Typical size of data you deal with on a daily basis
Processes

More than 161 Petabytes of raw data a day

On average, 1MB-2MB per minute

few times, exceeded a 3MB per-minute average (RARE)

its system processes 500+ terabytes of data each day

105 TB – data scanned(30 min)
RDBMS ???

- Fixed schema based design
- Good for small to medium volume applications
- Not designed to handle heterogeneous data
- Not designed to handle mixed workload
- Not designed to scale

Err... NO.
Hadoop

- Build based on Google infrastructure (GFS, Map-Reduce papers published 2003/2004), Hadoop is a software framework for distributed storage and distributed processing for large amounts of data. (needs to shortened to a single line)
- Java/Python/C interfaces, several projects built on top of it
- Open-source implementation of 2 key ideas
  - HDFS: Hadoop distributed file system
  - Map-Reduce: Programming Model
- Problems:
  - Data accessible only in a sequential manner and lacked random read/write access
What is Hbase?

- Open source clone of Google’s Bigtable and is written in Java.
- Distributed column-oriented wide column store on top of HDFS
- Real time read/write random-access
- Not Relational and does not support SQL

- But can work with very large datasets
  - Billions of rows, millions of columns

- Originally created in PowerSet in 2007
- Used in: Yahoo, Microsoft, Adobe, Twitter, …
<table>
<thead>
<tr>
<th>Year</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nov 2006</td>
<td>Google released the paper on BigTable.</td>
</tr>
<tr>
<td>Feb 2007</td>
<td>Initial HBase prototype was created as a Hadoop contribution.</td>
</tr>
<tr>
<td>Oct 2007</td>
<td>The first usable HBase along with Hadoop 0.15.0 was released.</td>
</tr>
<tr>
<td>Jan 2008</td>
<td>HBase became the sub project of Hadoop.</td>
</tr>
<tr>
<td>Oct 2008</td>
<td>HBase 0.18.1 was released.</td>
</tr>
<tr>
<td>Jan 2009</td>
<td>HBase 0.19.0 was released.</td>
</tr>
<tr>
<td>Sept 2009</td>
<td>HBase 0.20.0 was released.</td>
</tr>
<tr>
<td>May 2010</td>
<td>HBase became Apache top-level project.</td>
</tr>
</tbody>
</table>
Features

- Horizontally scalable
- Automatic Failover
- Automatic Shading
- Integration with Map Reduce framework
- Stores and retrieves data with random access
- It doesn't care about data types
- It doesn't enforce relationships within your data
- It is designed to run on a cluster of computers, built using commodity hardware
Storage Mechanism

• Hbase is a Column-Oriented Database.
• Tables in it are sorted by rows

<table>
<thead>
<tr>
<th>Row-Oriented Database</th>
<th>Column-Oriented Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>It is suitable for Online Transaction Process (OLTP).</td>
<td>It is suitable for Online Analytical Processing (OLAP).</td>
</tr>
<tr>
<td>Such databases are designed for small number of rows and</td>
<td>Column-oriented databases are designed for huge tables.</td>
</tr>
<tr>
<td>columns.</td>
<td></td>
</tr>
</tbody>
</table>
**Data Model**

- **Table** is a collection of rows.
- **Row** is a collection of column families.
- **Column family** is a collection of columns.
- **Column** is a collection of key value pairs.
- A **cell** is a combination of row, column family, and column qualifier, and contains a value and a timestamp, which represents the value’s version.
- A **timestamp** is written alongside each value, and is the identifier for a given version of a value.
Architecture
Regions

HBase tables are divided horizontally by row key range into “Regions”

Regions are assigned to the nodes in the cluster, called “Region Servers”
Region Server

- Region Server
  - BlockCache
  - Region
    - Memstore
    - HFile
  - Region
    - Memstore
    - HFile
- WAL
  - Write ahead log on disk- Used for recovery
- HDFS Data Node
  - Read Cache, LRU evicted
  - Write Cache, sorted map of KeyValues in memory
  - Hfile=sorted KeyValues on disk

Region Server

- WAL
- Blockcache
- Memstore
- Hfiles
The master is responsible for the following:

- Coordinating the region servers
- Admin functions
Zookeeper acts as a distributed coordination service to maintain server state in the cluster. It maintains which servers are alive and available, and provides server failure notification. Zookeeper has the following functions:

- Acts as a distributed coordination service to maintain server state in the cluster.
- Maintains which servers are alive and available, and provides server failure notification.
How do they work together?
META Table

META table is an HBase table that keeps a list of all regions in the system.
First Read or Writes
Log Structured Merge Tree Approach

Client issues put request

Data is written on WAL

Data is placed on Memstore

Ack returns to client
HBase Region Flush

Region Server

Region

Memstore  Memstore

HDFS Data Node

WAL

All memstores in region flushed to new HFiles on disk

HFile: sorted list of key → values on disk

Saves the last written sequence number so the system knows what was persisted so far.
Read Merge

1. First the scanner looks for the Row KeyValues in the Block cache
2. Next the scanner looks in the MemStore
3. If all row cells not in MemStore or blockCache, look in HFiles
Minor Compaction
Major Compaction
Load Balancing

If a region server is too busy...

- HMaster moves region to another server
- For load balancing

Data not local

HFile

Region 1

Key | colB | colC
---|------|------
val | val  | val  

Region 2

Key | colB | colC
---|------|------
val | val  | Val

Data will not be local on HDFS
- Until next major compaction
Crash Recovery

- Heartbeat failure
- ZooKeeper
- No heartbeat
- HMaster

- Starts recovery process
- Reassigns regions
- Splits WAL for regions
- Regions Replay WAL

Region Server
- Memstore
- Memstore
- HFile
- HFile

HDFS Data Node

Region Server
- Memstore
- Memstore
- HFile
- HFile

HDFS Data Node
Accessing hbase

- Hbase Shell
- JAVA API
- Apache Thrift – software framework allows other languages to access Hbase over thrift by connecting to thrift server that interfaces with the java client
Hbase api

Types of access:

- Get: Gets a row’s data based on the row key.
- Put: Inserts a row with data based on the row key.
- Scan: Finding all matching rows based on the row key.
- Delete: delete a specific cell in a table.
Creating a Table using HBase Shell

Create ‘<tablename>’, ‘<columnfamily>’

Create ‘emp’, ’personal data’, ’professional data’

<table>
<thead>
<tr>
<th>Row key</th>
<th>Personal data</th>
<th>Professional data</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Verification**: using list command

List

Table
emp
Inserting Data using HBase Shell

Put ‘<table_name>’, ‘row1’, ‘<column family:colname>’, ‘value’

<table>
<thead>
<tr>
<th>Row key</th>
<th>personal data</th>
<th>professional data</th>
</tr>
</thead>
<tbody>
<tr>
<td>empid</td>
<td>name</td>
<td>city</td>
</tr>
<tr>
<td>1</td>
<td>raju</td>
<td>hyderabad</td>
</tr>
<tr>
<td>2</td>
<td>ravi</td>
<td>chennai</td>
</tr>
<tr>
<td>3</td>
<td>rajesh</td>
<td>delhi</td>
</tr>
</tbody>
</table>

Put ‘emp’, ‘1’, ‘personal data:name’, ‘raju’
Updating Data using HBase Shell

Put ‘<table_name>’, ‘row1’, ‘<column family:colname>’, ’new value’

Put ‘emp’, ’1’, ‘personal data:city’, ’delhi’
Reading Data using HBase Shell

get‘<table_name>’ , ‘row1’
get‘emp’ , ‘1’
Output : All the information related to row1

Reading a Specific Column

get‘<table_name>’ , ‘rowid’, {COLUMN=‘columnfamily:columnname’}
get‘emp’ , ‘row1’, {COLUMN=‘personal:name’}
Output : raju

Scan --- also used to view Hbase table  scan ‘<tablename>’
Java API

Step 1: **Instantiate the Configuration Class** create()
   
   ```java
   Configuration conf = HbaseConfiguration.create();
   ```

Step 2: **Instantiate the HTable Class**
   
   ```java
   Htable htable = new Htable(conf, tableName);
   ```

**Instantiate the Get Class** get() method
   
   ```java
   Get get = new Get(toBytes("row1"));
   ```

**Read the Data** add() method
   
   ```java
   get.addfamily(personal data)
   get.addfamily(personal data, name)
   ```
HBase - Security

- **GRANT**
  
  ```
  grant <user> <permissions>
  grant ‘adb’ ‘RWXCA’
  ```

- **REVOKE**
  
  ```
  revoke <user>
  ```

- **USER_PERMISSION**
  
  ```
  user_permission <tablename>
  ```

R-Read Privilege
W-write privilege
X-execute privilege
C-create privilege
A-admin privilege
<table>
<thead>
<tr>
<th></th>
<th>RDBMS</th>
<th>Hbase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Layout</td>
<td>Row Oriented</td>
<td>Column Oriented</td>
</tr>
<tr>
<td>Transactions</td>
<td>Multi Row ACID</td>
<td>Single row Only</td>
</tr>
<tr>
<td>Query Language</td>
<td>SQL</td>
<td>Get/put/scan</td>
</tr>
<tr>
<td>Read/Write Throughput</td>
<td>1000s queries/second</td>
<td>Millions of queries/second</td>
</tr>
<tr>
<td>Limits</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
HBase is growing!!!
Ranking of Wide Column Stores

© September 2016, DB-Engines.com
When to use Hbase?

While dealing with large amounts of data.

Not a replacement for RDBMS.

Having a use case that fits.

If you’re ready for a lot of monitoring.
Why did Facebook use Hbase?

- Had over 350 million users
- Over 15 billion person to person messages per month
- To integrate email, IM, SMS, text messages.
- Hbase beat Apache Cassandra & MySQL.
- Hbase uses HDFS
- Haystack
- Zookeeper
- A major contributor too: Hydrabase
Evolution of Hbase: HydraBase

- To improve reliability
- It offers improved failover time
Why did TempoIQ ditch Hbase?

- Wrong use case
- Hbase as storage for sensor analytics data without proper configuration

Their Justification:
- Isolation
- Performance Variability – Compaction and Garbage collection
- Recovery scenarios

Piece of advice:
- Life could get a lot easier if you configure Hbase correctly from the beginning.
Some other use Cases:

- Runs 38 different hbase clusters
- Some clusters doing 5 million ops/sec
- To handle large number of queries per second
- Low read latency
- Wide Schema

- Hadoop Mapreduce + MySQL 😞
- Hbase 😊
- Wide Schema

- Started Customer profiling
- Hbase to the rescue
“HBase has earned a reputation for being rock-solid when properly stood up. In Cloudera’s regular support reviews, we can see that HBase has continually reduced the number of tickets per deployed cluster. In fact, in the past year we’ve greatly increased the number of HBase deploys with no proportional increase in support incidents.”

----Jonathan Hseieh, Tech Lead and Manager of the Apache HBase Team, Cloudera

“Our developers love working with HBase, since it uses really cool technology and involves working with “Big Data”. We’re working with something that most people can’t imagine or never get the chance to work with. This is the leading edge of data storage technology, and we really feel like we’re inventing the future in a lot of ways.”

----Ryan Rawson, System Architect, StumbleUpon
Any questions do you have?