**Session 31**

**Assignment 2**

**Problem Statements:**

**Part 1:**

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

HBase offers lot of good functionalities like a NoSQL DB, but it is still not a ‘Fit for All’ solution. Following are some of the key areas to be considered before finalizing HBase for the application.

* **Data volume:** The volume of data is the most common point to be considered. Size of the data should be of peta bytes 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.
* **Application Types:** HBase is not suitable for transactional applications, large volume Map Reduce 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 which 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 the application cab be built without these features, then HBase could be a good option.
* **Quick access to data:** If a random and real time access is needed to for data, then HBase is a suitable candidate. It is also a perfect fit for storing large tables with multiple 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.

Following are some of the recent improvements in HBase:

* Improved high availability
* HBase and YARN integration
* Block cache compression
* Support to data types
* Support to rolling upgrades

**Part 2:**

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

HBase has two run modes:

* Standalone HBase

### Distributed

### Out of the box, HBase runs in standalone mode. To set up a distributed deploy, you will need to configure HBase by editing files in the HBase ‘conf’ directory.

### Standalone HBase:

* This is the default mode.
* Standalone mode is the same as Quick Start mode.
* In standalone mode, HBase does not use HDFS, it uses the local file system and it runs all HBase daemons and a local Zookeeper all up in the same JVM.
* Zookeeper binds to a well-known port so the clients can 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

1. **Pseudo-Distributed:**

* A pseudo-distributed mode is simply a distributed mode run on a single host.
* Pseudo-distributed mode can run against the local file system or it can run against an instance of the Hadoop Distributed File System (HDFS).
* Use this configuration testing and prototyping on HBase but not for production or for evaluating HBase performance.

1. **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 more appropriate for production of the Hadoop applications.
* In distributed mode, multiple instances of HBase daemons run on multiple servers in the cluster.
* Like the pseudo-distributed mode, a fully distributed configuration requires the hbase-cluster.distributed property to be set as true. Typically, the hbase.rootdir is configured to point to a highly-available HDFS file system.
* In addition, the cluster is configured so that multiple cluster nodes enlist as RegionServers, ZooKeeper QuorumPeers, and backup HMaster servers.

### Part 3:

### Need and working of zookeeper in HBase?

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

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

### 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.

### HBase has a design policy of using ZooKeeper only for transient data. Thus if the ZooKeeper data is removed from the HBase, only the transient operations are affected — data can continue to be written and read to or from HBase.

* A distributed Apache HBase 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.