CS315: Database Systems
NoSQL and Big Data Systems

Arnab Bhattacharya
arnabb@cse.iitk.ac.in

Computer Science and Engineering,
Indian Institute of Technology, Kanpur
http://web.cse.iitk.ac.in/~cs315/

2nd semester, 2019-20

Tue, Wed 12:00-13:15

NoSQL

- NoSQL is

NoSQL

- NoSQL is *not* “no-SQL”
- It is not only SQL (originated as no-SQL, though)

NoSQL

- NoSQL is *not* “no-SQL”
- It is not only SQL (originated as no-SQL, though)
- Aims to provide
  - Scalability
  - Flexibility
  - Naturalness
  - Distribution
  - Performance

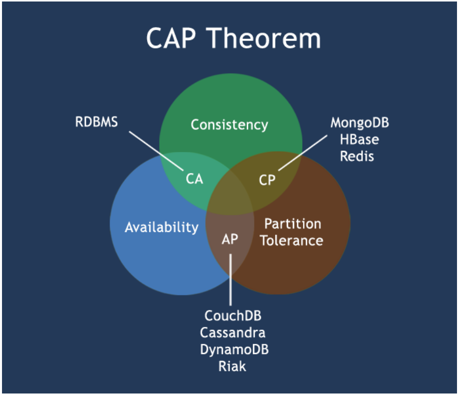
NoSQL

- NoSQL is *not* "no-SQL"
- It is not only SQL (originated as no-SQL, though)
- Aims to provide
  - Scalability
  - Flexibility
  - Naturalness
  - Distribution
  - Performance
- It does not aim to provide the ACID properties
  1. Atomicity: Either all operations of a transaction are reflected or none are reflected
  2. Consistency: If a database is consistent before the execution of the transaction, it must be consistent after it
  3. Isolation: Although multiple transactions may execute concurrently, each transaction must be unaware of others
  4. Durability: After a transaction finishes successfully, the changes must be permanent in the database despite subsequent failures

NoSQL

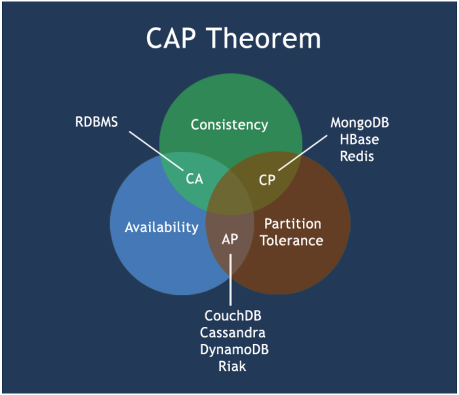
- NoSQL is *not* “no-SQL”
- It is not only SQL (originated as no-SQL, though)
- Aims to provide
  - Scalability
  - Flexibility
  - Naturalness
  - Distribution
  - Performance
- It does not aim to provide the ACID properties
  1. Atomicity: Either all operations of a transaction are reflected or none are reflected
  2. Consistency: If a database is consistent before the execution of the transaction, it must be consistent after it
  3. Isolation: Although multiple transactions may execute concurrently, each transaction must be unaware of others
  4. Durability: After a transaction finishes successfully, the changes must be permanent in the database despite subsequent failures
- Later changed since RDBMS is too powerful to always ignore
- NewSQL

CAP Theorem



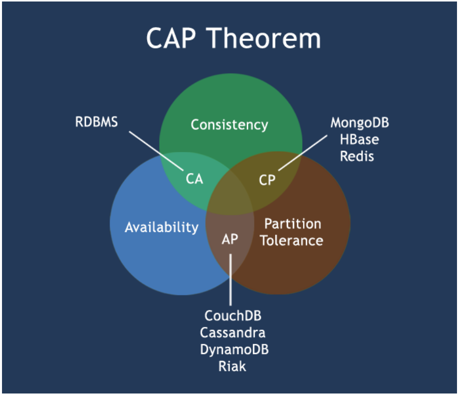
- All of C, A, P cannot be satisfied simultaneously

CAP Theorem



- All of C, A, P cannot be satisfied simultaneously
- CA: single-site; partitioning is not allowed
- CP: what is available is consistent
- AP: everything is available but may not be consistent

CAP Theorem



- All of C, A, P cannot be satisfied simultaneously
- CA: single-site; partitioning is not allowed
- CP: what is available is consistent
- AP: everything is available but may not be consistent
- Not a theorem, but a hypothesis

BASE Properties

BASE Properties

- Basically Available: System guarantees availability

BASE Properties

- Basically Available: System guarantees availability
- Soft state: State of system is soft, i.e., it may change without input to maintain consistency

BASE Properties

- Basically Available: System guarantees availability
- Soft state: State of system is soft, i.e., it may change without input to maintain consistency
- Eventual consistency: Data will be eventually consistent without any interim perturbation

BASE Properties

- Basically Available: System guarantees availability
- Soft state: State of system is soft, i.e., it may change without input to maintain consistency
- Eventual consistency: Data will be eventually consistent without any interim perturbation
- Sacrifices strong (immediate) consistency

BASE Properties

- Basically Available: System guarantees availability
- Soft state: State of system is soft, i.e., it may change without input to maintain consistency
- Eventual consistency: Data will be eventually consistent without any interim perturbation
- Sacrifices strong (immediate) consistency
- BASE is to counter ACID

Types

- Main types of NoSQL data stores:
  1. Columnar families
  2. Key-value stores
  3. Bigtable systems
  4. Document databases
  5. Graph databases

Columnar Storage

- Instead of rows being stored together, columns are stored consecutively
- A single disk block (or a set of consecutive blocks) stores a single column family
- A column family may consist of one or multiple columns
- This set of columns is called a super column

Columnar Storage

- Instead of rows being stored together, columns are stored consecutively
- A single disk block (or a set of consecutive blocks) stores a single column family
- A column family may consist of one or multiple columns
- This set of columns is called a super column
- Two main types
  - Columnar relational models
  - Key-value stores and/or big tables

Columnar Relational Models

- A type of *RDBMS*
- Column-wise storage on the disk

Columnar Relational Models

- A type of *RDBMS*
- Column-wise storage on the disk
- Allows faster querying when only few columns are touched on the entire data
- Allows compression of columns
- Provides better memory caching
- Joins are faster since they are mostly on similar columns from two tables

Columnar Relational Models

- A type of *RDBMS*
- Column-wise storage on the disk
- Allows faster querying when only few columns are touched on the entire data
- Allows compression of columns
- Provides better memory caching
- Joins are faster since they are mostly on similar columns from two tables
- Is not good for updates
- Is not good when many columns of a few tuples are accessed

Columnar Relational Models

- A type of *RDBMS*
- Column-wise storage on the disk
- Allows faster querying when only few columns are touched on the entire data
- Allows compression of columns
- Provides better memory caching
- Joins are faster since they are mostly on similar columns from two tables
- Is not good for updates
- Is not good when many columns of a few tuples are accessed
- Good for OLAP (online analytical processing)
- Not good for OLTP (online transaction processing)

Columnar Relational Models

- A type of *RDBMS*
- Column-wise storage on the disk
- Allows faster querying when only few columns are touched on the entire data
- Allows compression of columns
- Provides better memory caching
- Joins are faster since they are mostly on similar columns from two tables
- Is not good for updates
- Is not good when many columns of a few tuples are accessed
- Good for OLAP (online analytical processing)
- Not good for OLTP (online transaction processing)
- Example: MonetDB

Key-Value Stores

- Two columns: a key and a value
- Key is mostly text
- Value can be anything and is simply an object

Key-Value Stores

- Two columns: a key and a value
- Key is mostly text
- Value can be anything and is simply an object
- Essentially, actual data becomes “value” and an unique id is generated which becomes “key”

Key-Value Stores

- Two columns: a key and a value
- Key is mostly text
- Value can be anything and is simply an object
- Essentially, actual data becomes "value" and an unique id is generated which becomes "key"
- Whole database is then just one big table with these two columns
- Becomes schema-less

Key-Value Stores

- Two columns: a key and a value
- Key is mostly text
- Value can be anything and is simply an object
- Essentially, actual data becomes "value" and an unique id is generated which becomes "key"
- Whole database is then just one big table with these two columns
- Becomes schema-less
- Can be distributed and is, thus, highly scalable
- So, in essence a big distributed hash table

Key-Value Stores

- Two columns: a key and a value
- Key is mostly text
- Value can be anything and is simply an object
- Essentially, actual data becomes “value” and an unique id is generated which becomes “key”
- Whole database is then just one big table with these two columns
- Becomes schema-less
- Can be distributed and is, thus, highly scalable
- So, in essence a big distributed hash table
- All queries are on keys
- Keys are necessarily indexed

Key-Value Stores

- Two columns: a key and a value
- Key is mostly text
- Value can be anything and is simply an object
- Essentially, actual data becomes “value” and an unique id is generated which becomes “key”
- Whole database is then just one big table with these two columns
- Becomes schema-less
- Can be distributed and is, thus, highly scalable
- So, in essence a big distributed hash table
- All queries are on keys
- Keys are necessarily indexed
- Example: Cassandra, CouchDB, Tokyo Cabinet, Redis

BigTable Systems

- Started from Google's BigTable implementation
- Uses a key-value store
- Data can be replicated for better availability

BigTable Systems

- Started from Google's BigTable implementation
- Uses a key-value store
- Data can be replicated for better availability
- Uses a timestamp
- Timestamp is used to
  - Expire data
  - Delete stale data
  - Resolve read-write conflicts

BigTable Systems

- Started from Google's BigTable implementation
- Uses a key-value store
- Data can be replicated for better availability
- Uses a timestamp
- Timestamp is used to
  - Expire data
  - Delete stale data
  - Resolve read-write conflicts
- Same value can be indexed using multiple keys
- Map-reduce framework to compute

BigTable Systems

- Started from Google's BigTable implementation
- Uses a key-value store
- Data can be replicated for better availability
- Uses a timestamp
- Timestamp is used to
  - Expire data
  - Delete stale data
  - Resolve read-write conflicts
- Same value can be indexed using multiple keys
- Map-reduce framework to compute
- Example: BigTable, HBase, Cassandra, HyperTable, SimpleDB

Document Databases

- Uses document structure as the main storage format of data
- Popular document formats are XML, JSON, BSON, YAML
- Document itself is the key while the content is the value
- Document can be indexed by id or simply its location (e.g., URI)

Document Databases

- Uses document structure as the main storage format of data
- Popular document formats are XML, JSON, BSON, YAML
- Document itself is the key while the content is the value
- Document can be indexed by id or simply its location (e.g., URI)
- Content needs to be parsed to make sense
- Content can be organised further

Document Databases

- Uses document structure as the main storage format of data
- Popular document formats are XML, JSON, BSON, YAML
- Document itself is the key while the content is the value
- Document can be indexed by id or simply its location (e.g., URI)
- Content needs to be parsed to make sense
- Content can be organised further
- Extremely useful for insert-once read-many scenarios
- Can use map-reduce framework to compute

Document Databases

- Uses document structure as the main storage format of data
- Popular document formats are XML, JSON, BSON, YAML
- Document itself is the key while the content is the value
- Document can be indexed by id or simply its location (e.g., URI)
- Content needs to be parsed to make sense
- Content can be organised further
- Extremely useful for insert-once read-many scenarios
- Can use map-reduce framework to compute
- Example: MongoDB, CouchDB

Graph Databases

- Nodes represent entities
- Edges encode relationships between nodes
- Can be directed
- Can have hyper-edges as well

Graph Databases

- Nodes represent entities
- Edges encode relationships between nodes
- Can be directed
- Can have hyper-edges as well
- Easier to find distances and neighbors

Graph Databases

- Nodes represent entities
- Edges encode relationships between nodes
- Can be directed
- Can have hyper-edges as well
- Easier to find distances and neighbors
- Example: Neo4J, HyperGraph, Infinite Graph, Titan, FlockDB

Discussion

- NoSQL, although started as anti-SQL, is no more so
- More a realisation that for some cases
  - RDBMS does not scale or distribute, or
  - ACIDity is an overkill

Discussion

- NoSQL, although started as anti-SQL, is no more so
- More a realisation that for some cases
  - RDBMS does not scale or distribute, or
  - ACIDity is an overkill
- NoSQL is *not* good for every scenario
  - Not always that strong consistency can be sacrificed
- Most legacy systems still use RDBMS

Discussion

- NoSQL, although started as anti-SQL, is no more so
- More a realisation that for some cases
  - RDBMS does not scale or distribute, or
  - ACIDity is an overkill
- NoSQL is *not* good for every scenario
  - Not always that strong consistency can be sacrificed
- Most legacy systems still use RDBMS
- NoSQL horizon is shifting rapidly
- Trend is for NoSQL as cloud computing and big data relies on it

Discussion

- NoSQL, although started as anti-SQL, is no more so
- More a realisation that for some cases
  - RDBMS does not scale or distribute, or
  - ACIDity is an overkill
- NoSQL is *not* good for every scenario
  - Not always that strong consistency can be sacrificed
- Most legacy systems still use RDBMS
- NoSQL horizon is shifting rapidly
- Trend is for NoSQL as cloud computing and big data relies on it
- Many NoSQL systems are increasingly using features of RDBMS
- New paradigm of scalability with transaction support is NewSQL

Big Data

- What is big data?

Big Data

- What is big data?
- Data which is big
- How big is “big”?

Big Data

- What is big data?
- Data which is big
- How big is "big"?
- For sociologists, 100 subjects is big

Big Data

- What is big data?
- Data which is big
- How big is “big”?
- For sociologists, 100 subjects is big
- For social networks, terrabytes is normal

Big Data

- What is big data?
- Data which is big
- How big is “big”?
- For sociologists, 100 subjects is big
- For social networks, terrabytes is normal
- For astrophysicists, terrabytes is an hour’s work

Big Data

- What is big data?
- Data which is big
- How big is “big”?
- For sociologists, 100 subjects is big
- For social networks, terrabytes is normal
- For astrophysicists, terrabytes is an hour’s work
- So, no absolute definition or threshold

Big Data

- What is big data?
- Data which is big
- How big is "big"?
- For sociologists, 100 subjects is big
- For social networks, terrabytes is normal
- For astrophysicists, terrabytes is an hour's work
- So, no absolute definition or threshold
- When data is bigger than most standard machines can store or most algorithms can handle

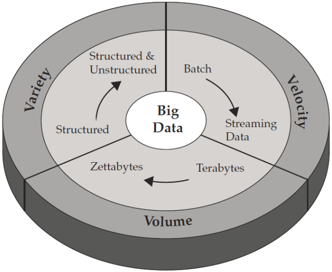
Characterization of Big Data

- Having large volumes of data requires
  - Newer techniques
  - Newer tools
  - Newer architectures

Characterization of Big Data

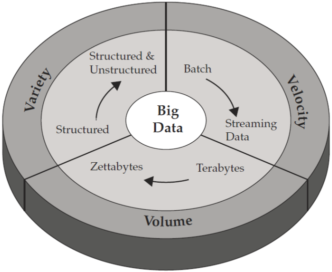
- Having large volumes of data requires
  - Newer techniques
  - Newer tools
  - Newer architectures
- Allows solving newer problems
  - Can also solve older problems better

Properties of Big Data



- 3 V's: volume, variety, velocity
- Volume: When data is extremely large in size, how to load it, index it or query it
- Variety: Data can be semi-structured or unstructured as well; how to query
- Velocity: Data can arrive at real time and can be streaming

Properties of Big Data



- 3 V’s: volume, variety, velocity
- Volume: When data is extremely large in size, how to load it, index it or query it
- Variety: Data can be semi-structured or unstructured as well; how to query
- Velocity: Data can arrive at real time and can be streaming
- Extended V’s: veracity, validity, visibility, variability

Enablers of Big Data

- Improved storage volume and type
- Improved processing power
- Improved data
- Improved network speed
- Improved architecture

Enablers of Big Data

- Improved storage volume and type
- Improved processing power
- Improved data
- Improved network speed
- Improved architecture
- Increased capital
- Increased business

Tools for Big Data

- Hosting: Distributed servers or Cloud
  - Amazon EC2

Tools for Big Data

- Hosting: Distributed servers or Cloud
  - Amazon EC2
- File system: Scalable and distributed
  - HDFS, Amazon S3

Tools for Big Data

- Hosting: Distributed servers or Cloud
  - Amazon EC2
- File system: Scalable and distributed
  - HDFS, Amazon S3
- Programming model: Distributed scalable processing
  - Map-reduce framework, Hadoop

Tools for Big Data

- Hosting: Distributed servers or Cloud
  - Amazon EC2
- File system: Scalable and distributed
  - HDFS, Amazon S3
- Programming model: Distributed scalable processing
  - Map-reduce framework, Hadoop
- Database: NoSQL
  - HBase, MongoDB, Cassandra

Tools for Big Data

- Hosting: Distributed servers or Cloud
  - Amazon EC2
- File system: Scalable and distributed
  - HDFS, Amazon S3
- Programming model: Distributed scalable processing
  - Map-reduce framework, Hadoop
- Database: NoSQL
  - HBase, MongoDB, Cassandra
- Operations: Querying, indexing, analytics
  - Data mining, Information retrieval
  - Machine learning: Mahout on top of Hadoop

Discussion

- Not so new term: Data Science

Discussion

- Not so new term: Data Science
- Many open-source tools

Discussion

- Not so new term: Data Science
- Many open-source tools
- Cloud, etc. can be rented

Discussion

- Not so new term: Data Science
- Many open-source tools
- Cloud, etc. can be rented
- Many applications still do *not* require big data