**31.2**

**Problem Statement:**

Explain in brief:

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

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

● Need and working of zookeeper in Hbase?

**Problem 1:**

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

**Answer:**

* **When should we use Hbase:**

The most important consideration when looking at HBase is that, while it is a great solution to many problems, it is not a silver bullet.

HBase is not optimized for classic transactional applications or even relational analytics

Following are some of the key areas to be considered before finalizing HBase for your application.

Volume of data:

It is the most common point to be considered. You should have 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. So, it will be a misuse of technology framework.

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.

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.

No requirement of relational features: Your application should not have any requirement for RDBMS features like transaction, triggers, complex query, complex joins etc

Quick access to data:

If you need a random and 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. Apart from the above points, HBase is also suitable when you need fault tolerant, fast and usable data management in a non-relational environment.

* **Scenarios for the HBase in real time:**

Apache HBase: Powered By HBase Wiki

Mozilla: Moving Socorro to HBase

Facebook: Facebook’s New Real-Time Messaging System: HBase

StumbleUpon: HBase at StumbleUpon

**Problem 2:**

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

**Answer:**

 HBase run modes:

* Standalone
* Distributed

HBase has two run modes:  Standalone and Distributed. Out of the box, HBase runs in standalone mode. Whatever your mode, you will need to configure HBase by editing files in the Hbase conf directory.

* **Standalone Hbase:**

This is the default mode. 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.

* **Distributed:**

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.

Pseudo-distributed mode can run against the local filesystem or it can run against an instance of the *Hadoop Distributed File System* (HDFS). Fully-distributed mode can ONLY run on HDFS.

**Pseudo-distributed:**

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.

**Fully-distributed:**

By default, HBase runs in standalone mode. Both standalone mode and pseudo-distributed mode are provided for the purposes of small-scale testing.

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

The conf file on the master server contains a list of hosts whose RegionServers are associated with this cluster. Each host is on a separate line.

All hosts listed in this file will have their RegionServer processes started and stopped when the master server starts or stops.

**Problem 3:**

Explain need and working of zookeeper in Hbase.

**Answer:**

* **Need and working of zookeeper in Hbase:**

A distributed Apache HBase (TM) installation depends on a running ZooKeeper cluster. All participating nodes and clients need to be able to access the running ZooKeeper ensemble.

Apache HBase by default manages a ZooKeeper "cluster" for you. It will start and stop the ZooKeeper ensemble as part of the HBase start/stop process.

This variable, which defaults to true, tells HBase whether to start/stop the ZooKeeper ensemble servers as part of HBase start/stop.

HBase is a NoSQL datastore that runs on top of your existing Hadoop cluster (HDFS). It provides you capabilities like random, real-time reads/writes, which HDFS being a FS lacks.

Since it is a NoSQL datastore it doesn't follow SQL conventions and terminologies. HBase provides a good set of APIs (includes JAVA and Thrift). Along with this HBase also provides seamless integration with MapReduce framework.

ZooKeeper is a high-performance coordination service for distributed applications (like HBase). It exposes common services like naming, configuration management, synchronization, and group services, in a simple interface so you don't have to write them from scratch.

You can use it off-the-shelf to implement consensus, group management, leader election, and presence protocols. And you can build on it for your own, specific needs.HBase relies completely on Zookeeper. HBase provides you the option to use its built-in Zookeeper which will get started whenever you start HBase.

But it is not good if you are working on a production cluster. In such scenarios it's always good to have a dedicated Zookeeper cluster and integrate it with your HBase cluster.

* + - **working of zookeeper in Hbase:**

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services.   
  
ZooKeeper has a hierarchal name space, much like a distributed file system. The only difference is that each node in the namespace can have data associated with it as well as children. It is like having a file system that allows a file to also be a directory.  
  
ZNodesEvery node in a ZooKeeper tree is refered to as a *znode*. Znodes maintain a stat structure that includes version numbers for data changes, acl changes. The stat  
structure also has timestamps. The version number, together with the timestamp allow ZooKeeper to validate the cache and to coordinate updates.  
  
WatchesClients can set watches on znodes. Changes to that znode trigger he watch and then clear the watch. When a watch triggers, ZooKeeper sends the client a notification.  
  
Data Access  
The data stored at each znode in a namespace is read and written atomically. Reads get all the data bytes associated with a znode and a write replaces all the data. Each node has an Access Control List (ACL) that restricts who can do what.

Ephemeral Nodes  
ZooKeeper also has the notion of ephemeral nodes. These znodes exists as long as the session that created the znode is active. When the session ends the znode is deleted. Because of this behavior ephemeral znodes are not allowed to have children.Sequence Nodes -- Unique NamingWhen creating a znode you can also request that ZooKeeper append a monotonicly increasing counter to the end of path.