**ASSIGNMENT 31.2**

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

**Data volume:** The volume of data 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. It is also suitable when you are going for a key dependent access to your stored data.

**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. If you can build your application without these features, then go for HBase.

**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. It gives ‘flashback’ support to queries, which makes it more suitable for fetching data in a particular instance of time.

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

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

HBase has two run modes:

* **Standalone HBase**

### Distributed

### Standalone HBase:

* This is the default mode.
* In standalone mode, HBase does not use HDFS, it uses the local filesystem 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 modes require an instance of the Hadoop Distributed File System (HDFS).

Distributed mode can be subdivided into

* pseudo-distributed - where all daemons run on a single node
* fully-distributed - where the daemons are spread across all nodes in the cluster

PSEUDO-DISTRIBUTED:

A pseudo-distributed mode is simply a 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:

Fully-distributed where the daemons are spread across all nodes in the cluster. This is the one which is used in the real time and for production of the Hadoop applications

### Need and working of zookeeper in Hbase?

### A distributed HBase relies completely on Zookeeper (for cluster configuration and management).

### In Apache HBase, ZooKeeper coordinates, communicates, and shares state between the Masters and Region Servers.

### HBase has a design policy of using ZooKeeper only for transient data (that is, for coordination and state communication). Thus if the HBase’s ZooKeeper data is removed, only the transient operations are affected — data can continue to be written and read to/from 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.
* You can also manage the ZooKeeper ensemble independent of HBase and just point HBase at the cluster it should use. To toggle HBase management of ZooKeeper, use the HBASE\_MANAGES\_ZK variable in conf/hbase-env.sh.
* This variable, which defaults to true, tells HBase whether to start/stop the ZooKeeper ensemble servers as part of HBase start/stop.
* Primarily, Zookeeper maintains a live cluster state
* The other common uses for ZooKeeper are storage of small bits of dynamic configuration data, locking, and group membership