**Ass-31.2**

**Explain in brief**

**● When should we use HBASE, list some of the scenarios for the same in real time.**

Points to consider before selecting hbase for a real time application

**1. Data volume:**

The volume of data should be in peta bytes of data to be processed in a distributed environment. Otherwise, for a small amount of data, it will be stored and processed in a single node, keeping other nodes idle which is a waste of memory.

**2. Application Types:**

HBase is not suitable for transactional applications, large volume MapReduce jobs, relational analytics, etc. It is preferred when you have a variable schema with slightly different rows. It is also suitable when you are going for a key dependent access to your stored data.

**3. Hardware environment**:

HBase runs on top of HDFS. And HDFS works efficiently with a large number of nodes (minimum 5). So, if you have good hardware support, then HBase can be a good selection.

**4. No requirement of relational features:**

Your application should not have any requirement for RDBMS features like transaction, triggers, complex query, complex joins etc. If you can build your application without these features, then go for HBase.

**5. Quick access to data:**

If you need a real time access to your data, then HBase is a suitable candidate. It is also a perfect fit for storing large tables with multi structured data. It gives ‘flashback’ support to queries, which makes it more suitable for fetching data in a particular instance of time.

-HBase is also suitable when you need fault tolerant, fast and usable data management in a non-relational environment.

- Mozilla generally store all crash data in HBase

- Facebook uses HBase storage to store real-time messages.

**● What are the different modes in which Hbase can be run?**

HBase has two run modes:

**1. Standalone mode:**

This is the default mode. Standalone mode is a “Quick Start - Standalone HBase” section. In standalone mode, HBase does not use HDFS -- it uses the local filesystem instead -- and it runs all HBase daemons and a local ZooKeeper all up in the same JVM. Zookeeper binds to a well known port,so clients may talk to HBase.

**2. Distributed mode:**

Distributed mode can be subdivided into distributed but all daemons run on a single node -- pseudo-distributed-- and fully-distributed where the daemons are spread across all nodes in the cluster.

**a) Pseudo-distributed mode:**

A pseudo-distributed mode is simply a fully-distributed mode run on a single host. Use this configuration testing and prototyping on HBase. Do not use this configuration for production nor for evaluating HBase performance.

**b) Fully-distributed mode:**

 For a production environment, distributed mode is appropriate. In distributed mode, multiple instances of HBase daemons run on multiple servers in the cluster

**● Need and working of zookeeper in Hbase?**

1. HBase uses ZooKeeper as a distributed coordination service for region assignments and to recover any region server crashes by loading them onto other region servers that are functioning.

2. ZooKeeper is a centralized monitoring server that maintains configuration information and provides distributed synchronization.

3. Whenever a client wants to communicate with regions, they have to approach Zookeeper first.

4. HMaster and Region servers are registered with ZooKeeper service, client needs to access ZooKeeper quorum in order to connect with region servers and HMaster.

5. In case of node failure within an HBase cluster, ZKquoram will trigger error messages and start repairing failed nodes.

6. bZooKeeper service keeps track of all the region servers that are there in an HBase cluster- tracking information about how many region servers are there and which region servers are holding which DataNode. HMaster contacts ZooKeeper to get the details of region servers.

7. Various services that Zookeeper provides include –

-Establishing client communication with region servers.

-Tracking server failure and network partitions.

-Maintain Configuration Information

-Provides ephemeral nodes, which represent different region servers.