MySQL Fabric

MySQL Global Business Unit
Sales Consulting Manager, JAPAC
梶山 隆輔 / Ryusuke Kajiyama
Safe Harbour Statement

The following 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.
BEST IN CLASS COMPONENTS

MySQL: Next Generation Web Applications
On-Premises, in the Cloud, Distributed Applications
MySQL@Oracle: 4 Years of MySQL Innovation

MySQL Fabric
MySQL Migration Wizard
Windows installer & Tools
MySQL Applier for Hadoop

MySQL 5.6
MySQL Workbench 6.1
MySQL 5.7
MySQL Cluster Manager
MySQL Enterprise Monitor 2.3 & 3.0
MySQL 5.5
MySQL Enterprise
MySQL 5.4
MySQL Utilities
MySQL Cluster 7.4
MySQL Workbench 5.2 & 6.0
MySQL Cluster 7.3
MySQL Enterprise
MySQL Cluster 7.2
MySQL Migration Wizard
MySQL Cluster 7.1
MySQL Enterprise
Oracle Certifications
MySQL Fabric
MySQL Migration Wizard
Windows installer & Tools
MySQL Applier for Hadoop

MySQL 5.6
MySQL Workbench 6.1
MySQL 5.7
MySQL Cluster Manager
MySQL Enterprise Monitor 2.3 & 3.0
MySQL 5.5
MySQL Enterprise
MySQL 5.4
MySQL Utilities
MySQL Cluster 7.4
MySQL Workbench 5.2 & 6.0
MySQL Cluster 7.3
MySQL Migration Wizard
MySQL Cluster 7.2
MySQL Cluster 7.1
MySQL Enterprise
Oracle Certifications
MySQL Fabric
MySQL Migration Wizard
Windows installer & Tools
MySQL Applier for Hadoop

MySQL 5.6
MySQL Workbench 6.1
MySQL 5.7
MySQL Cluster Manager
MySQL Enterprise Monitor 2.3 & 3.0
MySQL 5.5
MySQL Enterprise
MySQL 5.4
MySQL Utilities
MySQL Cluster 7.4
MySQL Workbench 5.2 & 6.0
MySQL Cluster 7.3
MySQL Migration Wizard
MySQL Cluster 7.2
MySQL Cluster 7.1
MySQL Enterprise
Oracle Certifications
MySQL Fabric
MySQL Migration Wizard
Windows installer & Tools
MySQL Applier for Hadoop

MySQL 5.6
MySQL Workbench 6.1
MySQL 5.7
MySQL Cluster Manager
MySQL Enterprise Monitor 2.3 & 3.0
MySQL 5.5
MySQL Enterprise
MySQL 5.4
MySQL Utilities
MySQL Cluster 7.4
MySQL Workbench 5.2 & 6.0
MySQL Cluster 7.3
MySQL Migration Wizard
MySQL Cluster 7.2
MySQL Cluster 7.1
MySQL Enterprise
Oracle Certifications
MySQL Fabric
MySQL Migration Wizard
Windows installer & Tools
MySQL Applier for Hadoop

MySQL 5.6
MySQL Workbench 6.1
MySQL 5.7
MySQL Cluster Manager
MySQL Enterprise Monitor 2.3 & 3.0
MySQL 5.5
MySQL Enterprise
MySQL 5.4
MySQL Utilities
MySQL Cluster 7.4
MySQL Workbench 5.2 & 6.0
MySQL Cluster 7.3
MySQL Migration Wizard
MySQL Cluster 7.2
MySQL Cluster 7.1
MySQL Enterprise
Oracle Certifications
MySQL Fabric
MySQL Migration Wizard
Windows installer & Tools
MySQL Applier for Hadoop
MySQL 5.7: DMR 4

MySQL 5.7 builds on MySQL 5.6 by improving:

- **InnoDB** for better transactional throughput, availability, IO
- **Replication** for better scalability and availability
- **Utilities** for dev/ops automation
- **Performance Schema** for better performance metrics
- **Optimizer** for better EXPLAINing, query performance, enhanced buffering and partition optimization
- **Connecting** at higher rates, improve session efficiency

Available Now! Get it here: dev.mysql.com/downloads/mysql/
MySQL 5.7 Sysbench Benchmark

Sysbench Point Select

630,000 QPS

MySQL 5.7: SysBench Read Only (Point Select)

2X Faster than MySQL 5.6
Over 3X Faster than MySQL 5.5

Intel(R) Xeon(R) CPU X7560 x86_64
5 sockets x 8 cores-HT (80 CPU threads)
2.27GHz, 256G RAM
Oracle Linux 6.5

Copyright Oracle Corporation 2014
MySQL 5.7: InnoDB Memcached

6x Faster than MySQL 5.6

1,150,000 QPS

Queries per Second vs Concurrent User Sessions

- 5.7.3
- 5.6

Intel(R) Xeon(R) CPU X7560 x86_64
8 sockets x 6 cores-HT (96 CPU threads)
2000Mhz, 256G RAM
Oracle Linux 6.2
MySQL Workbench 6.1
Performance and Status Dashboards

Network, Server, InnoDB

Analyze hotspots, costly SQL statements, wait times, locks, InnoDB stats, and more
MySQL Cluster 7.4
Better performance and operational simplicity

- Performance gain over 7.3
  - 47% (Read-Only)
  - 38% (Read-Write)

- Faster node restarts
  - Recovering nodes rejoin the cluster faster
MySQL Utilities 1.4

Powerful DevOps Management tools for MySQL

- Automate common Dev/Ops tasks
  - Replication: provisioning, testing, monitoring and failover
  - Database comparisons: consistency checking
  - Database administration: users, connections, tables
  - Auditing

- Python scripts
  - Now standalone or launched from MySQL Workbench
  - Extensible to include custom scripting; Python library for extensibility
## MySQL Utilities

$ mysqluc -e "help utilities"
Launching console ...

<table>
<thead>
<tr>
<th>Utility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mysqlauditadmin</td>
<td>audit log maintenance utility</td>
</tr>
<tr>
<td>mysqlauditgrep</td>
<td>audit log search utility</td>
</tr>
<tr>
<td>mysqldbcompare</td>
<td>compare databases for consistency</td>
</tr>
<tr>
<td>mysqldbcopy</td>
<td>copy databases from one server to another</td>
</tr>
<tr>
<td>mysqldbexport</td>
<td>export metadata and data from databases</td>
</tr>
<tr>
<td>mysqldbimport</td>
<td>import metadata and data from files</td>
</tr>
<tr>
<td>mysqlldiff</td>
<td>compare object definitions among objects where the difference is how db1.obj1 differs from db2.obj2</td>
</tr>
<tr>
<td>mysqldiskusage</td>
<td>show disk usage for databases</td>
</tr>
<tr>
<td>mysqlfailover</td>
<td>automatic replication health monitoring and failover</td>
</tr>
<tr>
<td>mysqlfrm</td>
<td>show CREATE TABLE from .frm files</td>
</tr>
</tbody>
</table>
# MySQL Utilities

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>mysqlindexcheck</td>
<td>check for duplicate or redundant indexes</td>
</tr>
<tr>
<td>mysqlmetagrep</td>
<td>search metadata</td>
</tr>
<tr>
<td>mysqlprocgrep</td>
<td>search process information</td>
</tr>
<tr>
<td>mysqlreplicate</td>
<td>establish replication with a master</td>
</tr>
<tr>
<td>mysqlrpladmin</td>
<td>administration utility for MySQL replication</td>
</tr>
<tr>
<td>mysqlrplcheck</td>
<td>check replication</td>
</tr>
<tr>
<td>mysqlrplms</td>
<td>establish multi-source replication</td>
</tr>
<tr>
<td>mysqlrplshow</td>
<td>show slaves attached to a master</td>
</tr>
<tr>
<td>mysqlrplsync</td>
<td>replication synchronization checker utility</td>
</tr>
<tr>
<td>mysqlserverclone</td>
<td>start another instance of a running server</td>
</tr>
<tr>
<td>mysqlserverinfo</td>
<td>show server information</td>
</tr>
<tr>
<td>mysqluserclone</td>
<td>clone a MySQL user account to one or more new users</td>
</tr>
</tbody>
</table>
MySQL Fabric
An extensible and easy-to-use framework for managing a farm of MySQL server supporting high-availability and sharding
MySQL Fabric 1.4
High Availability + Sharding-Based Scale-out

- High Availability:
  - Server monitoring with auto-promotion and transparent application failover
- Fabric-aware connectors rather than proxy: Python, Java & PHP
- Optionally scale-out through sharding
  - Application provides shard key
  - Range or Hash
  - Tools for resharding
  - Global updates & tables
- Available in MySQL Utilities 1.4
MySQL Fabric Framework

MySQL Fabric Node

State & Routing Info

Coordination and Control

SQL Queries

HA Group

Shard 2

Primary

Secondary

Extra Read Replicas

Copyright Oracle Corporation 2014
MySQL Fabric: Prerequisites

- MySQL Servers (version 5.6.10 or later)
  - Backing store database server
  - Application database servers
- Python 2.6 or 2.7
  - No support for 3.x yet
- Connector/Python 1.2.1 or later
- MySQL Utilities 1.4
  - Available at https://dev.mysql.com/downloads/tools/utilities
High-Level Components

- Fabric-aware Connectors
  - Python, PHP, and Java
  - Enhanced Connector API

- MySQL Fabric Node
  - Manage information about farm
  - Provide status information
  - Execute procedures

- MySQL Servers
  - Organized in High-Availability Groups
  - Handling application data
MySQL Replication & MySQL Fabric HA & how this effects failover

- MySQL Replication is the initial implementation used in HA Groups
  - PRIMARY = Replication Master & receives all writes
- Failover
  - MySQL Fabric detects failure of PRIMARY/Master
  - Selects a SECONDARY/Slave and promotes it
  - Updates State Store
  - Pushes state change to Fabric-aware connectors
MySQL Fabric: Configuration

- Backing Store
  - MySQL server
  - Persistent storage for state
  - Storage engine-agnostic
- Protocol
  - Address where node will be
  - Currently only XML-RPC
- Logging
  - Chatty: **INFO** (default)
  - Moderate: **WARNING**
  - URL for rotating log

```plaintext
[storage]
address = localhost:3306
user = fabric
password =
database = fabric

[servers]
user = fabric
password =

[protocol.xmlrpc]
address = localhost:32274
threads = 5
disable_authentication = yes

[logging]
level = INFO
url = file:///var/log/fabric.log
```
MySQL Replication & MySQL Fabric HA

& how this effects failover

- MySQL Replication is the initial implementation used in HA Groups
  - PRIMARY = Replication Master & receives all writes

- Failover
  - MySQL Fabric detects failure of PRIMARY/Master
  - Selects a SECONDARY/Slave and promotes it
  - Updates State Store
  - Pushes state change to Fabric-aware connectors
High-Availability Group Concept

- Abstract Concept
  - Set of servers
  - Server attributes

- Connector Attributes
  - Connection information
  - **Mode**: read-only, read-write, ...
  - **Weight**: distribute load

- Management Attributes
  - **State**: state/role of the server
  - **State**: Primary
  - **Mode**: Read-Write
  - **Host**: server-1.example.com
Create HA Groups and add Servers

- Define a group
  
  ```
  mysqlfabric group create my_group
  ```

- Add servers to group
  
  ```
  mysqlfabric group add my_group server1.example.com \ mats xyzzy
  mysqlfabric group add my_group server2.example.com \ mats xyzzy
  ```
Create HA Groups and add Servers

- Promote one server to be primary
  
  mysqlfabric group promote my_group

- Tell failure detector to monitor group
  
  mysqlfabric group activate my_group
The Path to Scalability
Scaling-Up can take you a long way

Scaling on dense, multi-core, multi-thread servers
- 10s - 100GBs RAM
- SSDs
Scale across cores within a single instance
You can get a long way with MySQL 5.6!

MySQL 5.6 vs 5.5 Read Write (Linux)
Benefits of Sharding

- Write scalability
  - Can handle more writes
- Large data set
  - Database too large
  - Does not fit on single server
- Improved performance
  - Smaller index size
  - Smaller working set
  - Improve performance
MySQL Fabric Features

- Connector API Extensions
  - Support Transactions
  - Support full SQL
- Decision logic in connector
  - Reducing network load
- Shard Multiple Tables
  - Using same key
- Global Updates
  - Global tables
  - Schema updates
- Sharding Functions
  - Range
  - (Consistent) Hash
- Shard Operations
  - Shard move
  - Shard split
MySQL Fabric Framework

MySQL Fabric Node

Coordination and Control

Server/Shard State & Mapping

SQL Queries

Master Group

Global Data

Primary

Secondary

Extra Read Replicas

Shard 1

Primary

Secondary

Extra Read Replicas

Shard 2

Primary

Secondary

Extra Read Replicas

Extra Read Replicas

HA Group

Master Group

PHP

Python

Java

Copyright Oracle Corporation 2014
Routing Transactions

- App Server
- Connector
- Cache
- State Store
- Shard #1
- Shard #2
- Shard #3
- Executor
Routing Transactions

App Server
Connector
Cache

App Server
Connector
Cache

State Store

Shard #1
Shard #2
Shard #3
Executor
MySQL Fabric: Sharding Setup

- Set up some groups
  - `my_global` – for global updates
  - `my_group.N` – for the shards
  - Add servers to the groups

- Create a shard mapping
  - A “distributed database”
  - Mapping keys to shards
  - Give information on what tables are sharded

- Add shards
MySQL Fabric: Moving and Splitting Shards

- Moving a shard from one group to another
  
  `mysqlfabric sharding move 5 my_group.8`

- Splitting a shard into two pieces (hash)
  
  `mysqlfabric sharding split 5 my_group.6`
Connector API: Shard Specific Query

- Provide tables in query
  - **Property:** tables
  - Fabric will compute map

- Provide sharding key
  - **Property:** key
  - Fabric will compute shard

```python
conn.set_property(tables=['employees.employees', 'employees.titles'],
                  key=emp_no)
cur = conn.cursor()
cur.execute("INSERT INTO employees VALUES (%s,%s,%s)",
            (emp_no, first_name, last_name))
cur.execute("INSERT INTO titles(emp_no, title, from_date)"
            " VALUES (%s, %s, CURDATE())",
            (emp_no, 'Intern'));
conn.commit()
```
Connector API: Shard Specific Query

- Provide tables in query
  - **Property:** tables
  - Fabric will compute map

```python
conn.set_property(tables=['employees.employees', 'employees.titles'],
                 key=emp_no)
cur = conn.cursor()
cur.execute(
    "SELECT first_name, last_name, title"
    " FROM employees JOIN titles USING (emp_no)"
    " WHERE emp_no = %d", (emp_no,))
for row in cur:
    print row[0], row[1], ",", row[2]
```

- Provide sharding key
  - **Property:** key
  - Fabric will compute shard
Connector API: Global Update

- Provide tables in query
  - **Property**: tables
  - Fabric will compute map
  - (Likely to not be needed)

- Set global scope
  - **Property**: scope
  - Query goes to global group

```python
conn.set_property(tables=['employees.titles'], scope='GLOBAL')
cur = conn.cursor()
cur.execute("ALTER TABLE employees.titles ADD nickname VARCHAR(64)")
```
MySQL Fabric Node
Extensible Architecture

MySQL Fabric Node
Connector
Connector
Connector

Protocols
AMQP
MySQL
XML-RPC

MySQL Fabric Framework
Executor
State Store (Persister)
Sh
HA

Extensions
Backling Store
MySQL
MySQL Fabric: Goals & Features

- Connector API Extensions
  - Support Transactions
  - Support full SQL

- Fabric-Aware Connectors at GA:
  - PHP + Doctrine, Python, Java + Hibernate

- Decision logic in connector
  - Reducing network load

- Load Balancing
  - Read-Write Split
  - Distribute transactions

- Global Updates
  - Global tables
  - Schema updates

- Shard Multiple Tables
  - Using same key

- Sharding Functions
  - Range
  - (Consistent) Hash

- Shard Operations
  - Shard move
  - Shard split
MySQL Fabric – Current Limitations

- Routing is dependent on Fabric-aware connectors
  - Currently Java (+ Hibernate), PHP (+ Doctrine) & Python
- MySQL Fabric node is a single (non-redundant process)
  - HA Maintained as connectors continue to route using local caches
- Establishes asynchronous replication
  - Manual steps to switch to semisynchronous
- Sharding not transparent to application (must provide shard key)
- No cross-shard joins or other queries
- Management in through CLI or XML/RPC API
  - No GUI
## Oracle MySQL HA & Scaling Solutions

<table>
<thead>
<tr>
<th>Feature</th>
<th>MySQL Replication</th>
<th>MySQL Fabric</th>
<th>Oracle VM Template</th>
<th>Solaris Cluster</th>
<th>Windows Cluster</th>
<th>DRBD</th>
<th>MySQL Cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>App Auto-Failover</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Data Layer Auto-Failover</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Zero Data Loss</td>
<td>MySQL 5.7</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Platform Support</td>
<td>All</td>
<td>All</td>
<td>Linux</td>
<td>Solaris</td>
<td>Windows</td>
<td>Linux</td>
<td>All</td>
</tr>
<tr>
<td>Clustering Mode</td>
<td>Master + Slaves</td>
<td>Master + Slaves</td>
<td>Active/Passive</td>
<td>Active/Passive</td>
<td>Active/Passive</td>
<td>Active/Passive</td>
<td>Multi-Master</td>
</tr>
<tr>
<td>Failover Time</td>
<td>N/A</td>
<td>Secs</td>
<td>Secs +</td>
<td>Secs +</td>
<td>Secs +</td>
<td>Secs +</td>
<td>&lt; 1 Sec</td>
</tr>
<tr>
<td>Scale-out</td>
<td>Reads</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Cross-shard operations</td>
<td>N/A</td>
<td>✗</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>✓</td>
</tr>
<tr>
<td>Transparent routing</td>
<td>✗</td>
<td>For HA</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Shared Nothing</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Storage Engine</td>
<td>InnoDB+</td>
<td>InnoDB+</td>
<td>InnoDB+</td>
<td>InnoDB+</td>
<td>InnoDB+</td>
<td>InnoDB+</td>
<td>NDB</td>
</tr>
<tr>
<td>Single Vendor Support</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

40 | Copyright Oracle Corporation 2014
MySQL Fabric Resources

- Download and try

- Documentation

- Forum (MySQL Fabric, Sharding, HA, Utilities)

- Tutorial: MySQL Fabric - adding High Availability and Scaling to MySQL


- Webinar Replays