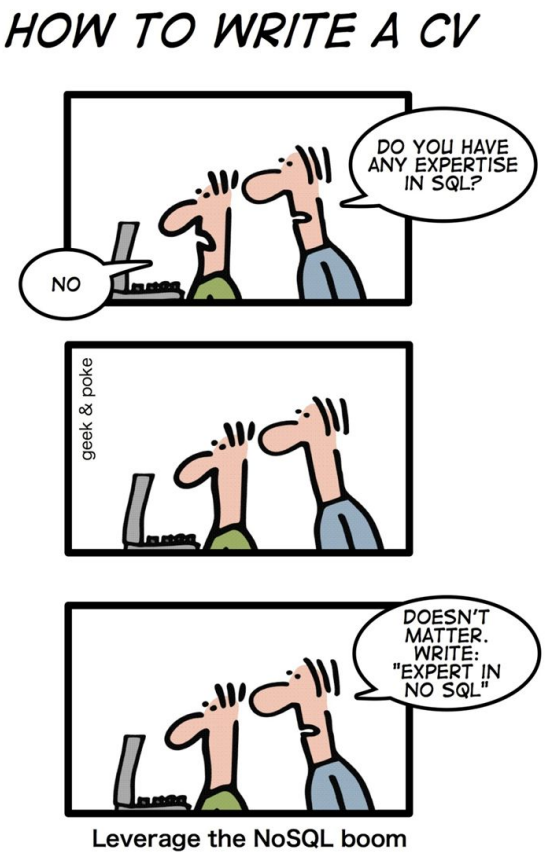
Unit 4 – Part 1
NoSQL

Credit for some of the slides in this lecture goes to Larson (Berkeley), Pattersen (UiA), Hoekstra (Perficient Inc.)

NoSQL!

NoSQL databases are currently a hot topic in some parts of computing, with over a hundred
different NoSQL databases.



RDBMS Characteristics

- Data stored in columns and tables
- Relationships represented by data
- Data Manipulation Language
- Data Definition Language
- Transactions
- Abstraction from physical layer
- Applications specify what, not how
- Physical layer can change without modifying applications
  - Create indexes to support queries
  - In Memory databases

Transactions – ACID Properties

- Atomic – All of the work in a transaction completes (commit) or none of it completes
  - a transaction to transfer funds from one account to another involves making a withdrawal operation from the first account and a deposit operation on the second. If the deposit operation failed, you don't want the withdrawal operation to happen either.
- Consistent – A transaction transforms the database from one consistent state to another consistent state. Consistency is defined in terms of constraints.
  - a database tracking a checking account may only allow unique check numbers to exist for each transaction
- Isolated – The results of any changes made during a transaction are not visible until the transaction has committed.
  - a teller looking up a balance must be isolated from a concurrent transaction involving a withdrawal from the same account. Only when the withdrawal transaction commits successfully and the teller looks at the balance again will the new balance be reported.
- Durable – The results of a committed transaction survive failures
  - A system crash or any other failure must not be allowed to lose the results of a transaction or the contents of the database. Durability is often achieved through separate transaction logs that can "re-create" all transactions from some picked point in time (like a backup).

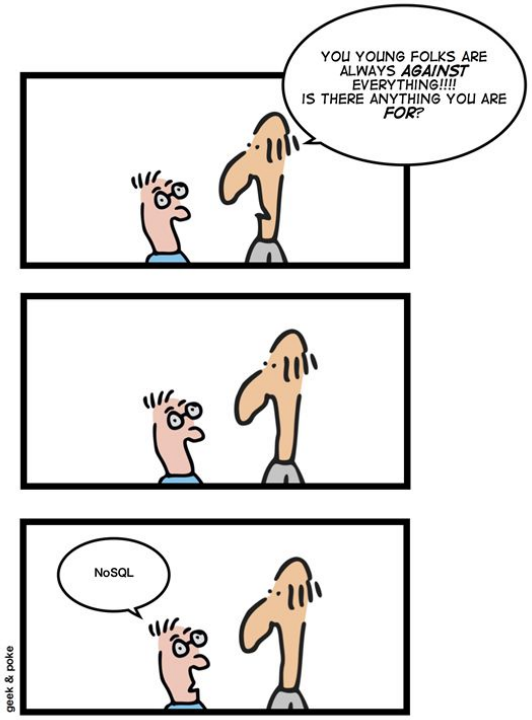
No SQL?

- NoSQL stands for:
  - No Relational
  - No RDBMS
  - Not Only SQL
- NoSQL is an umbrella term for all databases and data stores that don't follow the RDBMS principles
  - A class of products
  - A collection of several (related) concepts about data storage and manipulation
  - Often related to large data sets

NoSQL Definition

From www.nosql-database.org:

Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontal scalable. The original intention has been modern web-scale databases. The movement began early 2009 and is growing rapidly. Often more characteristics apply as: schema-free, easy replication support, simple API, eventually consistent / BASE (not ACID), a huge data amount, and more.



Where does NoSQL come from?

- Non-relational DBMSs are not new
- But NoSQL represents a new incarnation
  - Due to massively scalable Internet applications
  - Based on distributed and parallel computing
- Development
  - Starts with Google
  - First research paper published in 2003
  - Continues also thanks to Lucene's developers/Apache (Hadoop) and Amazon (Dynamo)
  - Then a lot of products and interests came from Facebook, Netfix, Yahoo, eBay, Hulu, IBM, and many more

Dynamo and BigTable

- Three major papers were the seeds of the NoSQL movement
  - BigTable (Google)
  - Dynamo (Amazon)
    - Distributed key-value data store
    - Eventual consistency
  - CAP Theorem
    - *The CAP theorem states that distributed databases can have **at most two of the three properties**: consistency, availability, and partition tolerance. As a result, database systems prioritize only two properties at a time*

NoSQL and Big Data

- NoSQL comes from Internet, thus it is often related to the “big data” concept
- How much big are “big data”?
  - Over few terabytes Enough to start spanning multiple storage units
- Challenges
  - Efficiently storing and accessing large amounts of data is difficult, even more considering fault tolerance and backups
  - Manipulating large data sets involves running immensely parallel processes
  - Managing continuously *evolving schema* and metadata for *semi-structured and un-structured* data is difficult

How did we get here?

- Explosion of social media sites (Facebook, Twitter) with large data needs
- Rise of cloud-based solutions such as Amazon S3 (simple storage solution)
- Just as moving to dynamically-typed languages (Python, Ruby, Groovy), a shift to dynamically-typed data with frequent schema changes
- Open-source community

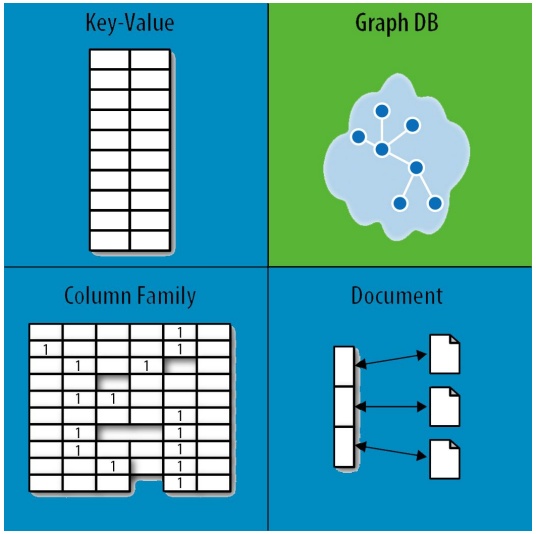
Why are RDBMS not suitable for Big Data

- The context is Internet
- RDBMSs assume that data are
  - Dense
  - Largely uniform (structured data)
- Data coming from Internet are
  - Massive and sparse
  - Semi-structured or unstructured
- With massive sparse data sets, the typical storage mechanisms and access methods get stretched

NoSQL Distinguishing Characteristics

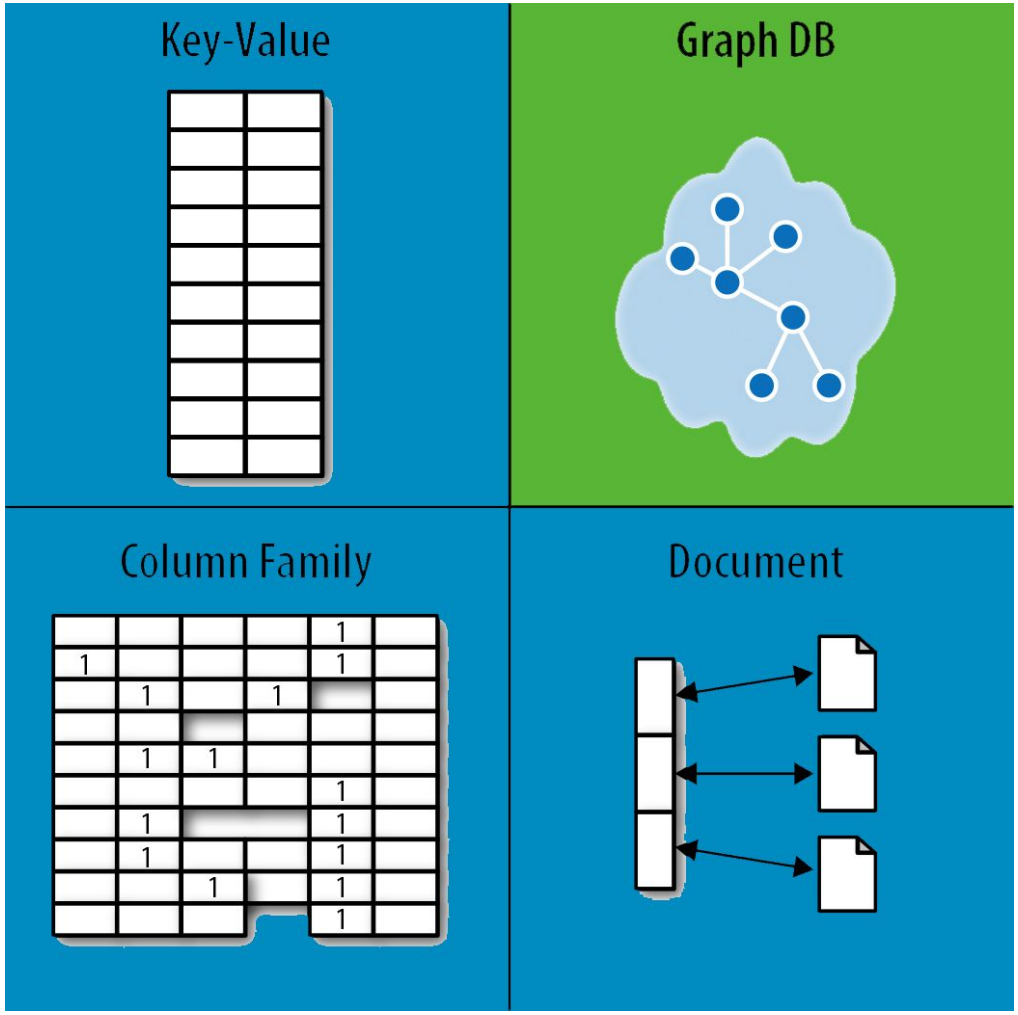
- Large data volumes
  - Google's "big data"
- Scalable replication and distribution
  - Potentially thousands of machines
  - Potentially distributed around the world
- Queries need to return answers quickly
- Mostly query, few updates
- Asynchronous Inserts & Updates
- Schema-less
- ACID transaction properties are not needed – BASE
- CAP Theorem
- Open source development

NoSQL Database Types



Discussing NoSQL databases is complicated because there are a variety of types:

- Sorted ordered Column Store
  - Optimized for queries over large datasets, and store columns of data together, instead of rows
- Document databases:
  - pair each key with a complex data structure known as a document.
- Key-Value Store :
  - are the simplest NoSQL databases. Every single item in the database is stored as an attribute name (or 'key'), together with its value.
- Graph Databases :
  - are used to store information about networks of data, such as social connections.


Key-Value
Graph DB
Column Family
Document

Document Databases (Document Store)

- Documents
  - Loosely structured sets of key/value pairs in documents, e.g., XML, JSON, BSON
  - Encapsulate and encode data in some standard formats or encodings
  - Are addressed in the database via a unique key
  - Documents are treated as a whole, avoiding splitting a document into its constituent name/value pairs
- Allow documents retrieving by keys or contents
- Notable for:
  - MongoDB (used in FourSquare, Github, and more)
  - CouchDB (used in Apple, BBC, Canonical, Cern, and more)

Document Databases (Document Store)

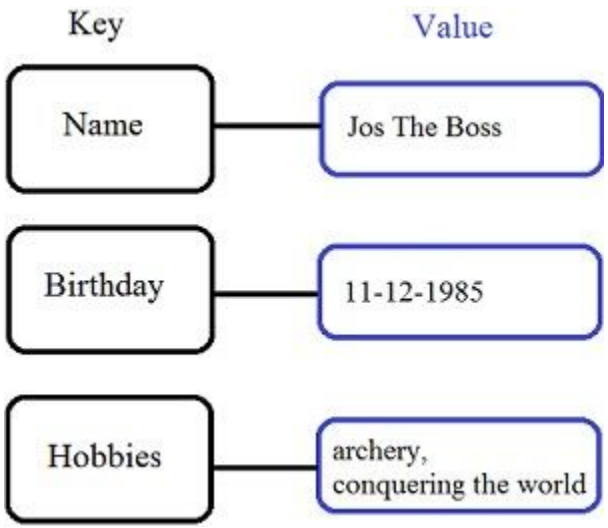
- The central concept is the notion of a "document“ which corresponds to a row in RDBMS.
- A document comes in some standard formats like JSON (BSON).
- Documents are addressed in the database via a unique *key* that represents that document.
- The database offers an API or query language that retrieves documents based on their contents.
- Documents are schema free, i.e., different documents can have structures and schema that differ from one another. (An RDBMS requires that each row contain the same columns.)

Document Databases, JSON

```
{
        _id: ObjectId("51156a1e056d6f966f268f81"),
        type: "Article",
        author: "Derick Rethans",
        title: "Introduction to Document Databases with MongoDB",
        date: ISODate("2013-04-24T16:26:31.911Z"),
        body: "This arti…"
},
{
        _id: ObjectId("51156a1e056d6f966f268f82"),
        type: "Book",
        author: "Derick Rethans",
        title: "php|architect's Guide to Date and Time Programming with PHP",
        isbn: "978-0-9738621-5-7"
}
```

Key/Value stores

- Store data in a schema-less way
- Store data as maps
  - HashMaps or associative arrays
  - Provide a very efficient average running time algorithm for accessing data
- Notable for:
  - Couchbase (Zynga, Vimeo, NAVTEQ, ...)
  - Redis (Craiglist, Instagram, StackOverfow, flickr, ...)
  - Amazon Dynamo (Amazon, Elsevier, IMDb, ...)
  - Apache Cassandra (Facebook, Digg, Reddit, Twitter,...)
  - Voldemort (LinkedIn, eBay, ...)
  - Riak (Github, Comcast, Mochi, ...)



Sorted Ordered Column-Oriented Stores

- Data are stored in a column-oriented way
  - Data efficiently stored
  - Avoids consuming space for storing nulls
  - Columns are grouped in column-families
  - Data isn't stored as a single table but is stored by column families
  - Unit of data is a set of key/value pairs
    - Identified by "row-key"
    - Ordered and sorted based on row-key
- Notable for:
  - Google's Bigtable (used in all Google's services)
  - HBase (Facebook, StumbleUpon, Hulu, Yahoo!, ...)

Sales

Product	Customer	Date	Sale
Beer	Thomas	2011-11-25	2 GBP
Beer	Thomas	2011-11-25	2 GBP
Vodka	Thomas	2011-11-25	10 GBP
Whiskey	Christian	2011-11-25	5 GBP
Whiskey	Christian	2011-11-25	5 GBP
Vodka	Alexei	2011-11-25	10 GBP
Vodka	Alexei	2011-11-25	10 GBP

Product

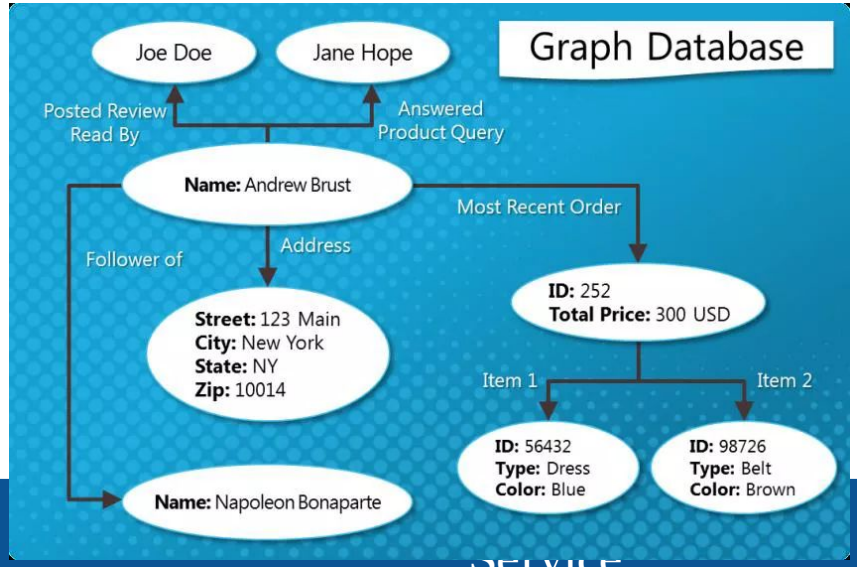
ID	Value
1	Beer
2	Beer
3	Vodka
4	Whiskey
5	Whiskey
6	Vodka
7	Vodka

Customer

ID	Customer
1	Thomas
2	Thomas
3	Thomas
4	Christian
5	Christian
6	Alexei
7	Alexei

Graph Databases

- Graph-oriented
- Everything is stored as an edge, a node or an attribute.
- Each node and edge can have any number of attributes.
- Both the nodes and edges can be labelled.
- Labels can be used to narrow searches.



Dealing with Big Data and Scalability

- Issues with scaling up when the dataset is just too big
- RDBMS were not designed to be distributed
- Traditional DBMSs are best designed to run well on a “single” machine
  - Larger volumes of data/operations requires to upgrade the server with faster CPUs or more memory known as ‘scaling up’ or ‘Vertical scaling’
- NoSQL solutions are designed to run on clusters or multi-node database solutions
  - Larger volumes of data/operations requires to add more machines to the cluster, Known as ‘scaling out’ or ‘horizontal scaling’
  - Different approaches include:
    - Master-slave
    - Sharding (partitioning)

Scaling RDBMS

- Master-Slave
  - All writes are written to the master. All reads performed against the replicated slave databases
  - Critical reads may be incorrect as writes may not have been propagated down
  - Large data sets can pose problems as master needs to duplicate data to
- Sharding
  - Any DB distributed across multiple machines needs to know in what machine a piece of data is stored or must be stored
  - A sharding system makes this decision for each row, using its key

NoSQL, No ACID

- RDBMSs are based on ACID (Atomicity, Consistency, Isolation, and Durability) properties
- NoSQL
  - Does not give importance to ACID properties
  - In some cases completely ignores them
- In distributed parallel systems it is difficult/impossible to ensure ACID properties
  - Long-running transactions don't work because keeping resources blocked for a long time is not practical

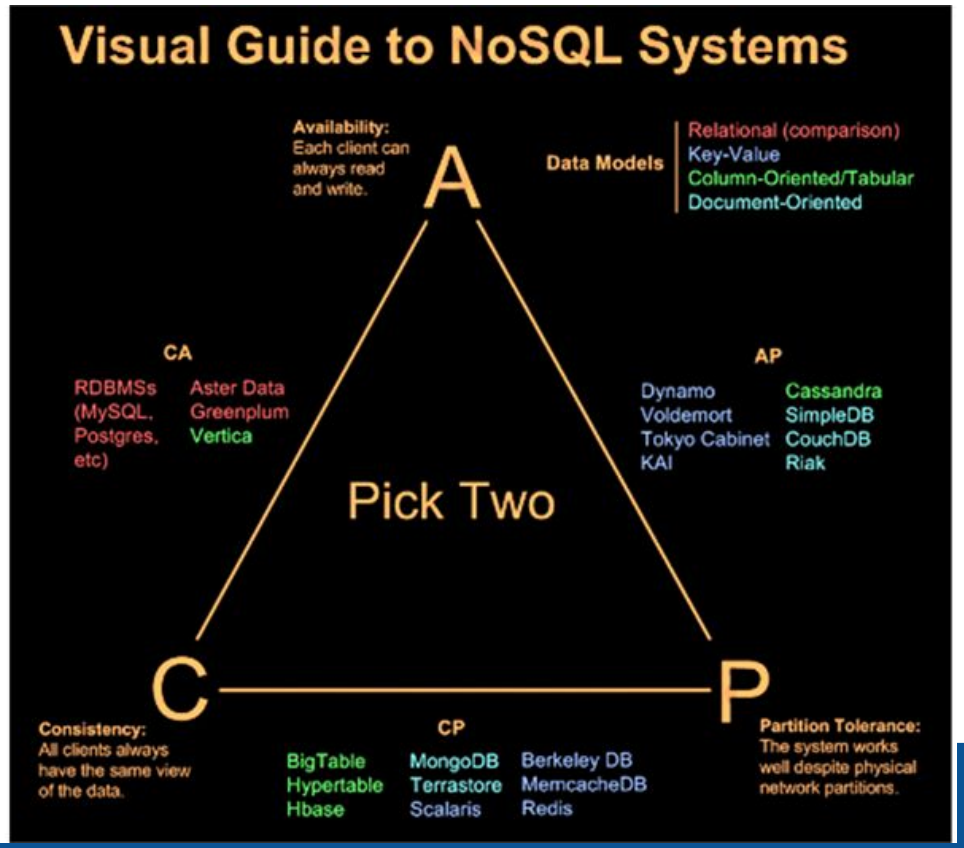
BASE Transactions

- Acronym contrived to be the opposite of ACID
  - **B**asically **A**vailable,
  - **S**oft state,
  - **E**ventually Consistent
- Characteristics
  - Weak consistency – stale data OK
  - Availability first
  - Best effort
  - Approximate answers OK
  - Aggressive (optimistic)
  - Simpler and faster

CAP Theorem

A congruent and logical way for assessing the problems involved in assuring ACID-like guarantees in distributed systems is provided by the CAP theorem
At most two of the following three can be maximized at one time

- Consistency
  - Each client has the same view of the data
- Availability
  - Each client can always read and write
- Partition tolerance
  - System works well across distributed physical networks



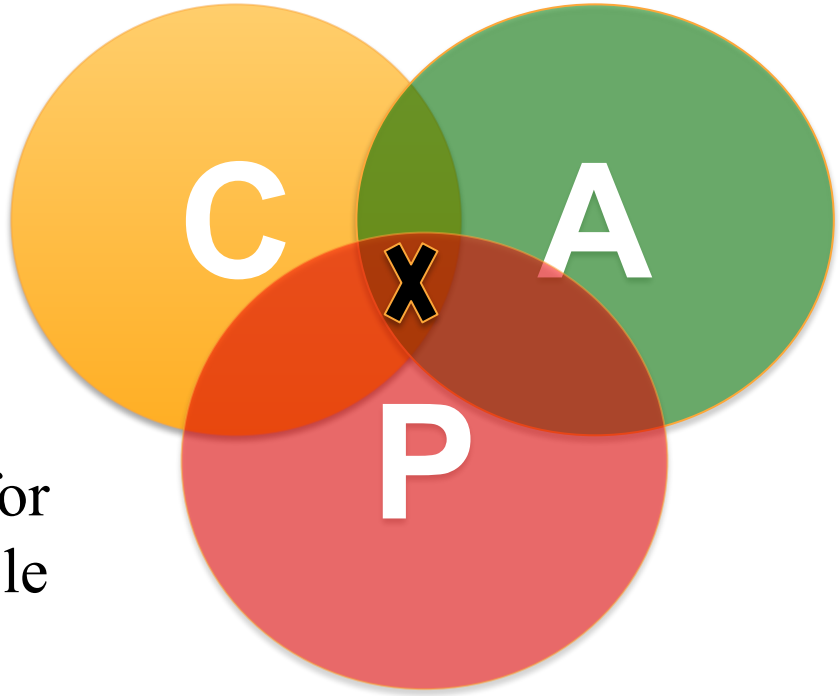
CAP Theorem: Two out of Three

- CAP theorem – At most two properties on three can be addressed
- The choices could be as follows:

1. Availability is compromised but consistency and partition tolerance are preferred over it
2. The system has little or no partition tolerance. Consistency and availability are preferred
3. Consistency is compromised but systems are always available and can work when parts of it are partitioned

Consistency or Availability

- Consistency and Availability is not "binary" decision
- AP systems relax consistency in favor of availability – but are not inconsistent
- CP systems sacrifice availability for consistency- but are not unavailable
- This suggests both AP and CP systems can offer a degree of consistency, and availability, as well as partition tolerance



Performance

- There is no perfect NoSQL database
- Every database has its advantages and disadvantages
  - Depending on the type of tasks (and preferences) to accomplish
- NoSQL is a set of concepts, ideas, technologies, and software dealing with
  - Big data
  - Sparse un/semi-structured data
  - High horizontal scalability
  - Massive parallel processing
- Different applications, goals, targets, approaches need different NoSQL solutions

Where would I use it?

- Where would I use a NoSQL database?
- Do you have somewhere a large set of uncontrolled, unstructured, data that you are trying to fit into a RDBMS?
  - Log Analysis
  - Social Networking Feeds (many firms hooked in through Facebook or Twitter)
  - External feeds from partners
  - Data that is not easily analyzed in a RDBMS such as time-based data
  - Large data feeds that need to be massaged before entry into an RDBMS

Don’t forget about the DBA

- It does not matter if the data is deployed on a NoSQL platform instead of an RDBMS.
- Still need to address:
  - Backups & recovery
  - Capacity planning
  - Performance monitoring
  - Data integration
  - Tuning & optimization
- What happens when things don’t work as expected and nodes are out of sync or you have a data corruption occurring at 2am?
- Who you gonna call?
  - DBA and SysAdmin need to be on board

The Perfect Storm

- Large datasets, acceptance of alternatives, and dynamically-typed data has come together in a perfect storm
- Not a backlash/rebellion against RDBMS
- SQL is a rich query language that cannot be rivaled by the current list of NoSQL offerings
  - So you have reached a point where a read-only cache and write-based RDBMS isn't delivering the throughput necessary to support a particular application.
  - You need to examine alternatives and what alternatives are out there.
  - The NoSQL databases are a pragmatic response to growing scale of databases and the falling prices of commodity hardware.

Summary

- Most likely, 10 years from now, the majority of data is still stored in RDBMS.
- Leading users of NoSQL datastores are social networking sites such as Twitter, Facebook, LinkedIn, and Digg.
- Not every problem is a nail and not every solution is a hammer.
- NoSQL has taken a field that was "dead" (database development) and suddenly brought it back to life.