NoSQL And XML Databases

Michal Valenta

Katedra softwarového inženýrství FIT
České vysoké učení technické v Praze

Java BootCamp, 10.11. 2011

What Are “Traditional DBs”?

- OLTP (at the beginning)
  - B-trees, write-itensive
  - row-level storage, views
- data warehouses (later)
  - bit-map indexes, query intensive
  - ad-hoc queries, materialized views
- data model extension (to overcome “semantic gap“) – ORDBMS
- but still DBMS’s like PostgreSQL, Oracle, MySQL, MS-SQL,... use
  - one source code
  - one interface (SQL)
  - cost-based optimization

M. Stonebraker, U. Centinemel article

“One Size Fits All”: An Idea Whose Time Has Come and Gone

Where ”Traditional DBs“ Are Not Sufficient Enough?

New areas of application of DBs

- data warehouses
- stream processing
- scientific databases
- XML storages, document storages
- web applications

What about ”Traditional DBs“ and additional technologies?

- SQL extensions (object references, text search, XML precessing, spatial querying, DW operations, ...)
- Data model extensions (LOBs, structures, sets, UDT, methods, object viesws, ...)
- OR mapping layers (Hibernate, Ibatis, ...)

NoSQL – Several Case Studies

There are many case studies, articles, blogs and talks pointing out weakness of ”traditional DB's“

We will very briefly present 3 of them:

1. Stream processing. Outbound versus inbound processing). According to article “One Size Fits All”: An Idea Whose Time Has Come and Gone by M. Stonebraker and U. Centinemel.
2. Web application. Redis Twitter Example. A talk by Karel Minařík and Tomáš Vondra on CSPUG meeting.
3. MapReduce principle for querying. An example of map-reduce to implement a query.

Stream Processing - An Example

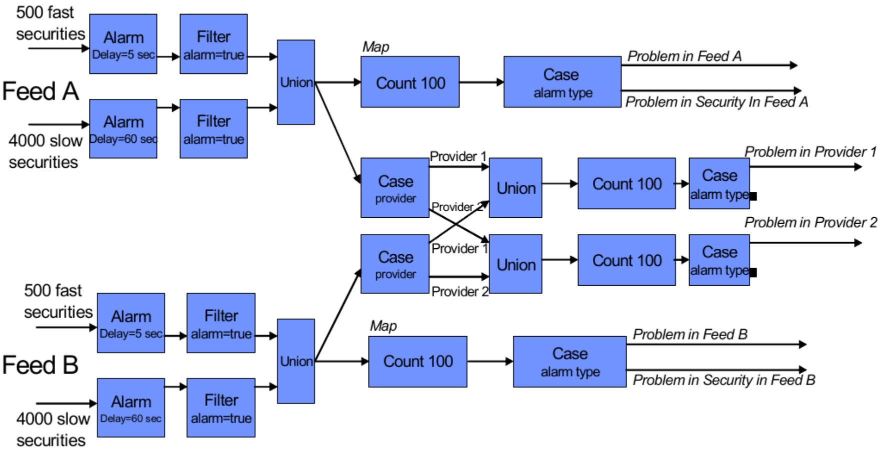


Figure: An Expriment by M. Stonebraker and U. Centinemel

An Expriment by M. Stonebraker and U. Centinemel

Implemented in traditional RDBMS and in streambase processing engine (SPE) on 2.8GHz Pent., 512MB, SCSI HD.

- SPE - 160.000 messages per second
- RDBMS - 900 messages per second

- outbound processing
  - stores data, execute queries – pull model – traditional RDBMS
- inbound processing
  - stores queries, passes data through – push model – SPE
- the end of aggregation in SQL ?
- windowing, loss messages detection in SQL ?
- client-server versus embeded architecture

- triggers in RDBMS partially implement push model
- stored procedures and OO partially implement embeded architecture

Twitter Example In Redis

Features (of Redis):

- key-value approach
- data structures (strings, lists, sets, sorted sets, hashes)
- very efficient operations on data structures
- denormalization

Objectives of example

- simulation of twitter operations: twitter, follower, messaging
- well-suited example for Redis (everything much more problematic in SQL)

See:
http://karmi.github.com/redis_twitter_example/ for commented example.
http://www.slideshare.net/karmi/redis-the-ak47-of-postrelational-databases for Redis overview.

Map-reduce Principle

- used in many (not all) NoSQL DBs (BigTable and CouchDB for example)
- naturally allows query distribution and parallel processing
- supports scalability of DB (large data sests on clusters)
- introduced by Google

map-reduce description

Map step The master node takes the input, partitions it up into smaller sub-problems, and distributes those to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes that smaller problem, and passes the answer back to its master node.

$$Map(k1, v1) \rightarrow list(k2, v2)$$

Reduce step The master node then takes the answers to all the sub-problems and combines them in some way to get the output — the answer to the problem.
All that is required is that all outputs of the map operation which share the same key are presented to the same reducer.

$$Reduce(k2, list(v2)) \rightarrow list(v3)$$

map-reduce – counting words – a canonical example

```
void map(String name, String document):
  // name: document name
  // document: document contents
  for each word w in document:
    EmitIntermediate(w, "1");

void reduce(String word, Iterator partialCounts):
// word: a word
// partialCounts: a list of aggreg. partial counts
  int result = 0;
  for each pc in partialCounts:
    result += ParseInt(pc);
  Emit(AsString(result));
```

See: *http://guide.couchdb.org/draft/cookbook.html* for more examples.

So, what are the basic features of NoSQL DBs?

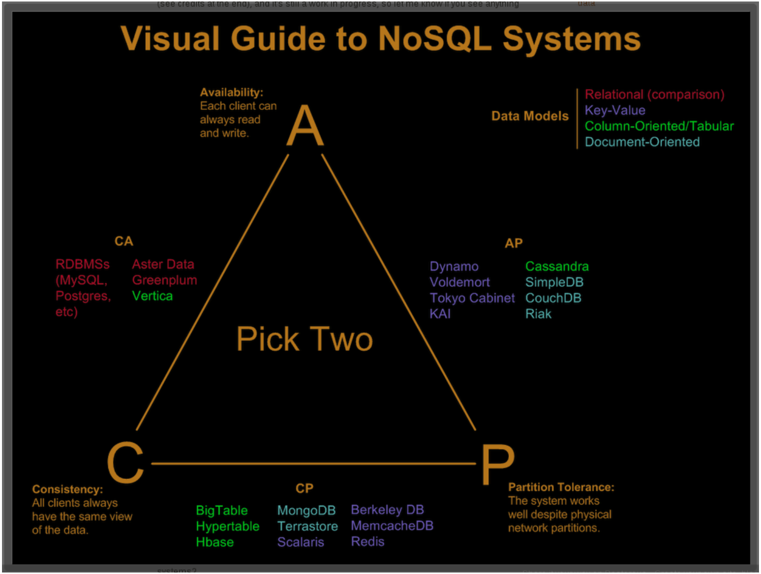
- non-relational
- distributed
- horizontal scalable
- schema-free
- easy replication support
- simple API
- eventually consistent / BASE (not ACID)
  - BASE (Basically Available, Soft state, Eventual consistency)
- huge data amount

- Term ”NoSQL“ is now usually translated as ”not only SQL“.

NoSQL DBs classification from datamodel point of view

- Wide Column Store / Column Families
- Document Store (also XML-native)
- Key Value / Tuple Store
- Eventually Consistent Key Value Store
- Graph Databases
- Stream processing DBs

PAC Theorem presented by Nathan Hurst



XML Basics

XML – Extensible Markup Language (1998)

- By World Wide Web Consortium (w3c)
- It's a *language* → described by a grammar

Example XML document

```
<?xml version="1.0" encoding="UTF-8"?>
<lectures>
  <lecture id="5">
    <speaker> Michal Valenta </speaker>
    <name> Native XML Databases </name>
    <difficulty> low </difficulty>
  </lecture>
</lectures>
```

XML & Data Models

Relational vs. XML models mismatch

- Informally, an XML document is a *tree* or *graph*
- More formal models for XML exist – DTD, XML Schema, Infoset, PSVI, XDM
- The *difference* between these models and the relational one is obvious and crucial
- XML document classification
  - data-centric documents
  - document-centric documents
  - hybrid documents (? loose boundary)
- Another XML partitioning
  - schema annotated (DTD, XML Schema, RELAX NG)
  - schema-free
- Result: **EVERYTHING** in native XML DBMS is **MORE COMPLEX.**

Storing XML Data

Common ways to store XML documents . . .

1. File system
2. Relational database
3. Native XML storage

. . . which one is the best? Depends.

- Volume of XML data
- Data characteristics – document/data-centric XML
- Schema-free or schema-based data
- Intended usage (long-term storage, heavy-loaded transactional system, fulltext-search oriented usage, . . .)
- Round-tripping
- . . .

Principal XDB Issues

Basically very similar to RDBMS . . .

- Storage, indexing
- Querying, query languages
- Application programming interfaces
- User rights
- Transactions, locking protocols
- Distributed data processing
- . . .

Query Languages and Querying

Recall the relational world

- Relational data model + algebra + calculus
- Industrial world-wide standard: SQL
- SQL := DDL + DML + DCL + TCL
- Multiple revisions: SQL-86, SQL-89, . . . , SQL:2008

XML & XDB world

- Multiple data models
- Standards set (almost exclusively) by W3C
- XPath, XQuery, XSLT, XML Schema
- Nowadays two versions from each spec exist, implemented usually only to some extent

APIs: Application Programming Interfaces

- Provide programming access to DBMS's functionality
- Standard XML equivalents to ODBC/JDBC do not exist yet
- Various proposals appear: XML:DB, XQJ
- Typical solution: proprietary API in common languages available

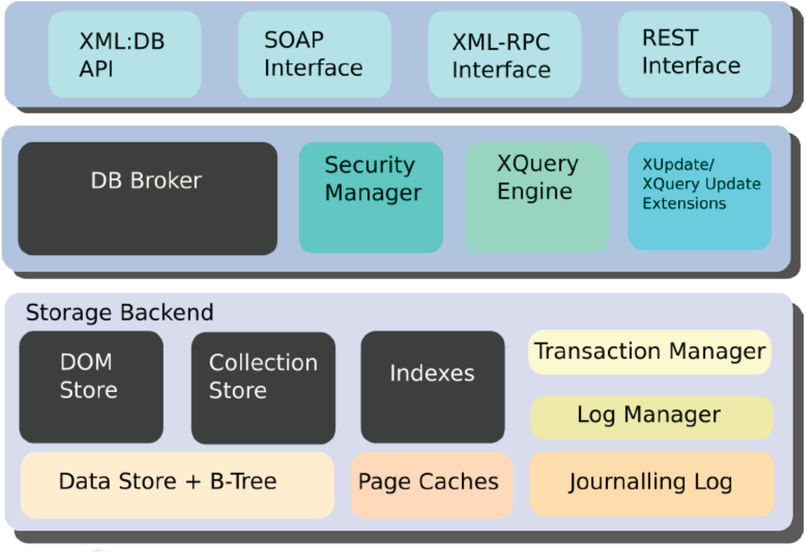
XML Database	License	Language	XQJ API	XML:DB API	RESTful API
BaseX	BSD License	Java	Yes	Yes	Yes
EMC xDB	Commercial	Java	No	No	Yes
eXist	LGPL License	Java	No	Yes	Yes
Gemfire Enterprise	Commercial	Unknown	No	Yes	No
MarkLogic Server	Commercial	C++	Yes	No	Yes
MonetDB/XQuery	Proprietary	C++	No	Yes	No
Oracle	Commercial	PL/SQL	Yes	No	No
Qizx	Commercial	Java	No	No	Yes
Sedna	Apache License	C++	Yes	Yes	No
Tamino	Commercial	Unknown	No	Partial	No
Xindice	Apache License	Java	No	Yes	No

eXist

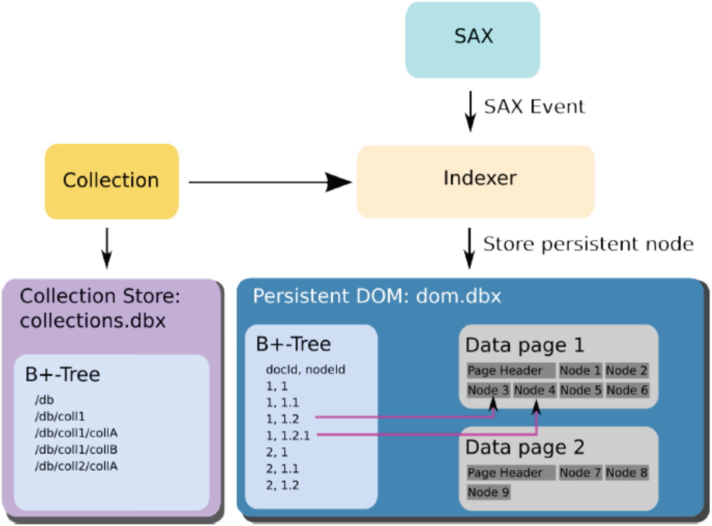
Overview and highlights

- Feature-rich open source XDB written in Java
- Uses B+ trees and paged files; document nodes are stored as persistent DOM
- Wide range of APIs: http REST, XML-RPC, SOAP, WebDAV, XML:DB API
- XQuery 1.0 processor with extensive function library
- Ideal for backing the XRX architecture (XForms-REST-XQuery)

eXist Architecture



eXist Data Storage



Conclusion – key properties, application domains

NoSQL Databases

- everything simple: data model, API
- scalability, huge amount of data, minimal latency
- **weakness:** joins, transactions, complex queries

XML Databases

- more flexible data model
- powerful query language
- **weakness:** efficiency, transactional processing

Traditional DBs

- strict data model
- efficiency on complex operations, transactional processing
- **weakness:** scalability, too universal

References:

- **XML technologie. Principy a aplikace v praxi.** Mlynkova, I., Pokorny , J., Richta, K., Toman, K., Toman, V.: Grada Publishing, a.s. Praha, 2008.
- **NoSQL Databáze.** J. Pokorný. DATAKON 2011
- **What the heck are you actually using NoSQL for?** Todd Hoff *http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql-for.html*
- **"One Size Fits All": An Idea Whose Time Has Come and Gone.** Michael Stonebraker and Ugur Cetintemel *http://www.cs.brown.edu/ ugur/fits_all.pdf*
- **Visual Guide to NoSQL Systems.** Nathan Hurst *http://blog.nahurst.com/visual-guide-to-nosql-systems*