HBase

Welcome to the world of NoSQL

Agenda

- Introduction to HBase
- Why to use HBase
- What HBase is not
- Characteristics
- Data Model
- Architecture
- HBase vs HDFS
- HBase vs RDBMS

What is HBase

- HBase is an open source, non-relational, distributed database modeled after Google's BigTable and written in Java.

- Think of it as a sparse, consistent, distributed, multidimensional, sorted map:
  - labeled tables of rows
  - row consist of key-value cells:

  - (row key, column family, column, timestamp) -> value

Why HBase

- Storing large amounts of data (TB/PB)
- Storing unstructured or variable column data
- Big Data with random read and writes
- Horizontally scalable
- Strongly consistent reads and writes
- Simple Java API
- Integration with Map/Reduce framework

What HBase is not

- Not an SQL database
- Not relational
- No joins
- No fancy query language and no sophisticated query engine
- No transactions out-of-the box
- No secondary indices out-of-the box
- Not a drop-in replacement for your RDBMS

Characteristics

- Non-relational (NoSQL)
- Column-Oriented
- Multi-Dimensional
- High Availability
- High Performance
- Open Source

NoSQL Database

- HBase is a "NoSQL" database.
- "NoSQL" is a general term meaning that the database doesn't supports SQL as its primary access language. There are many types of NoSQL databases:
  - **Key-Value Store** – It has a Big Hash Table of keys & values (Example- Riak, Amazon S3 (Dynamo))
  - **Document-based Store** - It stores documents made up of tagged elements. (Example- CouchDB)
  - **Column-based Store** - Each storage block contains data from only one column, (Example- HBase, Cassandra)
  - **Graph-based** - A network database that uses edges and nodes to represent and store data. (Example- Neo4J)

HBase Data Model

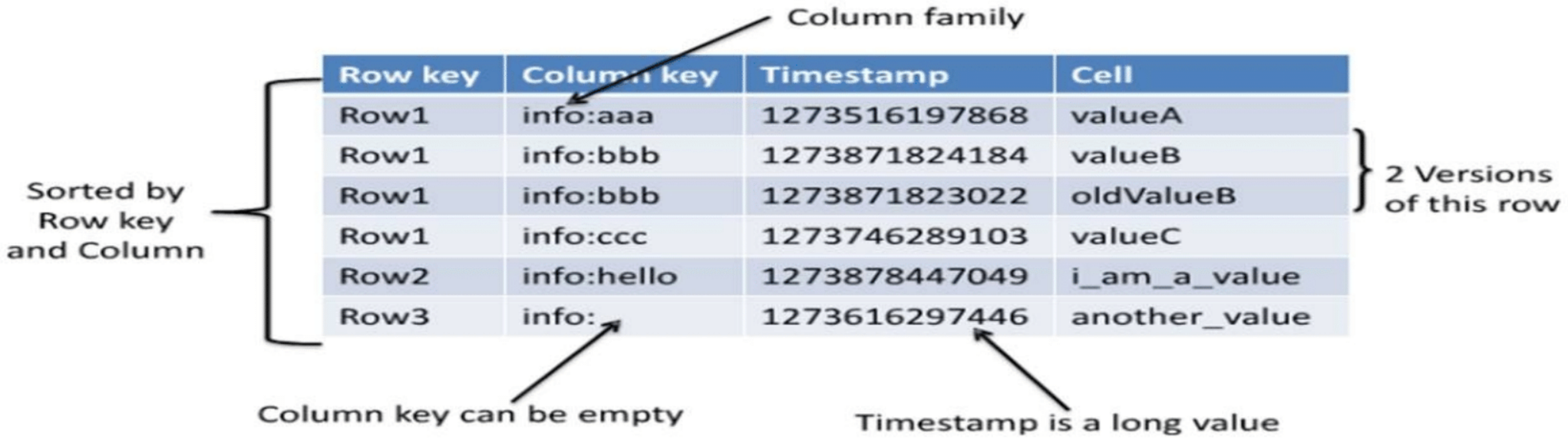
- Every row has a row key (analogous to a primary key)
- Rows are stored by row key for fast lookups

- A table may have 1 or more column families
- Common to have a small number of column families
- A column family can have number of columns

- Each row has a timestamp
- Multiple versions of a row can exist

HBase Data Model

- Table is a distributed sorted map
  - row key + column family + column + timestamp -> value



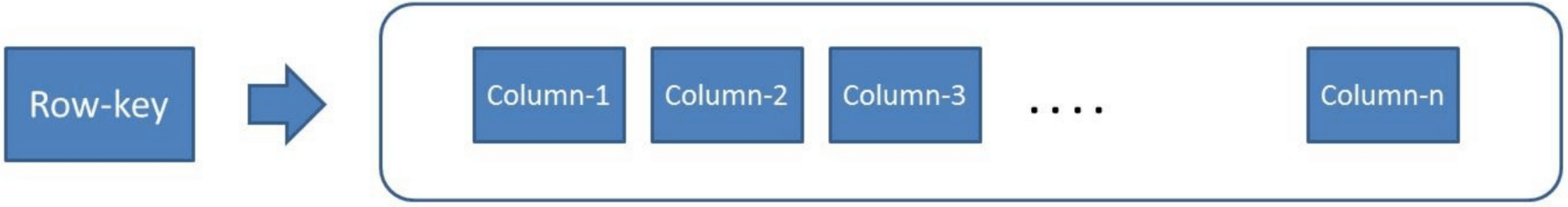
Versions of Data

- Stores multiple versions of columns
- With each version a timestamp is stored

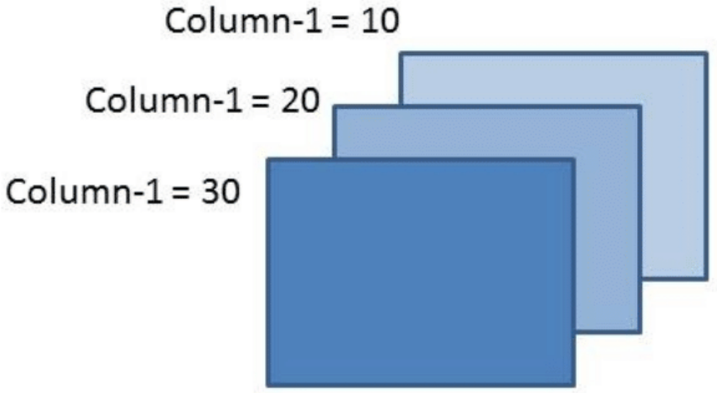
Key	Column	Value	Timestamp
rowA	Fam:foo	New value	1275340679713
rowA	Fam:foo	Old value	1275091706190
rowB	Fam:foo	Some value	1274999316683

} Sorted in descending order

Versions of Data

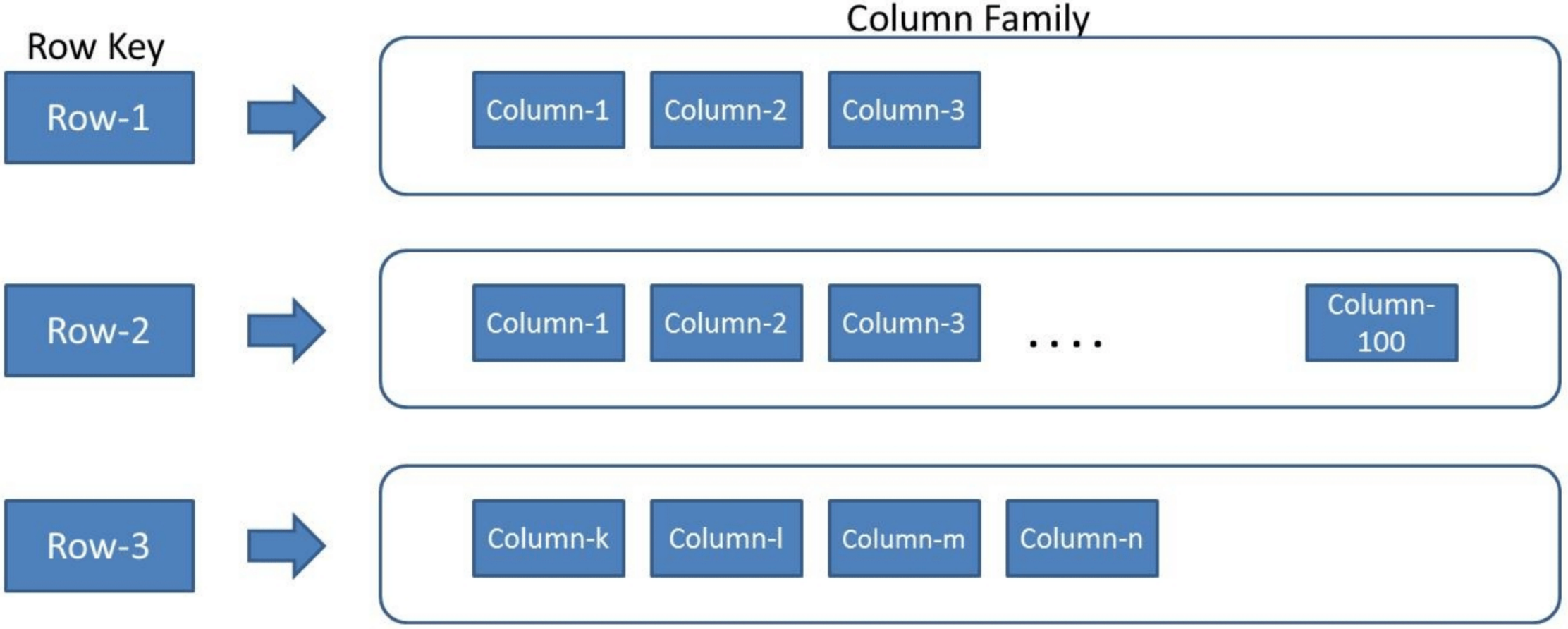


Column Family

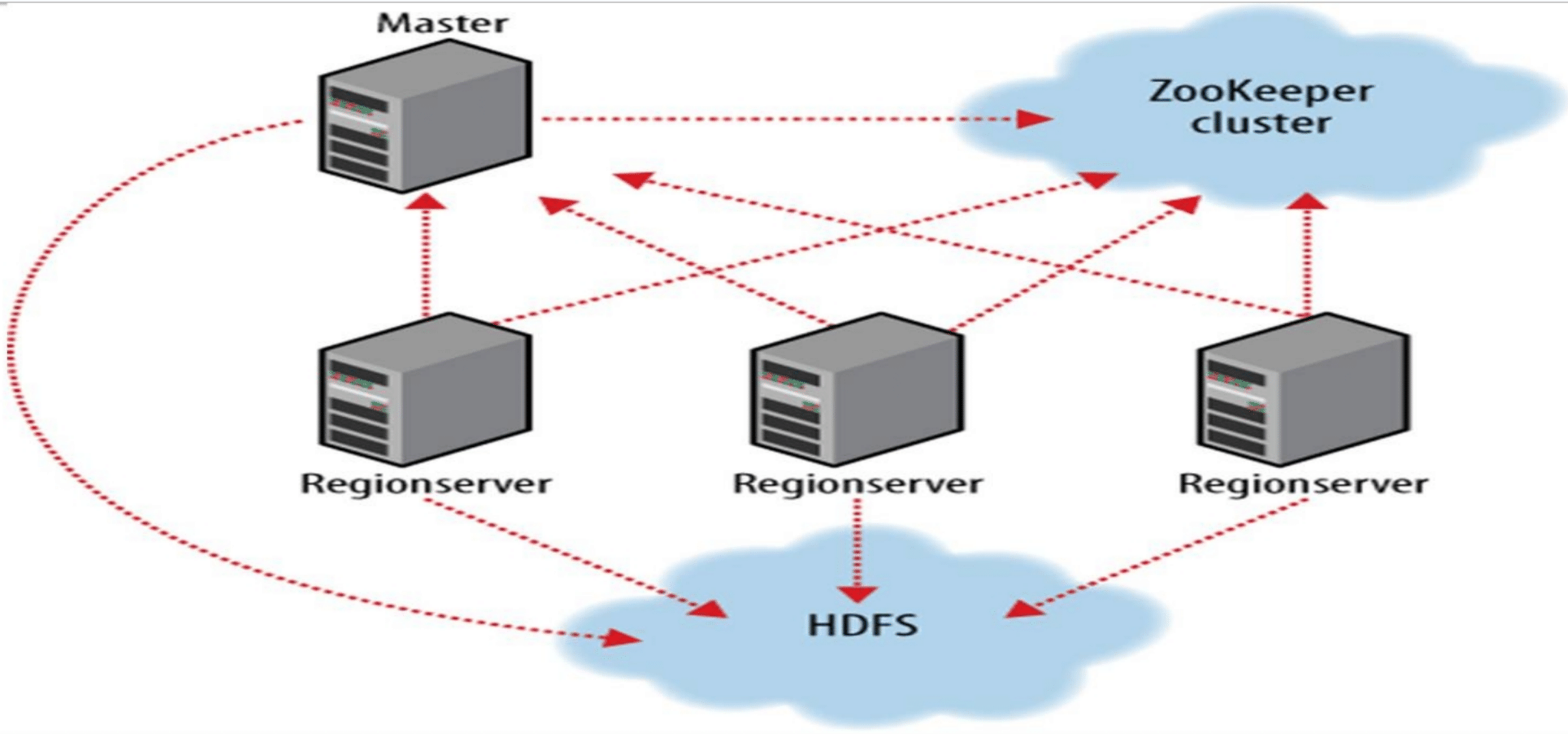


Schema-less

Create Table specify: Table-Name, Column-Family-Name



HBase Architecture



Storage Model

- Column – oriented database (column families)
- Table consists of Rows, each which has a primary key(row key)
- Each Row may have any number of columns
- Table schema only defines Column familes (column family can have any number of columns)
- Each cell value has a timestamp

Programming with HBase

- HBase shell
  - Scan, List, Create, Get, Put, Delete
- Java API
  - Get, Put, Delete, Scan,…
- Non-Java Clients
  - Thrift
  - REST
- HBase MapReduce API
  - hbase.mapreduce.TableMapper;
  - hbase.mapreduce.TableReducer;
- High Level Interface
  - Pig, Hive

HBase vs HDFS

- Both are distributed systems that scale to hundreds or thousands of nodes

- HDFS is good for batch processing (scans over big files)
  - Not good for record lookup
  - Not good for incremental addition of small batches
  - Not good for updates
  - Not good for Random read write

HBase vs HDFS

- HBase is designed to efficiently address the above points
  - Fast record lookup
  - Support for record-level insertion
  - Support for updates

- HBase updates are done by creating new versions of values

Use Cases

- Facebook Analytics
  - Real-time counters of URLs shared, preferred links
- Twitter
  - 25 TB of message every month
- Mozilla
  - Store crashes report, 2.5 million per day.