NYU/Poly

Enterprise Data Systems
MoT
Tandon School of Engineering

Why non-Relational?

Raman Kannan
rk1750@nyu.edu

RDBMS is hugely successful

- Market cap of oracle, db2, mysql,mssql,sybase
- Number of people employed
- Number of business workflows facilitated
- Entire industries
  - Finance
  - Healthcare
  - pos
  - Retail
  - travel

Relational databases are the building blocks for most digital information structures, not to mention the basis for big name software such as Oracle and NoSQL. Infoworld noticed that the high demand in Big Data requires

http://arnoldit.com/wordpress/?s=relational+db&submit=Search

IDC's Database Market Share Analysis

June 28, 2008

IDC's Chris Kanaracus has summarized the relational database market size in "Oracle Maintains Lead in Database Market". You can read the full round up here. The total market tallies an estimated $19 billion. For me, the most important data in the news story is:

> *Oracle once again took the top spot, capturing 44.3 percent of the market with revenue growth of 13.3 percent. IBM came in second with a 21 percent share, also logging a 13.3 percent revenue growth rate. It was followed by Microsoft, with 18.5 percent of the market and a 14 percent jump in revenue. Sybase and Teradata rounded out the top five, garnering market shares of 3.5 percent and 3.3 percent, respectively.*

So why then?

- World has moved on...
  - Data is not always tabular but also xml/html/fb/tweets
  - world is mostly unstructured and conversational not always structured and transactional
- Businesses interact in new and many different ways
  - Social media & Customer behavior analysis
- Technology improvements--> data volume explosion
  - Strain due to object to relational translation is amplified –

Interesting read

http://www.infoworld.com/article/2616126/

Application-development/

10-things-never-to-do-with-a-relational-database.html

If you need a plane, stapling a flock of birds to a

truck is not a strategy

http://arnoldit.com/wordpress/2008/06/28/

idcs-database-market-share-analysis/

RDBMS → 19 billion dollar ostrich (It aint dead yet..)

http://www.infoworld.com/article/2617876/database/

which-freaking-database-should-i-use-.html

http://download.microsoft.com/download/6/C/D/

6CDC123B-A159-438C-BBAD-7750EE5D0D67/

IDC-Server%20Workloads%20Forecast.pdf

What is NoSQL

- Not only SQL
- Many variant
  - Document oriented
  - Graph database
  - Key/value
  - Wide column
- Mongo – document oriented
  - Hu-***mongo***-us
  - No transaction / no acid / no schema / no relation

more

- CAP not ACID
- Eventually consistent
- Document --> record
- Operation on one document is atomic
  - No partial update to a single document

CAP Theorem

What is CAP about?

The *(CAP) theorem* (**C**onsistency, **A**vailability and **P**artitioning tolerance) was given by Eric Brewer, a professor at the University of California, Berkeley and one of the founders of Google, in 2001 in the keynote of Principles of Distributed Computing.

The theorem states:

> ***Though its desirable to have Consistency, High-Availability and Partition-tolerance in every system, unfortunately no system can achieve all three at the same time.***

In other words a system can have at most two of three desirable properties at the same time **in presence of errors**.

http://ivoroshilin.com/2012/12/13/brewers-cap-theorem-explained-base-versus-acid/

Definitions

Consistency: A service that is consistent should follow the rule of ordering for updates that spread across all replicas in a cluster – “what you write is what you read”, regardless of location. For example, Client A writes 1 then 2 to location X, Client B cannot read 2 followed by 1. This rule has another name “Strong consistency”.

Availability: A service should be available. There should be a guarantee that every request receives a response about whether it was successful or failed. If the system is not available it can be still consistent. However, consistency and availability cannot be achieved at the same time. This means that one has two choices on what to leave. Relaxing consistency will allow the system to remain highly available under the partitioning conditions (see next definition) and strong consistency means that under certain conditions the system will not be available.

Partition tolerance: The system continues to operate despite arbitrary message loss or failure of part of the system. A simple example, when we have a cluster of N replicated nodes and for some reason a network is unavailable among some number of nodes (e.g. a network cable got chopped). This leads to inability to synchronize data. Thus, only some part of the system doesn’t work, the other one does. If you have a partition in your network, you lose either consistency (because you allow updates to both sides of the partition) or you lose availability (because you detect the error and shut down the system until the error condition is resolved).

BASE

A simple meaning of this theorem is “It is impossible for a protocol to guarantee both consistency and availability in a partition prone distributed system”.

Eventual consistency
(BASE)

The term eventual consistency or as it is called BASE (Basically Available, Soft state, Eventual consistency) is the opposite of ACID (Atomicity, Consistency, Isolation and Durability). Where ACID is pessimistic and requires consistency at the end of every operation, BASE is optimistic and accepts that the database consistency will be in a state of flux. The eventual consistency is simply an acknowledgement that there is an unbounded delay in propagating a change made on one machine to all the other copies which might lead to stale data.

Node/Read/Write

NRW (Node, Read, Write) allows to analyse and tune how a distributed database will trade off consistency, read / write performance.

N = the number of nodes that keep copies of a record distributed to.
W = the number of nodes that must successfully acknowledge a write to be successfully committed.
R = the number of nodes that must send back the same value of a unit of data for it to be accepted as read by the system.
The majority of NoSQL databases use N>W>1 – more than one write must complete, but not all nodes need to be updated immediately.
When:

W < N – high write availability
R < N – high read availability
W+R > N – is a strong consistency, read/write are fully overlapped
W+R <= N – is an eventual consistency, meaning that there is no overlap in the read and write set;

Off to the races

http://docs.mongodb.org/manual/tutorial/query-documents/