Introduction to NoSQL and MongoDB

Kathleen Durant PhD
CS 3200
Northeastern University

Outline for today

- Introduction to NoSQL
    - Architecture
        - Sharding
        - Replica sets
    - NoSQL Assumptions and the CAP Theorem
    - Strengths and weaknesses of NoSQL
- MongoDB
    - Functionality
    - Examples

Taxonomy of NoSQL

- **Key-value**

redis riak

- **Graph database**

Neo4j the graph database

HyperGraphDB

- **Document-oriented**

mongoDB

CouchDB relax

- **Column family**

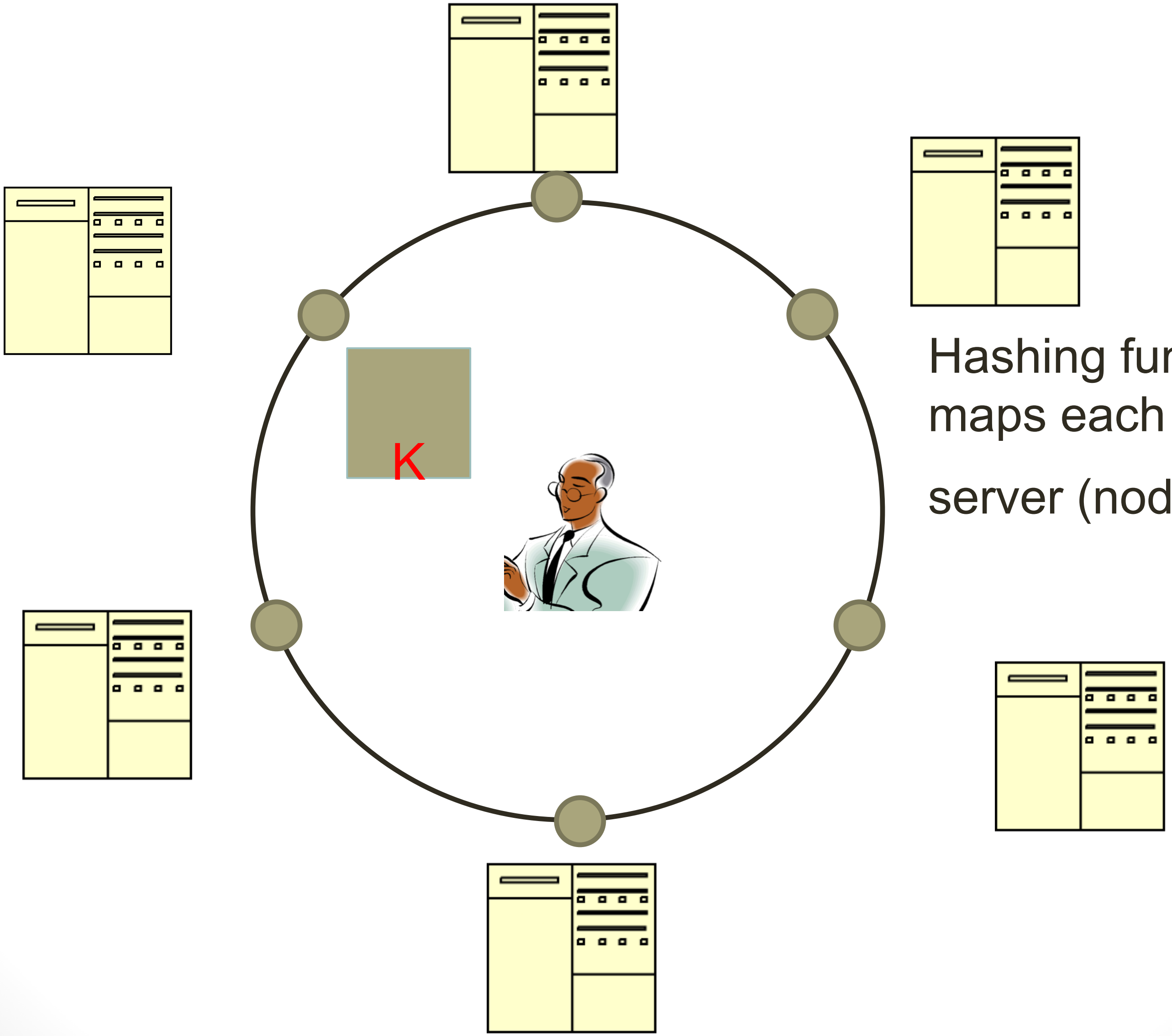
Cassandra

H·BASE

NoSQL Data Models

- **Key-value :** associate a data value with a specific key
- **Document-oriented :** associate a structured data value with a specific key. The structure is embedded in the object
- **Graph database :** consists of nodes and edges. Typically the nodes represent entities and the edges represent relationships.
- **Columnar database:** stores data by columns as oppose to rows. Columns are grouped into families. Typically a family corresponds to a real world object

Typical NoSQL architecture



Hashing function maps each key to a server (node)

CAP theorem for NoSQL

What the CAP theorem really says:

- If you cannot limit the number of faults and requests can be directed to any server and you insist on serving every request you receive then you cannot possibly be consistent

Eric Brewer 2001

How it is interpreted:

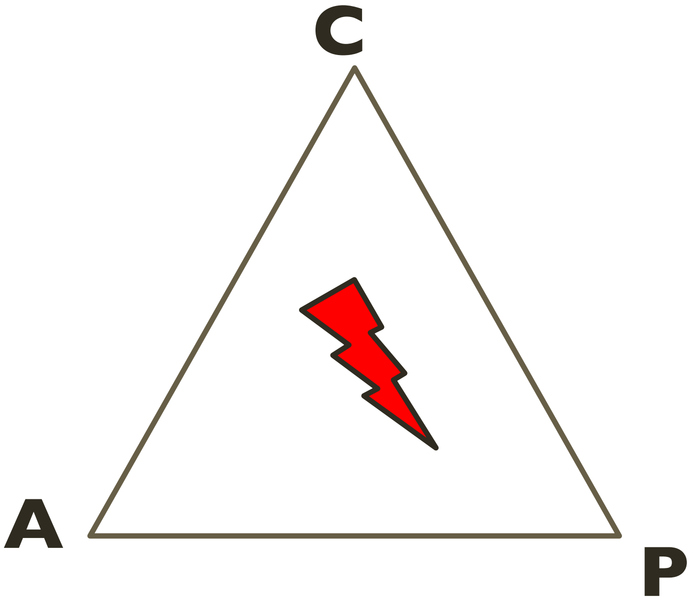
- You must always give something up: consistency, availability or tolerance to failure and reconfiguration

Theory of NoSQL: CAP

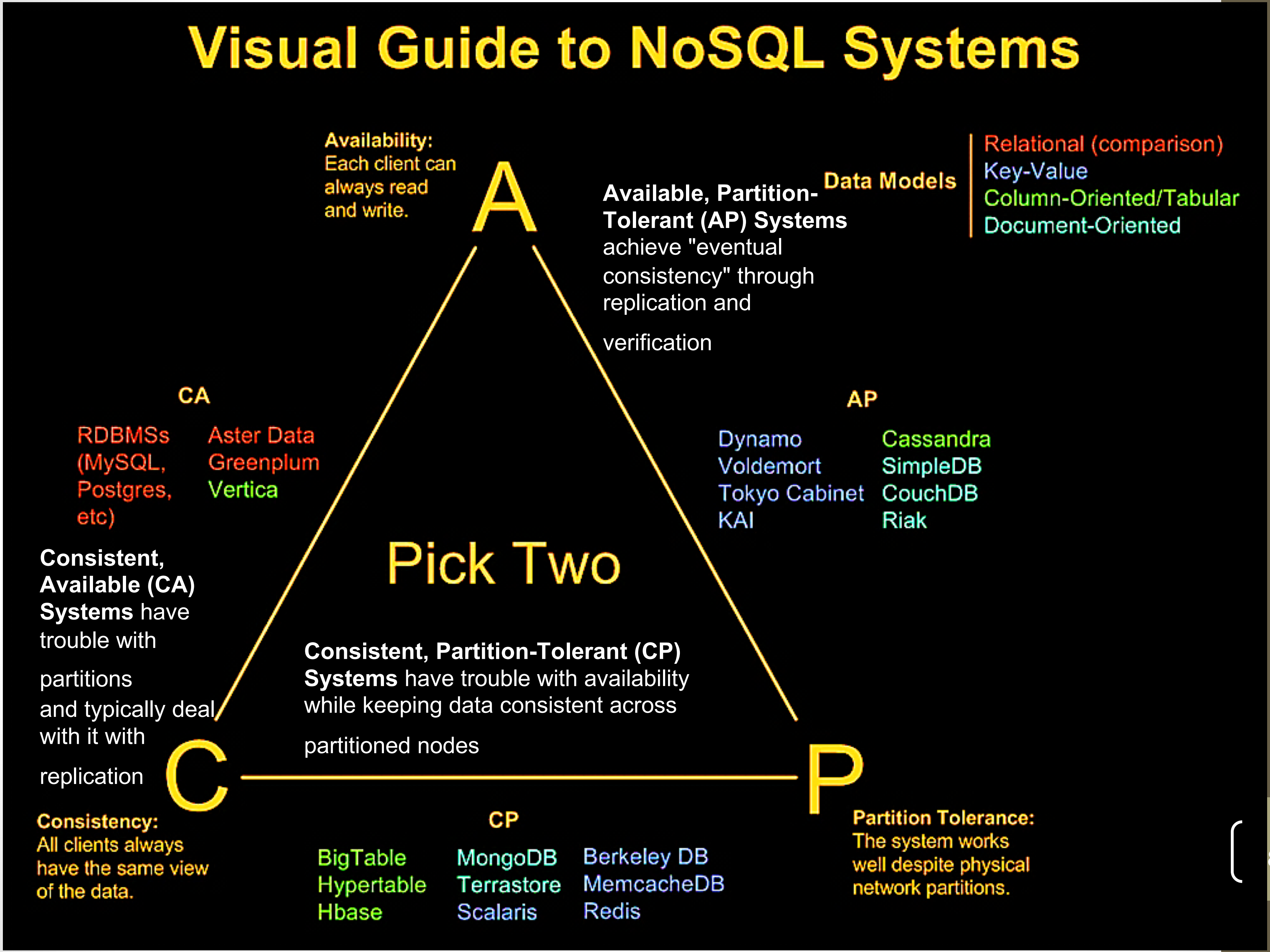
GIVEN:

- Many nodes
- Nodes contain ***replicas of partitions*** of the data

- Consistency
  - All replicas contain the same version of data
  - Client always has the same view of the data (no matter what node)
- Availability
  - System remains operational
  - All clients can always read and write
- Partition tolerance
  - multiple entry points
  - System remains operational on system split (communication malfunction)
  - System works well across physical network partitions



CAP Theorem: guarenteeing all three at the same time is impossible



Visual Guide to NoSQL Systems

Availability:
Each client can always read and write.

Data Models

Relational (comparison)
Key-Value
Column-Oriented/Tabular
Document-Oriented

Available, Partition-Tolerant (AP) Systems achieve "eventual consistency" through replication and verification

CA

RDBMSs (MySQL, Postgres, etc)	Aster Data Greenplum Vertica

AP

Dynamo	Cassandra
Voldemort	SimpleDB
Tokyo Cabinet	CouchDB
KAI	Riak

Pick Two

Consistent, Available (CA) Systems have trouble with partitions and typically deal with it with replication

Consistent, Partition-Tolerant (CP) Systems have trouble with availability while keeping data consistent across partitioned nodes

Consistency:
All clients always have the same view of the data.

CP

BigTable	MongoDB	Berkeley DB
Hypertable	Terrastore	MemcacheDB
Hbase	Scalaris	Redis

Partition Tolerance:
The system works well despite physical network partitions.

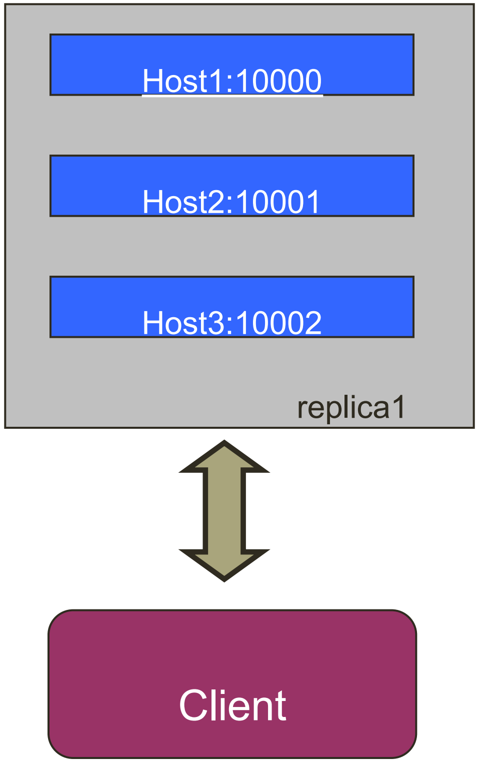
http://blog.nahurst.com/visual-guide-to-nosql-systems

Sharding of data

- Distributes a single logical database system across a cluster of machines
- Uses range-based partitioning to distribute documents based on a specific shard key
- Automatically balances the data associated with each shard
- Can be turned on and off per collection (table)

Replica Sets

- Redundancy and Failover
  - Failover: switching to another node for data access when a node fails
- Zero downtime for upgrades and maintenance
- Master-slave replication
  - Strong Consistency
  - Delayed Consistency
- Geospatial features



Host1:10000

Host2:10001

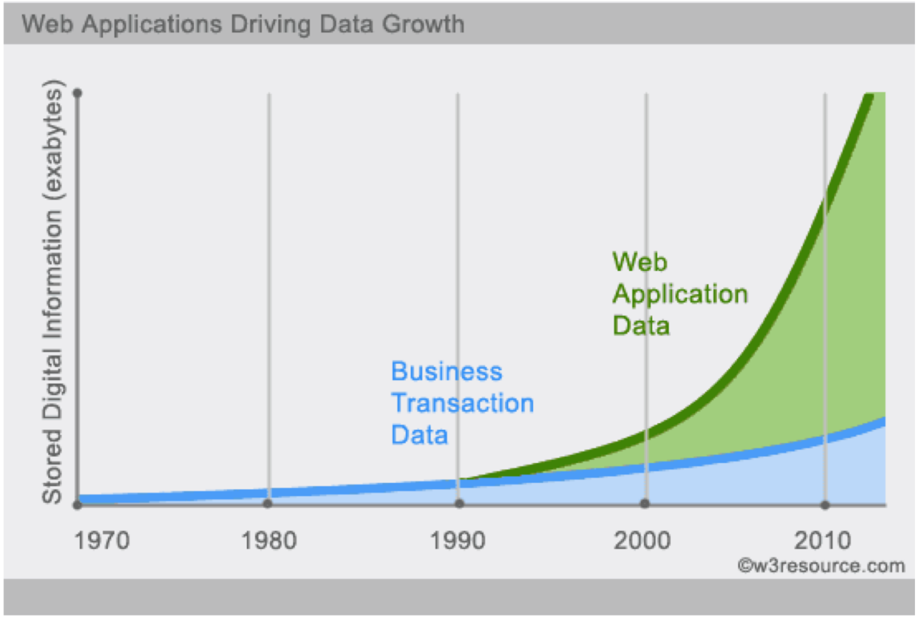
Host3:10002

replica1

Client

The Changing face of Data

- Big Data can be understood through the four V's of volume, variety, velocity, veracity:
  - **Volume**: enormous amounts of structured and unstructured data
  - **Variety**: multiple data types including documents, images, videos, and time series
  - **Velocity**: flow of data is continuous and increasing
  - **Veracity**: data contains biases, mistakes, noise, and abnormalities



How does NoSQL vary from RDBMS?

- Looser schema definition
- Applications written to deal with specific documents/ data
  - Applications aware of the schema definition as opposed to the data
- Designed to handle distributed, large databases
- Trade offs:
  - No strong support for ad hoc queries but designed for speed and growth of database
    - Query language through the API
  - Relaxation of the ACID properties

Benefits of NoSQL

Elastic Scaling

- RDBMS scale up – bigger load , bigger server
- NoSQL scale out – distribute data across multiple hosts seamlessly

DBA Specialists

- RDMS require highly trained expert to monitor DB
- NoSQL require less management, automatic repair and simpler data models

Big Data

- Huge increase in data RDMS: capacity and constraints of data volumes at its limits
- NoSQL designed for big data
  - Volume
  - Variety
  - Velocity
  - Veracity

Benefits of NoSQL

Flexible data models

- Change management to schema for RDMS have to be carefully managed
- NoSQL databases more relaxed in structure of data
  - Database schema changes do not have to be managed as one complicated change unit
  - Application already written to address an amorphous schema

Economics

- RDMS rely on expensive proprietary servers to manage data
- No SQL: clusters of cheap commodity servers to manage the data and transaction volumes
- Cost per gigabyte or transaction/second for NoSQL can be lower than the cost for a RDBMS

Drawbacks of NoSQL

- Support
  - RDBMS vendors provide a high level of support to clients
    - Stellar reputation
  - NoSQL – are open source projects with startups supporting them
    - Reputation not yet established

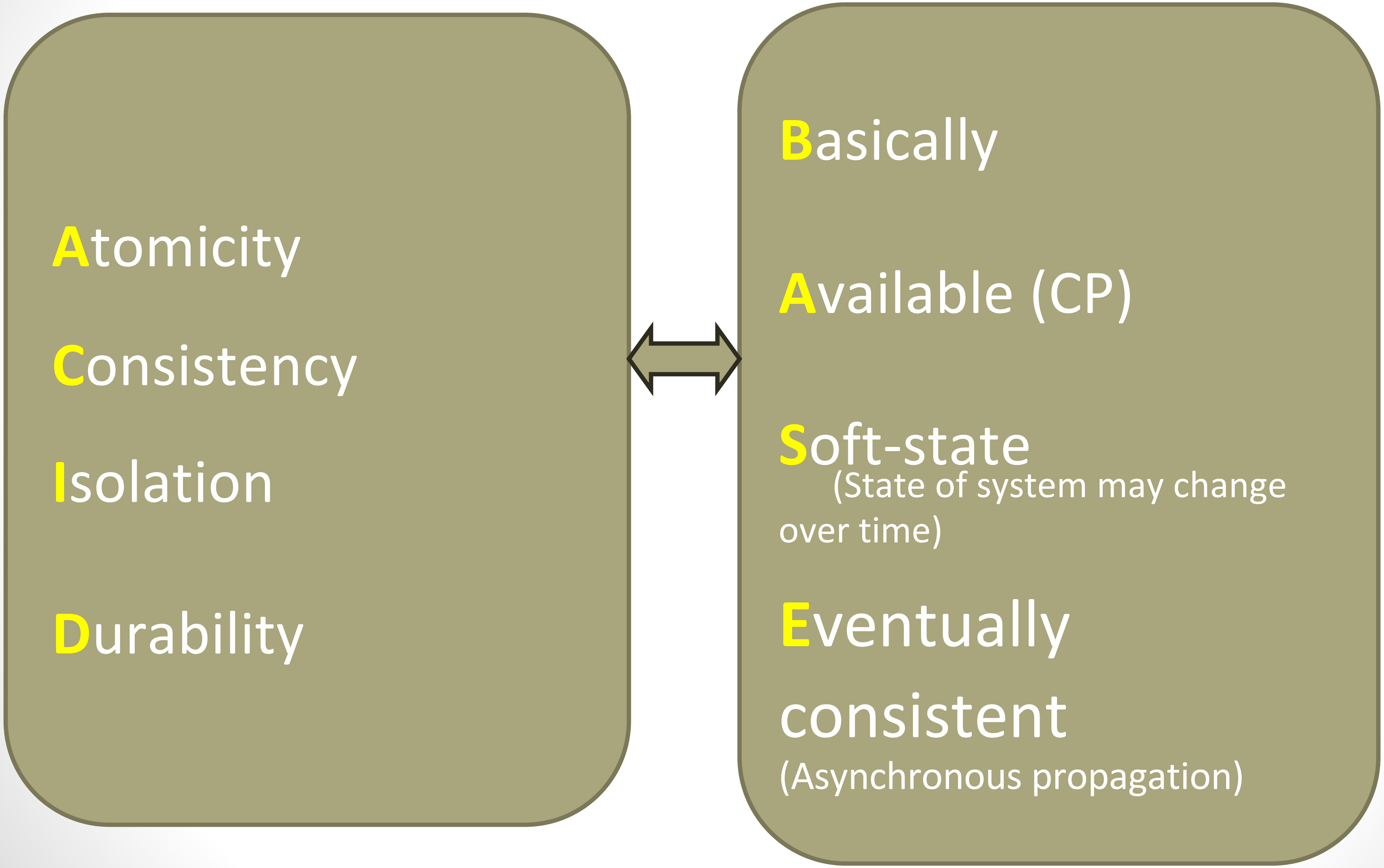
- Maturity
  - RDMS mature product: means stable and dependable
    - Also means old no longer cutting edge nor interesting
  - NoSQL are still implementing their basic feature set

Drawbacks of NoSQL

- **Administration**
  - RDMS administrator well defined role
  - NoSQL's goal: no administrator necessary however NO SQL still requires effort to maintain
- **Lack of Expertise**
  - Whole workforce of trained and seasoned RDMS developers
  - Still recruiting developers to the NoSQL camp

- **Analytics and Business Intelligence**
  - **RDMS designed to address this niche**
  - NoSQL designed to meet the needs of an Web 2.0 application - not designed for ad hoc query of the data
    - Tools are being developed to address this need

RDB ACID to NoSQL BASE



Atomicity

Consistency

Isolation

Durability

Basically

Available (CP)

Soft-state
(State of system may change over time)

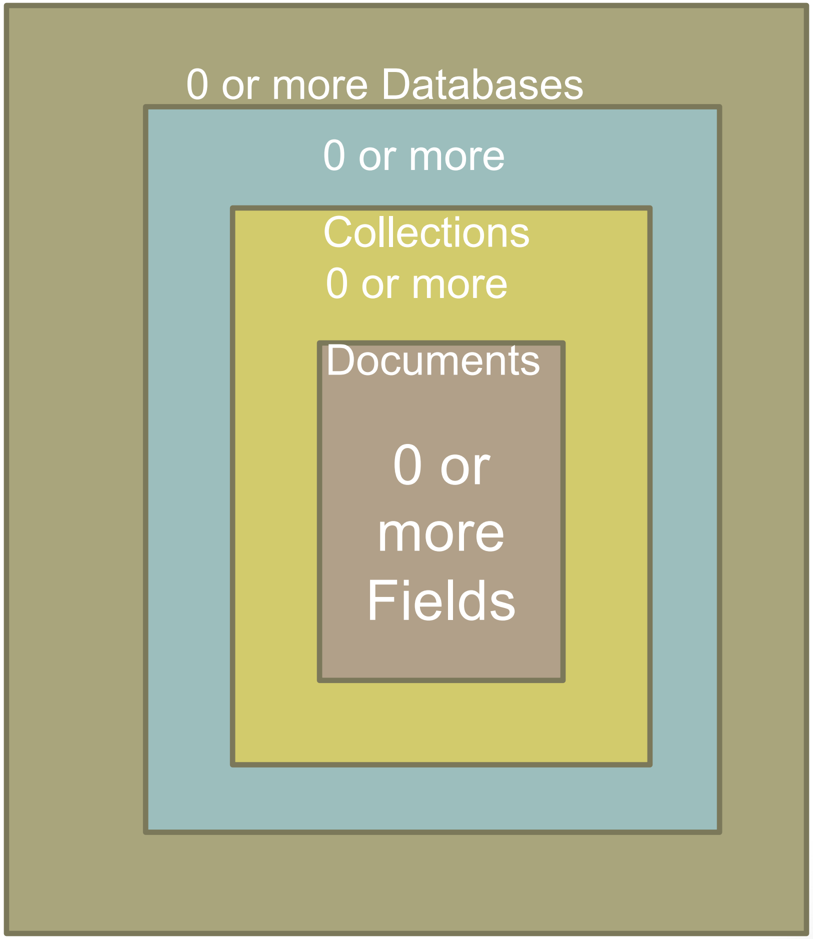
Eventually consistent
(Asynchronous propagation)

Pritchett, D.: BASE: An Acid Alternative (queue.acm.org/detail.cfm?id=1394128)

HTTPS://DOCS.MONGODB.ORG/MANUAL/

MongoDB: document store is a hierarchy

- A MongoDB instance may have zero or more ‘databases’
- A database may have zero or more ‘collections’.
- A collection may have zero or more ‘documents’.
- A document may have one or more ‘fields’.
- MongoDB ‘Indexes’ function much like their RDBMS counterparts.



RDB Concepts to NoSQL

RDBMS		MongoDB
Database	⇨	Database
Table, View	⇨	Collection
Row	⇨	Document (BSON)
Column	⇨	Field
Index	⇨	Index
Join	⇨	Embedded Document
Foreign Key	⇨	Reference
Partition	⇨	Shard

Collection is not strict about what it Stores

Schema-less

Hierarchy is evident in the design

Embedded Document to represent relation.