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?

Q1. When should we use HBASE, list some of the scenarios for the same in real time?

When to 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. It is also not a complete substitute for HDFS when doing large batch MapReduce.

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.

Use Cases

* Apache HBase: Powered By HBase Wiki
* Mozilla: Moving Socorro to HBase
* Facebook: Facebook’s New Real-Time Messaging System: HBase
* StumbleUpon: HBase at StumbleUpon

Q2. What are the different modes in which Hbase can be run?

**HBase run modes: Standalone and Distributed**

HBase has two run modes: “Standalone HBase” and  “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.

Whatever your mode, you will need to edit conf/hbase-env.sh to tell HBase which **java** to use. In this file you set HBase environment variables such as the heapsize and other options for the JVM, the preferred location for log files, etc. Set JAVA\_HOME to point at the root of your **java** install.

**1. Standalone Hbase:**

This is the default mode. Standalone mode is what is described in the “Quick Start” 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:**

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 [[9](http://hbase.apache.org/0.94/book/standalone_dist.html" \l "ftn.d1984e717)].

Distributed modes require an instance of the *Hadoop Distributed File System* (HDFS). See the Hadoop requirements and instructions for how to set up a HDFS. Before proceeding, ensure you have an appropriate, working HDFS.

Below we describe the different distributed setups.

**2.1. 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.

**2.2. Fully-distributed**

For running a fully-distributed operation on more than one host,

Q3. Need and working of zookeeper in Hbase?

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. ZooKeeper is a centralized monitoring server that maintains configuration information and provides distributed synchronization. Whenever a client wants to communicate with regions, they have to approach Zookeeper first. HMaster and Region servers are registered with ZooKeeper service, client needs to access ZooKeeper quorum in order to connect with region servers and HMaster. In case of node failure within an HBase cluster, ZKquoram will trigger error messages and start repairing failed nodes.

ZooKeeper 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. 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.