



## NoSQL (or Not Only SQL) Systems

Sharma Chakravarthy  
 Information Technology Laboratory  
 Computer Science and Engineering Department  
 The University of Texas at Arlington, Arlington, TX 76009  
 Email: [sharma@cse.uta.edu](mailto:sharma@cse.uta.edu)  
 URL: <http://itlab.uta.edu/sharma>

---


11/30/2022  © Sharma Chakravarthy 1

## Tutorial Outline




- *Before the cloud*
- *Cloud Definitions, trend*
- *Cloud computing characteristics*
- *Differences between cc, grid, and other forms of computing*
- *Cloud Architecture*
- *Map/reduce*
- *Hadoop*
- *Map/Reduce vs. DBMS*
- **NoSQL**
- **Conclusions**

---


11/30/2022  © Sharma Chakravarthy 2

## Acknowledgements




- These slides are put together from a variety of sources (both papers and slides/tutorials available on the web)
- Mostly I have tried to: provide my perspective, emphasize aspects that are of interest to this course, and have tried to put forth a consolidated view of Cloud computing

---


11/30/2022  © Sharma Chakravarthy 3

## RDBMSs



- Need Schema
  - EER modeling, normalization, and generation of schema
  - Schema is flat
  - Schema difficult to change
  - **No nesting of tables!**
- Do lots of joins
  - Joins are expensive, **though well-optimized** (over 40+ years)
  - Normalization increases number of joins
- Support limited Data types
  - Text and numeric data types
  - Not geared for others (voice, image, logs, graphs, objects etc.)
  - Lacks general search capability (e.g., find all occurrences in the database)


---

11/30/2022  © Sharma Chakravarthy 4

## RDMSSs iLab

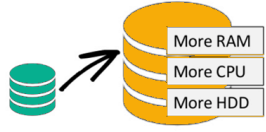
- Were optimized
  - For joins (*not for self-joins!* And not for *large number* of joins)
  - Fast retrieval (geared towards *reducing I/O*)
  - For *management* of data *over a long period of time*
  - Generating reports
  - OLTP, OLAP (dimensional analysis, data warehouses, marts)
  - *High concurrency*
  - *Guaranteed Recovery (ACID properties)*
- Limitations
  - Partitioning (or *distribution or scaling*) is *not automatic*
  - Query processing need redesign for taking advantage of partitioning
  - Distributed databases make *data unavailable* due to distributed commit (3 phase commit)

---

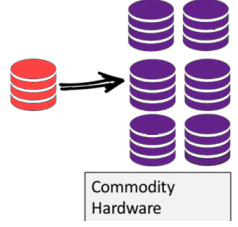
11/30/2022  © Sharma Chakravarthy 5

## Vertical and horizontal scaling iLab

**Scale-Up (vertical scaling):**




**Scale-Out (horizontal scaling):**



**Both are difficult with RDBMSs!**


---

11/30/2022  © Sharma Chakravarthy 6

## Newer applications iLab

- May not exactly fit the RDBMS model
  - No schema (or loose schema or need schema flexibility)
  - One shot computation (instead of management)
  - Weak concurrent usage (freshness is not critical)
  - Recovery is of a different nature (fault tolerance instead of strict consistency)
  - Custom computation; may be difficult to express in SQL
  - Needs custom optimization rather than general-purpose optimization
  - Data sizes beyond what DBMSs could handle
  - Quick response time
  - ACID properties were not needed as in an RDBMS
- **Answer: NoSQL systems**
  - **Data model, Customizable ACID properties (CAP), no recovery**


---

11/30/2022  © Sharma Chakravarthy 7

## Why NoSQL? iLab

- The concept of NoSQL databases became popular with Internet giants like Google, Facebook, Amazon, etc. who deal with **huge volumes of data**
  - The system response time becomes slow when you use RDBMS for massive volumes of data.
- To resolve this problem, we could "scale up" our systems by upgrading our existing hardware.
  - This **process is expensive**
- The alternative for this issue is to **distribute database load on multiple hosts** whenever the load increases.
  - This method is known as "scaling out."

---

11/30/2022  © Sharma Chakravarthy 8

## An example



- Suppose I want to keep track of customer orders, items in each order, how each order was paid.
- This is a complex objects that looks like:

### Objects representation

```
Order ID: 1001
Customer: mary
Line items:
item 1 qty per item total;
item 2 qty per item total;
...
Payment info:
cc: visa
cc #:
exp date:
```

### Normalized Relational representation

```
Orders table
Customers table
Items table
Credit_info table
```

To put together an object,  
You have to do **multiple**  
**joins** on the above tables!

Can avoid that of I store my data as objects!

11/30/2022



© Sharma Chakravarthy

9

## Brief History of NoSQL Databases



- 1998- Carlo Strozzi use the term NoSQL for his lightweight, open-source relational database
- 2000- Graph database **Neo4j** is launched
- 2004- **Google BigTable** is launched
- 2005- **CouchDB** is launched
- 2007- The research paper on **Amazon Dynamo** is released
- 2008- Facebook open sources the **Cassandra project**
- 2009- The term **NoSQL** was reintroduced

11/30/2022



© Sharma Chakravarthy

10

## NoSQL Databases



- **Google** came up with **BigTable** as an alternative to using **RDBMS**
- **Amazon** came up with **Dynamo**
- These did not use SQL (write your own code)
- The term was conjured up as a hashtag (#nosql) for a meeting!
- Initial group of systems
  - MongoDB
  - couchDB
  - Hypertable
  - Cassandra
  - Dynomo
  - Apache Hbase, voldemort,

11/30/2022



© Sharma Chakravarthy

11

## Definition of a NoSQL Database



- It is difficult to give a definition as there is no consensus
- Hence, **characteristics of NoSQL databases**
  - Non-relational (uses **different data models**)
  - Primarily open-source (as of now)
  - No complex features like query languages, query optimizers
  - Cluster friendly (shared-nothing architecture)
  - Schema less (or **loose schema**) no normalization
  - Elasticity (both storage and server capacity can be added on-the-fly)
  - Sharding (non-monolithic data representation)
  - BASE (Basically Available, Soft state, Eventual consistency) and CAP (Consistency, Availability, and Partition tolerance) instead of ACID
- If you think about **hierarchical and network databases**, they will qualify as NoSQL, but lacked some of the characteristics mentioned above!

11/30/2022



© Sharma Chakravarthy

12

## NoSQL in pictures

**SQL Database**

Relational

Analytical (OLAP)

**NoSQL Database**

Column-Family

Graph

Document

Key-Value

11/30/2022 © Sharma Chakravarthy 13

## Data Models

- **Documents**
  - mongoDB, couchDB, Riak, lotus notes,
  - Raven DB
- **Column-family**
  - Cassandra
  - Apache Hbase, HyperTable
- **Key-value**
  - Project Voldemort
  - Riak, Redis (based on Amazon Dynamo)
- **Graph**
  - Neo4j, GraphDB, FlockDB, OrientDB

11/30/2022 © Sharma Chakravarthy 14

## Key-Value store

10026 → Value

30075 → Value

50097 → Value

- Value can be anything you want (single number to complex object to an image)
  - String, json object, BLOB etc.
  - See how this will keep all information about an order together with order id.
- You can think of key-value store as a hash map!

11/30/2022 © Sharma Chakravarthy 15

## Key-Value store (2)

- A hash representation is used
- Caching is used to enhance performance
- Operations
  - Get(key)
  - Put(key)
  - Multi-get(key1, key2, ..., keyn)
  - Delete(key)
- As data size increases, additional performance issues come up
  - Keeping keys Unique may be difficult
- Examples: Oracle NoSQL, Riak and amazon Dynamo

11/30/2022 © Sharma Chakravarthy 16

### Key-Value store (3)



- You can envision how partitioning and scalability is achieved
- You can partition data and store separately
- If the keys are unique, updates can be processed in parallel and happens only in one partition!
- Scalability comes from hash-based approach and simplicity of management!
- May be good for some applications!

11/30/2022



© Sharma Chakravarthy

17

### Document store



Col1	Col2	Col3	Col4
Data	Data	Data	Data
Data	Data	Data	Data
Data	Data	Data	Data

Document 1	Document 2	Document 3
{ "prop1": data, "prop2": data, "prop3": data, "prop4": data }	{ "prop1": data, "prop2": data, "prop3": data, "prop4": data }	{ "prop1": data, "prop2": data, "prop3": data, "prop4": data }

- Key-value pair where the value part is stored as a document
- Inspired by Lotus notes!
- Documents in Jason or XML
- Jason or JavaScript Object Notation, is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application, as an alternative to XML.

11/30/2022



© Sharma Chakravarthy

18

### Document store (2)



- No schema or flexible schema
- Query or update portions of the document structure
- Implicit usage of attributes (metadata) in the absence of schema is important
  - "price" not "cost"
- Apache CouchDB
  - JSON to store data
  - JavaScript as its query language using Map/Reduce and HTTP for an API
- MangoDB is also popular
- Blogging platforms,

11/30/2022



© Sharma Chakravarthy

19

### Metadata and similarities



- Although key-value and document store are considered two data models, they have some similarities
  - Both have keys,
  - Some metadata (e.g., customer\_id: 6543 is used for indexing (otherwise, have to do sequential search)
- Both allow complex structures to be grouped and stored as a single structure
- Sometimes called aggregate-oriented databases

11/30/2022



© Sharma Chakravarthy

20

## Column-family Databases



- Inspired by Google's Big Table
- Different from RDBMS column storage such as Sybase IQ, vertica
- Column families are defined
- Columns can be defined dynamically
- Storage is more like an array; first dimension is the row identifier or row key, second is the combination of the column-family and column identifier. The third optional dimension is the timestamp and versions.

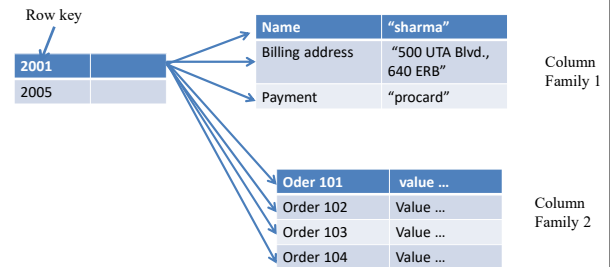
11/30/2022



© Sharma Chakravarthy

21

## Column-family Databases (2)



11/30/2022



© Sharma Chakravarthy

22

## Column-family Databases (3)



- It is also an aggregate-oriented representation
- Can easily distribute data
- Partitioning and distributing facilitates search
- However, in all of the above
  - Sorting and grouping is not easy
  - Report generation is not straight forward
  - Have to slice and dice data for this!
  - Can be done, but takes more computation and effort!
  - Examples: Apache Hbase, MariaDB, Hypertable

11/30/2022



© Sharma Chakravarthy

23

## Graph Databases



- Data models use nodes, relations, K-V on both
- Representation is different from that of tables
- Representation of graph as a table requires too many joins for traversal
- FlockDB developed and used by twitter is a good example
  - Keeps track of who follows whom
  - Traversal from a start node or a set of nodes
  - Ability to add more nodes and edges
  - Graph operations are optimized for this representation
  - Useful for social network analysis!

11/30/2022



© Sharma Chakravarthy

24

## Graph with attributes

iLab

- Captures entities and their relationships
- Mostly used for social networks

---

11/30/2022
© Sharma Chakravarthy
25

## An example

iLab

As of March 2009

---

11/30/2022
© Sharma Chakravarthy
26

## Graph Databases (2)

iLab

- May have their own query language for specifying queries
- Graph representation is becoming more popular as it is easy to see and understand relationships
- “Retrieve all founders of a company who attended stanford”
- “Query capabilities allow users to look for nodes, scan neighboring nodes, retrieve edges, and retrieve attribute values. Users can also perform more complex queries.”

---

11/30/2022
© Sharma Chakravarthy
27

## Disadvantages of NoSQL

iLab

- No standardization rules
- Limited query capabilities
- RDBMS databases and tools are comparatively mature
- It does not offer any traditional database capabilities, like consistency when multiple transactions are performed simultaneously.
- When the volume of data increases it is difficult to maintain unique values as keys
- Doesn't work as well as relational data
- The learning curve is stiff for new developers
- Open source options, hence not so popular for enterprises.

---

11/30/2022
© Sharma Chakravarthy
28

## ACID



- We are familiar with ACID properties
  - Atomicity, consistency, isolation, and durability
  - Recovery at the transaction level
  - **All of the above come at a cost!**
- However, when you update multiple aggregates, how does NoSQL handle that?

11/30/2022



© Sharma Chakravarthy

29

## ACID (2)



- However, when you update multiple aggregates, how does NoSQL handle that?
  - If the same aggregate is checked out by two users and they update it and post it back, there is a write-write conflict.
  - If we apply ACID, this whole transaction (read, update, and post) needs to be atomic.
  - This also means that the object is **locked for a longer period** especially if it is interactive! (that is why databases do not allow interactive transactions!)
  - Solution
    - Wrap each action as a Tx
    - Use time version stamp to resolve conflicts
    - By the way RCS, CVS systems (even optimistic concurrency control) have been doing this for a long time
    - Github does it as well

11/30/2022



© Sharma Chakravarthy

30

## Consistency



- Logical
  - About the data value in a **single copy**
- Replication
  - **Sharding** does not cause problems (**disjoint single copy**)
  - **distribution with replication** is a different story
    - Availability, resilience etc.
    - Whether all copies are consistent all the time
  - Choice between consistency and availability in the presence of communication failures and partition
  - This choice can be made by the application (or business) instead of the database vendor!
    - Compensation etc.

11/30/2022



© Sharma Chakravarthy

31

## CAP



- CAP
  - Consistency, Availability, and Partition tolerance
    - **Consistency means** value is consistent after the execution of an operation. All clients see the same state after an update
    - **availability means** no (or very little) down time
    - **Tolerance to partition means** the system will still be available for use after a network partition
  - **CAP theorem (or Brewer's theorem)**
    - For a distributed data store, it is impossible to offer more than two out of three guarantees (C, A, or P)

11/30/2022



© Sharma Chakravarthy

32



## CAP



- While traditional databases **make that decision for you** (which 2 out of 3), NoSQL databases **provide these guarantees as tuning/customizing options**. Database vendors must always decide which two to prioritize. The options are:
- Availability is compromised in favor of consistency and partition-tolerance (CP) – **used by Distributed DBMSs**
- Partition-tolerance is forfeited in favor of consistency and availability (CA) – mainly for non-distributed systems!
- Consistency is compromised but systems are always available and can work when parts are partitioned (AP)
  - Preferred by NOSQL systems

11/30/2022



© Sharma Chakravarthy

33

## CAP



- Traditional SQL databases place a high priority on **consistency** and **fault-tolerance (system failure)** and have, as a result, chosen to go with the first option (CP) and **forfeit high availability**.
- NoSQL databases frequently leave that decision to the application operations team (or business) and provide configuration options so that the preferred options can be chosen based on the application use case.
  - Atomicity is/can be done at the single update level (instead of Tx)
  - Eventual consistency is an option: means that given a sufficiently long period of time over which no changes are done, all updates can be expected to propagate eventually through the system and all the replicas will be consistent.
  - Consistency at different levels (influences response time)

11/30/2022



© Sharma Chakravarthy

34

## BASE



- Basically Available, Soft-state, Eventual consistency
  - Trades off strong consistency guarantees for availability and partition tolerance
  - Promotes availability over consistency by supporting eventual consistency
- BASE is a consistency model used in **distributed systems** (in contrast to distributed DBMSs)
  - NoSQL databases promote BASE

11/30/2022



© Sharma Chakravarthy

35

## Summary



- Vertical scaling – adding more juice to existing boxes
- Horizontal scaling – adding more boxes!
  - Involves data partitioning as well
- Databases are somewhat **ok with vertical scaling, but not good at horizontal scaling**
- SQL or NoSQL?
  - If you need horizontal scaling, NoSQL is a better alternative!
  - If your application has no schema or difficult to define a schema (e.g., web log processing, hierarchical documents)
  - If your applications does not need ACID as provided by a RDBMS
  - If you have graph applications and representation using RDBMS requires too many joins for computation
    - Social network applications

11/30/2022



© Sharma Chakravarthy

36

## Summary (2)



- **Good News**
  - We have more choices than before
  - You can choose the right model and computation needed to match your application
  - You can get scaling and quick setup
- **Bad news**
  - You have to write more code
  - NoSQL is still immature (very few utilities compared to RDBMS)
  - Architects have more difficulty in determining and defining what is best for their application
  - Tuning needs to be done on a case-by-case basis
    - Not many available tools
  - We will face Multi-DBMS problems!

11/30/2022



© Sharma Chakravarthy

37

## Summary (3)



- An example of using different DB technologies for different requirements
  - User sessions (Redis, **key-value**)
  - Financial data (**RDBMS**)
  - Shopping Cart (Risk, **key-value**)
  - Recommendations (Neo4J, **graph**)
  - Reporting (**RDBMS**)
  - Product catalog (MongoDB, **document store**)
- Truly, the amount of non-rdbms data is growing very fast as compared to rdbms data
- Applications/web services are used for accessing data unlike earlier way of accessing directly
- Applications can mask business-specific logic instead DBMSs

11/30/2022



© Sharma Chakravarthy

38

## Summary (4)



- **OldSQL** (pronounced old school)
  - Legacy RDBMSs with scalability and performance issues
- **NoSQL**
  - Give up SQL and ACID for scalability and availability
- **NewSQL** (pronounced new school)
  - Preserve SQL and ACID
  - Get performance from new architecture
  - Being proposed by old schoolers (from DBMS, e.g., Michael Stonebraker, ...)
    - Doubt whether this will succeed!

11/30/2022



© Sharma Chakravarthy

39

## What does this mean to a Designer



- With only/mainly the relational model the decision making was easy!
  - Choose RDBMS and a good vendor and you will be fine
  - This was like earlier thinking
    - you cannot get fired for choosing IBM!
- Now, **it is not that easy**
  - You have to look at many alternatives and choose the right one
    - **You can get fired if it turns out to be the wrong choice!**
- Bottom line:
  - Your life has become **more difficult, not easy**

11/30/2022



© Sharma Chakravarthy

40

Thank You !!!

