

**Solution-**

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

**Features-**

**Generic data model -**

* Heterogeneous containers, including sets, maps, and arrays.

**Dynamic type discovery and conversion-**

* NoSQL analytics systems support runtime type identification and conversion so that custom business logic can be used to dictate analytic treatment of variation.

**Non-relational and De-normalised** -

* Data is stored in single tables as compared to joining multiple tables.

**Commodity hardware**

* Adding more of the economical servers allows NoSQL databases to scale to handle more data

**Highly distributable**

* Distributed databases can store and process a set of information on more than one device

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

**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.

Some real life scenarios where HBase is used-

* Use of HBase by Mozilla: They generally store all crash data in HBase
* Use of HBase by Facebook: Facebook uses HBase storage to store real-time messages

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

**Soluion-**

* 1. 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.
  2. Distributed HBase- 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. Distributed modes require an instance of the Hadoop Distributed File System (HDFS).

1. **Need and working of Zookeeper in HBase.**

A Distributed applications like HBASE needs coordination service like zookeeper. ZooKeeper is a high-performance coordination service for distributed applications. 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.