1. Hardware architecture

Applications: Google Earth-like

Client Htable  MR Client Htable

HBASE

HDFS

Operating Sys

Hardware
2. Data Model:

3. BigTable and Hbase are really webtable (huge)
   Access to a row is atomic—strictly consistent
   Multiversion and timestamping can help with application consistency
   Basic unit of scalability and load balancing in Hbase is called a region. Initially there is only one region and as you start adding data, system monitors and make sure it does not exceed the limit of the size of a region. It is split into roughly two equal pieces or two regions: auto-sharding.
   Each region is served by a region server; a region server may server many regions. Auto-sharding is instantaneous.
   Regions are unit for fault-tolerance: can be moved around from region server to region server, when a RS fails.
4. Logical view of many rows together into regions served by different region servers

5. Storage API
   Operations to create and delete table and column families, to change the table and column metadata such as compression or block sizes.
   Scan rows, select columns, versions etc.
   Single-row transaction row-read-modify
   NO domain specific language, Use Java or Python; this is where PIG can help; pig has a HBaseStorage adapter to work on data in a Hbase table.
   Wrappers for MR.
   Get, put, delete, scan
6. Implementation
Uses existing components such as HDFS and Zookeeper.

Big-data application: EMR, healthcare, health

7. Cataloging the data: how to address when a linear space grows large: hierarchical space
-ROOT- → META table → Region server → Region

When a client request a read/write with a row key, Zookeeper locates ROOT file, that has the META DATA about Region servers → Region Server has details about Region it manages

User table Implemented Thru regionserver and regions: Rows, colfam, cols