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 $10^6$ mobile phone users
• $x 10^6$ operations per second
• micro - millisecond latency
• no downtime, even during upgrades
• low cost
SQL/NoSQL – shared nothing
no single point of failure

Clients

NDB API

MySQL Cluster Data Nodes

native  ClusterJ  REST/JSON  LDAP  memcached

MySQL
SQL, JDBC, ADO, ...

ORACLE
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

<table>
<thead>
<tr>
<th>Authid (PK)</th>
<th>Frame</th>
<th>Iname</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Albert</td>
<td>Camus</td>
<td>France</td>
</tr>
<tr>
<td>2</td>
<td>Ernest</td>
<td>Hemingway</td>
<td>USA</td>
</tr>
<tr>
<td>3</td>
<td>Johann</td>
<td>Goethe</td>
<td>Germany</td>
</tr>
<tr>
<td>4</td>
<td>Junichiro</td>
<td>Tanizaki</td>
<td>Japan</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Authid (PK)</th>
<th>Frame</th>
<th>Iname</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Albert</td>
<td>Camus</td>
<td>France</td>
</tr>
<tr>
<td>3</td>
<td>Johann</td>
<td>Goethe</td>
<td>Germany</td>
</tr>
<tr>
<td>2</td>
<td>Ernest</td>
<td>Hemingway</td>
<td>USA</td>
</tr>
<tr>
<td>4</td>
<td>Junichiro</td>
<td>Tanizaki</td>
<td>Japan</td>
</tr>
</tbody>
</table>
Availability with sub-second failover

<table>
<thead>
<tr>
<th>Authid (PK)</th>
<th>Frame</th>
<th>Iname</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Albert</td>
<td>Camus</td>
<td>France</td>
</tr>
<tr>
<td>2</td>
<td>Ernest</td>
<td>Hemingway</td>
<td>USA</td>
</tr>
<tr>
<td>3</td>
<td>Johann</td>
<td>Goethe</td>
<td>Germany</td>
</tr>
<tr>
<td>4</td>
<td>Junichiro</td>
<td>Tanizaki</td>
<td>Japan</td>
</tr>
</tbody>
</table>

**Application**

<table>
<thead>
<tr>
<th>Authid (PK)</th>
<th>Frame</th>
<th>Iname</th>
<th>Country</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Albert</td>
<td>Camus</td>
<td>France</td>
</tr>
<tr>
<td>2</td>
<td>Ernest</td>
<td>Hemingway</td>
<td>USA</td>
</tr>
<tr>
<td>3</td>
<td>Johann</td>
<td>Goethe</td>
<td>Germany</td>
</tr>
<tr>
<td>4</td>
<td>Junichiro</td>
<td>Tanizaki</td>
<td>Japan</td>
</tr>
</tbody>
</table>
## Key features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed</td>
<td>• Shared-nothing, clustered database server</td>
</tr>
<tr>
<td></td>
<td>• ACID-compliant relational database</td>
</tr>
<tr>
<td>Highly available</td>
<td>• Five 9s (99.999%) availability</td>
</tr>
<tr>
<td></td>
<td>• Self-healing, subsecond failover</td>
</tr>
<tr>
<td>Real-time performance</td>
<td>• High-load, real-time performance</td>
</tr>
<tr>
<td></td>
<td>• Predictable low latency, bounded access times</td>
</tr>
<tr>
<td>Dynamically scalable</td>
<td>• Incrementally scale up, out and on-line with application demands</td>
</tr>
<tr>
<td></td>
<td>• Linearly scale with distribution awareness</td>
</tr>
<tr>
<td>Open development</td>
<td>• Open source, multiple data access</td>
</tr>
<tr>
<td></td>
<td>• High-performance APIs (C++/Java), SQL, Memcached, REST/JSON, LDAP, Web Services</td>
</tr>
</tbody>
</table>
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-Replcation
- User-defined partitioning

MySQL Cluster 6.0:
- On-line add-node
- 4x performance
- Back-NDB for LDAP

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

MySQL 5.0:
- Batched API access
- Robustness & performance

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

MySQL Cluster 7.0:
- Scale-up (multi-threaded data nodes)

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

2004
MySQL 4.1.7 integrated with MySQL

2006
MySQL 5.1

2009
MySQL Cluster 6.X

2010
MySQL Cluster 7.1
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

GRANT mrfoo

MySQL Cluster Data Nodes
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

```cpp
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.

<table>
<thead>
<tr>
<th>Throughput Rate</th>
<th>4 nodes</th>
<th>8 nodes</th>
<th>16 nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>4M ops/s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3M ops/s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2M ops/s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1M ops/s</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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

```java
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

```javascript
$.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
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.