THE NOSQL MOUVEMENT (2)

GENOVEVA VARGAS SOLAR
FRENCH COUNCIL OF SCIENTIFIC RESEARCH, LIG-LAFMIA, FRANCE
Genoveva.Vargas@imag.fr
http://www.vargas-solar.com/bigdata-management
so now we have NoSQL databases

Data stores designed to scale simple OLTP-style application loads

- Data model
- Consistency
- Storage
- Durability
- Availability
- Query support

Read/Write operations by thousands/millions of users

examples include

mongoDB
CouchDB
riak
Cassandra
Neo4j
HBase
redis

We should also remember Google’s Bigtable and Amazon’s SimpleDB. While these are tied to their host’s cloud service, they certainly fit the general operating characteristics.
Use the right tool for the right job…

How do I know which is the right tool for the right job?

(Katsov-2012)
PROBLEM STATEMENT: HOW MUCH TO GIVE UP?

- CAP theorem\(^1\): a system can have two of the three properties
- NoSQL systems sacrifice *consistency*

## Comparing NoSQL & NewSQL Systems

<table>
<thead>
<tr>
<th>System</th>
<th>Concurrency Control</th>
<th>Data Storage</th>
<th>Replication</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Redis</td>
<td>Locks</td>
<td>RAM</td>
<td>Asynchronous</td>
<td>No</td>
</tr>
<tr>
<td>Scalaris</td>
<td>Locks</td>
<td>RAM</td>
<td>Synchronous</td>
<td>Local</td>
</tr>
<tr>
<td>Tokyo</td>
<td>Locks</td>
<td>RAM/Disk</td>
<td>Asynchronous</td>
<td>Local</td>
</tr>
<tr>
<td>Voldemort</td>
<td>MVCC</td>
<td>RAM/BDB</td>
<td>Asynchronous</td>
<td>No</td>
</tr>
<tr>
<td>Riak</td>
<td>MVCC</td>
<td>Plug in</td>
<td>Asynchronous</td>
<td>No</td>
</tr>
<tr>
<td>Membase</td>
<td>Locks</td>
<td>Flash+Disk</td>
<td>Synchronous</td>
<td>Local</td>
</tr>
<tr>
<td>Membase</td>
<td>Locks</td>
<td>Disk</td>
<td>Synchronous</td>
<td>Local</td>
</tr>
<tr>
<td>Dynamo</td>
<td>MVCC</td>
<td>Plug in</td>
<td>Asynchronous</td>
<td>No</td>
</tr>
<tr>
<td>SimpleDB</td>
<td>Non</td>
<td>S3</td>
<td>Asynchronous</td>
<td>No</td>
</tr>
<tr>
<td>MongoDB</td>
<td>Locks</td>
<td>Disk</td>
<td>Asynchronous</td>
<td>No</td>
</tr>
<tr>
<td>CouchDB</td>
<td>MVCC</td>
<td>Disk</td>
<td>Asynchronous</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System</th>
<th>Concurrency Control</th>
<th>Data Storage</th>
<th>Replication</th>
<th>Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Terrastore</td>
<td>Locks</td>
<td>RAM+</td>
<td>Synchronous</td>
<td>L</td>
</tr>
<tr>
<td>Hbase</td>
<td>Locks</td>
<td>HADOOP</td>
<td>Asynchronous</td>
<td>L</td>
</tr>
<tr>
<td>HyperTable</td>
<td>Locks</td>
<td>Files</td>
<td>Synchronous</td>
<td>L</td>
</tr>
<tr>
<td>Cassandra</td>
<td>MVCC</td>
<td>Disk</td>
<td>Asynchronous</td>
<td>L</td>
</tr>
<tr>
<td>BigTable</td>
<td>Locks+stamps</td>
<td>GFS</td>
<td>Both</td>
<td>L</td>
</tr>
<tr>
<td>FNIns</td>
<td>MVCC</td>
<td>Disk</td>
<td>Asynchronous</td>
<td>L</td>
</tr>
<tr>
<td>MySQL-C</td>
<td>ACID</td>
<td>Disk</td>
<td>Synchronous</td>
<td>Y</td>
</tr>
<tr>
<td>VoltDB</td>
<td>ACID/no Lock</td>
<td>RAM</td>
<td>Synchronous</td>
<td>Y</td>
</tr>
<tr>
<td>Clustrix</td>
<td>ACID/no Lock</td>
<td>Disk</td>
<td>Synchronous</td>
<td>Y</td>
</tr>
<tr>
<td>ScaleDB</td>
<td>ACID</td>
<td>Disk</td>
<td>Synchronous</td>
<td>Y</td>
</tr>
<tr>
<td>ScaleBase</td>
<td>ACID</td>
<td>Disk</td>
<td>Asynchronous</td>
<td>Y</td>
</tr>
<tr>
<td>NimbusDB</td>
<td>ACID/no Lock</td>
<td>Disk</td>
<td>Synchronous</td>
<td>Y</td>
</tr>
</tbody>
</table>

CONCLUSIONS

- Data are growing big and more heterogeneous and they need new adapted ways to be managed thus the NoSQL movement is gaining momentum
- Data heterogeneity implies different management requirements this is where polyglot persistence comes up
  - Consistency – Availability – Fault tolerance theorem: find the balance!
  - Which data store according to its data model?
  - A lot of programming implied …

Open opportunities if you’re interested in this topic!
POLYGLOT PERSISTENCE

GENOVEVA VARGAS SOLAR
FRENCH COUNCIL OF SCIENTIFIC RESEARCH, LIG-LAFMIA, FRANCE
Genoveva.Vargas@imag.fr
http://www.vargas-solar.com
Rapid access for reads and writes. No need to be durable

Needs transactional updates. Tabular structure fits data

Needs high availability across multiple locations. Can merge inconsistent writes

Rapidly traverse links between friends, product purchases, and ratings

Speculative Retailers Web Application

User sessions
Financial Data
Shopping Cart
Recommendations

Product Catalog
Reporting
Analytics
User activity logs

Lots of reads, infrequent writes. Products make natural aggregate

Large-scale analytics on large cluster

High volume of writes on multiple nodes

SQL interfaces well with reporting tools

This is a very hypothetical example, we would not make technology recommendations without more contextual information
THIS TALK IS ABOUT

Polyglot Persistence

alternative for managing multiform and multimedia data collections according to different properties and requirements
Polyglot Persistence

using multiple data storage technologies, chosen based upon the way data is being used by individual applications. Why store binary images in relational database, when there are better storage systems?

Polyglot persistence will occur over the enterprise as different applications use different data storage technologies. It will also occur within a single application as different parts of an application’s data store have different access characteristics.

http://martinfowler.com/bliki/PolyglotPersistence.html
Polyglot Programming: applications should be written in a mix of languages to take advantage of different languages are suitable for tackling different problems

Polyglot persistence: any decent sized enterprise will have a variety of different data storage technologies for different kinds of data

- a new strategic enterprise application should no longer be built assuming a relational persistence support
- the relational option might be the right one - but you should seriously look at other alternatives

DESIGNING AND BUILDING A POLYGLOT DATABASE
**OBJECTIVE**

Build a MyNet app based on a polyglot database for building an integrated directory of my contacts including their status and posts from several social networks.
Analysis on contacts networks, overlapping according to interests, posts topics
Top 10 most popular contacts

User sessions in different Social networks

Friend network

Lots of reads, infrequent writes. Products make natural aggregate

SQL interfaces well with reporting tools

User accounts activity in different social networks

Large-scale analytics on large cluster

Directory synchronisation
Integrating contacts’ information From all SN

Analysis on contacts networks, overlapping according to interests, posts topics
Top 10 most popular contacts

Users high availability across multiple locations. Can merge inconsistent writes

Integrating posts from all networks

Rapidly traverse links between friends, product purchases, and ratings

Contact graph traversal
For building groups out of Common characteristics

Rapid access for reads and writes. No need to be durable

Needs transactional updates. Tabular structure fits data

Synchronizing posts to all SN

High volume of writes on multiple nodes
DEPLOYING A POLYGLOT DATABASE
MANAGING A POLYGLOT DATABASE
QUERYING, INSERTING, MAINTAINING
GENERATING NOSQL PROGRAMS FROM HIGH LEVEL ABSTRACTS

High-level abstractions

UML class diagram application classes

Low-level abstractions

http://code.google.com/p/model2roo/
**Problem statement:**

- Evolution of the application: modification of classes, new classes, new relationships among classes
- Evolution of the “entities” managed in the polyglot database
  - Some change structure, change values, …
- The content of the stores start deriving from the application data structures
  - Which is the current structure of the entities stored?
  - Are there elements that are not being accessed because they do not longer correspond to the application data structures?
CRUD OPERATIONS

Consistent view of data

<table>
<thead>
<tr>
<th>Id</th>
<th>FirstName</th>
<th>LastName</th>
<th>Society</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
- Classic protocols are not an option (Ex: Two-phase commit)
  - Voting phase, Commit phase = **Scalability issues**

- Limited transactional support by NoSQL solutions
  - Neo4j is one of the few that truly supports ACID (**Atomicity, Consistency, Isolation, Durability**)
  - Others provide transactions limited to single entities (MongoDB), no roll-back (Redis), etc.
  - Rely on BASE (**Basic Availability, Soft-state, Eventual consistency**) instead of ACID
Use the right tool for a given job…

Lack of standardization of models and data storage technologies

(Katsov-2012)
Data stores designed to scale simply

OLTP-style application loads

Read/Write operations by thousands/millions of users

<table>
<thead>
<tr>
<th>CHARACTERISTIC</th>
<th>SUBCHARACTERISTIC</th>
<th>METRIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reliability</td>
<td>Maturity</td>
<td>API changes</td>
</tr>
<tr>
<td></td>
<td>Availability</td>
<td>Downtime $^3$</td>
</tr>
<tr>
<td></td>
<td>Fault tolerance</td>
<td>Node down throughput $^3$</td>
</tr>
<tr>
<td></td>
<td>Recoverability</td>
<td>Time to stabilize on node up $^3$</td>
</tr>
<tr>
<td>Performance and</td>
<td>Time behaviour</td>
<td>Throughput, latency $^2$</td>
</tr>
<tr>
<td>efficiency</td>
<td>Resource utilisation</td>
<td>CPU, Memory and disk usage $^4$</td>
</tr>
</tbody>
</table>

$^1$Yahoo Cloud Serving Benchmark, [https://github.com/brianfrankcooper/YCSB/wiki](https://github.com/brianfrankcooper/YCSB/wiki)


QUALITY DRIVEN BENCHMARK

- Workload executor
- DB interface layer
- Client threads
- Stats

Cloud serving store

Linked data & temporal streams

- Read/write mix
- Record size
- Popularity distribution

- DB to use
- Workload to use
- Target throughput
- Number of threads

Read latency
Throughput

YSCB Client

QDB

Linked data &
temporal streams
• **QDB benchmark** extends YCSB: *FaultTolerance, Recoverability and TimeBehaviour*
  • Pivot data model for representing NoSQL stores data models
  • Sample application: Shopping system¹ *(ProductInfo)*
  • Document data stores: MongoDB, Couchbase, VoltDB, Redis, Neo4J
    • Cluster of four Ubuntu 12.04 servers deployed with extra large VM instances (8 virtual cores and 14 GB of RAM) in Windows Azure²

• **Distributed polyglot (big) database** engineering
  • Model2Roo: engineering data storage solutions for given data collections
  • ExSchema for supporting the maintenance of a polyglot storage solution

² http://www.windowsazure.com/
³ http://forge.puppetlabs.com/puppetlabs/
⁴ Yahoo Cloud Serving Benchmark, https://github.com/brianfrankcooper/YCSB/wiki
IN BRIEF…

- Many proposals, but no definite solution yet…
- Research/Industry challenges
- **Open opportunities** if you’re interested in this topic!
WHEN IS POLYGLOT PERSISTENCE PERTINENT?

- Application essentially composing and serving web pages
  - They only looked up page elements by ID, they had different needs or availability, concurrency and no need to share all their data
- A problem like this is much better suited to a NoSQL store than the corporate relational DBMS
- Scaling to lots of traffic gets harder and harder to do with vertical scaling
- Many NoSQL databases are designed to operate over clusters
- They can tackle larger volumes of traffic and data than is realistic with a single server
Dr. Genoveva Vargas-Solar
CNRS, LIG-LAFMIA
France

Genoveva.Vargas@imag.fr
http://www.vargas-solar.com/bigdata-management

Javier Espinosa
University of Grenoble
France
REFERENCES

- C. Richardson, Developing polyglot persistence applications, http://fr.slideshare.net/chris.e.richardson/developing-polyglotpersistenceapplications-gluecon2013