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-managment

so now we have NoSQL databases

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

Data stores designed to scale simple

OLTP-style application loads

Read/Write operations
by **thousands/millions** of users

examples include

mongoDB
{name: "mongo", type: "DB"}

apache CouchDB relax

riak

Cassandra

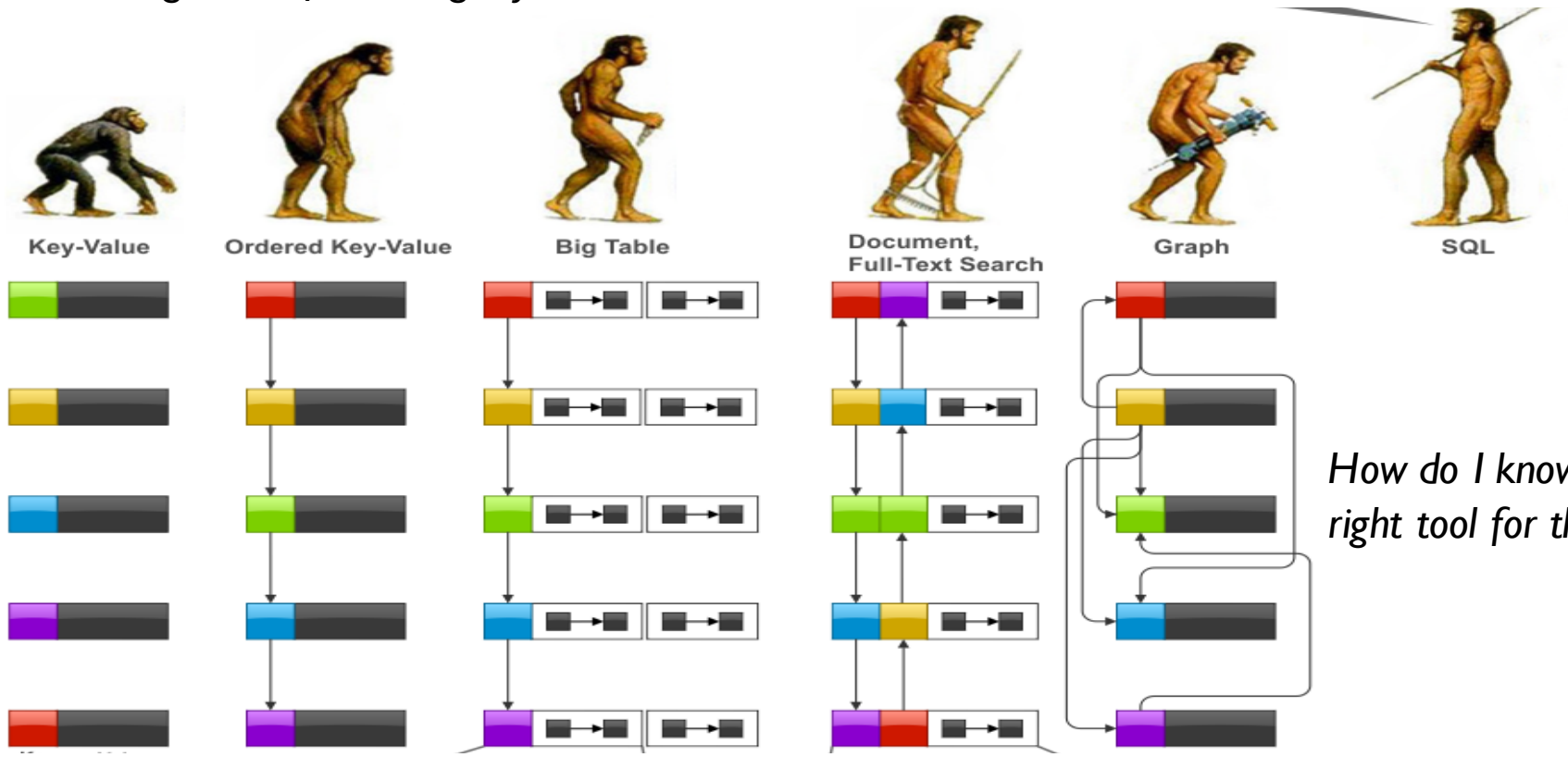
APACHE HBASE

Neo4j
NOSQL for the Enterprise

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…

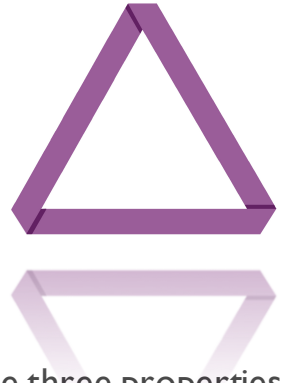


How do I know which is the right tool for the right job?

(Katsov-2012)

PROBLEM STATEMENT: HOW MUCH TO GIVE UP?

Fault-tolerant
partitioning



Availability

Consistency

- CAP theorem[1]: a system can have two of the three properties
- NoSQL systems sacrifice **consistency**

1 Eric Brewer, "Towards robust distributed systems." PODC. 2000 http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf

COMPARING NOSQL & NEWSQL SYSTEMS

	System	Concurrency control	Data storage	Replication	Transaction
Key-Value	Redis	Locks	RAM	Asynchronous	No
	Scalaris	Locks	RAM	Synchronous	Local
	Tokyo	Locks	RAM/Disk	Asynchronous	Local
	Voldemort	MVCC	RAM/BDB	Asynchronous	No
	Riak	MVCC	Plug in	Asynchronous	No
	Membrain	Locks	Flash+Disk	Synchronous	Local
	Membase	Locks	Disk	Synchronous	Local
Document	Dynamo	MVCC	Plug in	Asynchronous	No
	SimpleDB	Non	S3	Asynchronous	No
	MongoDB	Locks	Disk	Asynchronous	No
	CouchDB	MVCC	Disk	Asynchronous	No

	System	Concurrency control	Data storage	Replication	Transaction
	Terrastore	Locks	RAM+	Synchronous	L
Extended records	Hbase	Locks	HADOOP	Asynchronous	L
	HyperTable	Locks	Files	Synchronous	L
	Cassandra	MVCC	Disk	Asynchronous	L
	BigTable	Locs+stamps	GFS	Both	L
	PNuts	MVCC	Disk	Asynchronous	L
Relational	MySQL-C	ACID	Disk	Synchronous	Y
	VoltDB	ACID/no Lock	RAM	Synchronous	Y
	Clustrix	ACID/no Lock	Disk	Synchronous	Y
	ScaleDB	ACID	Disk	Synchronous	Y
	ScaleBase	ACID	Disk	Asynchronous	Y
	NimbusDB	ACID/no Lock	Disk	Synchronous	Y

Cattell, Rick. "Scalable SQL and NoSQL data stores." ACM SIGMOD Record 39.4 (2011): 12-27

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!

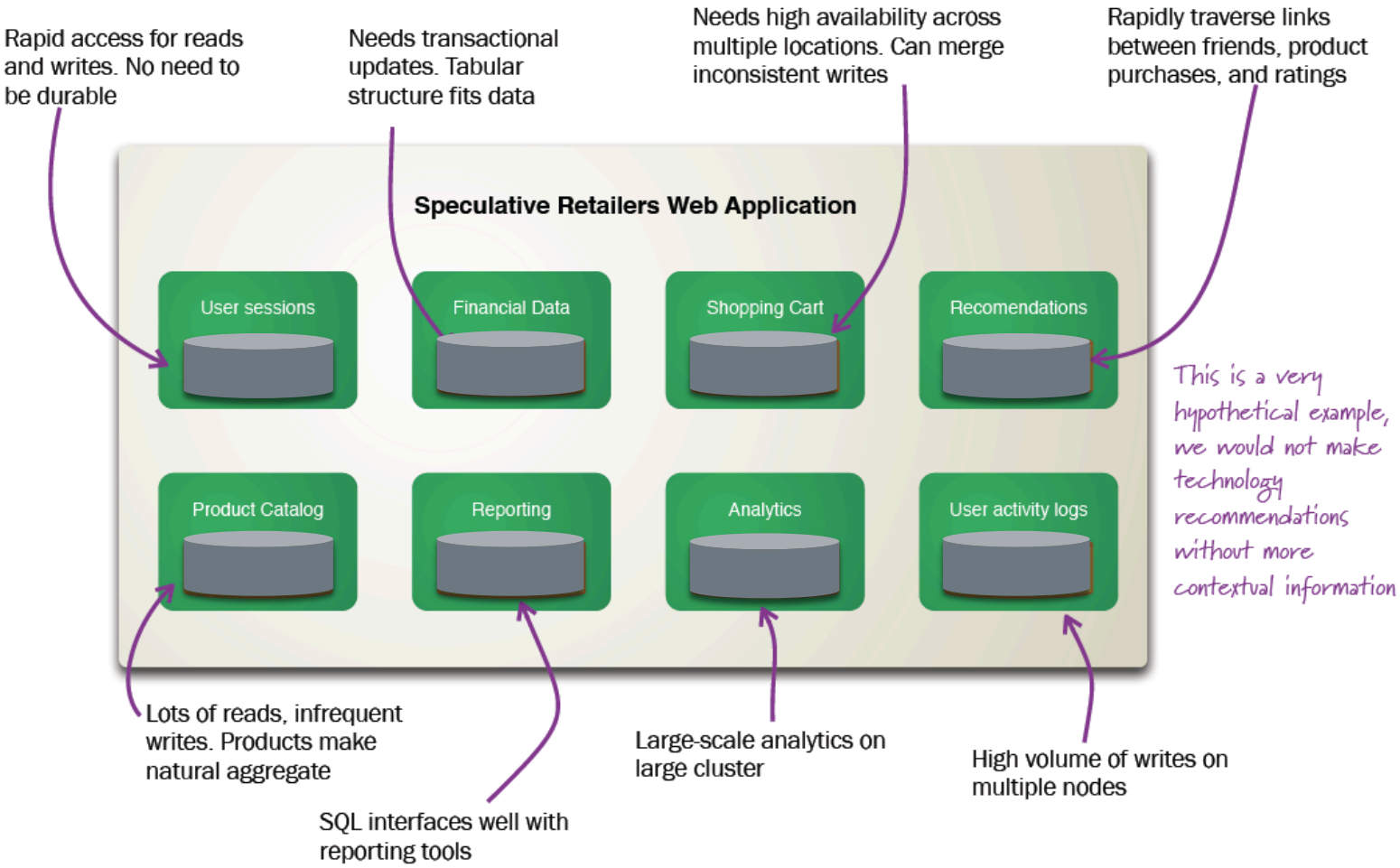
POLYGLOT PERSISTENCE

GENOVEVA VARGAS SOLAR

FRENCH COUNCIL OF SCIENTIFIC RESEARCH, LIG-LAFMIA, FRANCE

Genoveva.Vargas@imag.fr

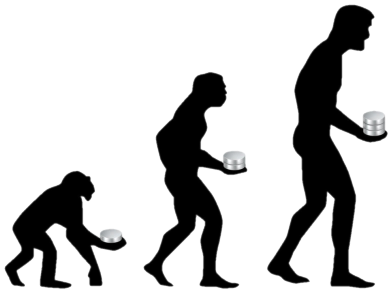
http://www.vargas-solar.com


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
Recomendations
Product Catalog
Reporting
Analytics
User activity logs
This is a very hypothetical example, we would not make technology recommendations without more contextual information
Lots of reads, infrequent writes. Products make natural aggregate
SQL interfaces well with reporting tools
Large-scale analytics on large cluster
High volume of writes on multiple nodes

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.

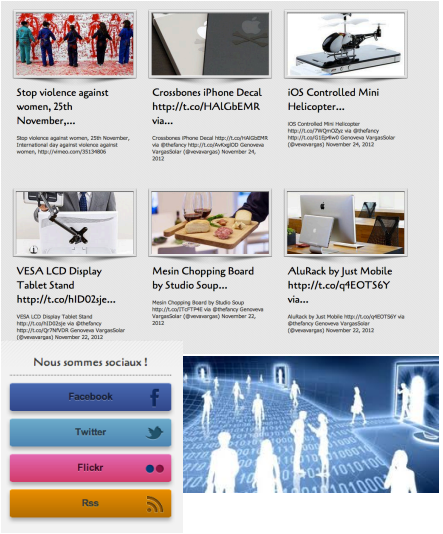
http://martinfowler.com/bliki/PolyglotPersistence.html

POLYGLOT PERSISTENCE

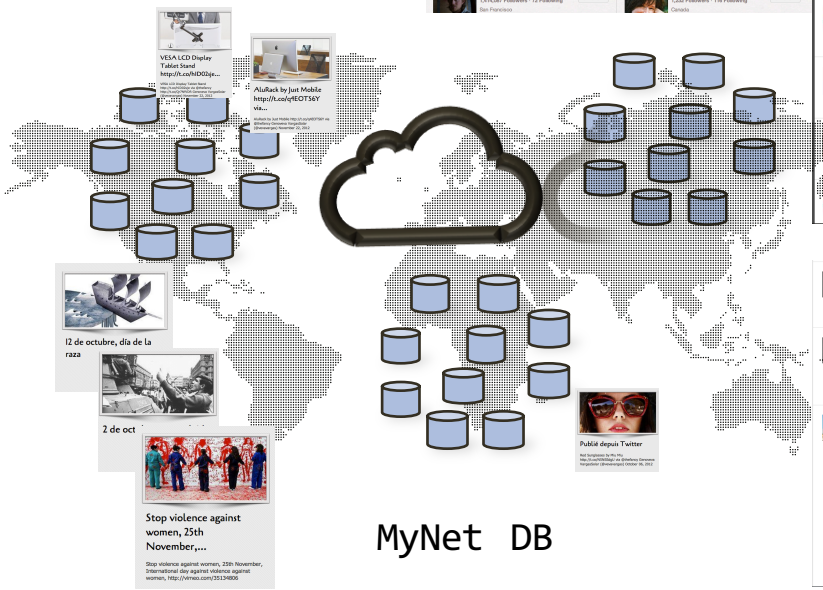
- ***Polyglot Programming:*** applications should be written in a mix of languages to take advantage of different languages are suitable for tackling different problems
- ***Polyglot persistence***: any decent sized enterprise will have a variety of different data storage technologies for different kinds 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

M. Fowler and P. Sadalage. *NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence*. Pearson Education, Limited, 2012

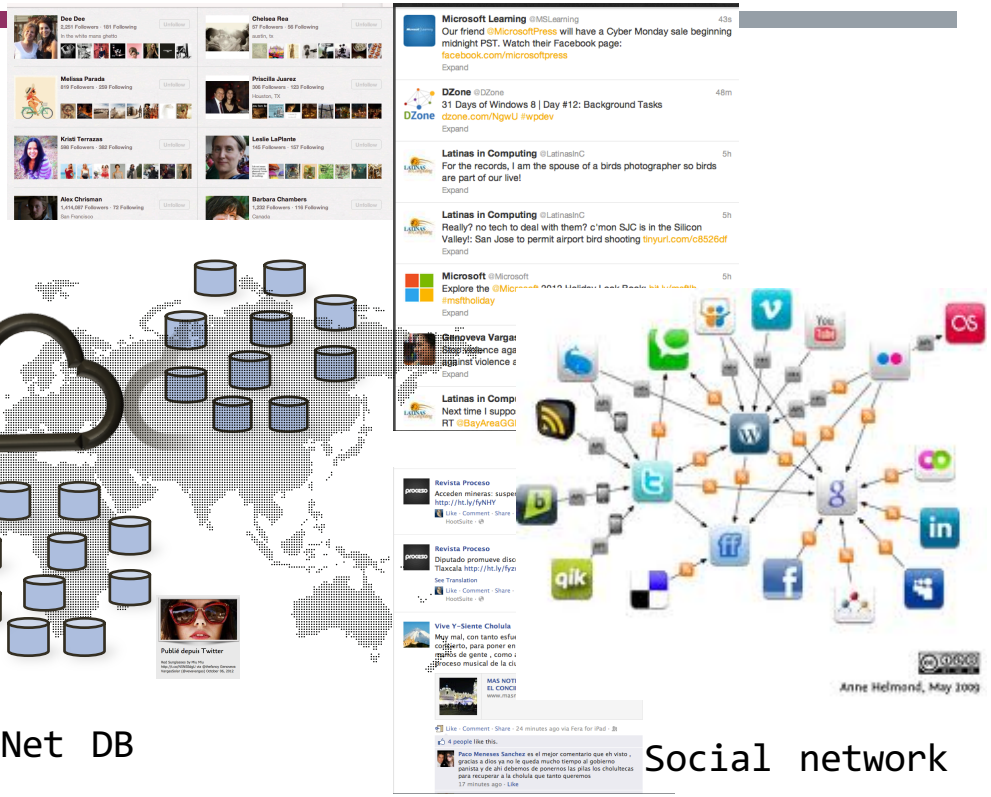
DESIGNING AND BUILDING A POLYGLOT DATABASE



MyNet App



MyNet DB



Social network

- 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

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

User sessions in different Social networks

Needs transactional updates. Tabular structure fits data

Directory synchronisation
Integrating contacts' information
From all SN

Needs 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

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

Friends network

Lots of reads, infrequent writes. Products make natural aggregate

User accounts activity
In different social networks

SQL interfaces well with reporting tools

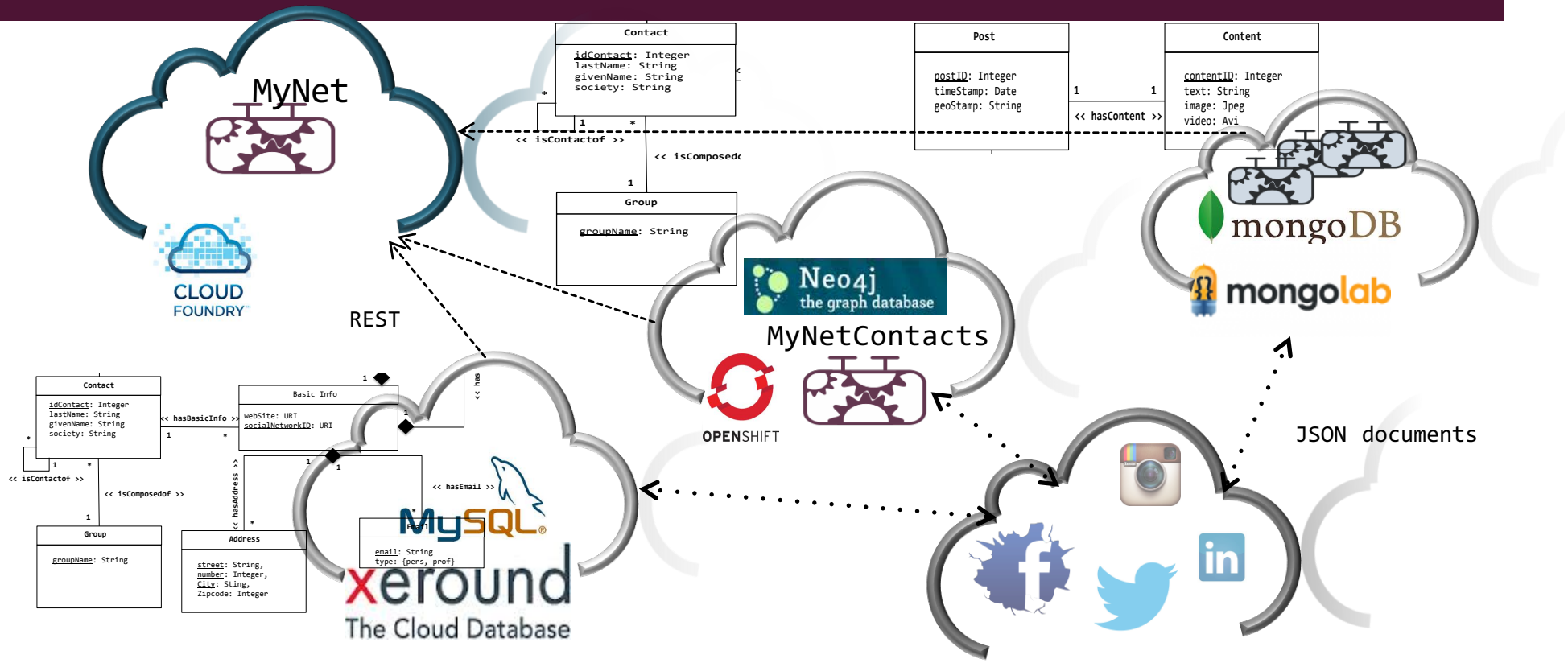
Large-scale analytics on large cluster

Synchronizing posts to all SN

High volume of writes on multiple nodes

DEPLOYING A POLYGLOT DATABASE

MULTI-CLOUD POLYGLOT DATABASE



MANAGING A POLYGLOT DATABASE

QUERYING, INSERTING, MAINTAINING

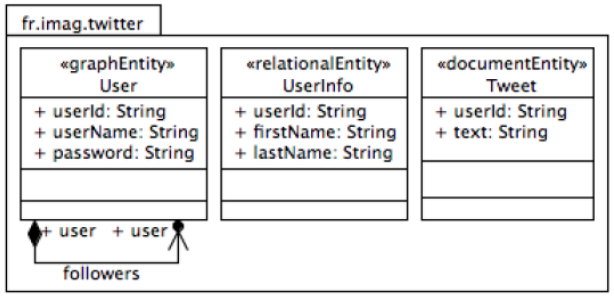
GENERATING NOSQL PROGRAMS FROM HIGH LEVEL ABSTRACTIONS

```
@org.springframework.data.neo4j.annotation.NodeEntity
public class User {}

@org.springframework.roo.addon.jpa.
activerecord.RooJpaActiveRecord
public class UserInfo {}

@org.springframework.roo.addon.layers.
repository.mongo.RooMongoEntity
public class Tweet {}
```

High-level abstractions



UML class diagram **application classes**

Spring Roo

Java web App

Spring Data

Low-level abstractions

Neo4j the graph database

heroku

Graph database

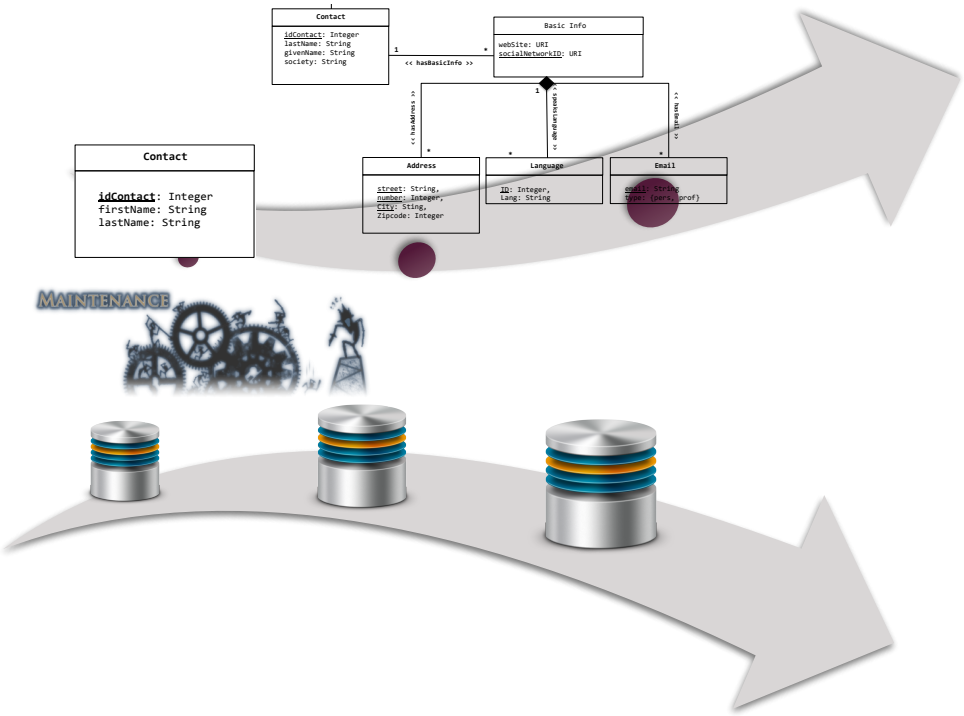
CLOUD FOUNDRY

MySQL

Relational database

http://code.google.com/p/model2roo/

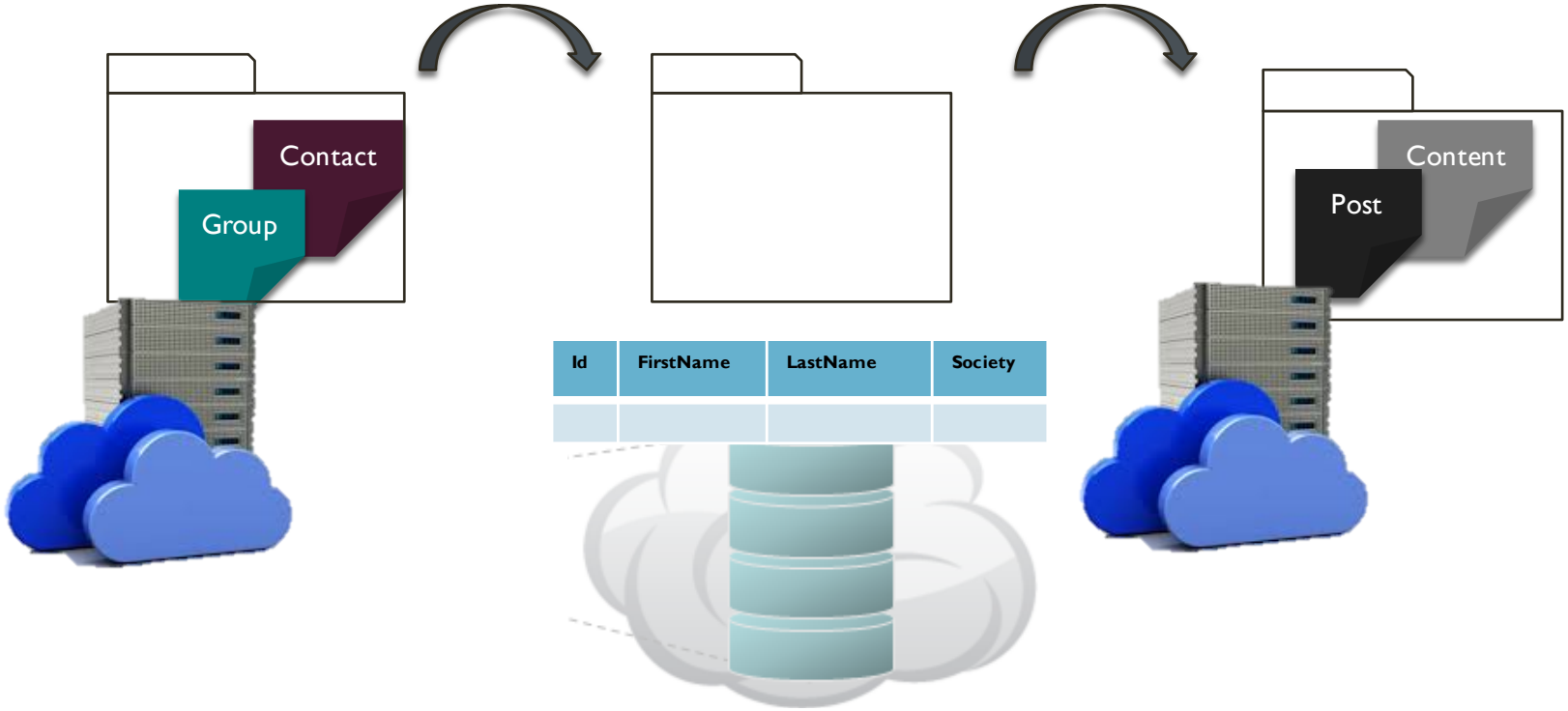
POLYGLOT DATABASE EVOLUTION



- *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

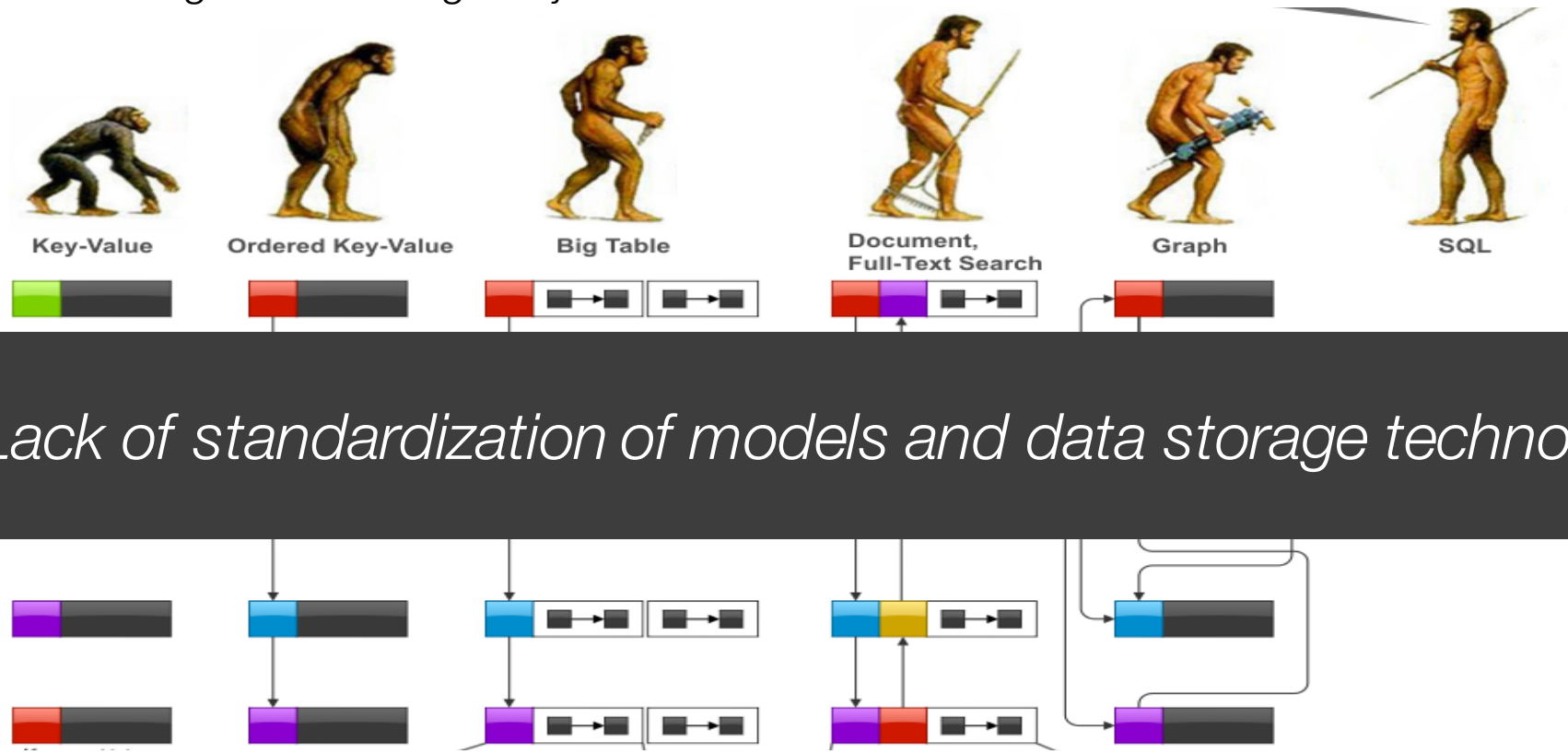


BACKGROUND

- 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, <u>Eventual consistency</u>**) instead of ACID

Use the right tool for a given job…



Lack of standardization of models and data storage technologies

(Katsov-2012)

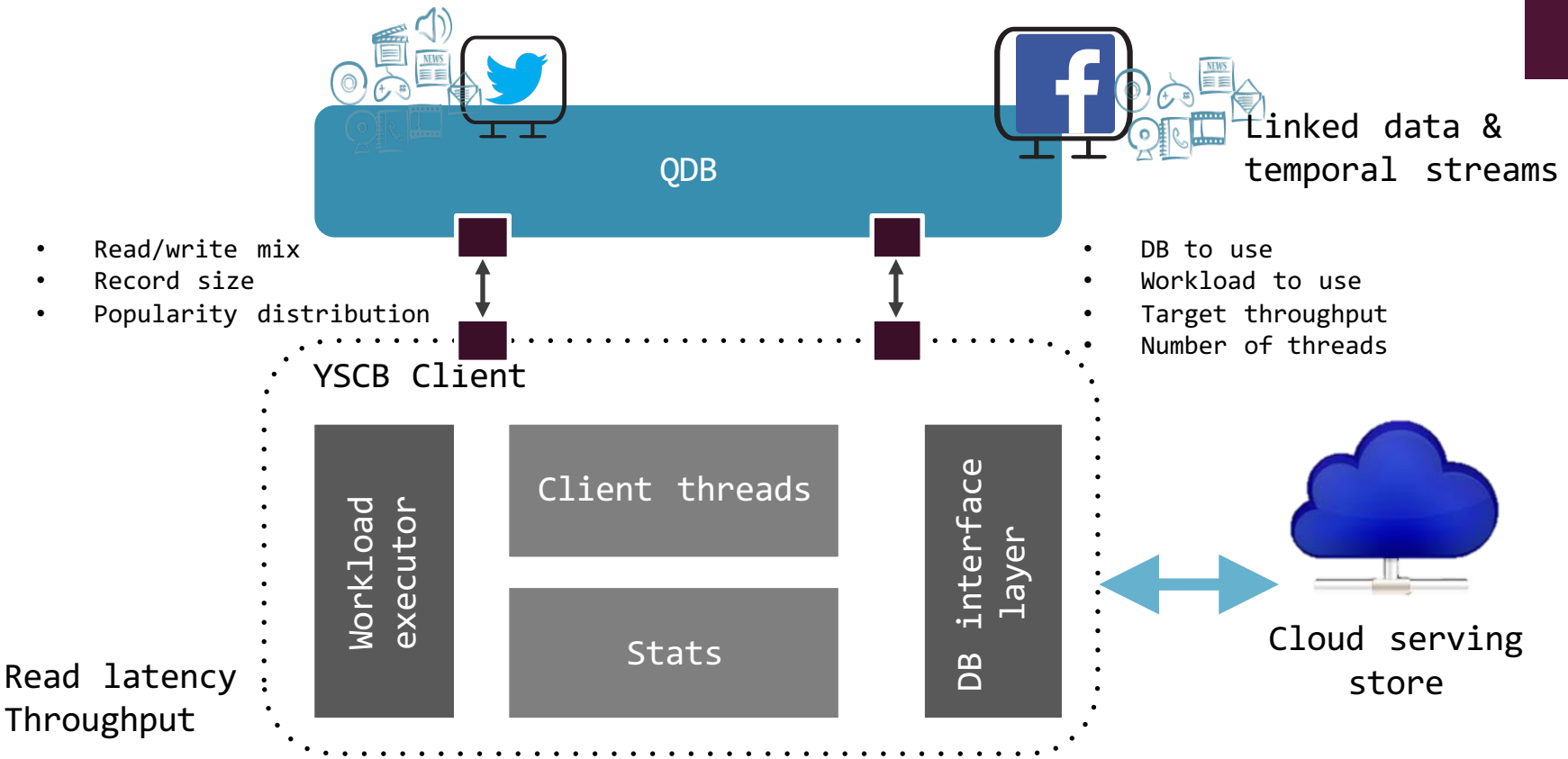
CHARACTERISTIC	SUBCHARACTERISTIC	METRIC
Reliability	Maturity	API changes
	Availability	Downtime [3]
	Fault tolerance	Node down throughput [3]
	Recoverability	Time to stabilize on node up [3]
Performance and efficiency	Time behaviour	Throughput, latency [2]
	Resource utilisation	CPU, Memory and disk usage [4]

Data stores designed to scale simply

OLTP-style application loads

Read/Write operations by **thousands/millions** of users

[1] Yahoo Cloud Serving Benchmark, https://github.com/brianfrankcooper/YCSB/wiki
[2] Cooper,B.F.,Silberstein,A.,Tam,E.,Ramakrishnan,R.,Sears,R.:Benchmarking cloud serving systems with YCSB. In: Proceedings of the 1st ACM symposium on Cloud computing. pp. 143–154. SoCC '10, ACM, New York, NY, USA (2010)
[3] Nelubin, D., Engber, B.: Failover Characteristics of leading NoSQL databases. Tech. rep., Thumbtack Technology (2013)
[4] Massie, M.L., Chun, B.N., Culler, D.E.: The Ganglia Distributed Monitoring Sys- tem: Design, Implementation, and Experience. Parallel Computing 30(7) (Jul 2004)


QDB
Linked data & temporal streams
Read/write mix
Record size
Popularity distribution
DB to use
Workload to use
Target throughput
Number of threads
YSCB Client
Workload executor
Client threads
Stats
DB interface layer
Cloud serving store
Read latency
Throughput

- **QDB benchmark** extends YCSB: *FaultTolerance, Recoverability and TimeBehaviour*
  - Pivot data model for representing NoSQL stores data models
  - Sample application: Shopping system[1] (`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 Azure[2]
- Distributed **polyglot (big) database** engineering
  - Model2Roo: engineering data storage solutions for given data collections
  - ExSchema for supporting the maintenance of a polyglot storage solution

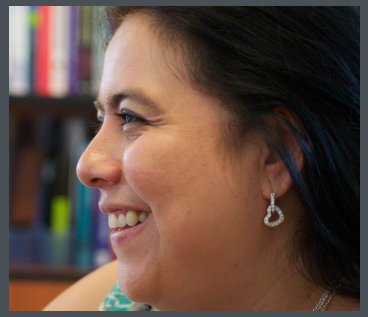
[1] McMurtry, D., Oakley, A., Sharp, J., Subramanian, M., Zhang, H.: Data Access for Highly-Scalable Solutions: Using SQL, NoSQL, and Polyglot Persistence Microsoft patterns & practices, Microsoft (2013)
[2] http://www.windowsazure.com/
[3] http://forge.puppetlabs.com/puppetlabs/
[4] Yahoo Cloud Serving 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

Javier Espinosa
University of Grenoble
France

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

REFERENCES

- Eric A., Brewer "Towards robust distributed systems." PODC. 2000
- Rick, Cattell "Scalable SQL and NoSQL data stores." ACM SIGMOD Record 39.4 (2011): 12-27
- Juan Castrejon, Genoveva Vargas-Solar, Christine Collet, and Rafael Lozano, ExSchema: Discovering and Maintaining Schemas from Polyglot Persistence Applications, In Proceedings of the International Conference on Software Maintenance, Demo Paper, IEEE, 2013
- M. Fowler and P. Sadalage. NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence. Pearson Education, Limited, 2012
- C. Richardson, Developing polyglot persistence applications, http://fr.slideshare.net/chris.e.richardson/developing-polyglotpersistenceapplications-gluecon2013