Explain in brief

● Differences between HBASE and HDFS.

HDFS is a distributed file system and has the following properties:  
1. It is optimized for streaming access of large files. You would typically store files that are in the 100s of MB upwards on HDFS and access them through MapReduce to process them in batch mode.  
2. HDFS is optimized for use cases where you write once and read many times like in the case of production logs. You can append to files in some of the recent versions but that is not a feature that is very commonly used. There is no concept of random writes.  
3. HDFS doesn’t do random reads very well.

HBase on the other hand is a distributed column oriented database. The filesystem of choice typically is HDFS owing to the tight integration between HBase and HDFS. Having said that, it doesn’t mean that HBase can’t work on any other filesystem. It’s just not proven in production and at scale to work with anything except HDFS.  
HBase provides you with the following:  
1. It gives you the ability to do random read/writes on your data which HDFS doesnt allow you to.  
2. HBase stores data in the form of key value pairs in a columnar fashion. HBase provides a flexible data model.  
3. Fast scans across tables.  
4. Scale in terms of writes as well as total volume of data.

● List and explain the main components of HBASE.

HBase architecture has 3 important components- HMaster, Region Server and ZooKeeper.

1. HMaster

HBase HMaster is a lightweight process that assigns regions to region servers in the Hadoop cluster for load balancing. Responsibilities of HMaster –

* Manages and Monitors the Hadoop Cluster
* Performs Administration (Interface for creating, updating and deleting tables.)
* Controlling the failover
* DDL operations are handled by the HMaster
* Whenever a client wants to change the schema and change any of the metadata operations, HMaster is responsible for all these operations.

1. Region Server

These are the worker nodes which handle read, write, update, and delete requests from clients. Region Server process, runs on every node in the hadoop cluster. Region Server runs on HDFS DataNode and consists of the following components –

* Block Cache – This is the read cache. Most frequently read data is stored in the read cache and whenever the block cache is full, recently used data is evicted.
* MemStore- This is the write cache and stores new data that is not yet written to the disk. Every column family in a region has a MemStore.
* Write Ahead Log (WAL) is a file that stores new data that is not persisted to permanent storage.
* HFile is the actual storage file that stores the rows as sorted key values on a disk.

1. Zookeeper

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.

● Does Hbase support sql?

HBase provides random, read-write access to Big Data stored in very large tables as a distributed columnar store. HBase thus gained immediately popularity as a Big Data technology that unlike Hadoop which was primarily used in the backend data warehousing infrastructure could be deployed to service online transactions as well.

**Phoenix**

Phoenix is a technology developed by Salesforce.com to put a SQL skin over HBase. This is not adding a new layer, rather it exposes HBase functionality through SQL using an embedded JDBC Driver that allows clients to run at native HBase speed. The JDBC driver compiles SQL into native HBase calls.

The claim is that performance of Phoenix is much better than that of Hive over HBase – upto 10x at 100 million rows. Performance is also more constant as the number of rows increases. Optimizations are transparent to the user.

**Impala**

Cloudera has been promoting Impala heavily since its announcement in Strata last year. Impala is a SQL engine that can run on HDFS or HBase or both. They too claim that Impala performance is much better than that of Hive – over 45 times! However this comparison seems to be based on native Hive, not Hive over HBase. The performance gains are achieved by not writing intermediate results to disk (like MapReduce), no spin up/down times, optimized code as Impala is written in C++ rather than Java. This means that it needs to use JNI to make calls into Java for HBase integrations.

Impala also includes authentication via Kerberos support, JDBC/ODBC drivers (similar to Phoenix), an interactive shell and support for popular hadoop file formats such as parquet, avro, thrift and sequence files. Parquet was specifically engineered for Impala to get the best performance.

The use cases for HBase and Impala include the following:

* Data is being streamed in
* Need to run queries on data as it is being created
* Need real time edits of data in-place
* Need random access to computed results (results can be placed in HBase)

**Drill**

Apache Drill (the only in this series which is an Apache project) was inspired by Google Dremel. It is the most interesting in terms of its emphasis on interactive analysis of large scale datasets. It is similar to Impala in many ways but is community driven. Although Hive is also a community project, the need for Drill rose because of the tight coupling of Hive to MapReduce which is based on a pessimistic execution principle i.e. all jobs will be long running and process lots of data. This causes a lot of overhead for short-running jobs which Drill addresses.

Drill does great when execution times of 100ms to 10 seconds are required. Phoenix provides even better response times – below 10ms.

Drill is unique in its focus on analytical queries, so a lot of emphasis is placed on bulk reads and aggregations. Phoenix is more focused on write optimizations as in the RDBMS world.

Drill also provides extensions to SQL for nested data types similar to Big Query. Many systems nest data structures inside of HBase cells (for example a cell can be a whole log record which in turn has many nested fields). It is much better to treat these nested cells and process them in the query language for optimal performance.

Drill leverages recent research approaches (late record materialization, vectorized operators, etc.) As such it definitely seems the most promising for analytical work.