DQ_2 & NoSQL Databases

Use cases and experiences

Vincent Garonne, Mario Lassnig, Donal Zang, Luca Canali, Gancho Dimitrov

@CERN.CH

Database Futures Workshop, June, 2011

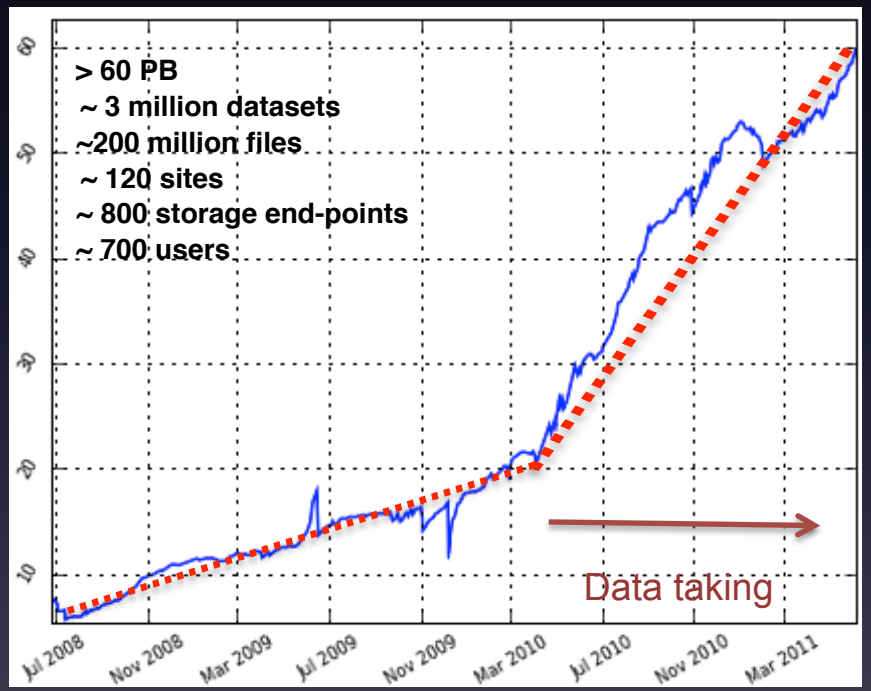
Outline

- $\mathcal{DQ}_2$ overview
- Relational Database Management System
- The NoSQL complementarity
- Experiences and results

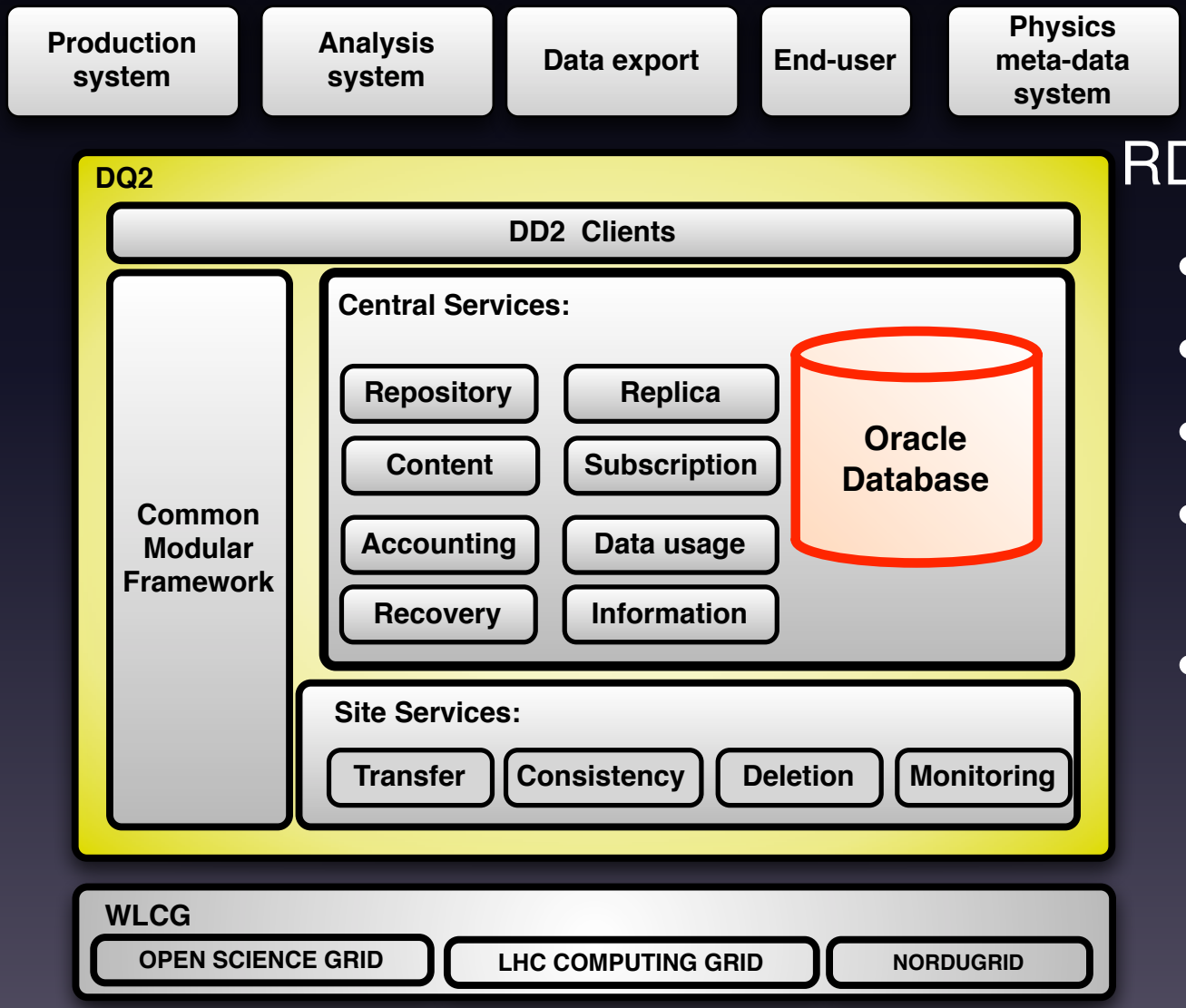
$\mathcal{DQ}_2$ in a nutshell

ATLAS $\mathcal{D}$istributed $\mathcal{D}$ata $\mathcal{M}$anagement $\mathcal{S}$ystem since 2004

- Enforces Computing Model
- Manages experiment's data
- Provides functionalities for
  - Data placement, deletion and organization
  - Bookkeeping & accounting
  - Data access



Relational Databases & $\mathcal{DQ}_2$



RDBMS - Oracle

- Critical dependency
- Proven technology
- Expertise@cern
- Great for enforcing data integrity
- Tool of choice for Online transaction processing applications (OLTP)

Why NoSQL?

$\mathcal{DQ}_2$ database is growing fast, beyond 3 TB scale

New use cases & applications

- Data warehousing
  - Query extremely large datasets with fast query speeds
- Accounting/Monitoring
  - Store non-predefined measurement results

Characteristics

- Lot of data
- No transactions and relaxed consistency
- Schemaless
- Multi dimensional queries

$\Rightarrow$ Relevant for NoSQL

Oracle implementation (IMHO)

- Hard to scale with data warehousing applications
  - As the databases grow larger, the queries start taking longer and longer
  - Non linear query execution time
  - Unstable query plans
  - Static schema

- Possible solutions
  - De-normalization
  - Data partitioning
  - New indexes

  $\Rightarrow$ Flat schema: Less tables, keys and table joins

- Contradictory with normalizing data (OLTP use case)

  $\Rightarrow$ More tables, keys and table joins

NoSQL evaluation - Modus operandi

Definition of the use cases

- 'SQL and NoSQL are complementary'
- Selected $\mathcal{DQ}_2$ 'costly' applications with Oracle

Implementation in SQL and NoSQL

- Collaboration between ATLAS and IT: DBAs, $\mathcal{DQ}_2$ team
- Dedicated Test-beds

Comparison

- R/W performances
- Data replication (Inter-backends)
- 24/7 production service
- Human operations/dev. and HW costs

Which NoSQL Databases ?

- Many open source projects
  - Cassandra vs MongoDB vs Hadoop Hbase vs Simpledb vs Dynamo vs Couchdb vs Hypertable vs Riak vs etc.
  - Wide Column / Document / Key-Value Store
  - Commodity hardware & Mixture of features
- Eventual consistency in favour of performance, scalability and availability
  - Brewer's CAP (Consistency, Availability, Partition-tolerance)
- Cassandra(AP), MongoDB(CP), Hadoop Hbase(CP)
  - Buzzword compliant: Facebook, Twitter
  - Large user community and support
  - Good responsiveness of developer team

Test-beds (mid-May)

NoSQL: 11 nodes - Intel Xeon(R) 2.27GHz

Application	NoSQL DBs	Test-bed
Accounting	MongoDB	2 * 2x8 cores/24GB, 2 disks
Tracer	Cassandra	9 * 2x8 cores/24GB, 4 disks

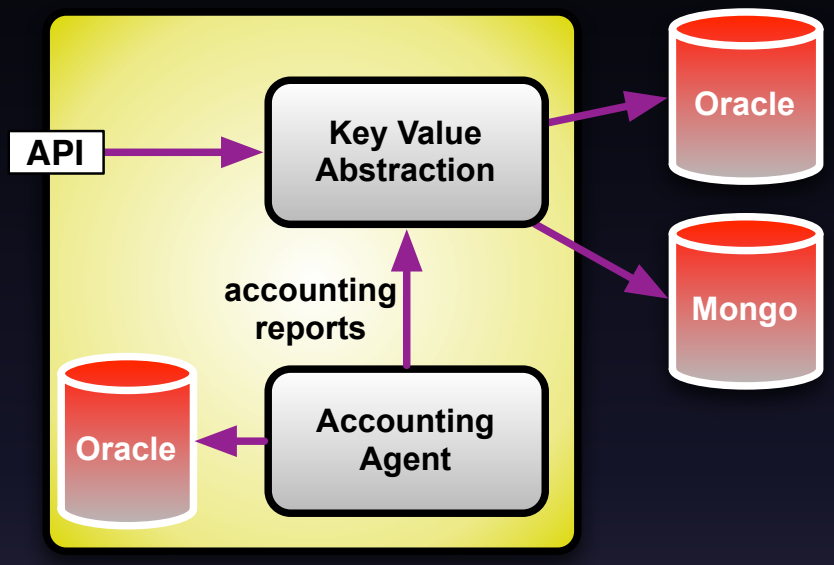
** Deployment with puppet [link]

Oracle: 4 nodes - Intel Xeon(R) 2.27GHz

- Dedicated Oracle 11g: 2 * 2x6 cores/ 48GB
- Storage: 24 SAS disks shared storage (8gbps FC), 32 SATA disks on shared NAS storage

- Shared ATLAS Oracle 10g integration DB: 2 * of 2x4 cores/24GB
- Storage: 36 SATA disks on 4Gbps FC

$\mathcal{DQ}_2$ Accounting service



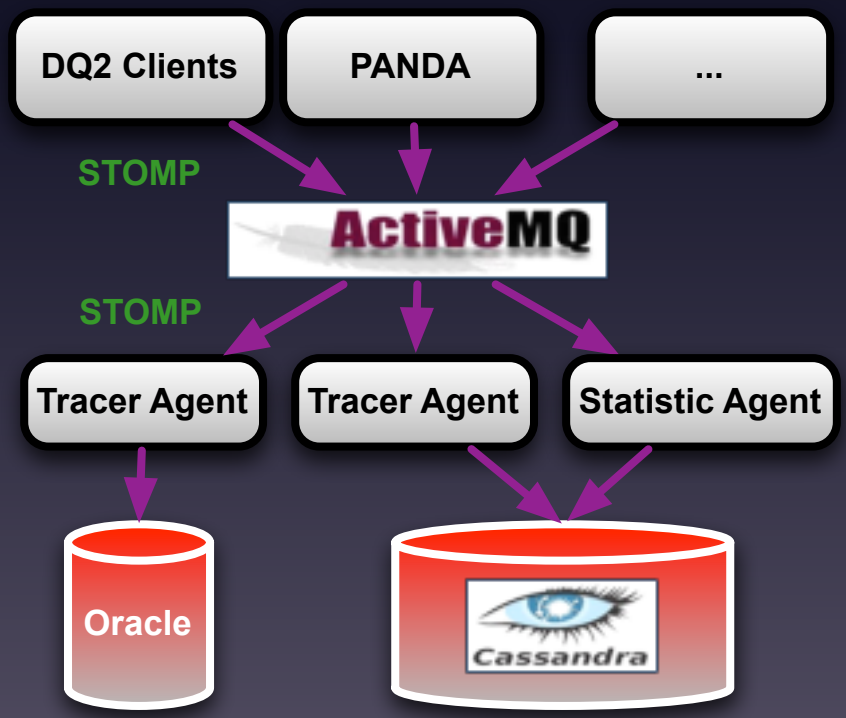
- Storage space and usage information
- Break down volumes by metadata information
  - E.g. location, datatype, custodiality
- Generic key-value approach
- Reports generated from Oracle and stored in two backends

	Summary retrieval	Dev. time
Oracle	7.04s	5 weeks + DBAs
MongoDB	0.16s	4 hours

Oracle	4 Tables, 243 Indexes, 2 Functions, 365 Partitions/Y+hints
MongoDB	1 Table, 1 Index

$\mathcal{DQ}_2$ Grid Tracer service

- Record relevant information about data Access and Usage
- Key and critical component for ATLAS
  - Automatic cleaning of grid storages based on popularity
- $\sim$ 70 traces/second, $\sim$ 90 millions traces/month



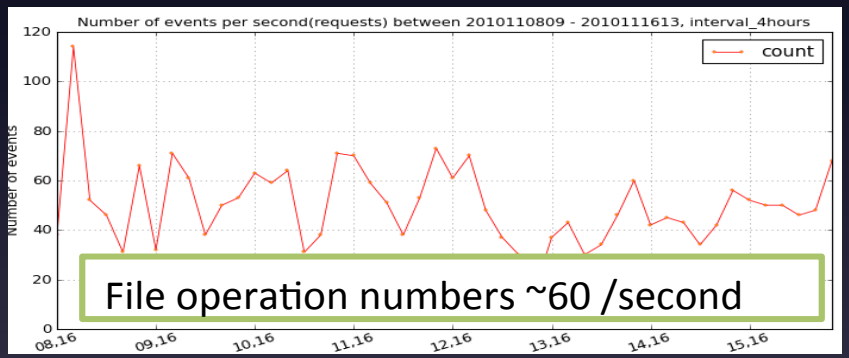
- Events stored two back-ends
- Inter-backend synchronization by Message queue
- Statistic metrics in Cassandra E.g.:

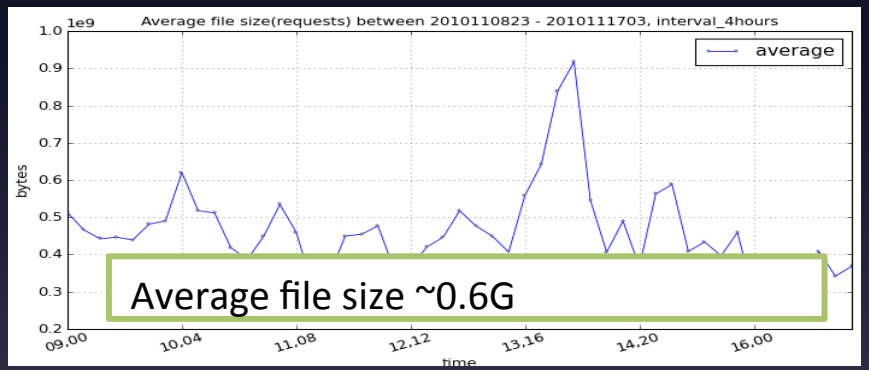
```
{'201011102105':
 {local_read:{
              count:11436,min:0.0,
              max:5507.0,avg:518.07
                 }
  }
 }
```

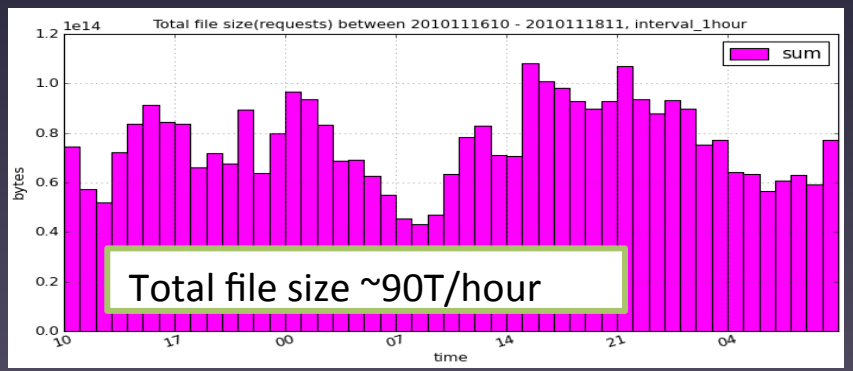
DQ_2 Tracer monitoring

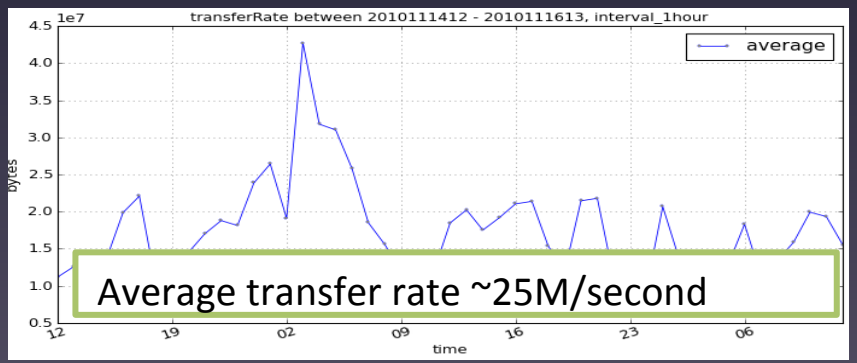
- Generic monitoring on thousands of metrics
  - Rate of requests/failures/transfer/etc.
  - Period: hour, day, month, year
  - Granularity: site, remotesite-localsite, users, etc

Tracer monitoring plots (based on statistic metrics in Cassandra)









Insertion speed

Workload

Concurrency: 10 threads
Run time: 600s
Ramp up: 5s
Row: Tracer event (~1 Kbytes)

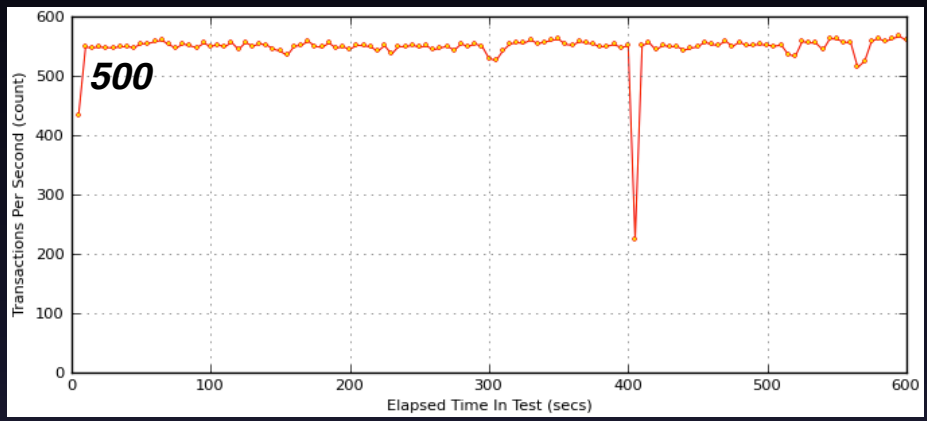
Python client

Oracle: cx_Oracle
MongoDB: pymongo
Cassandra: pycassa

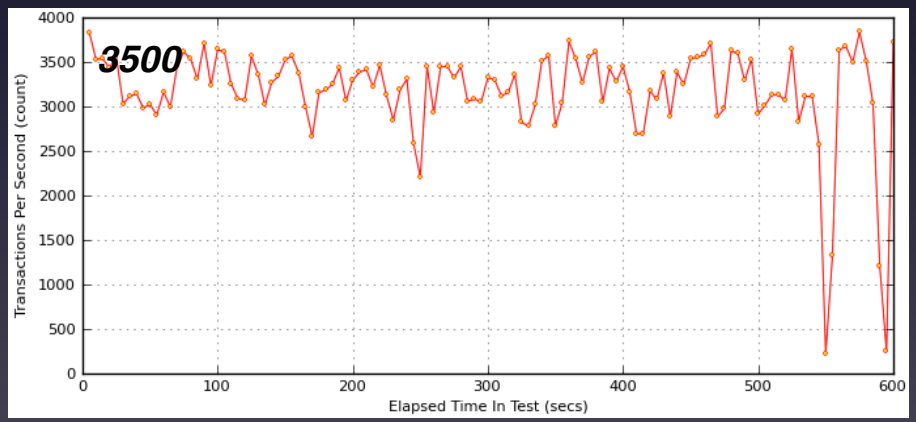
Setup

Oracle: 11g
MongoDB: 1 master, 1 slave, no sharding
Cassandra: random partition, consistencyLevel.quorum
Number of replicas 3, number of node write 2

Oracle Throughput: 327,948 inserted rows



MongoDB Throughput: 1,904,258 inserted rows



Cassandra Throughput: 936,124 inserted rows

High write speed with NoSQL

Query speed

- One month of traces - April 2011
  - $90,578,231$ rows / 34 G
- Oracle schema
  - De-normalized table for performance
  - Index on event timeentry
- Cassandra data model
  - Column family, row key, column, value
  - Analogy with persistent dictionnary

```
t_traces = {'1304514380628696':
{'eventType':'get', 'clientState':'DONE',...  },
...  }
```

  - Index column family / secondary indexes

Query speed - First results

	Oracle 10g	//	Oracle 11g	//	Cache	Cassandra
Data import	30min		30min			2.3h
Row key	0.5s	0.2s	0.03s	0.01s		0.02s
Count(*)	605s	62s	171s	39s	1s	132.5s
Range query	695s	41s	403s	30s	3s	1702s

More information

- Data Oracle2Oracle: Insert + Sub-select
- Data Oracle2NoSQL: 20 threads, 20 events per insert
- // Parallel hint with Oracle
- No live updates on the tables
- No map-reduce and parallelization on Cassandra queries

NoSQL Summary

- Complementary to Oracle, like caching, for certain needs
  - Schema-less approach useful for monitoring
  - More intuitive and flexible than Oracle
  - Save development time
- 2 NoSQL candidates: MongoDB, Cassandra
- First phase of tests focused on data modelling, performance and tunings
  - Still place for improvements with Cassandra
- Future plans
  - Horizontal scalability and resilience tests
  - Map-reduce with MongoDB, Hbase (Cloudera distribution)
  - Add use cases,e.g. Popularity and table joins