**Assignment 31.2**

**Problem Statement:**

**Explain in brief :**

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

If your application has a variable schema where each row is slightly different, then you should look at HBase. As an example, doing a modeling exercise using a standard relational schema; When you can’t add columns fast enough and most of them are NULL in each row, you should consider HBase. If you find that your data is stored in collections, for example some meta data, message data or binary data that is all keyed on the same value, then you should consider HBase. If you need key based access to data when storing or retrieving, then you should consider HBase.

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

HBase has two run modes:

1. Standalone mode
2. Distributed mode

**Standalone mode:**

This is the default mode. In standalone mode, HBase does not use HDFS .It uses the local filesystem. 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 mode:** Distributed mode can be subdivided into:

1) Pseudo-Distributed mode

2) Fully-Distributed mode

**Pseudo-Distributed mode:**

All daemons run on a single node. A pseudo-distributed mode is simply a distributed mode run on a single host.

This is used for prototyping HBASE .This should not be used for production and evaluating the HBASE performance.

**Pseudo-Distributed configuration file:**

<configuration>

...

<property>

<name>hbase.rootdir</name>

<value>hdfs://h-24-30.sfo.stumble.net:8020/hbase</value>

</property>

<property>

<name>hbase.cluster.distributed</name>

<value>true</value>

</property>

<property>

<name>hbase.zookeeper.quorum</name>

<value>h-24-30.sfo.stumble.net</value>

</property>

...

</configuration>

**Fully-Distributed mode:**

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.

**Fully-Distributed configuration file:**

<configuration>

...

<property>

<name>hbase.rootdir</name>

<value>hdfs://namenode.example.org:8020/hbase</value>

<description>The directory shared by RegionServers.

</description>

</property>

<property>

<name>hbase.cluster.distributed</name>

<value>true</value>

<description>The mode the cluster will be in. Possible values are

false: standalone and pseudo-distributed setups with managed Zookeeper

true: fully-distributed with unmanaged Zookeeper Quorum (see hbase-env.sh)

</description>

</property>

...

</configuration>

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

Apache HBase uses ZooKeeper to track the status of distributed data throughout the master and region servers with the help of **centralized configuration management** and **distributed mutex** mechanisms. Here are some of the use-cases of HBase −

* **Telecom** − Telecom industry stores billions of mobile call records (around 30TB / month) and accessing these call records in real time become a huge task. HBase can be used to process all the records in real time, easily and efficiently.
* **Social network** − Similar to telecom industry, sites like Twitter, LinkedIn, and Facebook receive huge volumes of data through the posts created by users. HBase can be used to find recent trends and other interesting facts.

There are a few supporting services that are important and one that’s required.

HBase uses ZooKeeper for various distributed coordination services such as master election. As HBase develops and grows it continues to rely on ZooKeeper for additional functionality, making it a key part of the system. In addition, you should have proper network services in place such as NTP and DNS. HBase depends on all nodes in the cluster having closely synchronized clocks and referring to each other consistently. Using NTP and DNS ensures that you won’t run into odd behaviors when one node A thinks that the time is tomorrow and node B thinks it’s yesterday. You’ll also prevent situations where the master node tells node C to serve a region but node C doesn’t know its own name and doesn’t answer. Using NTP and DNS will save a lot of headaches as you get started.