Advanced Database Technologies

NoSQL: *Not only SQL*

Christian Grün
Database & Information Systems Group
Universität Konstanz
NoSQL

Introduction

30, 40 years history of well-established database technology... all in vain? Not at all! But both setups and demands have drastically changed:

- *main memory* and CPU speed have exploded, compared to the time when *System R* (the mother of all RDBMS) was developed
- at the same time, *huge* amounts of data are now handled in real-time
- both data and use cases are getting more and more *dynamic*
- social networks (relying on *graph data*) have gained impressive momentum
- *full-texts* have always been treated shabbily by relational DBMS
NoSQL: Facebook

Statistics
royal.pingdom.com/2010/06/18/the-software-behind-facebook

- 500 million users
- 570 billion page views per month
- 3 billion photos uploaded per month
- 1.2 million photos served per second
- 25 billion pieces of content (updates, comments) shared every month
- 50 million server-side operations per second
- 2008: 10,000 servers; 2009: 30,000, ...
☞ One RDBMS may not be enough to keep this going on!
NoSQL: Facebook

Architecture

Memcached
- distributed memory caching system
- caching layer between web and database servers
- based on a distributed hash table (DHT)

HipHop for PHP
- developed by Facebook to improve scalability
- compiles PHP to C++ code, which can be better optimized
- PHP runtime system was reimplemented
NoSQL: Facebook

Architecture

Cassandra
- developed by Facebook for inbox searching
- data is automatically replicated to multiple nodes
- no single point of failure (all nodes are equal)

Hadoop/Hive
- implementation of Google’s MapReduce framework
- performs data-intensive calculations
- (initially) used by Facebook for data analysis
NoSQL: Facebook

Architecture

Varnish
• HTTP accelerator, speeds up dynamic web sites

Haystack
• object store, used for storing and retrieving photos

BigPipe
• web page serving system; serves parts of the page (chat, news feed, ...)

Scribe
• aggregates log data from different servers
NoSQL: Facebook

Architecture: Hadoop Cluster

hadoopblog.blogspot.com/2010/05/facebook-has-worlds-largest-hadoop.html

- 21 PB in Data Warehouse cluster, spread across 2000 machines:
- 1200 machines with 8 cores, 800 machines with 16 cores
- 12 TB disk space per machine, 32 GB RAM per machine
- 15 map-reduce tasks per machine

Workload

- daily: 12 TB of compressed data, and 800 TB of scanned data
- 25,000 map-reduce jobs and 65 million files per day
NoSQL: Facebook

Architecture

HBase
• Hadoop database, used for e-mails, IM and SMS
• has recently replaced MySQL, Cassandra and few others
• built on Google’s BigTable model

Conclusion
• classical database solutions have turned out to be completely insufficient
• heterogeneous software architecture is needed to match all requirements
NoSQL: *Not only SQL*

...do we need a Facebook-like architecture for conventional tasks?

**Thoughts**

Depends on problems to be solved:

- RDBMS are still a great solution for centralized, tabular data sets
- NoSQL gets interesting if data is heterogeneous and/or too large
- most NoSQL projects are open source and have open communities
- code bases are up-to-date (no 30 years old, closed legacy code)
- they are subject to rapid development and change
- cannot offer general-purpose solutions yet, as claimed by RDBMS
NoSQL: *Not only SQL*

10 Things: Five Advantages
www.techrepublic.com/blog/10things/10-things-you-should-know-about-nosql-databases/1772

Elastic Scaling
- *scaling out:* distributing data instead of buying bigger servers

Big Data
- opens *new dimensions* that cannot be handled with RDBMS

Goodbye DBAs (see you later?)
- *automatic* repair, distribution, tuning, ...

Economics
- based on *cheap commodity* servers
- less costs per transaction/second

Flexible Data Models
- non-existing/relaxed data schema
- structural changes cause no overhead
NoSQL: *Not only SQL*

10 Things: Five Challengees
www.techrepublic.com/blog/10things/10-things-you-should-know-about-nosql-databases/1772

Maturity
- still in *pre-production* phase, key features yet to be implemented

Support
- mostly open source, *start-ups*, limited resources or credibility

Analytics and Business Intelligence
- focused on *web apps* scenarios
- limited *ad-hoc* querying

Expertise
- few number of NoSQL experts available in the market

Administration
- require lot of *skill to install* and effort to maintain
NoSQL: Concepts

Definition

nosql-database.org

“Next Generation Databases mostly addressing some of the points: being non-relational, distributed, open-source and horizontally 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.”

Stefan Edlich

History

1998 term was coined by NoSQL RDBMS, developed by Carlo Strozzi
2009 reintroduced on a meetup for “open source, distributed, non relational databases”, organized by last.fm developer Johan Oskarsson
NoSQL: Concepts

Scalability

System can handle growing amounts of data without losing performance.

Vertical Scalability (*scale up*)
- add resources (more CPUs, more memory) to a single node
- using more threads to handle a local problem

Horizontal Scalability (*scale out*)
- add nodes (more computers, servers) to a distributed system
- gets more and more popular due to low costs for commodity hardware
- often surpasses scalability of vertical approach
NoSQL: Concepts

CAP Theorem: Consistency, Availability, Partition Tolerance
Brewer [2000]: Towards Robust Distributed Systems

Consistency
- after an update, all readers in a distributed system see the same data
- all nodes are supposed to contain the same data at all times

Example
- single database instance will always be consistent
- if multiple instances exist, all writes must be duplicated before write operation is completed
NoSQL: Concepts

CAP Theorem: Consistency, Availability, Partition Tolerance
Brewer [2000]: Towards Robust Distributed Systems

**Availability**
- all requests will be answered, regardless of crashes or downtimes

**Example**
- a single instance has an availability of 100% or 0%
- two servers may be available 100%, 50%, or 0%

**Partition Tolerance**
- system continues to operate, even if two sets of servers get isolated

**Example**
- system gets partitioned if connection between server clusters fails
- failed connection won’t cause troubles if system is tolerant
NoSQL: Concepts

CAP Theorem: Consistency, Availability, Partition Tolerance

Brewer [2000]: Towards Robust Distributed Systems

• Theorem (proven 2002): only 2 of the 3 guarantees can be given in a “shared-data” system

• (Positive) consequence: we can concentrate on two challenges

• ACID properties needed to guarantee consistency and availability

• BASE properties come into play if availability and partition tolerance is favored
NoSQL: Concepts

ACID: Atomicity, Consistency, Isolation, Durability
Härder & Reuter [1983]: Principles of Transaction-Oriented Database Recovery

Atomicity
• all operations in the transaction will complete, or none will

Consistency
• before and after transactions, database will be in a consistent state

Isolation
• operations cannot access data that is currently modified

Durability
• data will not be lost upon completion of a transaction
NoSQL: Concepts

BASE: Basically Available, Soft State, Eventual Consistency
Fox et al. [1997]: Cluster-Based Scalable Network Services

Basically Available
• an application works basically all the time (despite partial failures)

Soft State
• is in flux and non-deterministic (changes all the time)

Eventual Consistency
• will be in some consistent state (at some time in future)

Why is BASE a reasonable paradigm for e.g. web search?
NoSQL: Concepts

MapReduce Framework
Dean & Ghemawat [2004]: MapReduce: Simplified Data Processing on Large Clusters

• developed by Google to replace old, centralized index structure
• supports distributed, parallel computing on large data sets
• inspired by (...not equal to...) the map and fold functions in functional programming: no side effects, deadlocks, race conditions
• divide-and-conquer paradigm:
  Map recursively breaks down a problem into sub-problems and distributes it to worker nodes; Reduce receives and combines the sub-answers to solve the problem
NoSQL: Concepts

Map/Fold in XQuery 3.0

- map() function:
  
  ```xquery
  let $func := function($a){$a * $a}
  return map($func, 1 to 4)
  ```

  Result: 1 4 9 16

- fold-left() function:
  
  ```xquery
  let $func := function($a, $b){$a * $b}
  return fold-left($func, 1, 1 to 4)
  ```

  Same as: (((((1 * 1) * 2) * 3) * 4)
  
  Result: 24
NoSQL: Concepts

MapReduce: Functions

Input & Output: each a set of key/value pairs.
Programmer specifies two functions:

- \( \text{map}(\text{in\_key}, \text{in\_value}) \rightarrow \text{list}(\text{out\_key}, \text{intermed\_value}) \): processes input key/value pair, produces set of intermediate pairs

- \( \text{reduce}(\text{out\_key}, \text{list}(\text{intermed\_value})) \rightarrow \text{list}(\text{out\_value}) \): combines all intermediate values for a particular key, produces a set of merged output values (usually just one)
NoSQL: Concepts

MapReduce: Example

All-time favorite: count word occurrence

map(String input_key, String input_value):

# input_key: document name
# input_value: document contents
for each word w in input_value:
    EmitIntermediate(w, "1")

reduce(String output_key, Iterator intermed_values):

# output_key: a word
# output_values: a list of counts
int result = 0
for each v in intermed_values:
    result += ParseInt(v)
Emit(AsString(result))

• programmer can focus on map/reduce code
• framework cares about parallelization, fault tolerance, scheduling, ...
NoSQL: Concepts

MapReduce: Example

blog.jteam.nl/2009/08/04/introduction-to-hadoop
NoSQL: Concepts

MapReduce: Execution
Dean & Ghemawat [2004]

- input files are stored on map worker disks
- master assigns tasks to workers
- map workers write on local disk
- reduce workers retrieve data and write result (...may be recursively repeated)
NoSQL: Concepts

MapReduce: File System

- often based on a custom file system, which is optimized for distributed access (Google: GFS, Hadoop: HDFS)
- uses few large files, as data throughput >> access time

Summary

- MapReduce framework can be written for different environments: shared memory, distributed networks, ...
- used in a wide range of scenarios: distributed search (grep), creation of word indexes, sorting distributed data, count page links, ...