application will fail or Node manager will kill the Application Master and you will get similar massage as mentioned below.

Application application_1424873694018_3023 failed 2 times due to AM Container for appattempt_1424873694018_3023_000002 exited with exitCode: 143 due to: Container [pid=25108,containerID=container_1424873694018_3023_02_000001] is running beyond physical memory limits. Current usage: 1.0 GB of 1 GB physical memory used; 1.5 GB of 2.1 GB virtual memory used. Killing container.

5. What is HDFS Snapshot how it helps you to recover?
ANS: Fundamentally snapshot means taking a Xerox copy of the content from entire file level or sub tree of the file system till certain time and its read only. Snapshot is handling data corruption of user or application and accidental delete. It is always quicker to recovery from snapshot as compare to restore of whole FSImage and it is easy to create a snapshot of important directory before changing anything to it. Snapshot can be taken on any directory once you can marked as "snapshottable" , to doing the same you have to provide the command as "Hdfs dfsadmin -allowSnapshot <Path>" .Once snapshottable directory has been created than under that, subdirectory has been created as .snapshot, It is the place where snapshots are stored. There is no limit on the number of snapshottable directories, any number of directory can create and snapshottable directory can contain 65536 snapshots simultaneously. We can change the name of snapshot or we can use the default one (based on timestamp: "s'yyyyMMdd-HHmmss.SSS"). If there are any snapshots in snapshottable directory then neither you can delete the directory nor rename the directory. Deleting the snapshottable directory you have to delete all the snapshots under that directory. During the upgrading version of HDFS,".snapshot" need to first be renamed or deleted to avoid conflicting with the reserved path.
Snapshots are easily created with *hdfs dfsadmin* command, please find the few commands related to snapshot.

a.
Create directory structure
hdfs dfs -mkdir /my_dir_bibhu
b.
Allow snapshots creation for /my_dir_bibhu
hdfs dfsadmin -allowSnapshot /my_dir_bibhu
Allowing snapshot on /my_dir_bibhu succeeded
c.
Create the first snapshot
hdfs dfs -createSnapshot /my_dir_bibhu snaptest1
Created snapshot /my_dir_bibhu/.snapshot/snaptest1
d.
.snapshot can be read directly using below command
hdfs dfs -ls /my_dir_bibhu/.snapshot
Found 1 item
drwxr-xr-x - bibhu supergroup 0 2016-12-03 09:52 /my_dir/.snapshot/snaptest1
e.
Create new snapshot - this time for directory containing a file
hdfs dfs -createSnapshot /my_dir_bibhu snaptest2
Created snapshot /my_dir_bibhu/.snapshot/snaptest2
f.
This command serves to compare snapshots
hdfs snapshotDiff /my_dir_bibhu .snapshot/snaptest1 .snapshot/snaptest2
g.
Restore snapshot directory to a temporary place and check if file is there or not
hdfs dfs -cp /my_dir_bibhu/.snapshot/snaptest2 /tmp/dir_from_snapshot
hdfs dfs -ls /dir_from_snapshot

6. Map reduce runs on top of yarn and utilizes YARN containers to schedule and execute its map and reduce tasks. When configuring Map Reduce resource utilization on yarn, what are the characteristics to be considered?

ANS:
When configuring Map reduce resource utilization on YARN, There are three aspects to be considered
a. Physical RAM limit for each Map and Reduce Task
b. JVM heap size limit for each task
C. the amount of virtual memory each task will get
Physical RAM limit for each Map and Reduce Task
****
in your Map reduce Job, you can mention how much maximum memory each Map and Reduce task will take, since each Map and each Reduce task will run in a separate container, these maximum memory settings should be at least equal to or more than the YARN minimum Container allocation which it has mentioned in the configuration parameter(yarn.scheduler.minimum-allocation-mb) .
In mapred-site.xml:

<name>mapreduce.map.memory.mb</name>
<value>4096</value>
<name>mapreduce.reduce.memory.mb</name>
<value>8192</value>
The JVM heap size limit for each task
****
The JVM heap size should be set to lower than the Map and Reduce memory defined above, so that they are within the boundaries of the Container memory which is allocated by YARN.
In mapred-site.xml:
<name>mapreduce.map.java.opts</name>
<value>-Xmx3072m</value>
<name>mapreduce.reduce.java.opts</name>
<value>-Xmx6144m</value>
the amount of virtual memory each task will get
****
virtual memory is determined based on the upper limit of the physical RAM that each Map and Reduce task will use. By default the value is 2.1, for example if Total physical RAM allocated as 4 GB than Virtual memory upper limit is 4*2.1 = 8.2 GB

7. You have a setup of YARN cluster where the total application memory available is 18cpu and 36GB RAM there are two JOB such as JOB1 and JOB2 , JOB1 has 2 cpu and 8 GB RAM allocated and JOB2 has 6 cpu and 2 GB RAM allocated. How does the fair scheduler assign the available memory resources under the Dominant Resource Fairness (DRF) scheduler?

ANS:
Prior to discuss about the scenario, which has mentioned in the question. We need to understand how Fair and Capacity scheduler are working in the cluster.
Fair scheduler is a pool based it means the fair scheduler is allow job to be submitted across different pools so that job will get fair resources to run and complete it .for example you are using fair scheduler and you submit the job number 1 during the time no other jobs are running in the scheduler in this scenario job number 1 will take whatever resources required and tried to complete the job so alternatively 100% resource can utilize to complete the job. Now suppose while job number 1 is running during the time job number 2 and 3 are submitted so resources are equally divided to 3 jobs. approximately 33% resources are allocated to each job but if the job number 2 is required 20% resources then another 13% resources are free so whomever either job 2 or 3 can use the 13% resources based on their requirement.
Fair Scheduler can apply fairness scheduling using any of FIFO Policy, Fair Policy or Dominant Resource Fairness Policy.
Capacity scheduler is more of fixed and defines queues with resource quotas. In capacity scheduler suppose job number 1 is submitted in a queue which is using 70% of resources and during the time job number 2 has submitted which is approximately 30% resources are used. In this scenario 100% resources are used now when we submit job number 3 there is no resources available so job number 3 has to wait till the job number 1 or 2 complete. Capacity scheduler can apply fairness scheduling using either FifoPolicy or FairPolicy.
As, I have mentioned earlier Fair Scheduler can use different scheduling policies. The default scheduling policy is fair sharing, here in fair sharing we can use memory as a resource. There's

also a FIFO policy first in first out which is not much use. Some time we are using third type of scheduling policy as DRF, which allocates both memory and CPU resources to applications fairly.
JOB1 --> Each Task require 2 CPU and 8 GB RAM
JOB2 --> Each Task require 6 CPU and 2 GB RAM
Total application memory available --> 18cpu and 36GB RAM
Question is when you are trying to run both the job at a time how fairly each task will get required resources, So DRF will mitigate this to some extent.
JOB1
******
Each Task of Job1 consumes % of total CPUs = 2/18 =1/9
Each Task of Job1 consumes % of total RAM = 8/36 = 2/9
Here in above example 1/9 is less than 2/9 so for job1 dominant resource is RAM. JOB1 is more Memory intensive then CPU intensive.
JOB2
******
Each Task of Job2 consumes % of total CPUs = 6/18 =6/18
Each Task of Job2 consumes % of total RAM = 2/36 = 1/18
Here in above example 1/18 is less than 6/18 so for job2 dominant resource is CPU. JOB2 is more CPU intensive then Memory intensive.

Different job having different resource type, based on this DRF having below properties
1. % of dominant resource type is equally divided to each job it means DRF of job 1 is RAM and DRF of Job2 is CPU so job 1's % of RAM get = job 2's % of CPU get.
2. DRF will solved based on some linear equation.

Solution for our Above Example:
Based on linear equation the ideal distribution would be
JOB 1 gets 3 task Each with 2 CPUs and 8 GB Ram
JOB 2 gets 2 task Each with 6 CPU and 2 GB Ram

JOB 1's % RAM = Number of tasks * RAM per task /Total cluster RAM = 3*8/36 = 2/3
JOB 2's % CPU = Number of tasks * CPU per task /Total cluster CPU = 2*6/18 = 2/3

This confirms that jobs 1's % RAM = job 2's % CPU

8. Each node in your hadoop cluster with running YARN and has 140GB memory and 40 cores .your yarn-site.xml has the configuration as shown below. You want YARN to launch a maximum of 100 containers per node. Enter the property value that would restrict YARN from launching more than 100 containers per node.
<property>
<name>yarn.nodemanager.resource.memory-mb</name>
<value>102400</value>
</property>
<property>
<name>yarn.nodemanager.resource.cpu-vcores</name>
<value>48</value>

</property>
<name>yarn.scheduler.minimum-allocation-mb</name>
<value> what is the correct value here</value>
ANS:
Usually YARN is taking all of the available resources on each machine in the cluster into consideration. Based on the available resources, YARN negotiate the resources as requested from application or map reduce running in the cluster. YARN is allocating container based on how much resources are required to application. A container is the basic unit of processing capacity in YARN, and resource element included memory CPU etc. In Hadoop cluster it is required to balance the usage of memory(RAM),processors (CPU cores) and disks so that processing is not controlled by any one of the these cluster resources. As per the best practice, it allowing for two containers per disk and one core gives the best balance for cluster utilization.
when you are considering the appropriate YARN and MAPREDUCE memory configurations for a cluster node ,at that time it is an ideal situation to consider the below values in each node.
a. RAM (Amount of memory)
b. CORES (Number of CPU cores)
c. DISKS (Number of disks)
Prior to calculate how much RAM, how much CORE and how much disks are required, you have to be aware of the below parameters.
1. Approximately how much data is required to store in you cluster for example 200TB.
2. What is the retention policy of the data for example 1 year
3. What kind of work load you have whether it is CPU intensive for example complex query or query which is computing billion records, I/O Intensive for example Ingestion of data, Memory intensive for example spark processing.
4. What kind of storage mechanism for the data for example whether the data format is plain Text or AVRO or Parque,ORC or compress GZIP or snappy
Total memory available ==> 102400
No of containers ==> 100
Minimum memory required for container ==> 102400 MB total RAM/100 = 1024MB minimum per container.
The next calculation is to determine the maximum number of containers allowed per node.
no of containers = Minimum of (2*cores,1.8* disks,(Total available RAM/MIN_CONTAINER_SIZE)
RAM-per -container = Maximum of (MIN_CONATINER_SIZE, (total available RAM)/CONTAINER))

9. You observed that the number of spilled records from map tasks far exceeds the number of map output records. Your child heap size is 1 GB and your io.sort.mb value is set to 1000mb. How would you tune your io.sort.mb value to achieve maximum memory to I/O ratio?
ANS:

Basically "mapreduce.task.io.sort.mb" is the total amount of buffer memory which is to use while sorting files. It is representing in megabytes.
Tune or provide the io.sort.mb value in such a way that the number of spilled records equals or is as close to equal the number of map output records.

Map reduce job makes the assurance that, the input to every reducer is sorted by key. The process by which the system performs the sort and then transfers the mapper output to the reducers as inputs is known as shuffle. In the Map-reduce job, shuffle is an area of the code where fine-tuning and improvements are continually being made. In many ways the shuffle is the heart of map reduce job. When the map function starts producing output during that time the process is taking advantage of buffering and writes in memory and doing some presorting for more efficiency as well.
Each map task has a circular memory buffer that it writes the output too. The buffer is 100mb by default, a size which can be tuned by changing the io.sort.mb property when the contents of the buffer reaches a certain threshold size. Usually default threshold size of io.sort.spill is 0.8 or 80% when it reaches the threshold a background thread will start to spill the contents to disk. Mapper output will continue to be written to the buffer while the spill takes place, but if the buffer fills up during this time the map will block until the spill is complete. Spills are written in round-robin fashion to the directories specified by the mapred.local.dir property in a job specific subdirectory. Each time when the memory buffer reaches the spill threshold at that time a new spill file is created, so after the map task has written its last output record there could be several spill files before the task is finished. The spill files are merged into single partitioned and sorted the output file. The configuration property io.sort.factor control the maximum number of streams to merge at once. The default value of io.sort.factor is 10.
Just want to brief about how io.sort.factor is working, When the Mapper task is running it continuously writing data into Buffers, to maintain the buffer we have to set up a parameter called io.sort.spill.percent .
The value of io.sort.spill.percent will indicate, after what point the data will be written into disk instead of buffer which is filling up. All of this spilling to disk is done in a separate thread so that the Map can continue running. There may be multiple spills on the task tracker after map task finished. Those files have to be merged into one single sorted file per partition which is fetched by reducer. The property io.sort.factor says how many of those spill files will be merged into one file at a time.

10. You have installed a cluster HDFS and Map Reduce version 2 on YARN. You have no dfs.hosts entries in your hdfs-site.xml configuration file. You configure a new worker node by setting fs.default.name in its configuration files to point to the Name Node on your cluster, and you start the Data Node daemon on that worker node. What do you have to do on the cluster to allow the worker node to join, and start sorting HDFS blocks?
ANS:
Basically DFS.HOST file contain all the data node details and it allows access to all the nodes mentioned in the DFS.HOST file. This is the default configuration used by the name node. DFS.HOST and DFS.HOST.EXCLUDE will help to re-commission and decommission the data nodes.
Hadoop provide the decommission feature to exclude a set of existing data nodes, the nodes to be taken out, should be included into exclude file and the exclude file name should be specified as a configuration parameter as dfs.hosts.exclude. You can find the example as mentioned below.
Examples:
Modify the conf/mapred-site.xml, add:
<property>
<name>dfs.hosts</name>

```
                <value>/opt/hadoop/Bibhu/conf/datanode-allow.list</value>
</property>
<property>
                <name>dfs.hosts.exclude</name>
                <value>/opt/hadoop/Bibhu/conf/datanode-deny.list</value>
</property>
```

Decommission cannot happen immediately because it requires replication of potentially a large number of blocks and we do not want the cluster to be overwhelmed with just this one job. The decommission progress can be monitored on the name-node web UI or cloudera UI. Till all blocks are replicated, the status of nodes will be in "Decommission in progress" state. When decommission is done the state will change to "Decommissioned". The node can be removed whenever decommission is finished.

We can use below commands without creating a dfs.hosts file or making any entries, run the commands hadoop.dfsadminrefreshModes on the Name Node.
$HADOOP_HOME/bin/hadoop dfsadmin -refreshNodes
-refreshNodes, It will update the name node with set of data nodes so that data nodes are allowed to connect Name node.

11. In your Mapreduce job you consistently see that Mapreduce map tasks on your cluster are running slowly because of excessive garbage collection of JVM. How do you increase JVM heap size property to 3GB to optimize performance?

ANS: Java garbage collection is the process by which Java programs perform automatic memory management. When we are talking about automatic memory management, it is a technique which is automatically manage to allocation and deallocation of memory. Java programs compile to bytecode that can be run on a Java Virtual Machine alternatively Byte code is the compiled format of java program, once java program has been converted to byte code afterwards it will execute by JVM and transferred across network. While Java programs is running on the JVM, JVM is consumed memory which is called heap memory to do the same. Heap memory is a part of memory dedicated to the program.
Hadoop mapper is a java process and every java process has its own heap memory. Heap memory maximum allocation settings configured as mapred.map.child.java.opts or mapreduce.map.java.opts in Hadoop2 .If mapper process runs out of heap memory then the mapper throws a java out of memory exceptions as mentioned below.
Error: java.lang.Runtimeexception:Java.lang.OutofMemoryError
The java heap settings or size should be smaller than the Hadoop container memory limit because we need to reserve some memory for java code. Usually it is recommended to reserve 20% memory for code. So if the settings are correct then java-based Hadoop tasks will never get killed by Hadoop so you will not see the "Killing container" error like above.
To execute the actual map or reduce task, YARN will run a JVM within the container. The hadoop property mapreduce.{map|reduc}.java.opts is proposed to pass to this JVM.This could include -Xmx to set max heap size of the JVM.
Example: hadoop jar<jarName> -Dmapreduce.reduce.memory.mb=4096 -Dmapreduce.map.java.opts=-Xmx3276

12. How HIVE Database and IMPALA are working together in CLOUDERA?

ANS:
Hive works on structured data provide a SQL like layer on top of HDFS, Map reduce task will execute for each query of Hive which is trying to do some compute of HDFS data. Impala is Massive parallel processing SQL query engine which is capable enough to handle huge volume of data. Impala is faster than Hive because Impala is not storing the intermediate query results on disk, it process the SQL query in Memory without running any Map reduce.
Please find some of the hive components as mentioned below.
a. Hive Clients
b. Hive Services
a. Hive Clients:
Hive clients are helping hive to perform the queries. There are three types of clients we can use to perform the queries 1. Thrift Clients 2. JDBC clients 3. ODBC clients
Thrift clients: Basically Apache Thrift is a protocol which will help to get connected in between client and server. Apache Hive uses Thrift to allow remote users to make a connection with HiveServer2 (The thrift server) to connect to it and submit queries. Thrift protocols are written in different languages like C++, Java, and Python so user can query the same source in different languages.
JDBC Clients: Apache hive allow Java applications to connect hive using JDBC driver. It is defined the class as apache.hadoop.hive.jdbc.HiveDriver
ODBC Clients: ODBC driver allows applications that support Open database connectivity protocol to connect to the hive.
b. Hive Services
Apache hive provides below services.
1. CLI (Command Line Interface)**:** This is the default hive shell which will help query and command to execute directly.
2. Web interface: Hive also provides web based GUI for executing Hive queries and commands for example HUE in cloudera.
3. Hive server/Thrift server: Different clients are submit their request to hive and get the result accordingly.
4. Hive Driver: once queries are submitted from Thrift/ JDBC/ODBC/CLI/Web UL, driver is responsible to receive those queries, then it will process through compiler, optimizer and executor. Compiler will verify the syntax check with the help of schema present in the metastore then optimizer generates the optimized logical plan in the form of Directed Acyclic Graph of Map reduce and HDFS tasks. The Executor executes the tasks after the compilation and the optimization steps. The Executor directly interacts with the Hadoop Job Tracker for scheduling of tasks to be run.
5. Metastore: Metastore is the central repository of Hive Metadata like schema and locations etc.
Impala components are 1. Impala daemon (*Impalad*) 2. Impala State Store 3. Impala Catalog Service.
Impala daemon (*Impalad*): Impala daemon is running where impala is installed, Impalad accepts the queries from various interfaces like impala shell, hue etc and process the queries to get the result.
Whenever query submitted in any impala daemon, related node is considered “central coordinator node" for that query. After accepting the query, IMPALAD logically divides the query into smaller parallel queries and distribute them to different nodes in the impala cluster. All the Impalad gather

all the intermediate result and send it to central coordinator node, accordingly central coordinator node construct the final query output.
Impala State Store: Impala daemons are continuously communicating with statstore to identify the nodes that are healthy and capable enough to accept the new work. These information will convey to the Daemons by Statstore component. Due to any reason if any node is getting failed then *Statestore* updates all other nodes about this failure and once this notification is available to the other *impalad*, no other Impala daemon assigns any further queries to the affected node.
Impala Catalog Service: Catalog service provide the information about the metadata changes from Impala sql statement to all the impala daemons in the cluster.Impala is use the data stored in Hive so Impala refer the Metastore Table to get connect the Database and Tables created in Hive. Once Table is created through Hive shell, prior to available this table for Impala queries we need to invalidate the metadata so that Impala reloads the corresponding metadata before the query is processed.
Example: INVALIDATE METADATA [[db_name.]table_name];
REFRESH [*db_name*.]*table_name*];

13. Brief about the few optimizing techniques for the Hive performance.
ANS:
As we know that most of the Hive tables are containing billions and millions of records and for any computation, hive query will process with the help of Mapper and Reducer and it will consume more time and memory. Few of the optimization techniques which will always help hive query to do perform better. Please find few of the below techniques.
a. **Use Tez Engine(Option in Horton works):**
Apache TEZ is an execution engine used for faster query execution. Tez will allow you to launch a single Application Master for each session for multiple job, condition is that jobs are relatively small so that Tez engine can use for those jobs. You have to set up the processing engine as Tez instead of default Map-Reduce execution engine providing below parameter.
Set hive.execution.engine=tez;
If you are using Horton works, then you will find TEZ option in the Hive query editor as well.
b. **Enable compression in Hive**
Basically Compression techniques, It reduce the amount of data size transferring , so that it reduces the data transfer between mappers and reducers and compression is not suggestible if your data is already compressed because the output file size might be larger than the original. For better result, you need to perform compression at both mapper and reducer side separately. There are many compression formats are available out of which GZIP is taking more CPU resources than Snappy or LZO but compression ratio is more in GZIP. It is not appropriate for splittable table.
Other formats are snappy, lzo, bzip, etc. You can set compression at mapper and reducer side using below configuration change:
set mapred.compress.map.output = true;
set mapred.output.compress= true;
Users can also set the following properties in hive-site.xml and map-site.xml to get permanent effects.
<property>
<name>mapred.compress.map.output</name>
<value>true</value>

```
</property>
<property>
 <name>mapred.map.output.compression(for MR)/compress(for Yarn).codec</name>
  <value>org.apache.hadoop.io.compress.SnappyCodec</value>
</property>
```

c. Use ORC file format

ORC (optimized record columnar) is an appropriate format for hive performance tuning,query performance can improve using ORC file format easily. We can use ORC file format for all kind of table whether it is partitioned or single and in response, you get faster computation and compressed file size.

Create table sales (id int, name string, address string) stored as ORC tblproperties (“sales.compress”= “SNAPPY”);

Now simply you can also insert the data like

Insert overwrite table sales select * from sale-imp;

d. Optimize your joins

If your table is having large data then it is not suggestable to just use normal joins which we use in SQL. There are many other joins like Map Join; bucket joins, etc. which will help you to improve Hive query performance.

- Use Map Join

When we are talking about Map join,It is a process where join is happening in between two tables during the Map phase without involving reducer phase.Map join is allow one of the tables to get loaded into the memory so that join with other table will be much faster. Hive has a property which can do auto-map join when enabled. Set hive.auto.convert.join to true for enabling the auto map join. We can set this from the command line as well as from the hive-site.xml file

```
<property>
<name>hive.auto.convert.join</name>
<value>true</value>
<description>Whether Hive enables the optimization about converting common join into mapjoin based on the input file size</description>
</property>
```

For performing Map join, there should be two files, one is of larger size and the other is of smaller size. You can set the small file size by using the following property:

hive.mapjoin.smalltable.filesize=(default it will be 25MB)

e. Bucketed Map Join

If tables are bucketed by a particular column, you can use bucketed map join to improve the hive query performance. For enabling the bucketed map join in Hive you have to set up the below parameter.

```
<property>
<name>hive.optimize.bucketmapjoin</name>
<value>true</value>
<description>Whether to try bucket mapjoin</description>
</property>
<property>
<name>hive.optimize.bucketmapjoin.sortedmerge</name>
<value>true</value>
<description>Whether to try sorted bucket merge map join</description>
</property>
```

f. Use partition

Partition is always helpful for huge data. It is used to segregate the large table based on certain column so that the whole data can be divided into small chunks. When we are saying partition the table, basically it allows you to store the data under sub-directory inside a table or directory.

Selecting the partition table is always a critical decision, and you need to take care of future data and volume of data as well. For example, if you have data of a particular location then you can partition the table based on state, you can partition the data in month wise as well. You can define the partition column based on your requirement.

Here is the syntax to create partition table

```
CREATE TABLE countrysales_partition
(Id int, countryname string, population int, description string)
PARTITIONED BY (country VARCHAR(64), state VARCHAR(64))
row format delimited
fields terminated by '\t'
stored AS textfile;
```

There are two types of partition in Hive.

- Static partition
- Dynamic partition

By default the partition is static in hive. In static partition usually we are providing the parameter as "PARTITIONED BY (department String) ". When loading big files into hive, static partition is preferred.

Single insert to partition table is known as dynamic partition and it load the data from non-partitioned Table. If you don't know about how many columns are available in your table in

this scenario also dynamic partition is suitable. To use dynamic partition in Hive, you need to set the following property-

set hive.exec.dynamic.partition=true;

set hive.exec.dynamic.partition.mode=nonstrict;

set hive.exec.max.dynamic.partitions=1000;

set hive.exec.max.dynamic.partitions.pernode=100;

g. Use Vectorization

A standard query is executed one row at a time. vectorized query execution ,it improves performance of operation like scan, aggregation, filter and joins and it is considering 1024 rows at a time to perform the operation. To use Vectorization you can use the below parameter.

set hive.vectorized.execution.enabled=true

set hive.vectorized.execution.reduce.enabled=true

14. How LDAP,Active directory and Kerberos will help Hadoop environment to get secure?
ANS:
LDAP and Active Directory are providing a centralized security system for managing both servers and users, it is managing for all user accounts and associated privileges for your employee. Kerberos is handled Authentication it means when a user trying to connect any Hadoop services, Kerberos will authenticate the user first then it will authenticate service too. When you are considering AD, LDAP and Kerberos in this scenario Kerberos will only provide authentication, all Identity Management is handled outside of Kerberos that is in AD and LDAP.

In high level when a new employee joins, his/her id has to be added in Active directory first then LDAP and Kerberos because AD is a directory service, owned by Microsoft and AD supports several standard protocols such as LDAP and Kerberos.

LDAP and AD communicating each other based on what user ID belongs to which group, for example user Bibhu is a member of which groups and what kind of access permission he is having in different directories or files These are the information is managed differently in AD and Linux system. In Window we have concept which is called SID or Window security identifiers and in Linux we do have User ID or Group ID. SSSD can use the SID of an AD user to algorithmically generate POSIX IDs in a process called ID mapping. ID mapping creates a map between SIDs in AD and UID/GID on Linux.
AD can create and store POSIX attributes such as uidNumber, gidNumber, unixHomeDirectory, or loginShell

There are two ways to mapping these SID and UID/GID using SSSD.
1. To connect AD and LDAP using SSSD you can add the below line in sssd.conf file
ldap_id_mapping = true
2. POSIX permissions are the standards that define how Unix interacts with applications. POSIX stands for Portable Operating System Interface. We need to configure for each user in AD related to each UID/GID of LDAP using POSIX. Especially when we have several domains. In this case we write following in sssd.conf to Disable ID Mapping in SSSD
ldap_id_mapping = False

Below are few concepts need to know to understand the Integration of AD/LDAP/Kerberos.
a. There is no such built in authentication mechanism in LINUX. You can find password details in /etc/passwd file.
b. There are two important modules which are having important role in providing security features at Linux level 1. PAM 2.NSS
PAM: PAM stands for pluggable authentication Module, which allow integration of authentication technology such as Unix,Linux,LDAP etc into system services such as password,login,ssh etc. alternatively When you're prompted for a password, that's usually PAM's doing. PAM provides an API through which authentication requests are mapped into technology specific actions. This kind of mapping is done by PAM configuration files. Authentication mechanism is providing for each service.
NSS: NSS uses a common API and a configuration file (/etc/nsswitch.conf) in which the name service providers for every supported database are specified**.** Here Names include host names, user names, group names such as /etc/passwd, /etc/group, and /etc/hosts.
c. Below are the few components related to Kerberos.

- Key Distribution Center (KDC): It's a Kerberos server which contains encrypted database where it stores all the principal entries related to user, hosts and services including domain or Realm information.
- Authentication Server: Once a user successfully authenticates to the Authenticate server, Authenticate server grants TGT to client .Principal will use the TGT and request access for the Hadoop service.
- Ticket granting server: Ticket granting server validates a TGT in return grant the service ticket to the client, which the client can use to access the Hadoop service.
- Keytab File: It's a secure file which contains the password of all the service principal in a domain.
- Realm: A realm is the Domain name which has to mention in Upper case letters. For example: HADOOP.COM
- Principal: A principal may be a user or service or host which is part of Realm. For example: pbibhu@HADOOP.COM

d. NTP (Network Time Protocol): It is an internet protocol which is used for synchronizing the computer clock time in a network alternatively NTP client initiates a time request exchange with NTP server.
e. PAM_SSS:Kerberos(pam_sss) : One of the design principles of SSSD's PAM module pam_sss was that it should not do any decisions on its own but let SSSD do them. pam_sss cannot decide which type of password prompt should be shown to the user but must ask SSSD first. Currently the first communication between pam_sss and SSSD's PAM responder happens after the user entered

the password. Hence a new request, a pre-authentication request, to the PAM responder must be added before the user is prompted for the password.

f. NSS_SSS: LDAP (nss_sss_) SSSD provides a new NSS module as nss_sss , so that you can configure your system to use SSSD to retrieve user information.

Below are 3 ways of integrating Linux with AD for Authentication
a. Using LDAP/Kerberos PAM and NSS Module
b. Using Winbind
c. Using SSSD that is system services daemon for Integrating with Active Directory

a. Using LDAP/Kerberos PAM and NSS Module:

PAM is configured to use Kerberos for authentication and NSS is to use LDAP protocol for querying UID or GID information. nss_ldap, pam_ldap, and pam_krb5 modules are available to support.
Here Problem is no caching of the credentials and there is no such offline support available here.

b. Using Winbind:

Samba Winbind was a traditional or usual way of connecting Linux systems to AD. Basically Winbind copy a Windows client on a Linux system and is able to communicate to AD servers alternatively we have winbind daemon which will receive calls from PAM and NSS, Once it is received it will translate into corresponding Active directory calls using either LDAP,KERBEROS or Remote protocol(RPC) depending on requirement. The current versions of the System Security Services Daemon (SSSD) closed a feature gap in between Samba Winbind and SSSD so Samba Winbind is no longer the first choice in general.

C. using SSSD that is system services daemon for Integrating with Active Directory:

The System Security Services Daemon (SSSD) is an intermediary between local clients and any Remote Directories and Authentication Mechanism. The local clients connect to SSSD and then SSSD contacts the external providers that is AD, LDAP server. So here SSSD is working as a Bridge which will help you to Access the AD, LDAP.
Basically System authentication is configured locally which means initially services check with a local user store to determine users and credentials. SSSD is allow a local service to check with local cache in SSSD so Local cache information might have taken from an LDAP directory or AD or Kerberos Realm.

Below are the few advantages related to SSSD

- It will Reduce the load on identification/authentication servers.
  despite of connecting AD or LDAP directly, all of the local clients can contact SSSD which can connect to the identification server or check its cache.

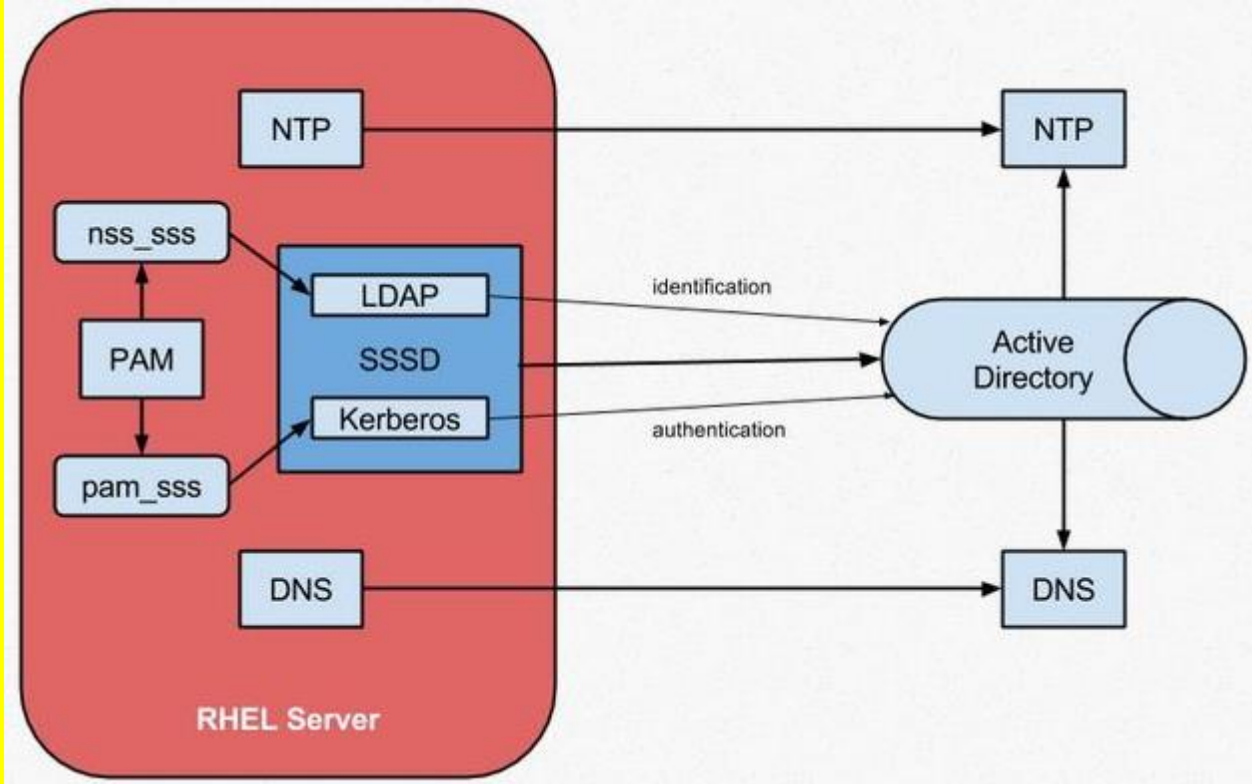
- Permitting offline authentication.

SSSD also caches those users and credentials, so that user credentials are still available if the local system *or* the identity provider go offline.

- Using a single user account.

Usually Remote users have two (or even more) user accounts, such as one for their local system and one for the organizational system. In this scenario it is necessary to connect to a virtual private network (VPN). Because SSSD supports caching and offline authentication, remote users

can connect to network resources simply by authenticating to their local machine and then SSSD maintains their network credentials.



sssd daemon provides different services for different purposes. we have a configuration file called sssd.conf which determine what tasks sssd can do. The file has 2 main parts as we can see here:

```
[sssd]
domains = WIN.EXAMPLE.COM
services = nss, pam
config_file_version = 2

[domain/WINDOWS]
id_provider = ad
auth_provider = ad
access_provider = ad
```

In the first part we have clearly mentioned that what services on the system must use sssd, here in the above example nss and Pam has mentioned. The second part, domain/WINDOWS defines directory services also called identity provider for example AD, LDAP server. SSSD connecting AD/LDAP for querying the information, authentication, password change etc.

In brief below are the steps how SSSD is working or brief about the above diagram

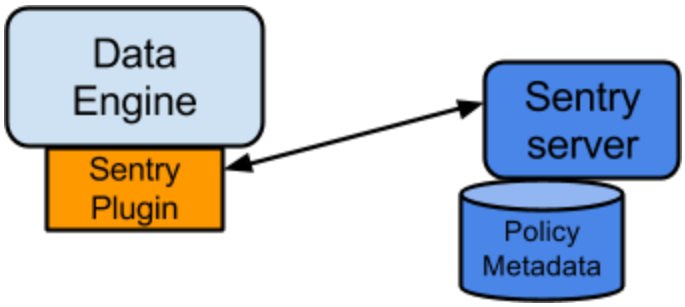
- Once user id and password has provided LIBC opens nss_sss module as per the nsswitch.conf and passes the request.
- The nss_sss memory-mapped cache is consulted first to check the user id and corresponding password
- If not found in nss_sss cache the request is passed to the sssd_nss module or SSSD
- Then sssd_nss checks the SSSD on-disk LDB cache, all cache files are named for the domain. For example, for a domain named exampleldap, the cache file is named cache_exampleldap.ldb. If the data is present in the cache and valid, the nss responder returns it
- If the data is not present in the LDB cache or if it is expired then it connects to the remote server and runs the search, here Remote server indicates AD or LDAP
- The sssd.conf is configured with multiple domains; “domains = AD, LDAP”.
- Active Directory is searched first, and if not found
- LDAP searched next
- When the search is finished the LDB cache is updated
- The sssd_be or SSSD back end control provide signals back to the NSS responder to check the cache again.
- The sssd_nss responder returns the cached data. If there is no data in cache then no data is returned.

15. How sentry Architecture is helping Hadoop services to get secure? How Hive, Impala, HDFS and search activities are working with Sentry

ANS:

Sentry is a role-based authorization to both data and metadata stored on Hadoop cluster for a user. Prior to know more about Sentry, below are the components based on which sentry is working.

1. Sentry server 2. Data engine 3. Sentry plugin



Sentry server: sentry server is a RPC (Remote protocol) server that stores all the authorization meta data details in an underlying relational database. RPC interface is help to retrieve or control the privileges.

Data engine: Data engines which are providing access to the data. Here we can consider data engine as Hive, Impala and Hadoop HDFS.

Sentry Plug-in: sentry plug-in runs in each data engine. The plug-in interfaces will help to manipulation of authorization metadata which is stored in the Apache Sentry Server. Whatever access request come from data engine (Hive,Impala,Hdfs) those are validated by plug-in authorization policy engine referring authorization metadata.

Sentry server only help you to get the metadata information. The actual authorization decision is made by a Data engine which runs in data processing applications such as Hive or Impala. Each component loads the Sentry plug-in it means for each services like Hive/Hdfs/Impala/solr, each sentry plug-in has to be installed for dealing with the Sentry services and the policy engine to validate the authorization request.

Below are the few capability which sentry is having.

1. Fine-Grained Authorization:

It means Permissions on object hierarchies for example Server level, Database level, Table level, view (Row/column level authorization) ,URI and permissions hierarchies will be Select/insert/All this is called Fine-Grained Authorization.

2. Role-Based Authorization (RBAC):

Sentry is providing role based authorization where it is supporting a set of privileges and it support for role templates which is combines multiple access rules for a large set of users and data objects(Database, Table etc).

For example If Bibhu joins the Finance Department, all you need to do is add him to the finance-department group in Active Directory. This will give Bibhu access to data from the Sales and Customer tables.

You can create a role called Analyst and grant SELECT on tables Customer and Sales to this role.

CREATE ROLE Analyst;

GRANT SELECT on table Customer TO ROLE Analyst;

Now Bibhu who is member of the finance-department group get SELECT privilege to the Customer and Sales tables.

GRANT ROLE Analyst TO GROUP finance-department;

3. Multi Tanent Administration or Delegate Admin responsibilities:

It is having capability to delegate or assign the admin responsibilities for a subset of resources. Delegate admin responsibility it means Delegated-Admin Privilege is assigned on a specific set of resources for a specific set of users/groups by a person who has already Delegated-Admin privilege on the specific set of resources.

4. User Identity and Group Mapping: Sentry depend on Kerberos or LDAP to identify the user. It also uses the group mapping mechanism configured in Hadoop to ensure that Sentry sees the same group mapping as other components of the Hadoop ecosystem.

For example considering that users Bibhu and Sibb who belong to an Active Directory (AD) group called finance-department. Sibb also belongs to a group called finance-managers. In Sentry, Create the roles first and then grant required privileges to those roles. For example, you can create a role called Analyst and grant SELECT on tables Customer and Sales to this role.

CREATE ROLE Analyst;

GRANT SELECT on table Customer TO ROLE Analyst;

The next step is to join these authentication entities (users and groups) to authorization entities (roles). This can be done by granting the Analyst role to the finance-department group. Now Bibhu and Sibb who are members of the finance-department group get SELECT privilege to the Customer and Sales tables.

GRANT ROLE Analyst TO GROUP finance-department;

Below are some scenarios where Hive,Imapala,HDFS and search activities are working with Sentry. Considering few examples we will try to understand how it works.

1. **Hive and Sentry:**

If ID "Bibhu" submits the following Hive query:

select * from production.status

Here in the above query Hive will identify that user Bibhu is requesting SELECT access to the Status table. At this point Hive will ask the Sentry plugin to validate access request of Bibhu. The plugin will retrieve Bibhu's privileges related to the Status table and policy engine will determine if the request is valid or not.

2. Impala and Sentry:

Authorization processing in Impala is more or less same as Hive. The main difference is caching of privileges. Usually Impala's Catalog server is managing caching roles and privileges or metadata, and spread it to all Impala server nodes. As a result, impala daemon can authorize queries much faster referring the metadata from the cache memory. Only drawback related to performance is it will take some time for privilege changes to take effect, it might take few seconds.

3. Sentry-HDFS Synchronization:

When we are talking about Sentry and HDFS authorization, it basically talking about Hive warehouse data. Ware house data, It means whether it is Hive or Impala data related to Table. The main objective is when other components like Pig,Mapreduce or Spark trying to access hive table at that time similar authorization check will be occur. At this point this feature does not replace HDFS ACLs. The tables which are not associated with sentry those are retain their old ACLs.
The mapping of Sentry privileges to HDFS ACL permissions is as follows:

- SELECT privilege -> Read access on the file.
- INSERT privilege -> Write access on the file.
- ALL privilege -> Read and Write access on the file

When NameNode loads a Sentry plugin that caches Sentry privileges as well Hive metadata. It helps HDFS to keep file permissions and Hive tables privileges in sync. The Sentry plugin periodically communicate the Sentry and Metastore to keep the metadata changes are in sync.
For example, if Bibhu runs a Pig job, which is reading from the Sales table data files, anyhow data files will be stored in HDFS. Sentry plugin on the Name Node will figure out that data file is part of Hive data and cover Sentry privileges on top of the file ACLs, It means HDFS will get the same privileges for this Pig client that Hive would have applied for a SQL query.
For HDFS-Sentry synchronization to work, for doing the same you *must* use the Sentry service, not policy file authorization.

4. Search and Sentry:

Sentry can apply restriction on search tasks which are coming from a browser or command line or through admin console.
With Search, Sentry stores its privilege policies in a policy file (for example, sentry-provider.ini) which is stored in an HDFS location such as hdfs://ha-nn-uri/user/solr/sentry/sentry-provider.ini.
Multiple policy files for multiple database is not supported by Sentry with Search .However, you must use a separate policy file for each Sentry-enabled service.

5. Disabling Hive CLI:

To execute the hive queries you have to use beeline. When you will disable Hive CLI, Hive CLI is not supported with Sentry and Hive Metastore also be disabled. This is especially necessary if the Hive metastore has sensitive metadata.
To do the same, you have to modify the hadoop.proxyuser.hive.groups in core-site.xml on the Hive Metastore host.
For example, to give the hive user permission to members of the hive and hue groups, set the property to:

```
<property>
<name>hadoop.proxyuser.hive.groups</name>
<value>hive,hue</value>
</property>
```

If more user groups that require access to the Hive Metastore can be added to the comma-separated list as needed.