# *Cloud Computing*

- PaaS Techniques
  - Database

# Agenda

- Overview
  - Hadoop & Google
- PaaS Techniques
  - File System
    - GFS, HDFS
  - Programming Model
    - MapReduce, Pregel
  - Storage System for Structured Data
    - Bigtable, Hbase

Database Overview
Relational Database (SQL)
Non-relational Database Introduction (NOSQL/NOREL)
Google Bigtable
Hadoop (Hbase)

# *STORAGE SYSTEM FOR STRUCTURED DATA*

# Unstructured Data

- Data can be of any type
  - Not necessarily follow any format or sequence
  - Not follow any rules, so is not predictable
- Two Categories
  - Bitmap Objects
    - Inherently non-language based, such as image, video or audio files
  - Textual Objects
    - Based on a written or printed language, such as Microsoft Word documents, e-mails or Microsoft Excel spreadsheets

# Structured Data

- Data is organized in semantic chunks (entities)
- Similar entities are grouped together (relations or classes)
- Entities in the same group have the same descriptions (attributes)
- Descriptions for all entities in a group (schema) , e.g., Flight(f-num, date, time, price)
  - The same defined format
  - A predefined length
  - All present
  - The same order

| First Name | Last Name | Address | City | Age |
|---|---|---|---|---|
| Mickey | Mouse | 123 Fantasy Way | Anaheim | 73 |
| Bat | Man | 321 Cavern Ave | Gotham | 54 |
| Wonder | Woman | 987 Truth Way | Paradise | 39 |
| Donald | Duck | 555 Quack Street | Mallard | 65 |
| Bugs | Bunny | 567 Carrot Street | Rascal | 58 |
| Wiley | Coyote | 999 Acme Way | Canyon | 61 |
| Cat | Woman | 234 Purrfect Street | Hairball | 32 |
| Tweety | Bird | 543 | Itotitaw | 28 |

# *Semi-Structured Data*

- Organized in semantic entities
- Similar entities are grouped together
- Entities in same group may not have same attributes
  - Order of attributes not necessarily important
  - Not all attributes are required
  - Size of same attributes in a group may different
  - Type of same attributes in a group may different

# *Example of Semi-Structured Data*

- Name: Computing Cloud
- Phone_home: 035715131

- Name: TA Cloud
- Phone_cell: 0938383838
- Email: cloudTA@gmail.com

- Name: Student Cloud
- Email: hiCloud@hotmail.com

# *Database, and Database Management System*

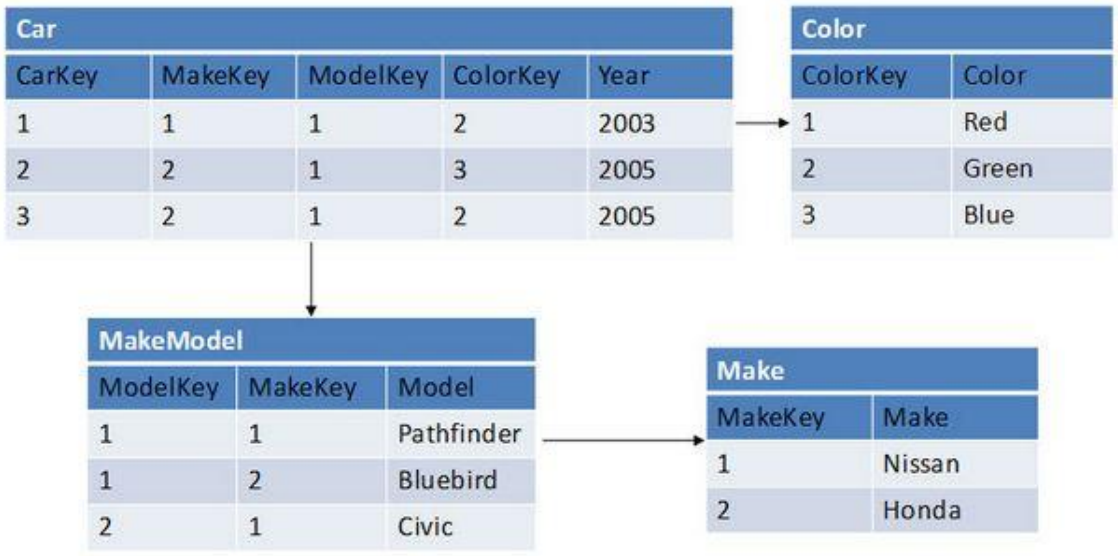
- Database
  - A system intended to organize, store, and retrieve large amounts of data easily
- Database management system (DBMS)
  - Consists of software that operates databases
  - Provides storage, access, security, backup and other facilities

Database Overview
Relational Database (SQL)
Non-relational Database Introduction (NOSQL/NOREL)
Google Bigtable
Hadoop (Hbase)

# *STORAGE SYSTEM FOR STRUCTURED DATA*

# Relational Database(1/4)

- Essentially a group of tables (entities)
  - Tables are made up of columns and rows (tuples)
  - Tables have constraints, and relationships defined between them
- Facilitated through Relational Database Management Systems (RDBMS)



**Car**

| CarKey | MakeKey | ModelKey | ColorKey | Year |
|---|---|---|---|---|
| 1 | 1 | 1 | 2 | 2003 |
| 2 | 2 | 1 | 3 | 2005 |
| 3 | 2 | 1 | 2 | 2005 |

**Color**

| ColorKey | Color |
|---|---|
| 1 | Red |
| 2 | Green |
| 3 | Blue |

**MakeModel**

| ModelKey | MakeKey | Model |
|---|---|---|
| 1 | 1 | Pathfinder |
| 1 | 2 | Bluebird |
| 2 | 1 | Civic |

**Make**

| MakeKey | Make |
|---|---|
| 1 | Nissan |
| 2 | Honda |

Example of a Typical Relational Data Model

# Relational Database(2/4)

- Multiple tables being accessed in a single query are "joined" together
- Normalization is a data-structuring model used with relational databases
  - Ensures data consistency
  - Removes data duplication
- Almost all database systems we use today are RDBMS
  - Oracle
  - SQL Server
  - MySQL
  - DB2
  - …

# Relational Database(3/4)

- Advantages
  - Simplicity
  - Robustness
  - Flexibility
  - Performance
  - Scalability
  - Compatibility in managing generic data
- However,
  - To offer all of these, relational databases have to be incredibly complex internally

# Relational Database(4/4)

- It’s a problem in a different situation
  - Large-scale Internet application services
    - Their scalability requirements can, first of all, change very quickly and, secondly, grow very large.
    - Relational databases scale well, but usually only when that scaling happens on a single server node.
    - This is when the complexity of relational databases starts to rub against their potential to scale.
  - Cloud services to be viable
    - A cloud platform without a scalable data store is not much of a platform at all

Database Overview
Relational Database (SQL)
Non-relational Database Introduction (NOSQL/NoREL)
Google Bigtable
Hadoop (Hbase)

# *STORAGE SYSTEM FOR STRUCTURED DATA*

NOSQL Overview

Related Theorem

Distributed Database System

# NON-RELATIONAL DATABASE INTRODUCTION

# What is NOSQL

- Not Only SQL
  - A term used to designate database management systems
  - Differ from classic relational database management systems
  - The most common interpretation of "NoSQL" is “Non-relational“ (NoREL, not widely used)
- Some NOSQL examples
  - Google Bigtable
    - Open Source - Apache Hbase
  - Amazon Dynamo
  - Apache Cassandra
- Emphasizes the advantages of Key/Value Stores, Document Databases, and Graph Databases

# Key/Value Database(1/4)

- No official name yet exists, so you may see it referred to
  - Document-oriented
  - Internet-facing
  - Attribute-oriented
  - Distributed database (this can be relational also)
  - Sharded sorted arrays
  - Distributed hash table
  - Key/value database (datastore)

# Key/Value Database(2/4)

- No Entity Joins
  - Key/value databases are item-oriented
  - All relevant data relating to an item are stored within that item
  - A domain (a table) can contain vastly different items
  - This model allows a single item to contain all relevant data
    - Improves scalability by eliminating the need to join data from multiple tables
    - With a relational database, such data needs to be joined to be able to regroup relevant attributes

# Key/Value Database(3/4)

- Advantages of key/value DBs to relational DBs
  - Suitability for Clouds
    - Key/Value DBs are simple and thus scale much better than relational databases
    - Provides a relatively cheap data store platform with massive potential to scale
  - More Natural Fit with Code
    - Relational data models and Application Code Object Models are typically built differently
    - Key/value databases retain data in a structure that maps more directly to object classes used in the underlying application code

# Key/Value Database(4/4)

- Disadvantages of key/value DBs to relational DBs
  - Data integrity issues
    - Data that violate integrity constraints cannot physically be entered into the relational DB
    - In a key/value DB, the responsibility for ensuring data integrity falls entirely to the application
  - Application-dependent
    - Relational DBs modeling process creates a logical structure that reflects the data it is to contain, rather than reflecting the structure of the application
    - Key/value DBs can try replacing the relational data modeling exercise with a class modeling exercise
  - Incompatibility

NOSQL Overview

Related Theorem

Distributed Database System

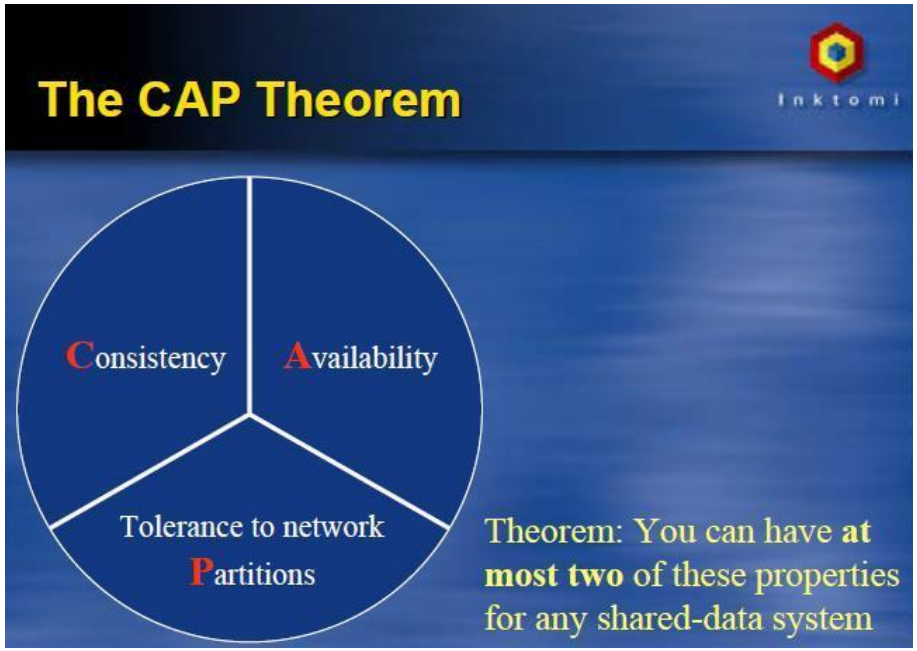
# *NON-RELATIONAL DATABASE INTRODUCTION*

# CAP Theorem(1/2)

- When designing distributed data storage systems, it's very common to invoke the CAP Theorem
  - Consistency, Availability, Partition-tolerance
- Consistency
  - The goal is to allow multisite transactions to have the familiar all-or-nothing semantics.
- Availability
  - When a failure occurs, the system should keep going, switching over to a replica, if required.
- Partition-tolerance
  - If there is a network failure that splits the processing nodes into two groups that cannot talk to each other, then the goal would be to allow processing to continue in both subgroups.

# CAP Theorem(2/2)

- Consistency, availability, partition tolerance. Pick two.
  - If you have a partition in your network, you lose either consistency (because you allow updates to both sides of the partition) or you lose availability (because you detect the error and shutdown the system until the error condition is resolved).



NOSQL Overview

Related Theorem

Distributed Database System

# NON-RELATIONAL DATABASE INTRODUCTION

# Introduction

- Distributed database system = distributed database + distributed DBMS
  - Distributed database
    - a collection of multiple inter-correlated databases distributed over a computer network
  - Distributed DBMS
    - manage a distributed database and make the distribution transparent to users
- Consists of
  - query nodes: user interface routines
  - data nodes: data storage
- Loosely coupled: connected with network, each node has its own storage / processor / operating system

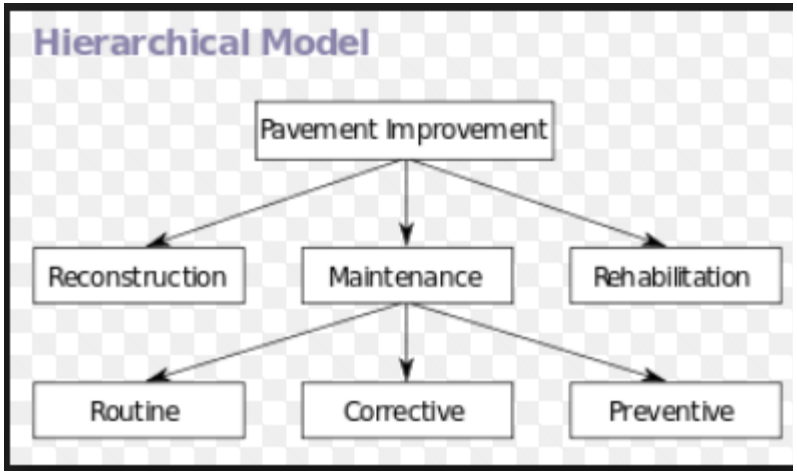
# System Architectures

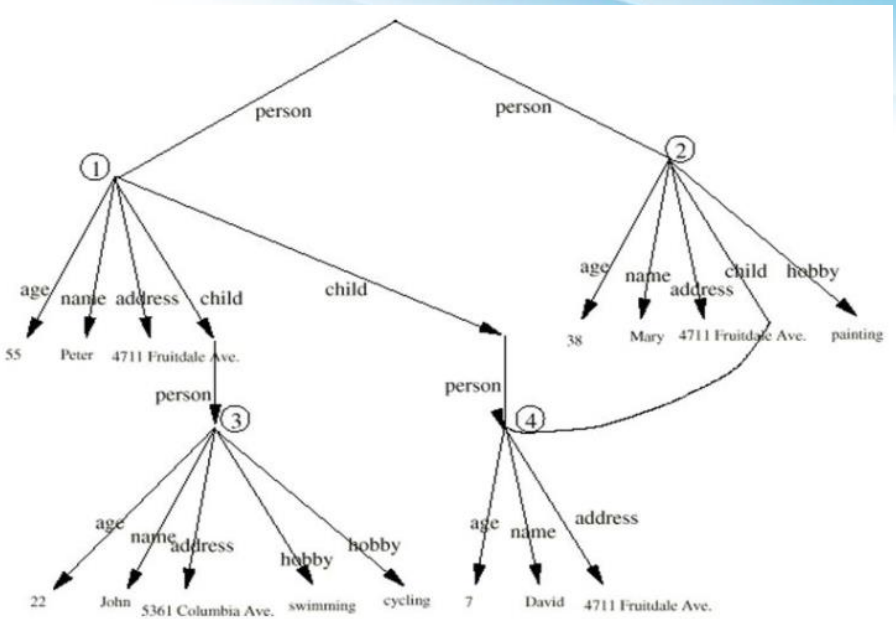
- Centralized
  - one host for everything, multi-processor is possible but a transaction gets only one processor
- Parallel
  - a transaction may be processed by multiple processors
- Client-Server
  - database stored on one server host for multiple clients, centrally managed
- Distributed
  - database stored on multiple hosts, transparent to clients
- Peer to Peer
  - each node is a client and a server; requires sophisticated protocols, still in development

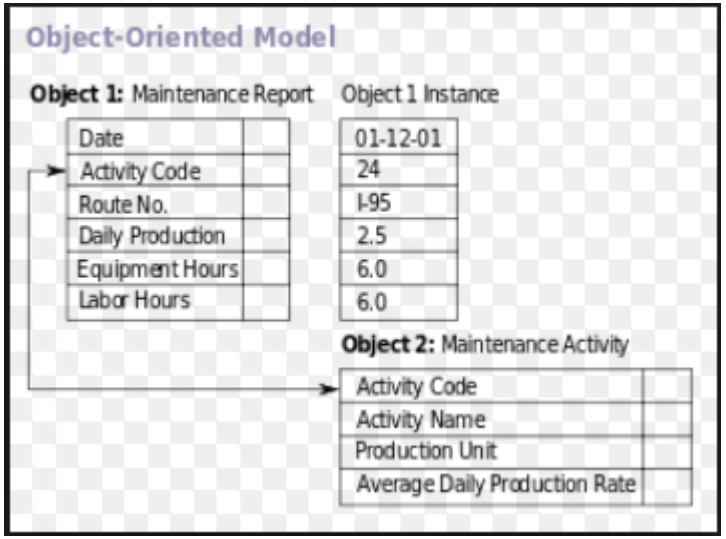
# Data Models

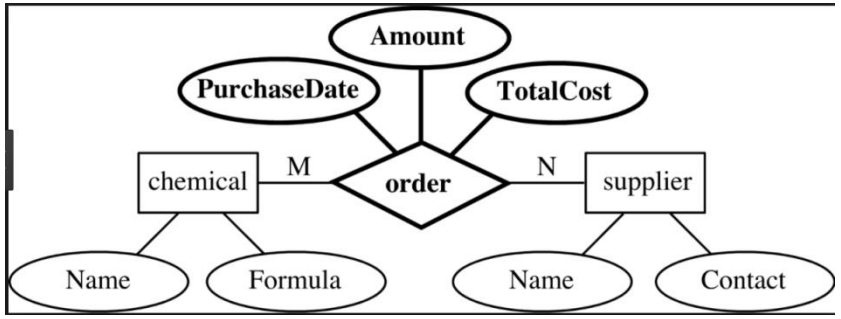
- Hierarchical Model
  - Data organized in a tree namespace
- Network Model
  - Like Hierarchical Model, but a data may have multiple parents
- Entity-Relationship Model
  - Data are organized in entities which can have relationships among them
- Object-Oriented Model
  - Database capability in an object-oriented language
- Semi-structured Model
  - Schema is contained in data (often associated with “self-describing” and “XML”)

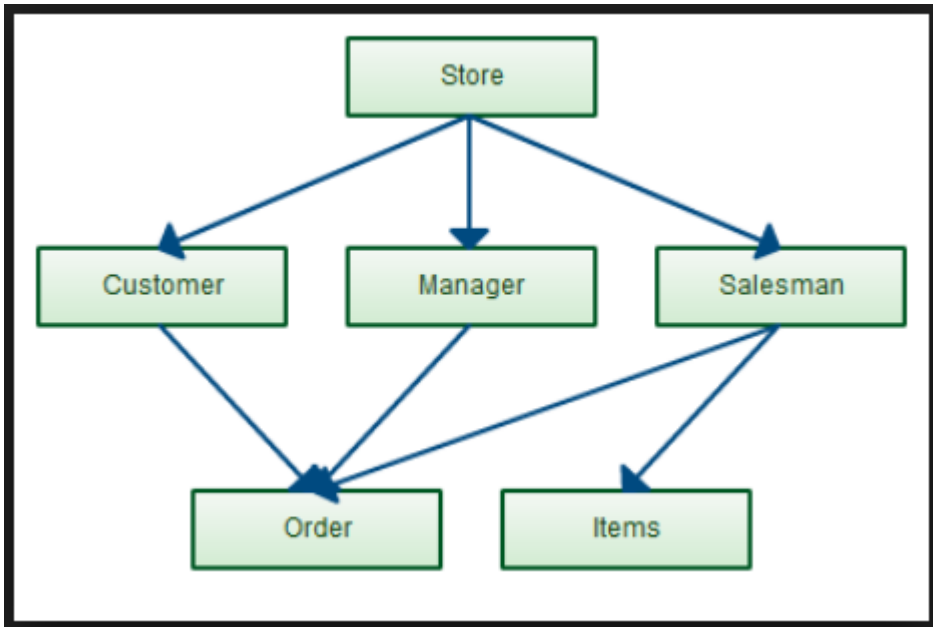
# Hierarchical Model, Network Model, Entity-Relationship Model, Object-Oriented Model, Semi-structured Model




Store
Customer
Manager
Salesman
Order
Items

# Data distribution

- Data is physically distributed among data nodes
  - Fragmentation: divide data onto data nodes
  - Replication: copy data among data nodes
- Fragmentation enables placing data close to clients
  - May reduce size of data involved
  - May reduce transmission cost
- Replication
  - Preferable when the same data is accessed from applications that run at multiple nodes
  - May be more cost-effective to duplicate data at multiple nodes rather than continuously moving it between them
- Many different schemes of fragmentation and replication

# Fragmentation

- Horizontal fragmentation
  - split by rows based on a fragmentation predicate
- Vertical fragmentation
  - split by columns based on attributes
- Also called “partition” in some literature

| Last name | First name | Department | ID |
|---|---|---|---|
| Chang | Three | Computer Science | X12045 |
| Lee | Four | Law | Y34098 |
| Chang | Frank | Medicine | Z99441 |
| Wang | Andy | Medicine | S94717 |

# Properties

- Concurrency control
  - Make sure the distributed database is in a consistent state after a transaction
- Reliability protocols
  - Make sure termination of transactions in the face of failures (system failure, storage failure, lost message, network partition, etc)
- One copy equivalence
  - The same data item in all replicas must be the same

# Query Optimization

- Looking for the best execution strategy for a given query
- Typically done in 4 steps
  - query decomposition: translate query to relational algebra (for relational database) and analyze/simplify it
  - data localization: decide which fragments are involved and generate local queries to fragments
  - global optimization: finding the best execution strategy of queries and messages to fragments
  - local optimization: optimize the query at a node for a fragment
- Sophisticated topic

Database Overview
Relational Database (SQL)
Non-relational Database Introduction (NOSQL/NoREL)
Google Bigtable
Hadoop (Hbase)

# *STORAGE SYSTEM FOR STRUCTURED DATA*

How to manage structured data in a distributed storage system that is designed to scale to a very large size …

***Bigtable***

Database Overview
Relational Database (SQL)
Non-relational Database Introduction (NOSQL/NoREL)
Google Bigtable
Hadoop Hbase

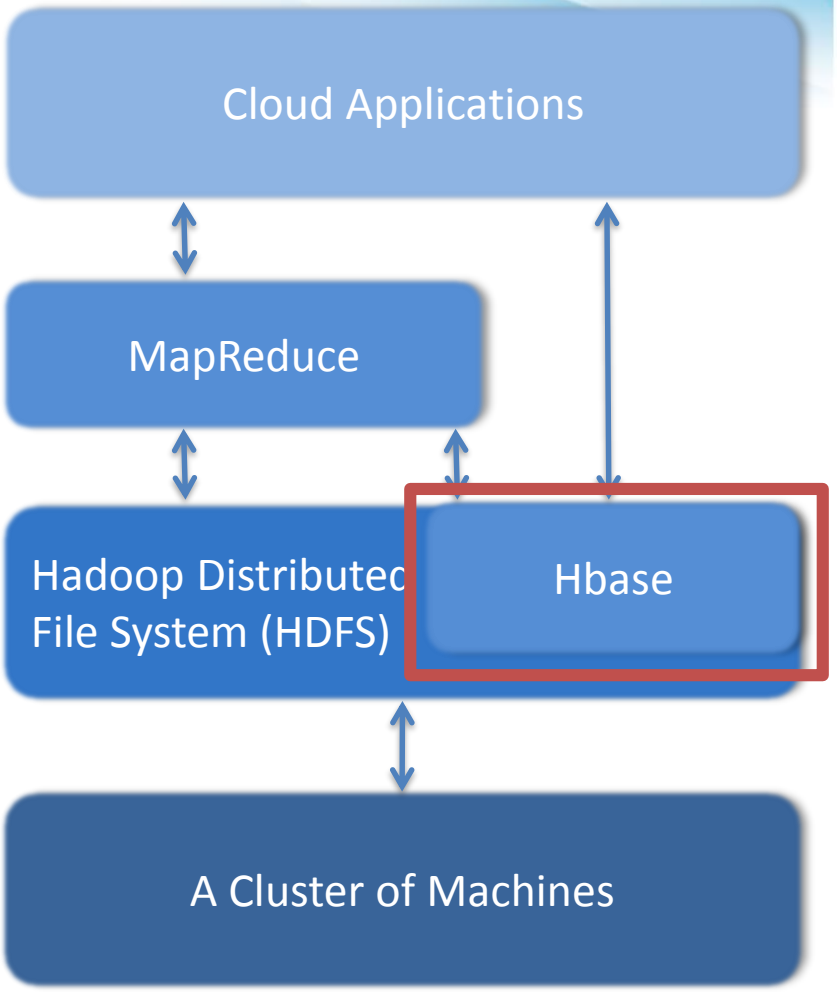
# *STORAGE SYSTEM FOR STRUCTURED DATA*

# *Hbase*

- Overview
- Architecture
- Data Model
- Different from Bigtable

# What's Hbase

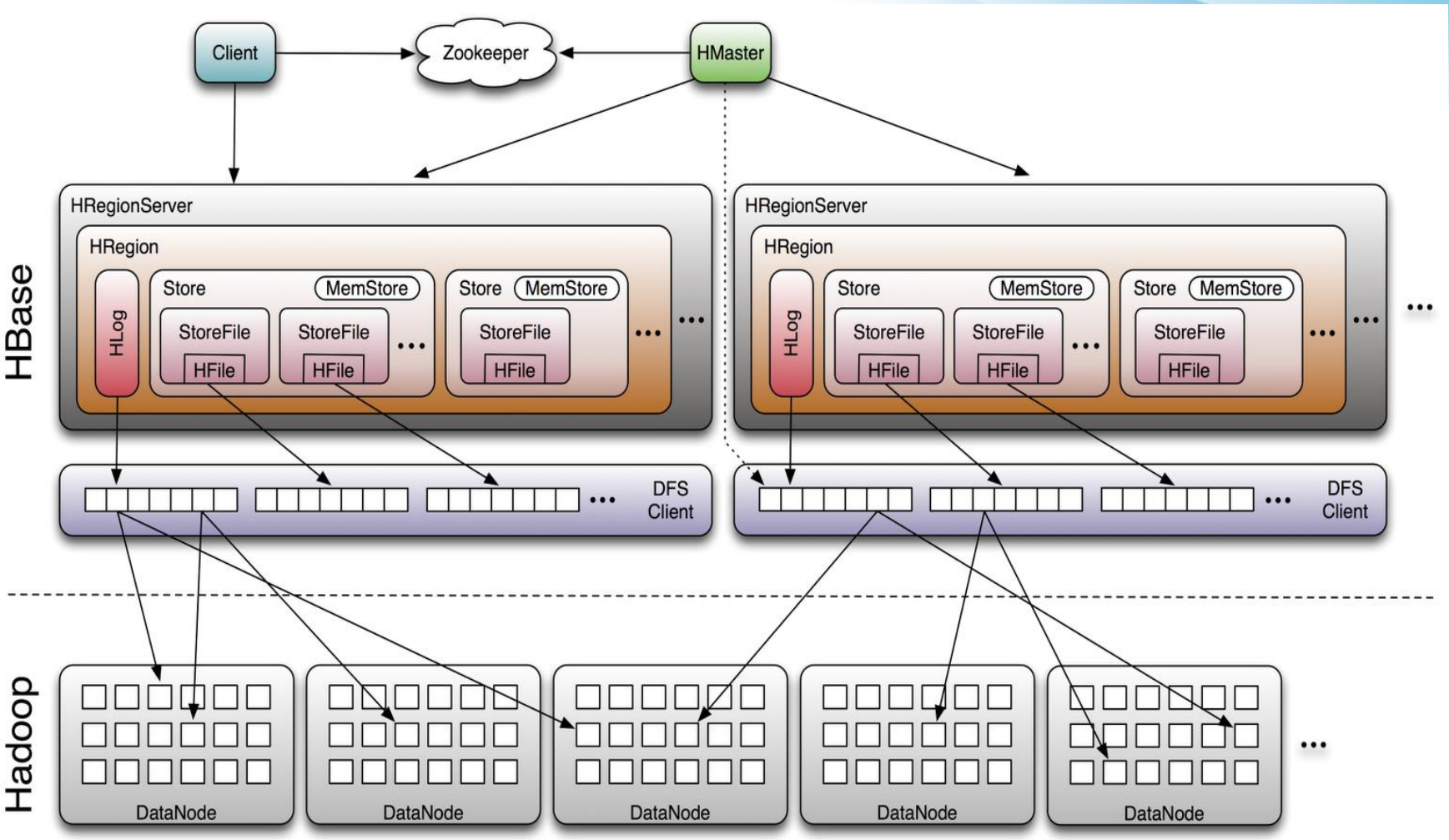
- Distributed Database modeled on column-oriented rows
- Tables of column-oriented rows
- Scalable data store (scales horizontally)
- Apache Hadoop subproject since 2008



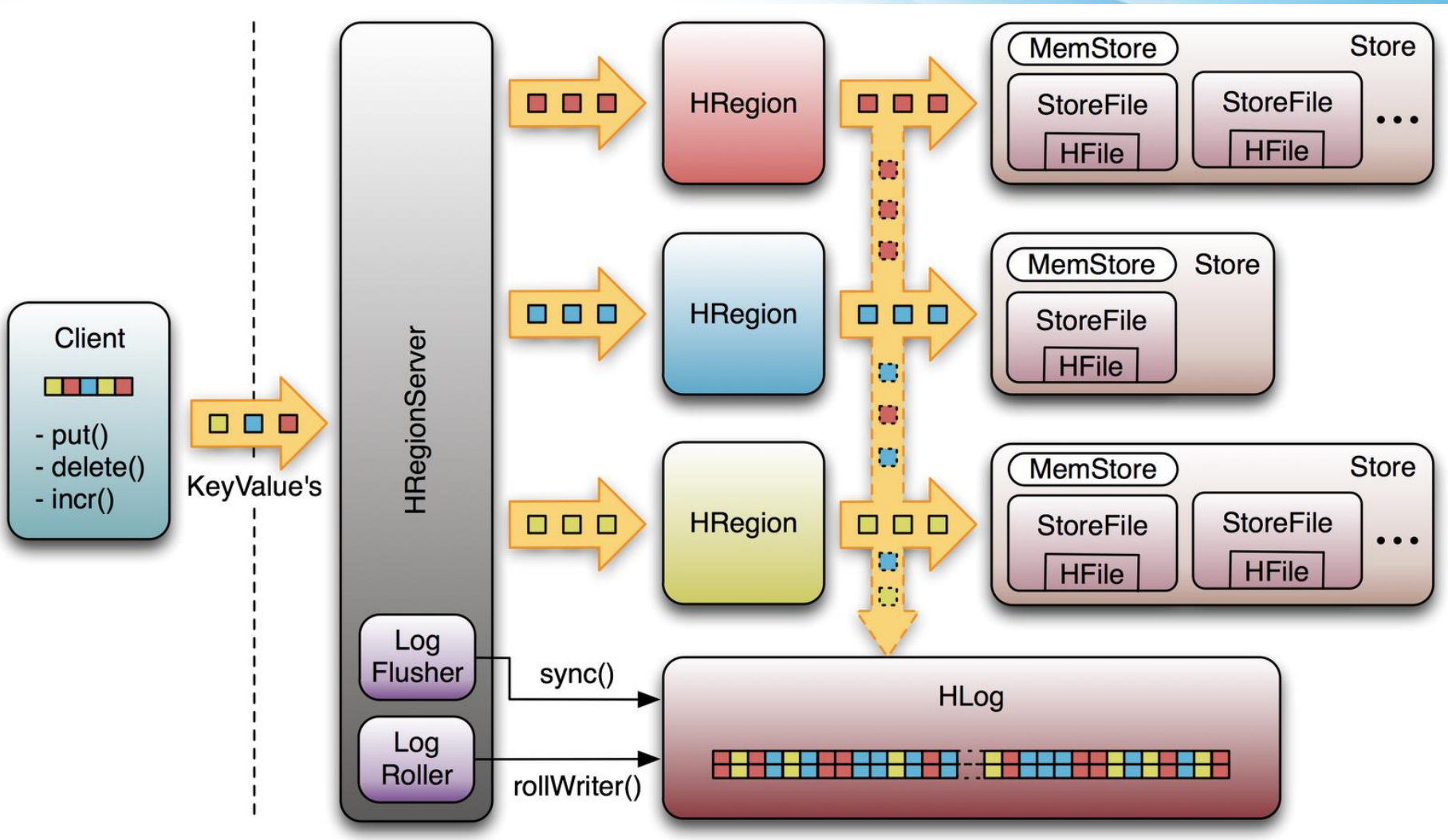
# *Hbase*

- Overview
- Architecture
- Data Model
- Different from Bigtable

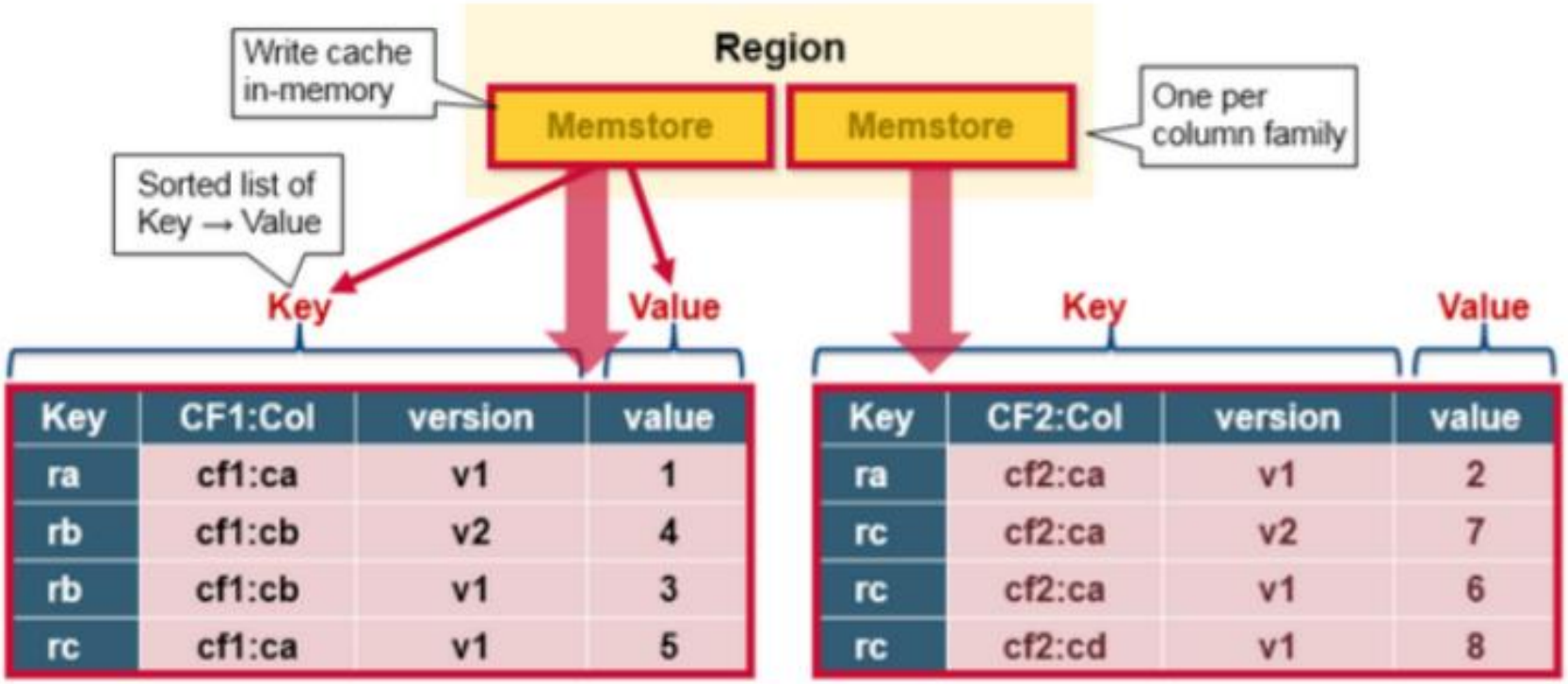
# Hbase Architecture



# How does Hbase work?



# How does Hbase work?




| Key | CF1:Col | version | value |
|---|---|---|---|
| ra | cf1:ca | v1 | 1 |
| rb | cf1:cb | v2 | 4 |
| rb | cf1:cb | v1 | 3 |
| rc | cf1:ca | v1 | 5 |

| Key | CF2:Col | version | value |
|---|---|---|---|
| ra | cf2:ca | v1 | 2 |
| rc | cf2:ca | v2 | 7 |
| rc | cf2:ca | v1 | 6 |
| rc | cf2:cd | v1 | 8 |

# Roles in Hbase(1/2)

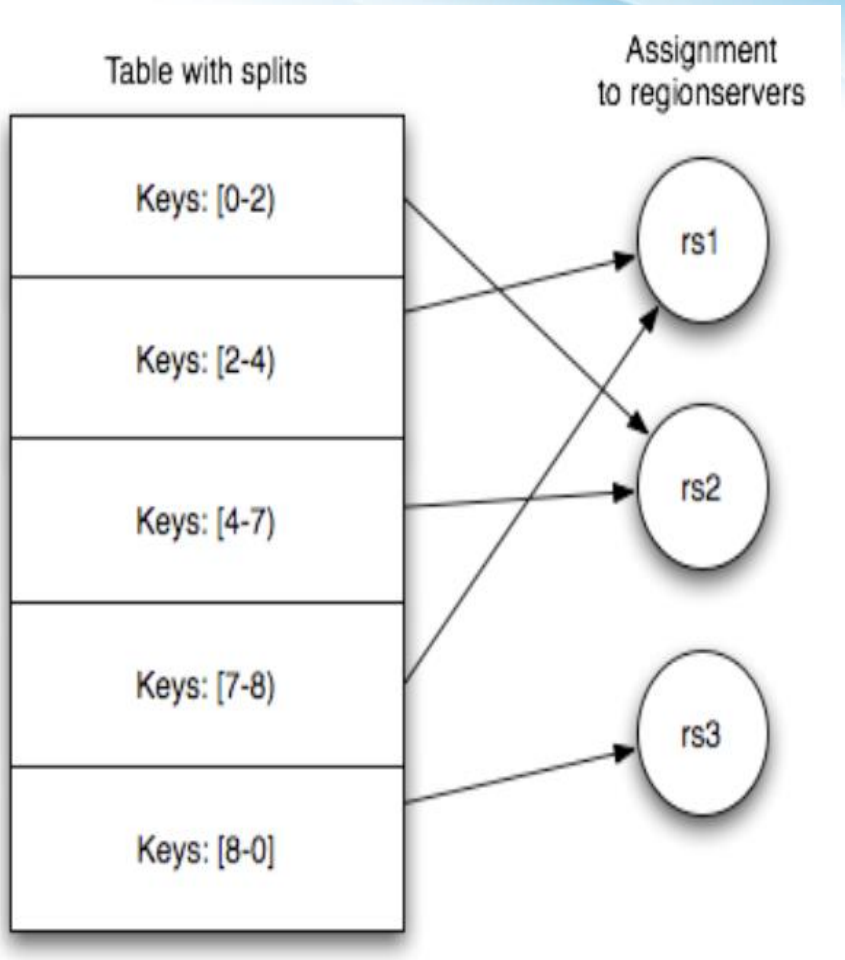
- Master
  - Cluster initialization
  - Assigning/unassigning regions to/from Regionservers (unassigning is for load balance)
  - Monitor the health and load of each Regionserver
  - Changes to the table schema and handling table administrative functions
- Regionservers
  - Serving Regions assigned to Regionserver
  - Handling client read and write requests
  - Flushing cache to HDFS
  - Keeping Hlog
  - Compactions
  - Region Splits

# Roles in Hbase(2/2)

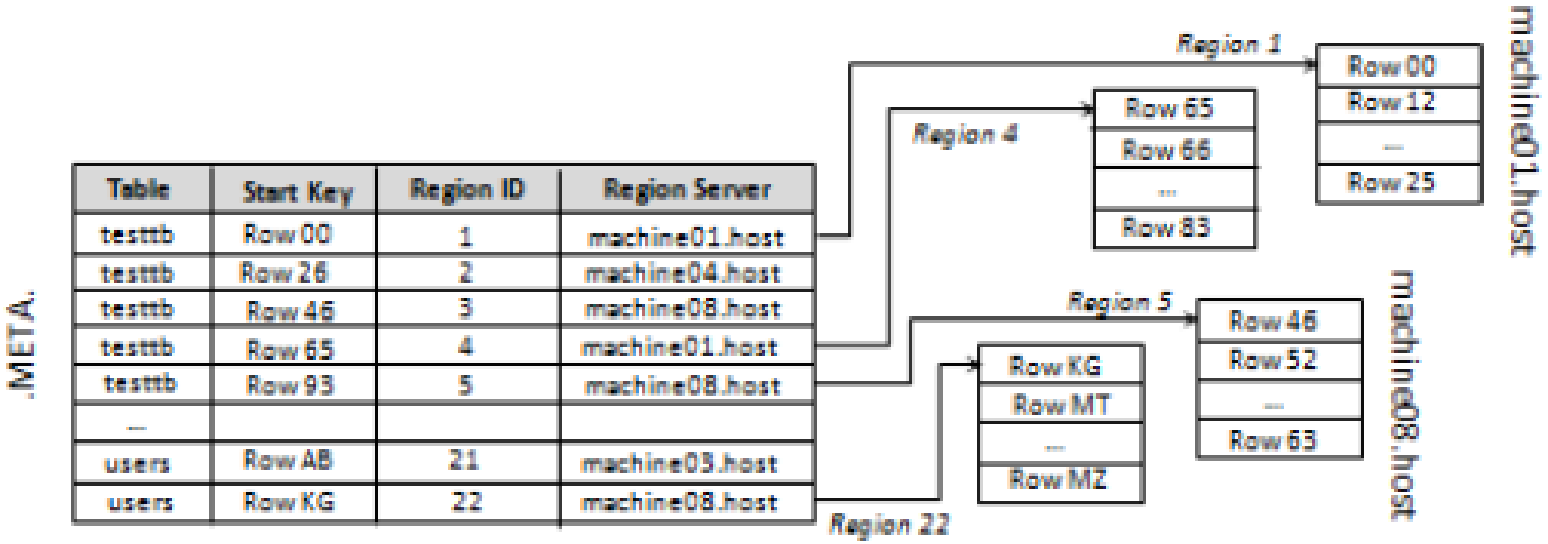
- Zookeeper
  - Master election and recovery
  - Store membership info
  - Locate -ROOT- region
- HDFS
  - All persistence Hbase storage is on HDFS (HFile)
  - HDFS reliability and performance are key to Hbase reliability and performance

# Table & Region

- Rows stored in byte-lexicographic sorted order
- Table dynamically split into “regions”
- Each region contains values [startKey, endKey)
- Regions hosted on a regionserver



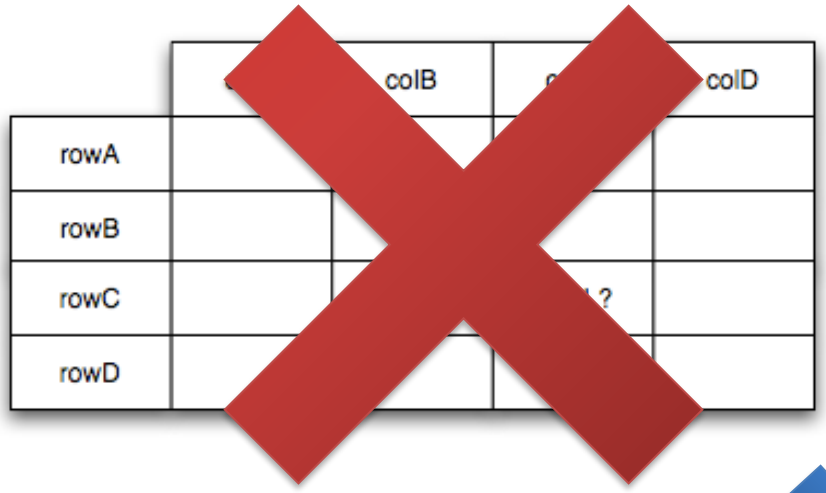
# META in Zookeeper

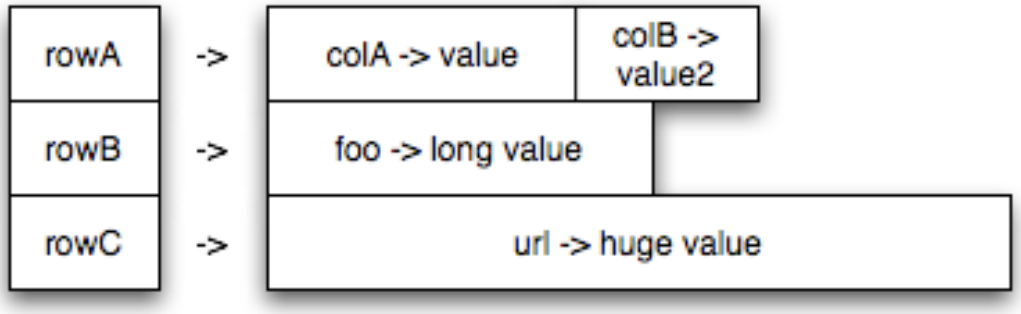


# *Hbase*

- Overview
- Architecture
- Data Model
- Different from Bigtable

# Data Model


colB
colD
rowA
rowB
rowC
rowD


rowA
->
colA -> value
colB -> value2
rowB
->
foo -> long value
rowC
->
url -> huge value

# Data Model (cont.)

- Data are stored in tables of rows and columns
  - Columns are grouped into column families
    - A column name has the form “<family>:<label>”
    - Table consists of 1+ “column families”
    - Column family is unit of performance tuning
  - Rows are sorted by row key, the table's primary key
- Cells are ”versioned”
  - Each row id + column – stored with timestamp
    - Hbase stores multiple versions
    - (table, row, <family>:<label>, timestamp) ⟶ value
  - Can be useful to recover data due to bugs
  - Use to detect write conflicts/collisions

# Example

| Row Key | Time Stamp | Column "contents:" | Column "anchor:" | | Column "mime:" |
|---|---|---|---|---|---|
| "com.cnn.www" | t9 | | "anchor:cnnsi.com" | "CNN" | |
| | t8 | | "anchor:my.look.ca" | "CNN.com" | |
| | t6 | "<html>..." | | | "text/html" |
| | t5 | "<html>..." | | | |
| | t3 | "<html>..." | | | |

## Conceptual View

| Row Key | Time Stamp | Column "contents:" |
|---|---|---|
| "com.cnn.www" | t6 | "<html>..." |
| | t5 | "<html>..." |
| | t3 | "<html>..." |

| Row Key | Time Stamp | Column "mime:" |
|---|---|---|
| "com.cnn.www" | t6 | "text/html" |

## Physical Storage View

| Row Key | Time Stamp | Column "anchor:" | |
|---|---|---|---|
| "com.cnn.www" | t9 | "anchor:cnnsi.com" | "CNN" |
| | t8 | "anchor:my.look.ca" | "CNN.com" |

# Hbase w/ Hadoop

- Easy integration with Hadoop MapReduce(MR)
- Look from HDFS (HDFS Requirements Matrix)

| Requirement | MR | HBase |
|---|---|---|
| Scalable storage | ✓ | ✓☺ |
| System fault tolerance | ✓ | ✓☺ |
| Large streaming writes | ✓ | ✓☺ |
| Large streaming reads | ✓ | ✓☺ |
| Small random reads | - | ✓☹ |
| Single client fault tolerance | - | ✓☹ |
| Durable record appends | - | ✓☹ |

# Summary

- Scalability
  - Provide scale-out storage capability of handling very large amounts of data.
- Availability
  - Provide the scheme of data replication based on a reliable google file system to support high availability for data store.
- Manageability
  - Provide mechanism for the system to automatically monitor itself and manage the massive data transparently for users.
- Performance
  - High sustained bandwidth is more important than low latency.

# References


- Chang, F., et al. “Bigtable: A distributed storage system for structured data.” In OSDI (2006).
- Hbase.
  - http://hbase.apache.org/
- NCHC Cloud Computing Research Group.
  - http://trac.nchc.org.tw/cloud
- NTU course- Cloud Computing and Mobile Platforms.
  - http://ntucsiecloud98.appspot.com/course_information
- Wiki.
  - http://en.wikipedia.org/wiki/Database#Database_management_systems

- How HBase Works
- https://www.youtube.com/watch?v=lSrNUyMR_Ek

- What is Hbase in Hadoop
- https://www.youtube.com/watch?v=VEmy3I5eq74

- What is HBase? How is it different from Hadoop? | HDFS and HBase Architecture
- https://www.youtube.com/watch?v=hs8QnQvwyCM