Big Data Analytics

Rasoul Karimi

Information Systems and Machine Learning Lab (ISMLL)
Institute of Computer Science
University of Hildesheim, Germany

Big Data Analytics

Outline

NoSQL Databases

HBase (Hadoop database)

Outline

NoSQL Databases

HBase (Hadoop database)

Distributed File System (DFS)

- A DFS (Hadoop, GFS,...) is the opposite of a virtual machine. A virtual machine chops one machine into multiple virtual machines. A distributed file system combines many virtual machines into one virtual machine.
- It includes a NoSQL database
- Relational databases are like sports cars, Hadoop is like a freight train.
- Relational database is for online transaction processing (OLTP), ACID transactions, etc. DSF is good for analyzing massive amounts of data, handling unstructured data, handling very complex analyses, etc.

Relational Databases

- Advantages of relational DB.
  - persistence
  - integration
  - transaction
  - reporting
- The disadvantage of relational databases is that one object in the program is split into many tables in the data base (impedance mismatch).

Object-Oriented Databases

- The motivation was to solve the impedance mismatch issue.
- It could not replace the relational data bases because the integration aspect of relational data bases was very powerful.
- Therefore, relational databases dominated the market for 20 years (1990-2010).

The Rise of Internet

- Solution 1: centralized solution ×
- Problems:
  - cost
  - limited
- Solution 2: Distributed solution √
- SQL does not work well for big distributed databases.
- NoSQL databases were innovated for big distributed databases.

Characteristics of NoSQL Databases

- Non-relational
- Cluster-friendly
- Open source
- Web-oriented
- Schema-less

What is missing in NoSQL Databases

- No joint operation
- No complex transaction
- No constrains

The objectives of NoSQL Databases

- horizontal scalability
- availability
- high performance

When to use NoSQL Databases

- Large amount of data
- Relationships are not important
- Dealing with growing list of elements
- The data is not structured or the structure is changing
- Prototype and fast applications
- No need for constraints or validation rules in data bases

When NOT to use NoSQL Databases

- Complex transactions
- Joint
- Constrains in data bases

NoSQL Databases

- Key-value
- Document
- Column
- Graph

Key–Value stores

- Key–Value stores use the associative array as their fundamental data model.
- In this model, data is represented as a collection of key–value pairs.
- The key–value model is one of the simplest non-trivial data models.

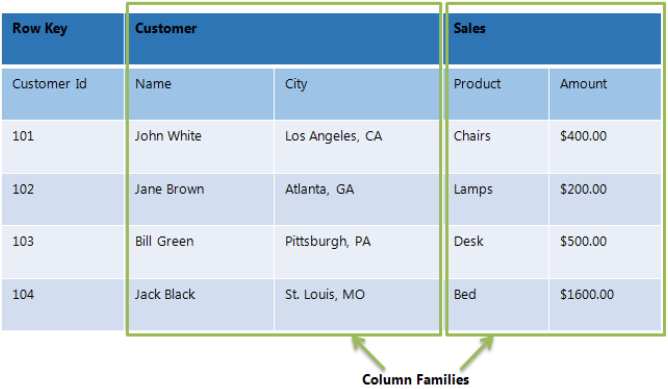
Example:

```
{
"Great Expectations": "John",
"Pride and Prejudice": "Alice",
"Wuthering Heights": "Alice"
}
```

Document Databases

- Documents are addressed in the database via a unique key that represents that document.
- Documents can be retrieved by their
  - key
  - content
- Documents are organized through
  - Collections
  - Tags
  - Non-visible Metadata
  - Directory hierarchies
  - Buckets

Elements of Column Databases



Row Key	Customer		Sales	
Customer Id	Name	City	Product	Amount
101	John White	Los Angeles, CA	Chairs	$400.00
102	Jane Brown	Atlanta, GA	Lamps	$200.00
103	Bill Green	Pittsburgh, PA	Desk	$500.00
104	Jack Black	St. Louis, MO	Bed	$1600.00

Column Families

- Column
- Column Family
- KeySpace

Column

- Columns are a tuple (a key-value pair) consisting of three elements:
  - Unique name: Used to reference the column
  - Value: The content of the column.
  - Timestamp: The system timestamp used to determine the valid content.
- Example:

{
street: name: ”street”, value: ”1234 x street”, timestamp: 123456789,
city: name: ”city”, value: ”san francisco”, timestamp: 123456789,
zip: name: ”zip”, value: ”94107”, timestamp: 123456789,
}

Column Family

- A set of column sets that are about related data.
- It also includes a key-value pair, where the key is mapped to a value that is a column set.
- In analogy with relational databases, a standard column family is as a "table", each key-value pair being a "row".

KeySpace

- A keyspace is an object that holds together all column families in a design.
- It resembles to the schema concept in Relational database management systems
- Column Family 1:
  UserProfile = {
  Cassandra = { emailAddress:"casandra@apache.org" , age:"20"}
  TerryCho = { emailAddress:"terry.cho@apache.org" , gender:"male"}
  Cath = { emailAddress:"cath@apache.org" , age:"20", gender:"female", address:"Seoul"}
  }

KeySpace

- Column Family 2:
  Products = {
  book = { name:” introduction to Java” , year:”2001”}
  movie = { name:”The Lord of the Rings” , director:”Peter Jackson”}
  Shirt = { Brand:”lacoste ” , color:”blue”, price:”100”}
  }

KeySpace

$$< KeyspaceName = ''Company'' >$$
$$< KeysCachedFraction > 0.01 < /KeysCachedFraction >$$
$$< ColumnFamilyCompareWith = ''UTF8Type'' \; Name = ''UserProfiles'' / >$$
$$< ColumnFamilyCompareWith = ''UTF8Type'' \; Name = ''Products'' / >$$
$$< /Keyspace >$$

Aggregation in Column Databases

- As an example, a relational table could consist of the columns UID, first name, surname, birthdate, gender.
- Suppose that we are searching people who are male and were born between 1950 and 1960.
- In the relational database, all the table has to be read.
- In Column datasets, we only read the two columns.
- The Columns in a family are stored together in a low level storage file known.

Graph Database

- Key-value, Document, and Column databases are aggregate databases.
- Graph databases are not an aggregate database.
- Graph databases are used for the data that is not aggregated and there are many relationships
- Graph databases are schema-less like aggregate databases.
- Graph databases support transactions.

Summary of Databases

- Aggregate databases (Key-value, Document, and Column).
- Graph databases
- Relational Databases

ACID transactions

- In relational databases, ACID transactions guarantee that the database is consistent.
- Aggregate databases do not have ACID transactions.
- Graph databases have ACID transactions.
- In aggregate databases, time stamps are used to resolve the conflicts.

Big data and consistency

- Replication leads to more availability
  - More nodes are able to execute requests
  - If one nodes goes down, other nodes can still process requests
- But it also causes more problems in consistence
- Consistence or availability? which one is more important?

Example

- There is an international booking website with several servers all over the world.
- Two persons from different continents request a particular room of a hotel in the same time.
- Two servers communicate and in the end the room is assigned to one of the persons.

Example

- What about if the connection between the servers go down?
- If consistency is more important, both requests are rejected.
- If availability is more important, both requests are accepted.
- The choice between availability and consistency depends on the business rules of the application.

CAP theorem

It is impossible for a distributed computer system to simultaneously provide all three of the following guarantees:

- Consistency (all nodes see the same data at the same time)
- Availability (a guarantee that every request receives a response about whether it was successful or failed)
- Partition tolerance (the system continues to operate despite arbitrary message loss or failure of part of the system)

Consistency vs. Availability

- The choice is not binary, but a spectrum.
- In practice, the trade off is not always between consistency and availability or partitioning, but consistency and response time.

The reasons of going for NoSQL databases

- Big Data
- Easy and rapid development
- The importance of integration issue has deceased due to service-oriented programming.
- Analytics

Challenges of NoSQL databases

- Organizational change
- Immaturity
- Eventual consistency

SQL database will still play important role in databases at least for one decade.

Outline

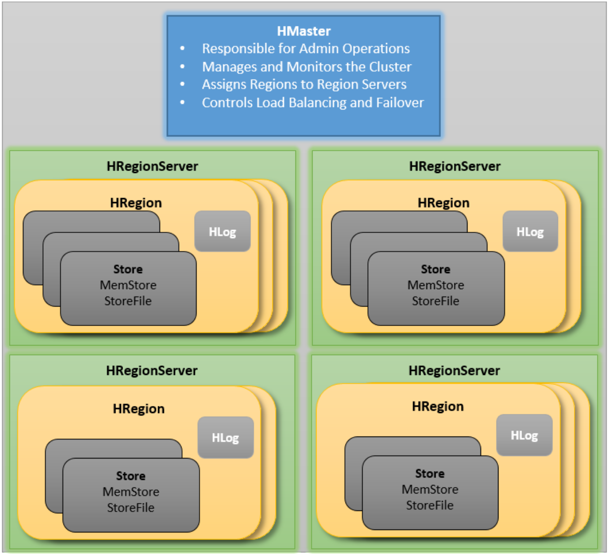
NoSQL Databases

HBase (Hadoop database)

HBase (Hadoop database)

- it is a distributed column-oriented database, which lays between HDFS and MapReduce.
- It is open-source implementation of Google's Big Table database.
- has some built-in features such as
  - Scalability
  - Versioning
  - Compression
  - Garbage collection
  - Fault tolerance

The architecture of HBase



HMaster

- Performing Administration
- Managing and Monitoring the Cluster
- Assigning Regions to the Region Servers
- Controlling the Load Balancing and Failover

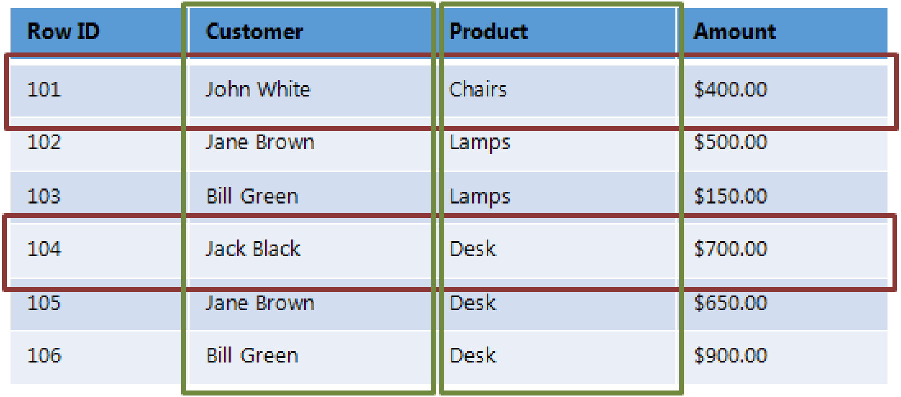
HRegionServer

- Hosting and managing Regions
- Splitting the Regions automatically
- Handling the read/write requests
- Communicating with the Clients directly

HRegionServer

- Each Region Server contains a Log (called HLog) and multiple Regions.
- Each Region in turn is made up of a MemStore and multiple StoreFiles (HFile).
- The data lives in these StoreFiles in the form of Column Families.
- The MemStore holds in-memory modifications to the Store (data).

Column vs. row-oriented database



Row ID	Customer	Product	Amount
101	John White	Chairs	$400.00
102	Jane Brown	Lamps	$500.00
103	Bill Green	Lamps	$150.00
104	Jack Black	Desk	$700.00
105	Jane Brown	Desk	$650.00
106	Bill Green	Desk	$900.00

Characteristics of Row-oriented database

- Data is stored and retrieved one row at a time and hence could read unnecessary data if only some of the data in a row is required.
- Easy to read and write records
- Well suited for OLTP systems
- Not efficient in performing operations applicable to the entire dataset and hence aggregation is an expensive operation
- Typical compression mechanisms provide less effective results than those on column-oriented data stores
- In the hard disk, each row is stored together.

Characteristics of column-oriented database

- Data is stored and retrieved in columns and hence can read only relevant data if only some data is required
- Read and Write are typically slower operations
- Well suited for OLAP systems
- Can efficiently perform operations applicable to the entire dataset and hence enables aggregation over many rows and columns
- In the hard disk, each column is stored together.

RDBMS vs. HBase

- HBase is schema-less, but RDBMS is based on a fixed schema
- HBase is for big data (million columns, billion rows), but RDBMS was not initially created to store big data
- HBase stores denormalized data, but RDBMS stores normalized data
- HBase supports automatic partitioning, but RDBMS does not.

HBase vs. HDFS

- HBase is is built for Low Latency operations, but HDFS is suited for High Latency operations batch processing
- HBase Data is accessed through shell commands, Client APIs in Java, etc. , but in HDFS Data is primarily accessed through MapReduce
- HBase Provides access to single rows from billions of records, but HDFS is designed for batch processing and hence doesn't have a concept of random reads/writes

Powered by HBase

- **Facebook** uses HBase to power their Messages infrastructure.
- **Twitter** runs HBase across its entire Hadoop cluster
- **Yahoo!** uses HBase to store document fingerprint.
- **Adobe** use HBase in several areas from social services to structured data and processing for internal use.