



ORACLE®



MySQL Cluster – sometimes SQL

Bernd Ocklin

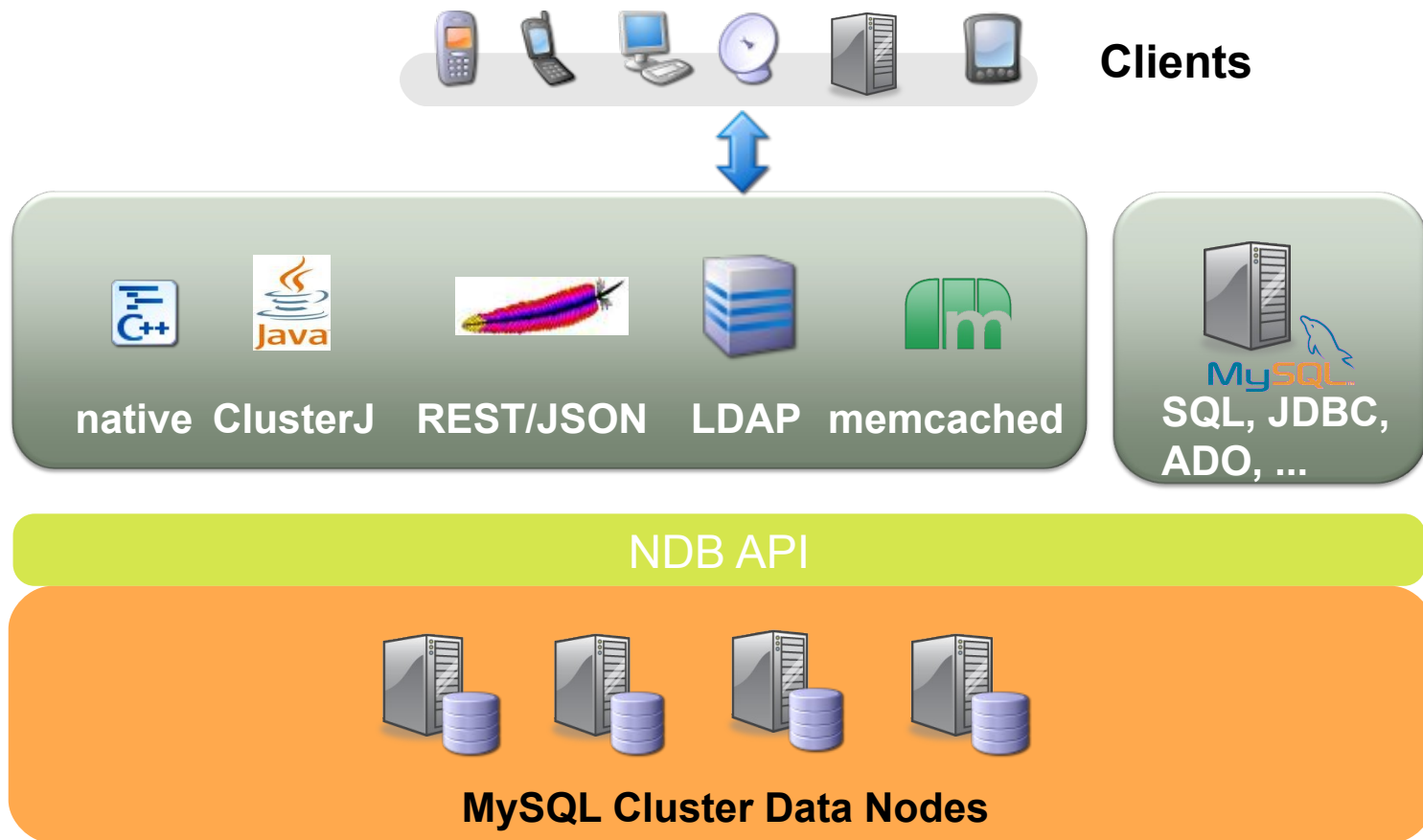
MySQL Cluster

- High Performance: Write Scalability & Low Latency
- 99.999% Availability
- Flexible Data Access Methods (SQL & NoSQL)
- Low TCO (Open Source + Commodity Hardware)

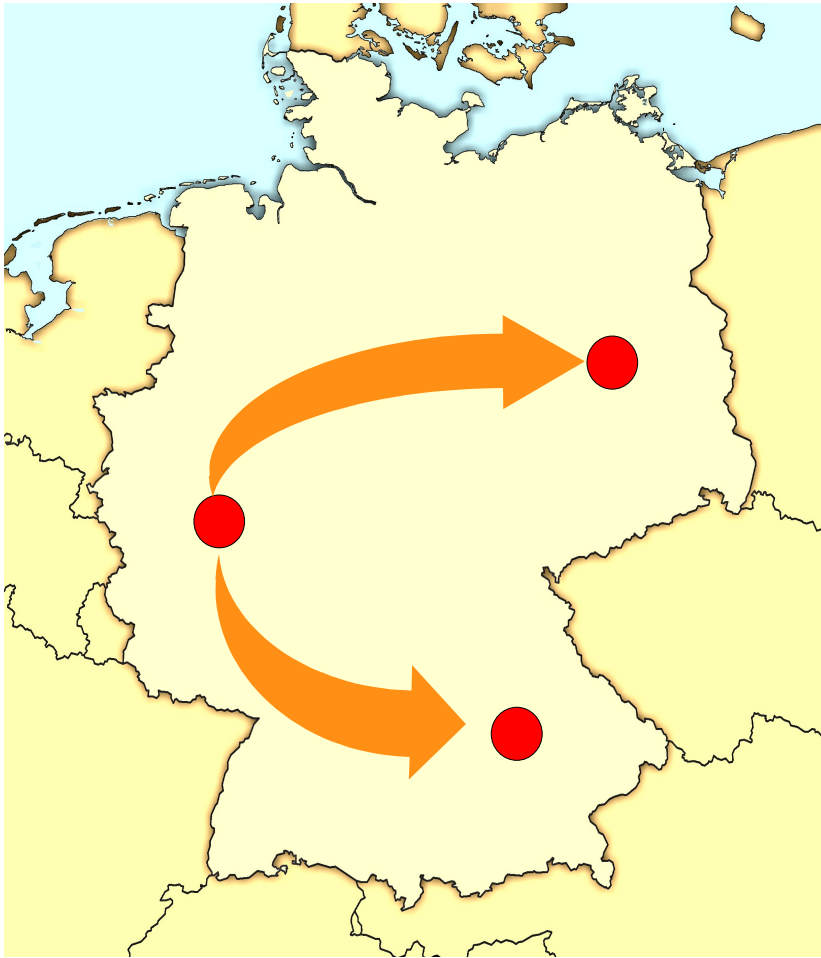
Coming from mobile networks with tough requirements ...

- 50, 60, 100 x 10⁶ mobile phone users
- x 10⁶ operations per second
- micro - millisecond latency
- no downtime, even during upgrades
- low cost

SQL/NoSQL – shared nothing no single point of failure



Geographic Redundancy



- synchronous replication locally
- async replication geographically
- master-slave or multi-master
- automated conflict detection and resolution

Out of the box scalability

distributed hash & data partitioning

Application

Authid (PK)	Frame	Iname	Country
1	Albert	Camus	France
2	Ernest	Hemingway	USA
3	Johann	Goethe	Germany
4	Junichiro	Tanizaki	Japan



Authid (PK)	Frame	Iname	Country
1	Albert	Camus	France
3	Johann	Goethe	Germany

Authid (PK)	Frame	Iname	Country
2	Ernest	Hemingway	USA
4	Junichiro	Tanizaki	Japan

Availability with sub-second failover

Application

Authid (PK)	Frame	Iname	Country
1	Albert	Camus	France
2	Ernest	Hemingway	USA
3	Johann	Goethe	Germany
4	Junichiro	Tanizaki	Japan



Authid (PK)	Frame	Iname	Country
1	Albert	Camus	France
3	Johann	Goethe	Germany

Authid (PK)	Frame	Iname	Country
2	Ernest	Hemingway	USA
4	Junichiro	Tanizaki	Japan

Key features

Distributed

- Shared-nothing, clustered database server
- ACID-compliant relational database

Highly available

- Five 9s (99.999%) availability
- Self-healing, subsecond failover

Real-time performance

- High-load, real-time performance
- Predictable low latency, bounded access times

Dynamically scalable

- Incrementally scale up, out and on-line with application demands
- Linearly scale with distribution awareness

Open development

- Open source, multiple data access
- High-performance APIs (C++/Java), SQL, Memcached, REST/JSON, LDAP, Web Services

12 years and counting ... rock solid

Ericsson Network
DataBase (NDB):
Real-Time
99.99% availability
Auto-failover
In-memory
Scale-out
On-line backups
NDB API direct
C++ Access

MySQL 4.1.7:
NDB/Cluster
integrated
with MySQL
SQL Access

MySQL 5.1:
Disk data
Geo-Replication
User-defined
partitioning

MySQL Cluster 6.1:
255 nodes
NDB/J direct Java access

MySQL Cluster 7.0:
Scale-up (multi-
threaded data nodes)
On-line add-node
4x performance
Back-NDB for LDAP

MySQL Cluster 7.1:
MySQL Cluster
Manager
NDBINFO – real-
time monitoring
Java Connector (ClusterJ)

1998

2004

2006

2009

2010

MySQL 5.0:
Batched API
access
Robustness &
performance

MySQL Cluster 6.X:
On-line schema changes
Enhanced NDB API
mod_ndb REST/JSON

Who uses cluster?

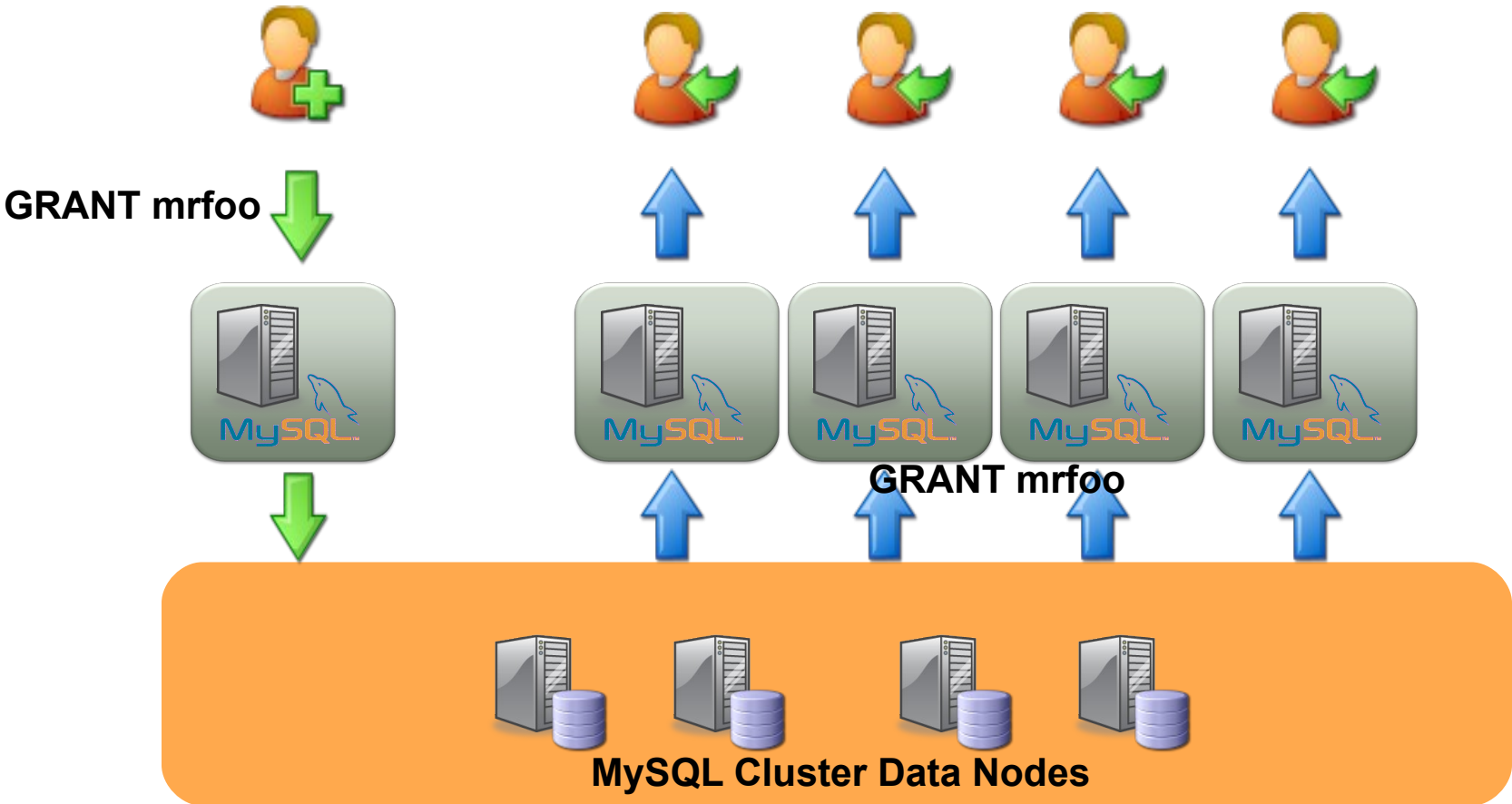
Web & telecoms

- Telecoms
 - Subscriber Databases (HLR/HSS)
 - Service Delivery Platforms
 - VoIP, IPTV & VoD
 - Mobile Content Delivery
 - On-Line app stores and portals
 - IP Management
 - Payment Gateways
- Web
 - User profile management
 - Session stores
 - eCommerce
 - On-Line Gaming
 - Application Servers



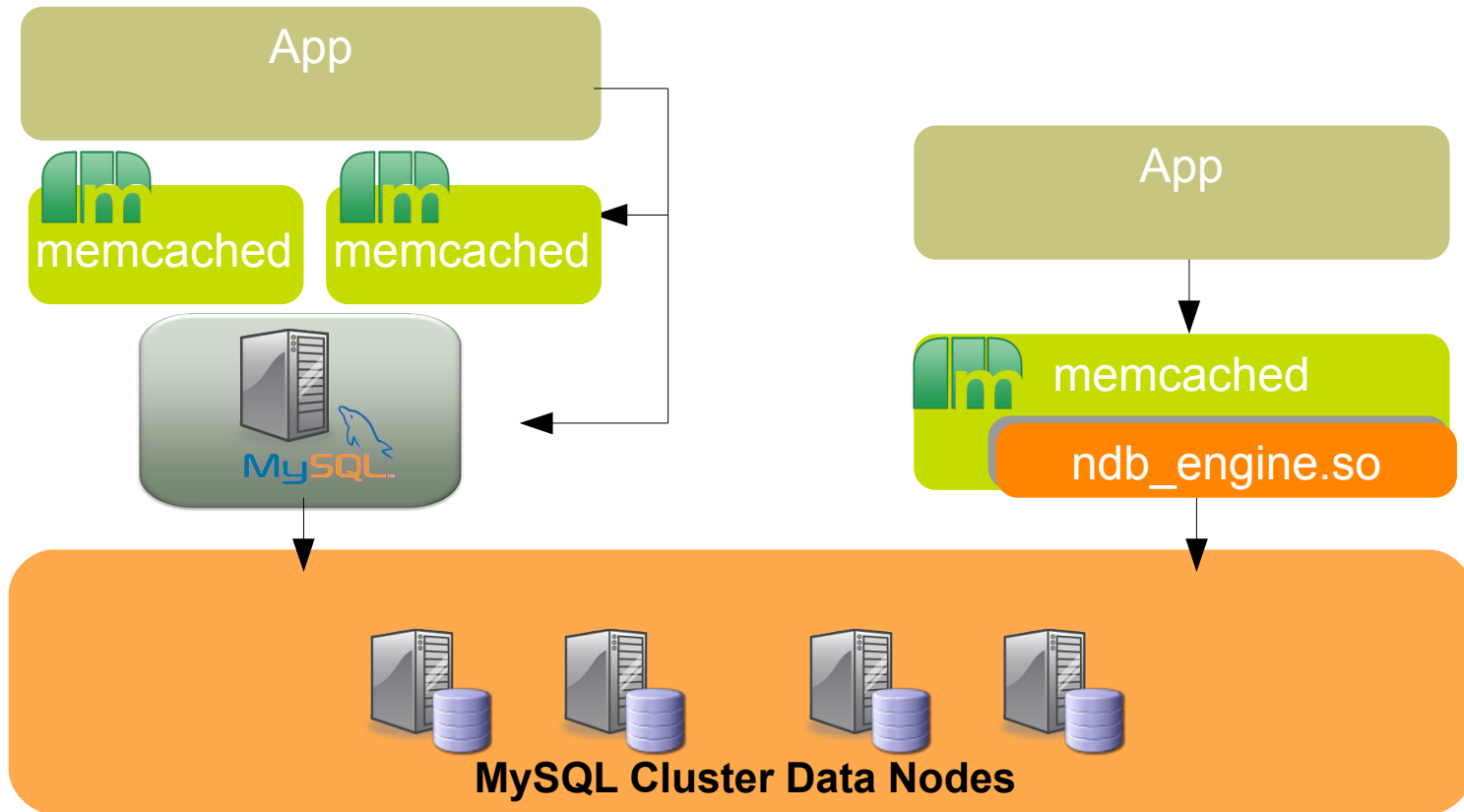
<http://www.mysql.com/customers/cluster/>

Consolidated authentication



Simplify the stack

Native memcached ndb_engine



native cluster memcached ndb_engine

- Native cluster access through memcache protocol
- Direct write/read-through to cluster
- Using memcached 1.6 plug-in engine architecture
- Many programming languages
- Reduces system complexity
- Eliminates data inconsistencies between several memcached instances and the db backend

NDBAPI example

```
NdbTransaction * tx = ndb->startTransaction();

NdbOperation * op = tx->getNdbOperation(myTable);

op->readTuple(NdbOperation::LM_CommittedRead);

op->equal("code", code);

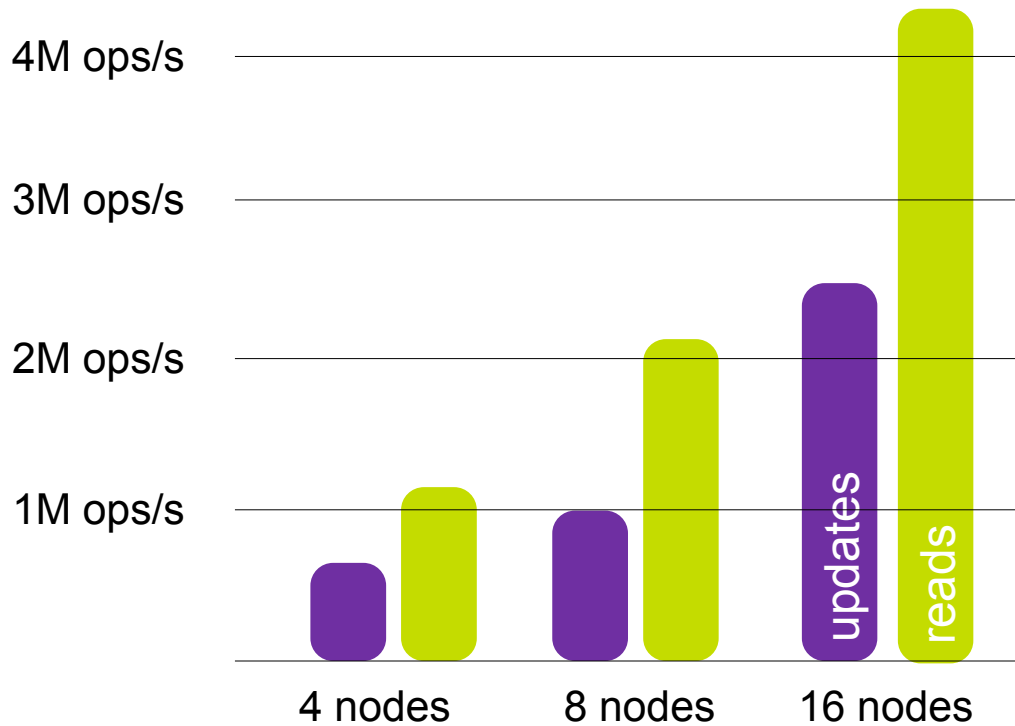
op->getValue("name", name);

tx->execute( NdbTransaction::Commit );
cout << "name = " << name << endl;

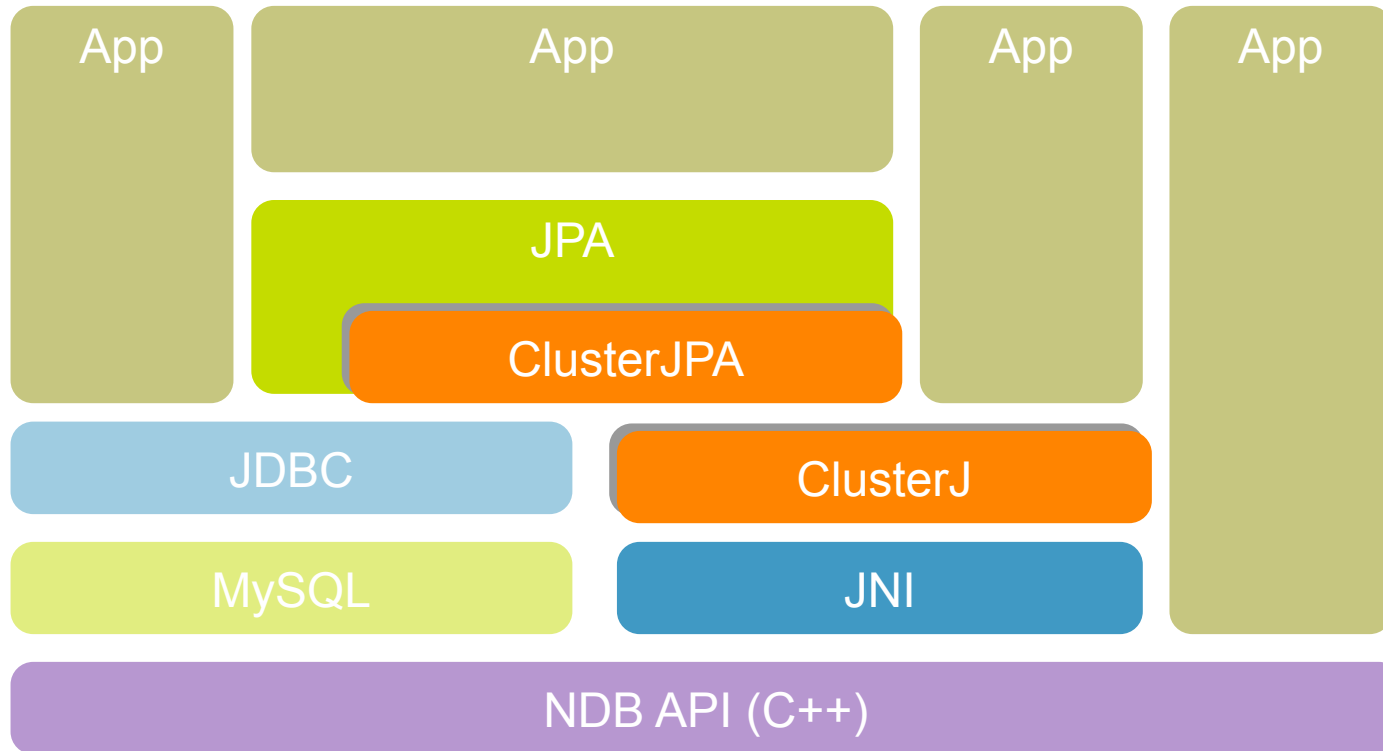
ndb->closeTransaction(tx);
```

Throughput benchmark 2011

Note: last minute benchmark before the conference to show some new numbers.



Java Connector



ClusterJ annotated interface

```
@PersistenceCapable(table="t_fish_food")  
public interface Fish {
```

```
@PrimaryKey
```

```
int getId();
```

```
void setId(int id);
```

```
@Column(name = "Fishname")
```

```
@Index(name="idx_unq_hash_name")
```

```
String getName();
```

```
void setName(String name);
```

```
...
```

```
};
```

```
CREATE TABLE t_fish_food(  

```

```
id int primary key,  

```

```
Fishname varchar(255)
```

```
unique index using hash,  

```

```
...
```

```
) ENGINE = ndbcluster;
```

ClusterJ find() data

```
Employee theEmployee =
```

```
    session.find(Employee.class, 988);
```

ClusterJ persist()

```
// Create and initialise an Employee  
Employee newEmployee = session.newInstance(Employee.class) ;  
newEmployee.setId(988);  
newEmployee.setFirst("John");  
newEmployee.setLast("Jones");  
newEmployee.setStarted("1 February 2009");  
newEmployee.setDepartment(666);
```

```
// Write the Employee to the database  
session.persist(newEmployee) ;
```

mod_ndb

- Community feature
- Apache module allowing native access to cluster
- REST API speaking JSON or XML

```
> GET /flights/airports?iata=SFO
```

```
{iata : SFO,  
  Long : "San Francisco Intl Airport",  
  Loc  : US}
```

mod_ndb and AJAX

```
$.get('http://localhost/demo/iata',  
    { "code": value },  
    function(data) {  
        var n = JSON.parse(data);  
        output(n.name);  
    });
```

SQL, memcached, ClusterJ, C++, REST/JSON ... which to choose?

- Answer: Use either or both, whatever best fits your application
 - Different APIs into MySQL Cluster
 - Alternative data stores
- Factors to consider:
 - Performance & Scalability
 - Developer skills & Familiarity with APIs
 - Levels of support
 - Access patterns (joins needed? Key/value sufficient?)
 - Schema changes (online or schema-less)
- Mix & Match!
 - MySQL Cluster allows the same data to be accessed simultaneously through SQL & NoSQL interfaces

SQL, memcached, ClusterJ, C++, REST/JSON ... which to choose?

SQL

- Using a standard
- Joins & complex queries
- Relational model

Memcached

- simple to use API
- key/value
- driver for many languages
- ideal as e.g. PHP proxy

mod_ndb

- REST/JSON
- HTML
- using apache

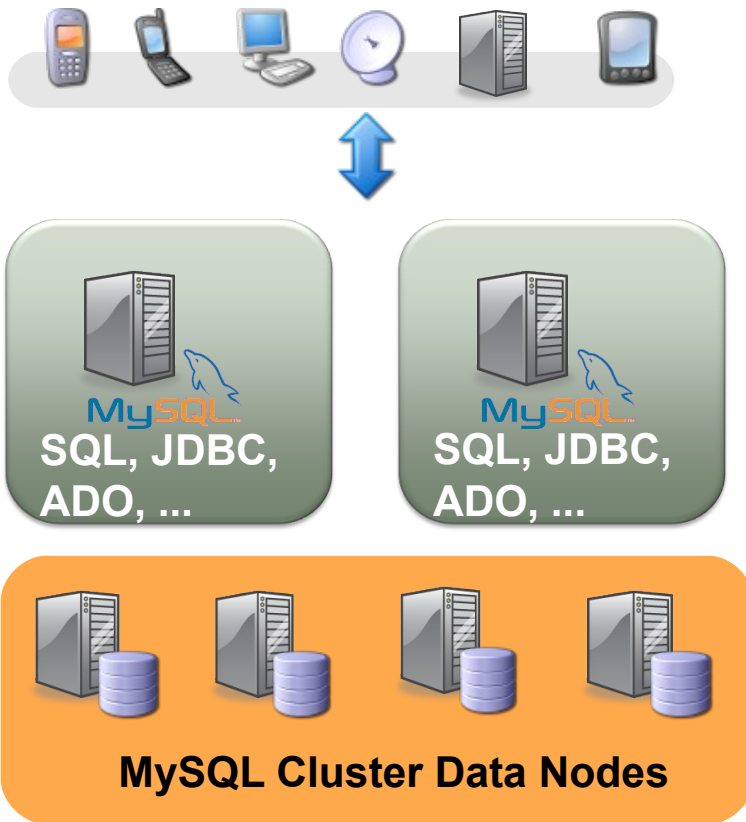
ClusterJ

- simple to use Java API
- Web & telco
- Object Relational Mapping
- native & fast access to cluster

C++

- knowledged developer
- super low latency / real-time

Sometimes SQL!



- Allows most complex business questions
- Standard – most know it
- But cluster was traditionally not optimal with joins

Scaling distributed joins in cluster!

7.2

Traditionally

- Very fast primary key operations
- Certain optimized clauses (e.g. IN(...))
- Distributed joins were hard to scale

New in 7.2

- Pushing the execution down into the storage layer, greatly reduces network trips
- Makes joins scale and up to 40x faster
- expand the use of MySQL Cluster into a broader range of services and applications



COMPANY OVERVIEW

- Division of DocuDesk
- Deliver Document Management SaaS

CHALLENGES / OPPORTUNITIES

- Provide a single repository for customers to manage, archive, and distribute documents
- Implement scalable, fault tolerant, real time data management back-end
- PHP session state cached for in-service personalization
- Store document meta-data, text (as BLOBs), ACL, job queues and billing data
- Data volumes growing at 2% per day

SOLUTION

- MySQL Cluster deployed on EC2

USER PERSPECTIVE

“MySQL Cluster exceeds our requirements for low latency, high throughput performance with continuous availability, in a single solution that minimizes complexity and overall cost.”

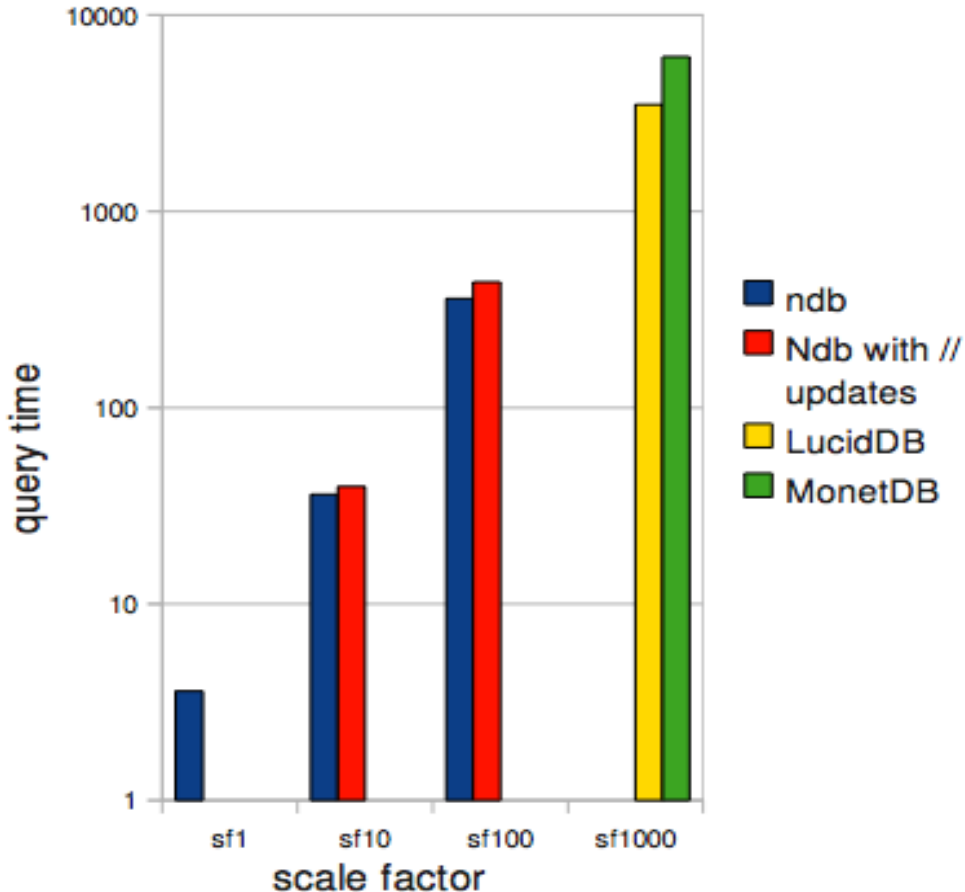
-- Casey Brown, Manager of Dev & DBA Services,
DocuDesk



RESULTS

- Successfully deployed document management solution, eliminating paper trails from legal processes
- Integrate caching and database into one layer, reducing complexity & cost
- Support workload with 50:50 read/write ratio
- Low latency for real-time user experience and document time-stamping
- Continuous database availability

Star Schema Q1.1 with distributed joins



- Likely never as fast as specialized databases
- But only moderately slower with parallel 50k updates / sec make a unique combination.

Download 7.2 Dev Milestone now

Download the software:

7.1 GA: www.mysql.com/downloads/cluster/

7.2 DMR: dev.mysql.com/downloads/cluster/

Memcached: labs.mysql.com

Blogs on latest developments: clusterdb.com, planet.mysql.com

MySQL Cluster Quick Start Guides

www.mysql.com/products/database/cluster/get-started.html#quickstart

MySQL Cluster 7.1, Architecture and New Features

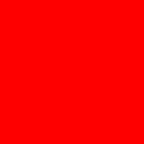
www.mysql.com/why-mysql/white-papers/mysql_wp_cluster7_architecture.php

MySQL Cluster on the Web

www.mysql.com/products/database/cluster



Thank you!



The presentation is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

ORACLE®