Please, take a seat,
I always have time for a ridiculous query
Complex Querying with NoSQL

Nested Queries, Projection, Transactions and more

Ron Zavner
Technical Director, EMEA @ GigaSpaces
Agenda

• Moving to fast data
• Data stores and queries evolution
• Facebook & Twitter case studies
• The challenges and the game changer
• Demo
The Real Time World...

Social

User Tracking & Engagement

Homeland Security

eCommerce

Financial Services

Real Time Search
Let’s Look at Some Solutions
RDBMS

- Centralized
- Transactional
- Support ad hoc queries
- Model first, query later
- Scaling is a problem ...
Key/Value

- Got the key? Get the value
- Very fast, low latency
- Good for side cache
- Easy to scale
- Some are highly available
No SQL

- Flexible schema
- Support ad hoc queries
- Support multi TB of data
- Eventual consistency
- Disk based solution
IMDG

• Scalable & Highly available
• Async persistency
• Co-location of data & logic
• Support for Complex query
• Small data?
What are the big guys doing?
Facebook Real Time Analytics
Facebook keeps 80% of its data in Memory (Stanford research)

RAM is **100-1000x** faster than Disk (Random seek)

- Disk: 5 -10ms
- RAM: ~0.001msec
Twitter Real Time Analytics

- **All data**
  - Hadoop: Precomputed batch view
  - Storm: Precomputed realtime view

- **New data stream**
  - Storm: Query
Storm

• Streams
  – Sequence of tuples

• Spouts
  – Source of streams (Queues)

• Bolts
  – Functions, Filters, Joins

• Topologies
Comparing the two approaches ...

Facebook

- Rely on Hadoop for Real Time and Batch
- RT = 10’s Sec
- Suits for Simple processing
- Low parallelization

Twitter

- Use Hadoop for Batch and Storm for real time
- RT = Msec, Sec
- Suits for Complex processing
- Extremely parallel
The Challenges?

Challenge Accepted
Challenge #1
No SQL Solutions are disk based - it is not REAL TIME
Challenge #2
Multiple APIs for different data store solutions
Wanted: All-In-One

- Big Data – multi TB
- Highly available & scalable cluster
- Transactional & strong consistency?
- Intuitive API for complex query
- Real time ad-hoc query
SSD Can Change the Game

IMC Multi-layered Storage Architecture: DRAM vs. NAND Flash/SSD vs. HDD

- IMC Application
- IMC Enabling Infra (IMDBMS, IMDG, CEP)
- Hot Data (DRAM)
- Warm Data (NAND Flash/SSD)
- Cold Data (HDD)

- Flash as additional memory type
- Persistence
- Backup
- Post-processing
- Archiving
- Logging
- Overflow
- Persistence
- Recovery
Flash is not just a fast disk
Demo Time!
Complex Query, Fast Update ...

// Query for a Person who lives in New York:
... = new SQLQuery<Person>(Person.class, "address.city = 'New York'" Abs);
// Query for a Dealer which sales a Honda:
... = new SQLQuery<Dealer>(Dealer.class, "cars[*] = 'Honda'" Abs);

IdQuery<Account> idQuery = new IdQuery<Account>(Account.class, id, routing);
space.change(idQuery, new ChangeSet().increment("balance.euro", 5.2D));

IdQuery<MyPojo> idQuery = new IdQuery<MyPojo>(MyPojo.class, id, routing);
space.change(idQuery, new ChangeSet().increment("someIntProperty", 1)
    .set("someStringProperty", "newStringValue")
    .putInMap("someNestedProperty.someMapProperty", "myKey", 2));

Remains with strong consistency!
Transactions support

```java
@Transactional
public void updateFoo(Foo foo) {
    // do something
}
```
Summary

- RDBMS
- Data Grid
- Big Data
- Fast Data
Thank You!

Ron Zavner
Ronz@gigaspaces.com