MySQL Fabric

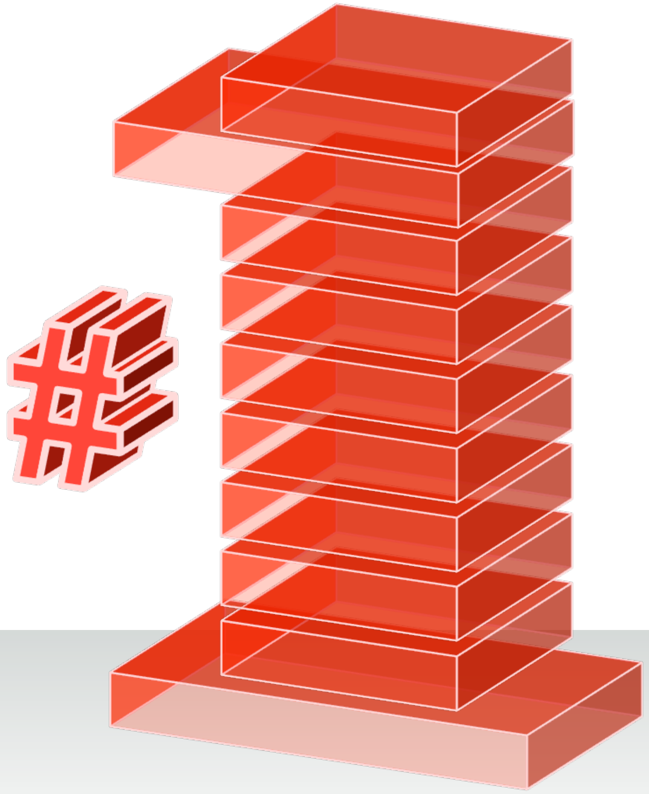
MySQL Global Business Unit
Sales Consulting Manager, JAPAC
梶山 隆輔 / Ryusuke Kajiyama

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 Workbench 6.1

MySQL Migration Wizard

MySQL 5.6

MySQL 5.5

Windows installer & Tools

MySQL 5.7

MySQL Cluster Manager

MySQL Applier for Hadoop

MySQL Enterprise Monitor 2.3 & 3.0

MySQL Cluster 7.4

MySQL Enterprise

- Backup
- Security
- Scalability
- HA
- Audit

MySQL Utilities

MySQL Cluster 7.3

MySQL Workbench 5.2 & 6.0

MySQL Cluster 7.2

MySQL Enterprise Oracle Certifications

MySQL Cluster 7.1

ORACLE

MySQL 5.7: DMR 4

MySQL 5.7 builds on MySQL 5.6 by improving:

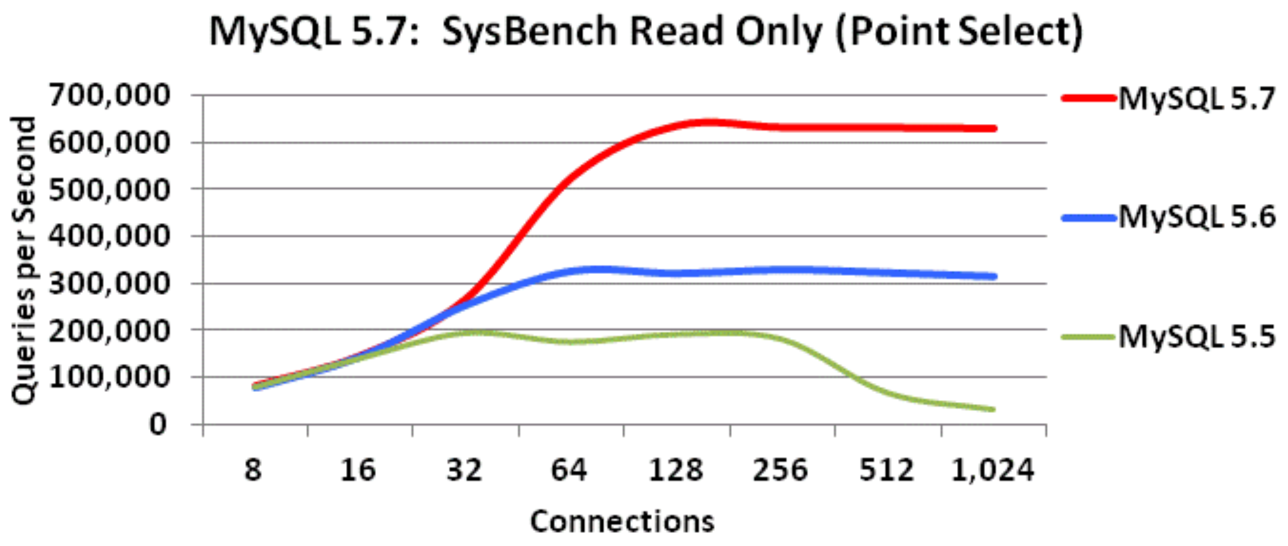
- **InnoDB** for better transactional throughput, availability, IO
- **Replication** for better scalability and availability
- **Utilities** for dev/ops automation
- **Performance Schema** for better performance metrics
- **Optimizer** for better EXPLAINing, query performance, enhanced buffering and 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



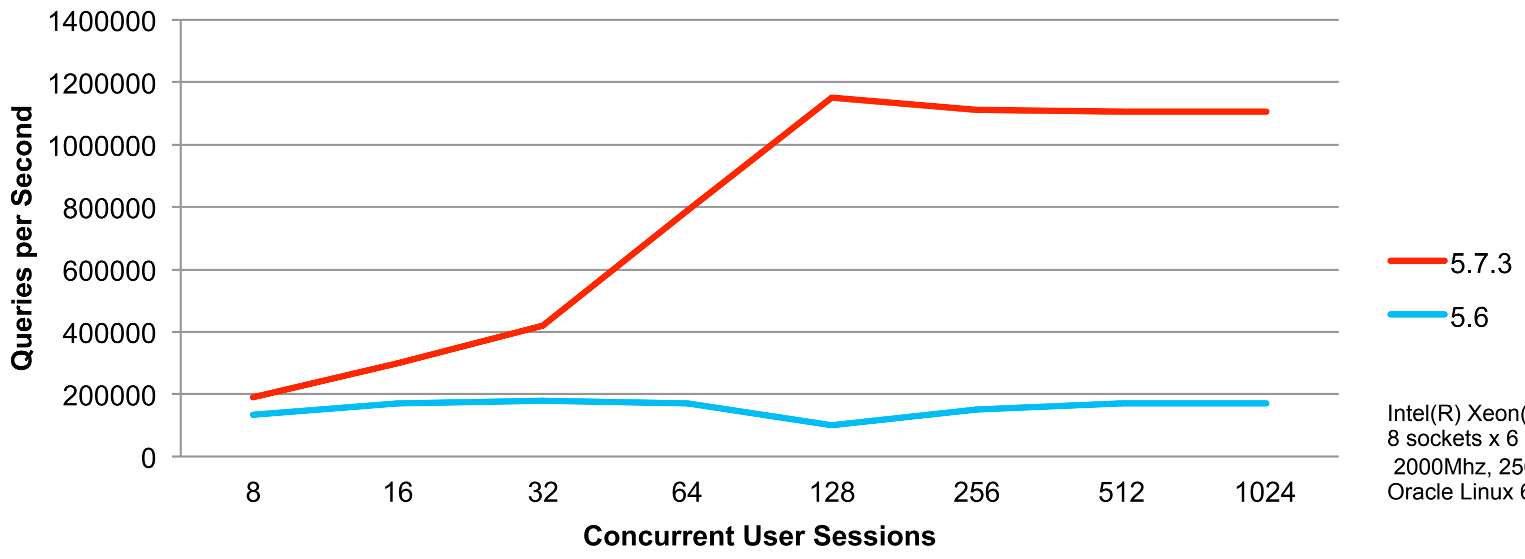
Intel(R) Xeon(R) CPU X7560 x86_64
5 sockets x 8 cores-HT (80 CPU threads)
2.27GHz, 256G RAM
Oracle Linux 6.5

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

ORACLE

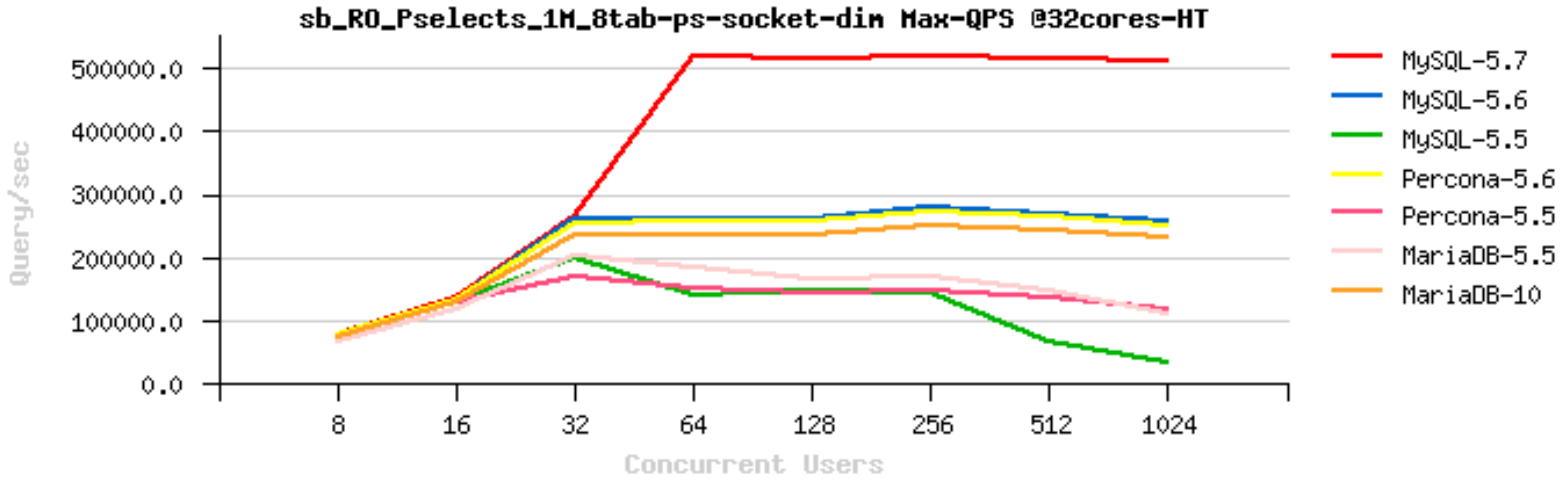
MySQL 5.7: InnoDB Memcached

1,150,000 QPS



Intel(R) Xeon(R) CPU X7560 x86_64
8 sockets x 6 cores-HT (96 CPU threads)
2000Mhz, 256G RAM
Oracle Linux 6.2

6x Faster than MySQL 5.6

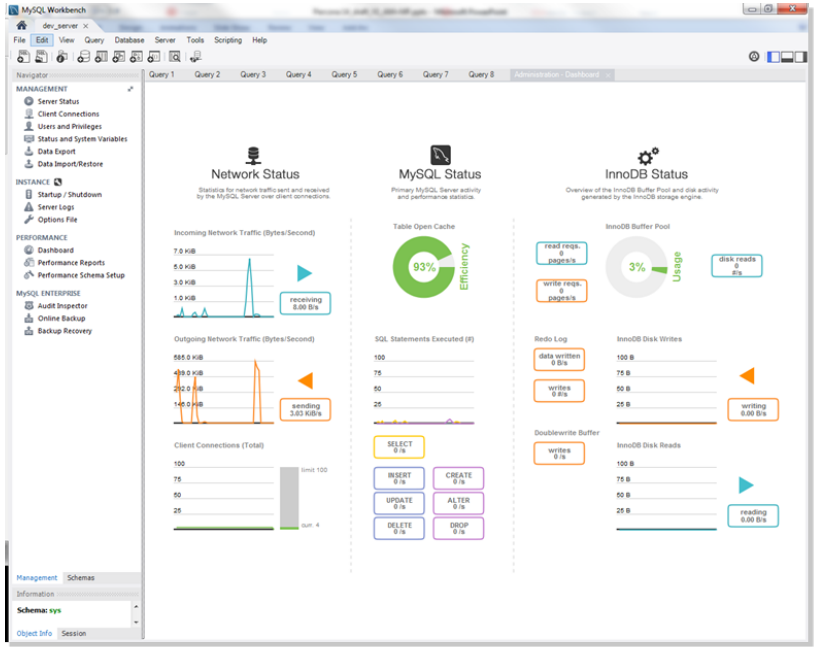


http://dimitrik.free.fr/blog/archives/2014/04/mysql-57-just-rocks.html

MySQL Workbench 6.1

Performance and Status Dashboards

GA



Network, Server, InnoDB

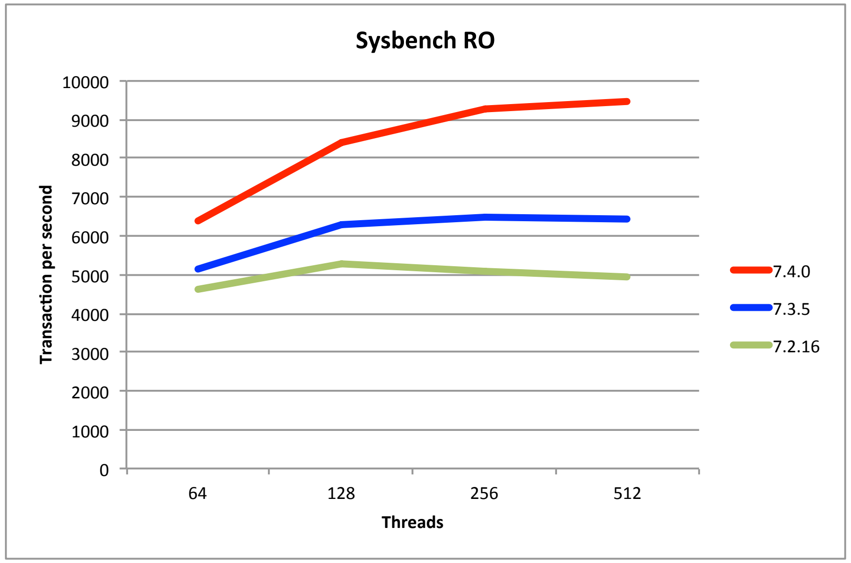
Analyze hotspots, costly SQL statements, wait times, locks, InnoDB stats, and more

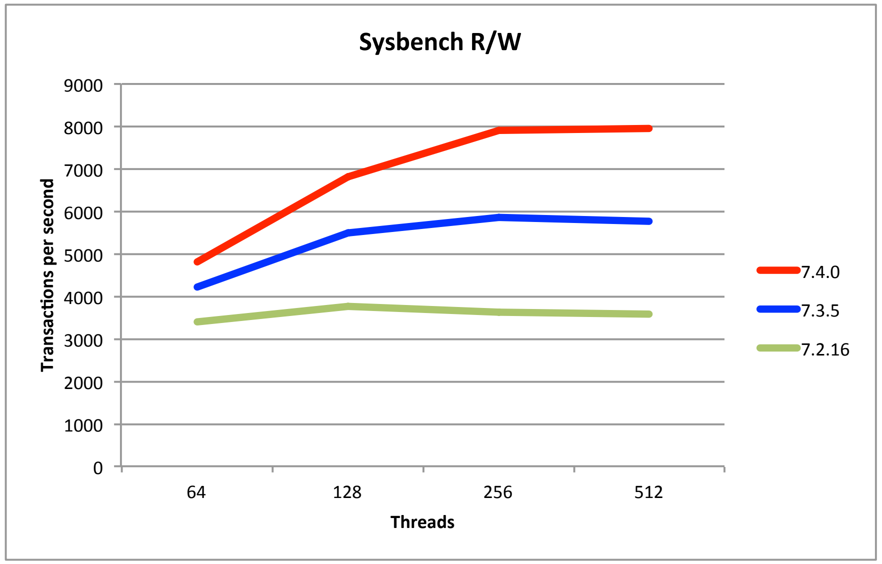
ORACLE

MySQL Cluster 7.4

Better performance and operational simplicity

labs.mysql.com





- Performance gain over 7.3
  - 47% (Read-Only)
  - 38% (Read-Write)
- Faster node restarts
  - Recovering nodes rejoin the cluster faster

ORACLE

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 ...

Utility             Description
----------------    ---------------------------------------------------------
mysqlauditadmin     audit log maintenance utility
mysqlauditgrep      audit log search utility
mysqldbcompare      compare databases for consistency
mysqldbcopy         copy databases from one server to another
mysqldbexport       export metadata and data from databases
mysqldbimport       import metadata and data from files
mysqldiff           compare object definitions among objects where the
                    difference is how db1.obj1 differs from db2.obj2
mysqldiskusage      show disk usage for databases
mysqlfailover       automatic replication health monitoring and failover
mysqlfrm            show CREATE TABLE from .frm files
...
```

MySQL Utilities

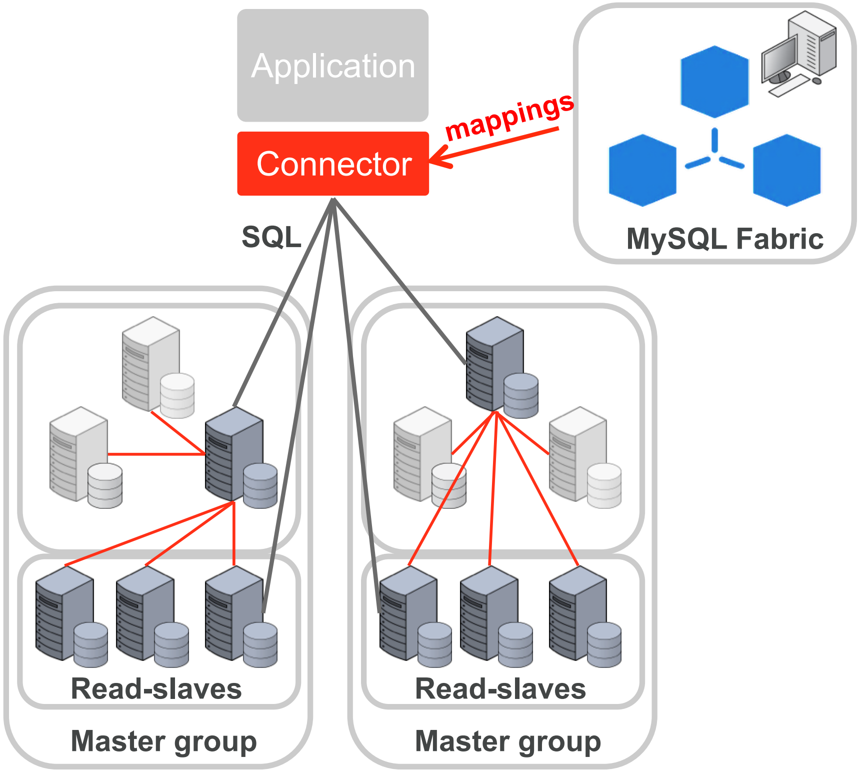
```
...
mysqlindexcheck   check for duplicate or redundant indexes
mysqlmetagrep     search metadata
mysqlprocgrep     search process information
mysqlreplicate    establish replication with a master
mysqlrpladmin     administration utility for MySQL replication
mysqlrplcheck     check replication
mysqlrplms        establish multi-source replication
mysqlrplshow      show slaves attached to a master
mysqlrplsync      replication synchronization checker utility
mysqlserverclone  start another instance of a running server
mysqlserverinfo   show server information
mysqluserclone    clone a MySQL user account to one or more new users
```

MySQL Fabric

An extensible and easy-to-use framework for managing a farm of MySQL server supporting high-availability and sharding

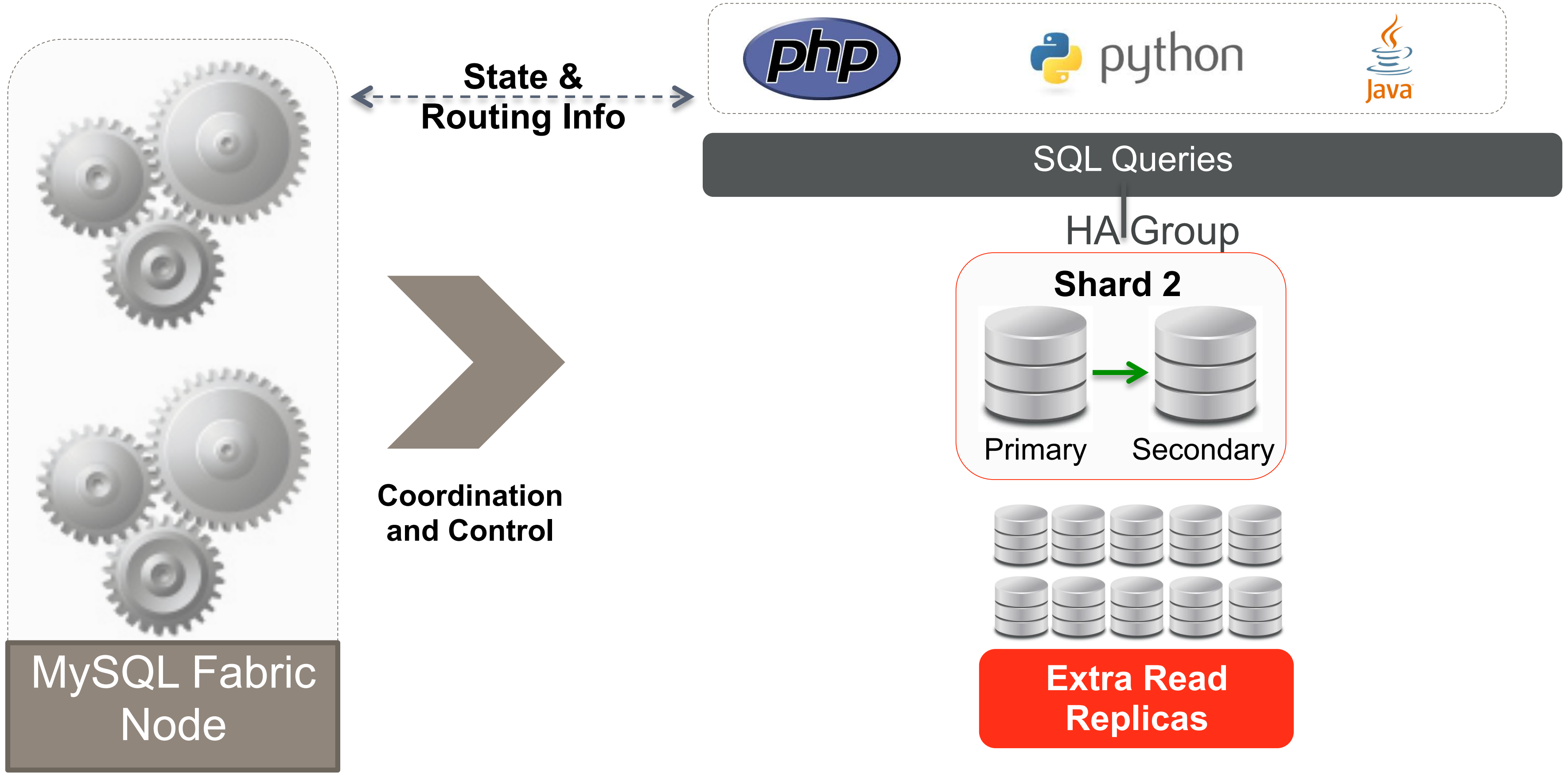
MySQL Fabric 1.4

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

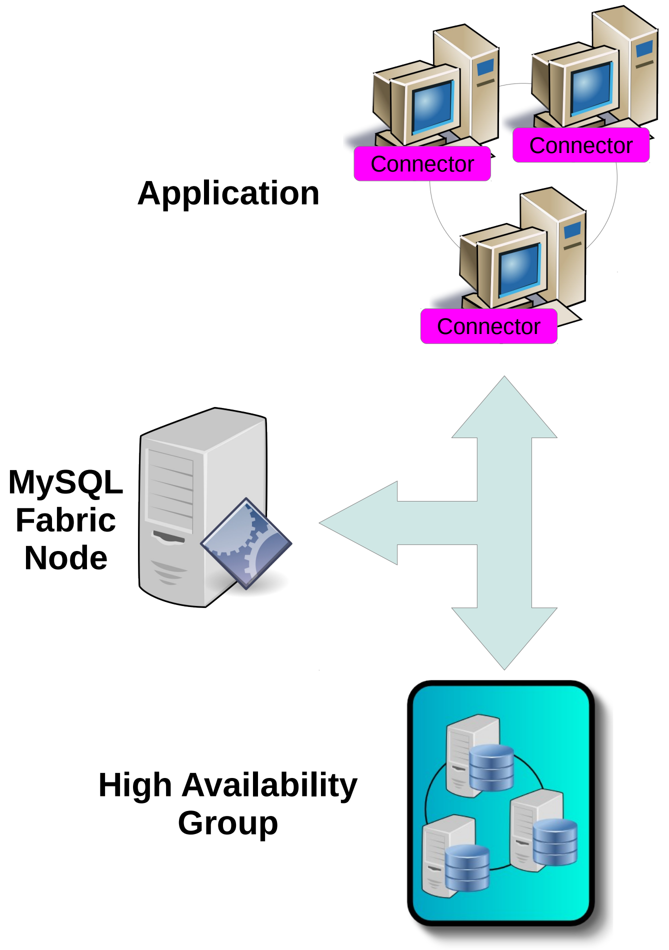


MySQL Fabric: Prerequisites

- 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



ORACLE

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

```
[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
```

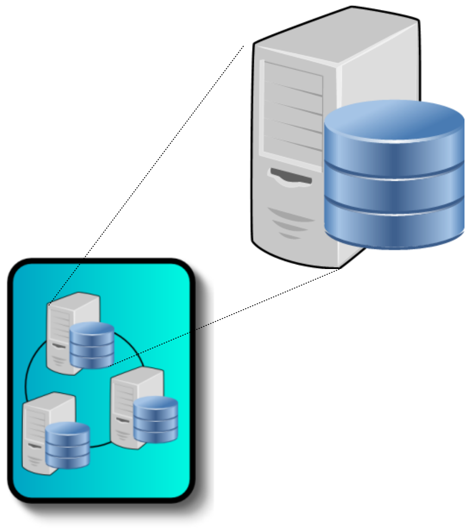
MySQL Replication & MySQL Fabric HA

& 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

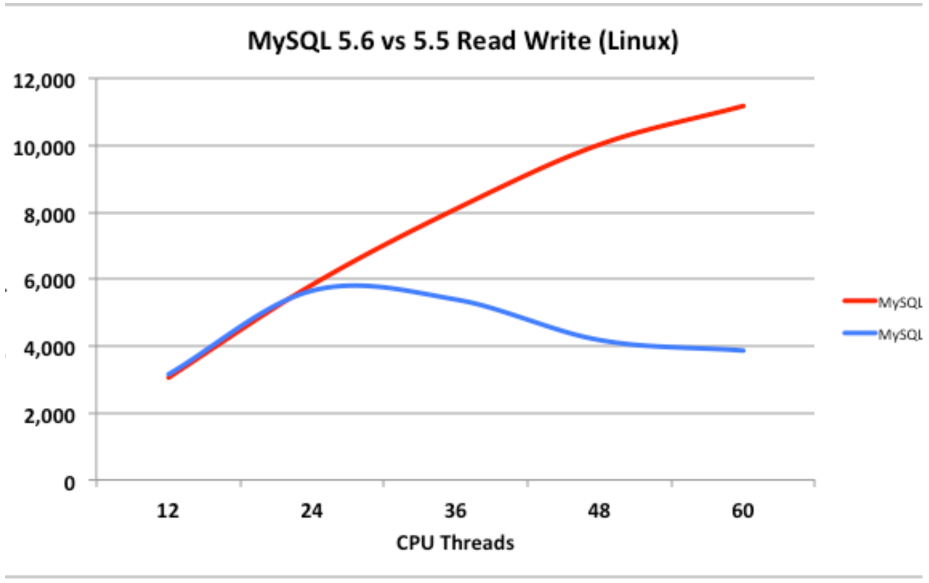
Scaling-Up can take you a long way

Scaling on dense, multi-core, multi-thread servers

- 10s - 100GBs RAM
- SSDs

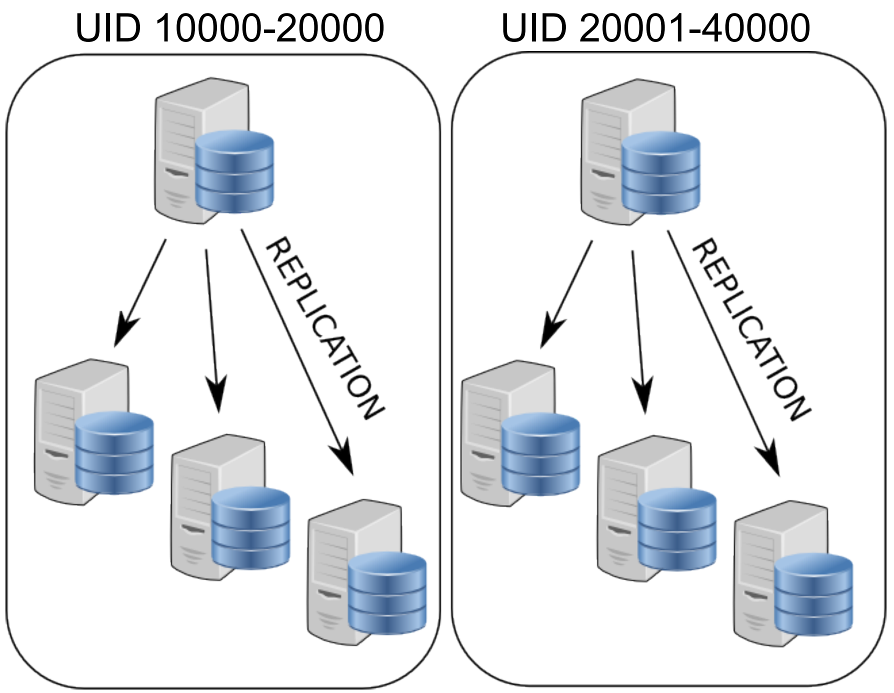
Scale across cores within a single instance

You can get a long way with MySQL 5.6!



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

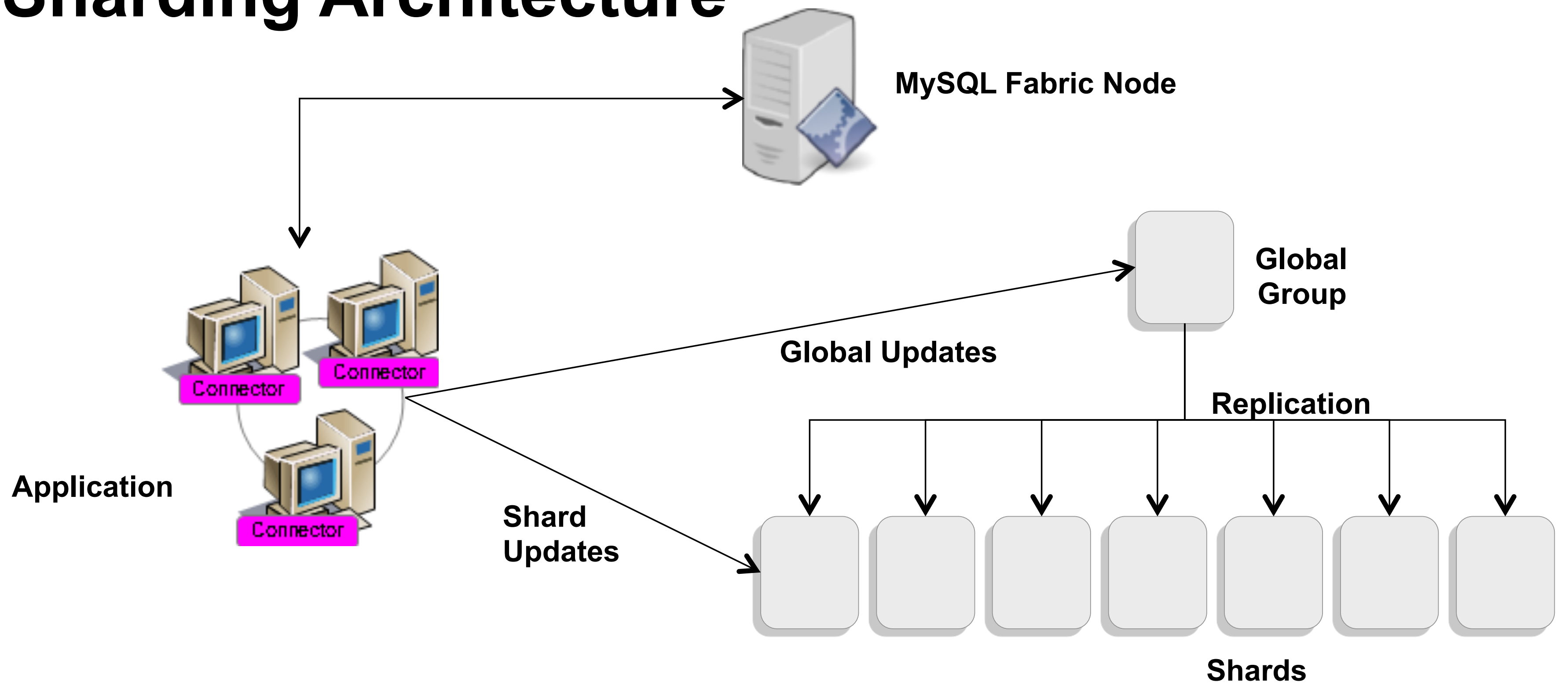


ORACLE

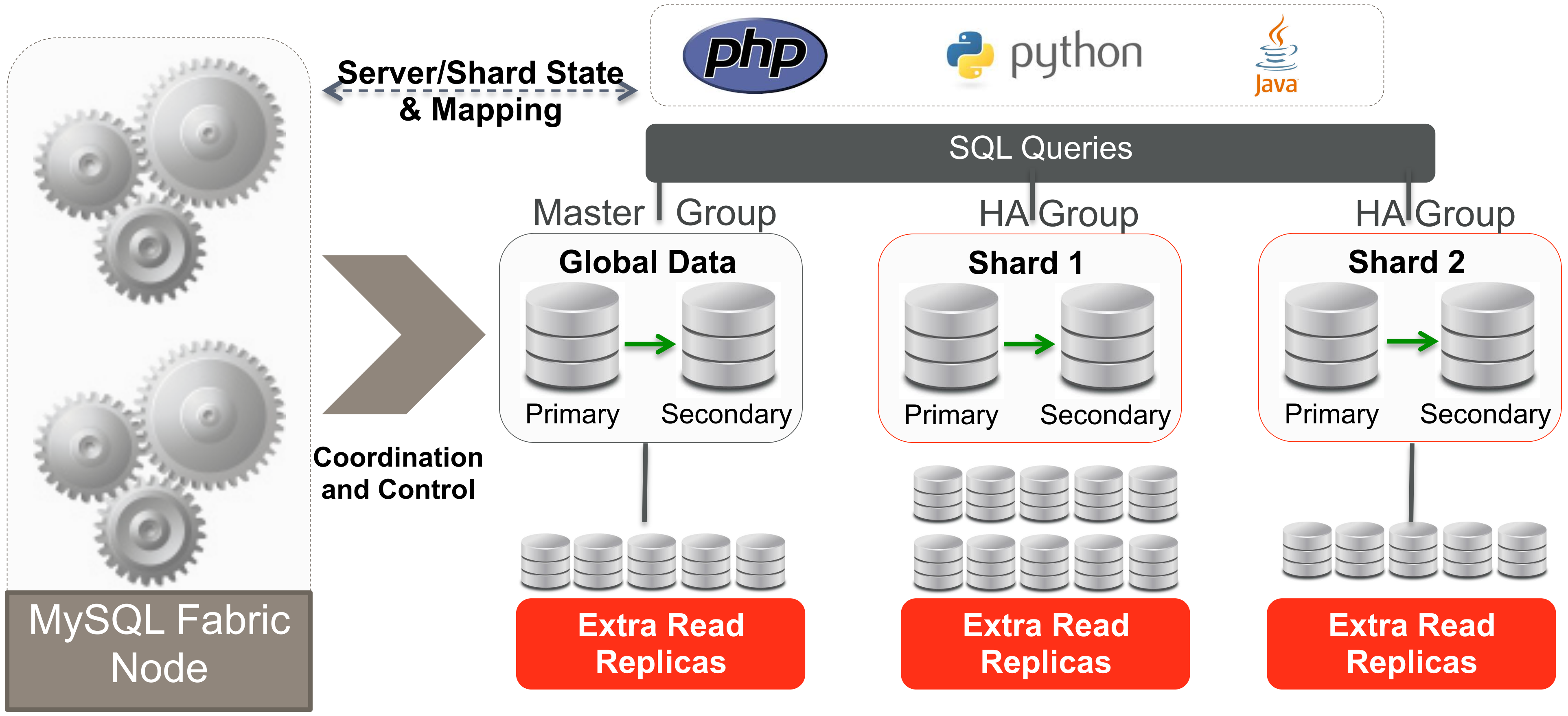
MySQL Fabric Features

- 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

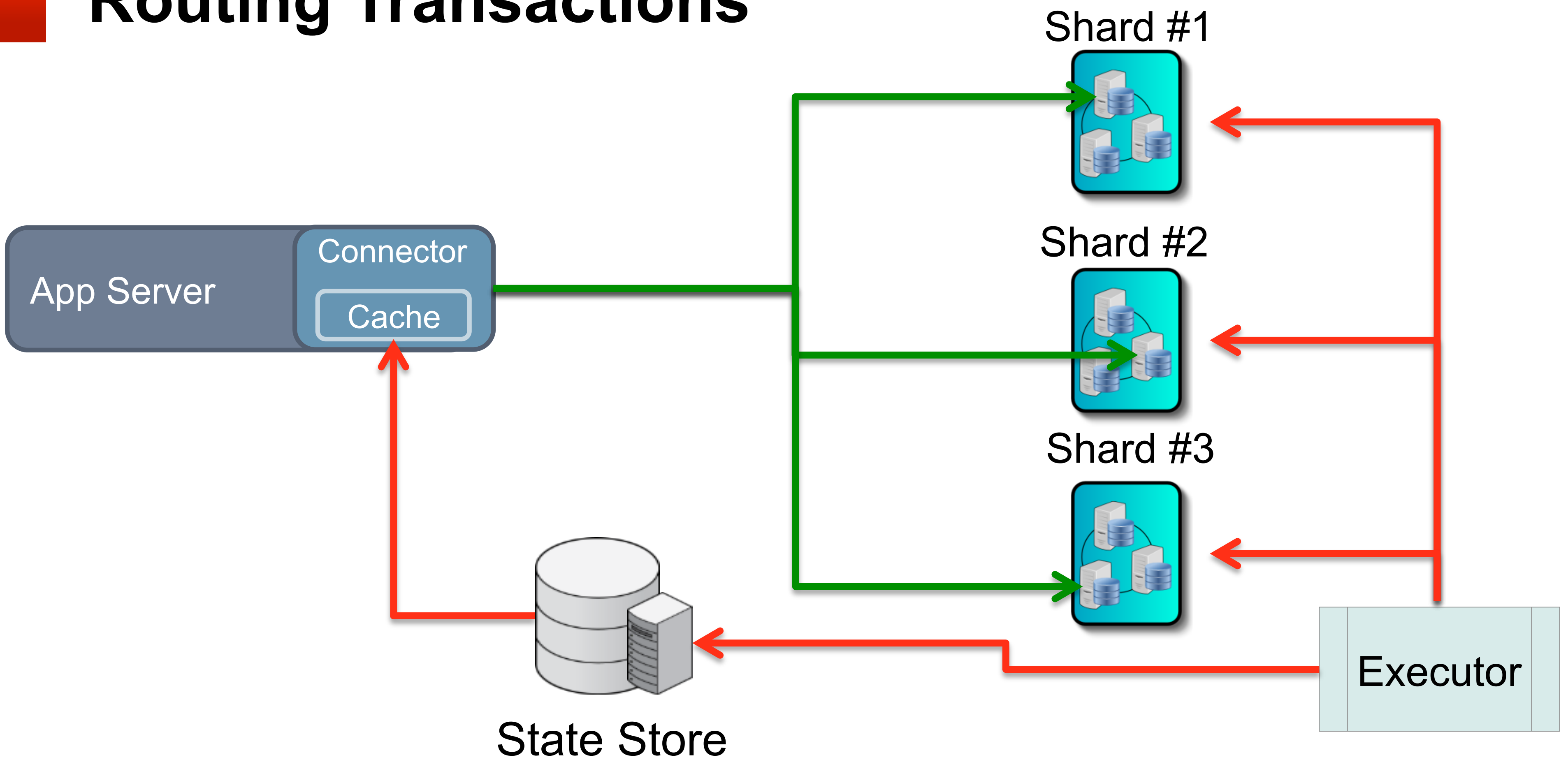
Sharding Architecture



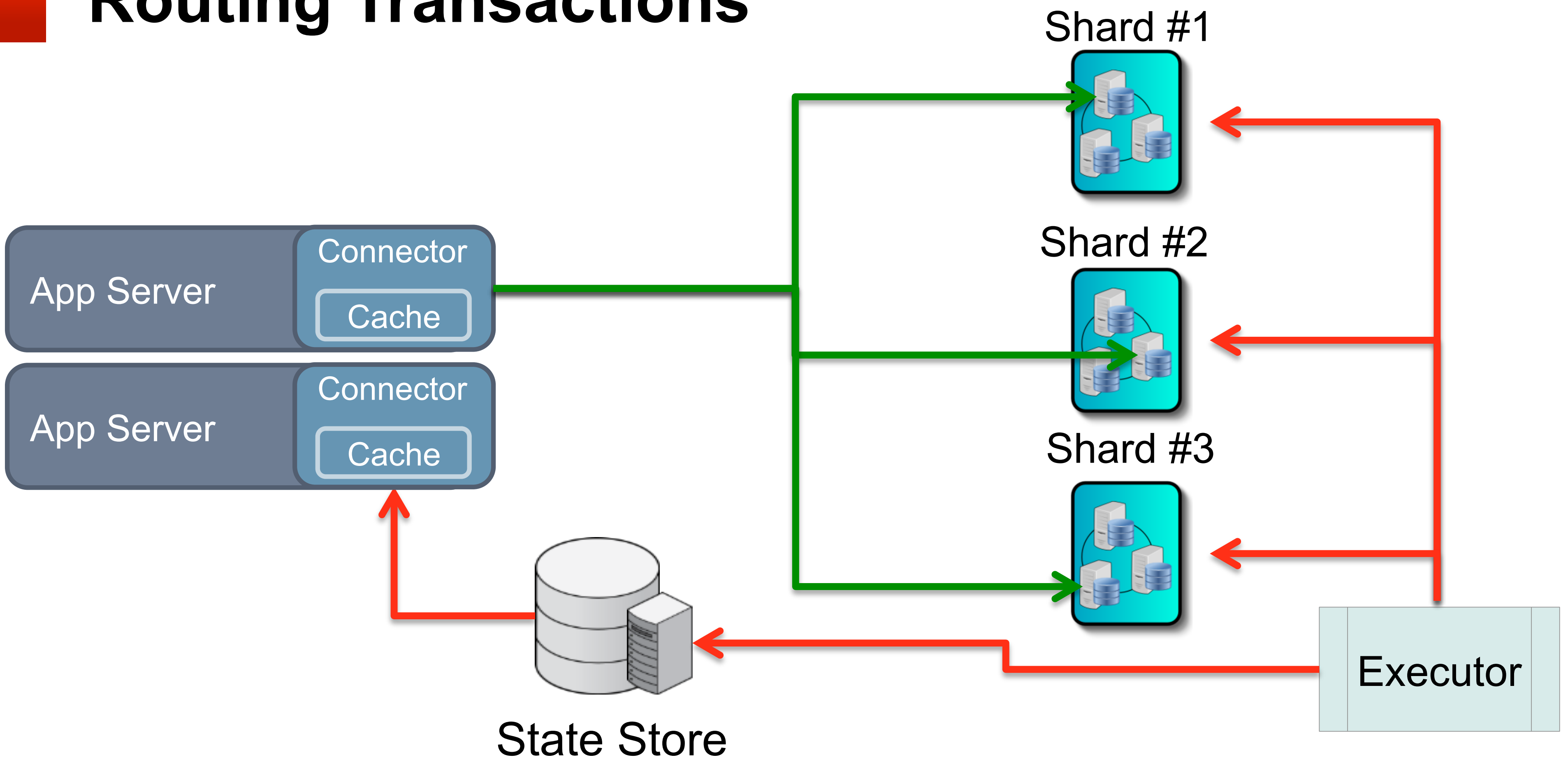
MySQL Fabric Framework



Routing Transactions



Routing Transactions



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

```
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()
```

ORACLE

Connector API: Shard Specific Query

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

```
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]
```

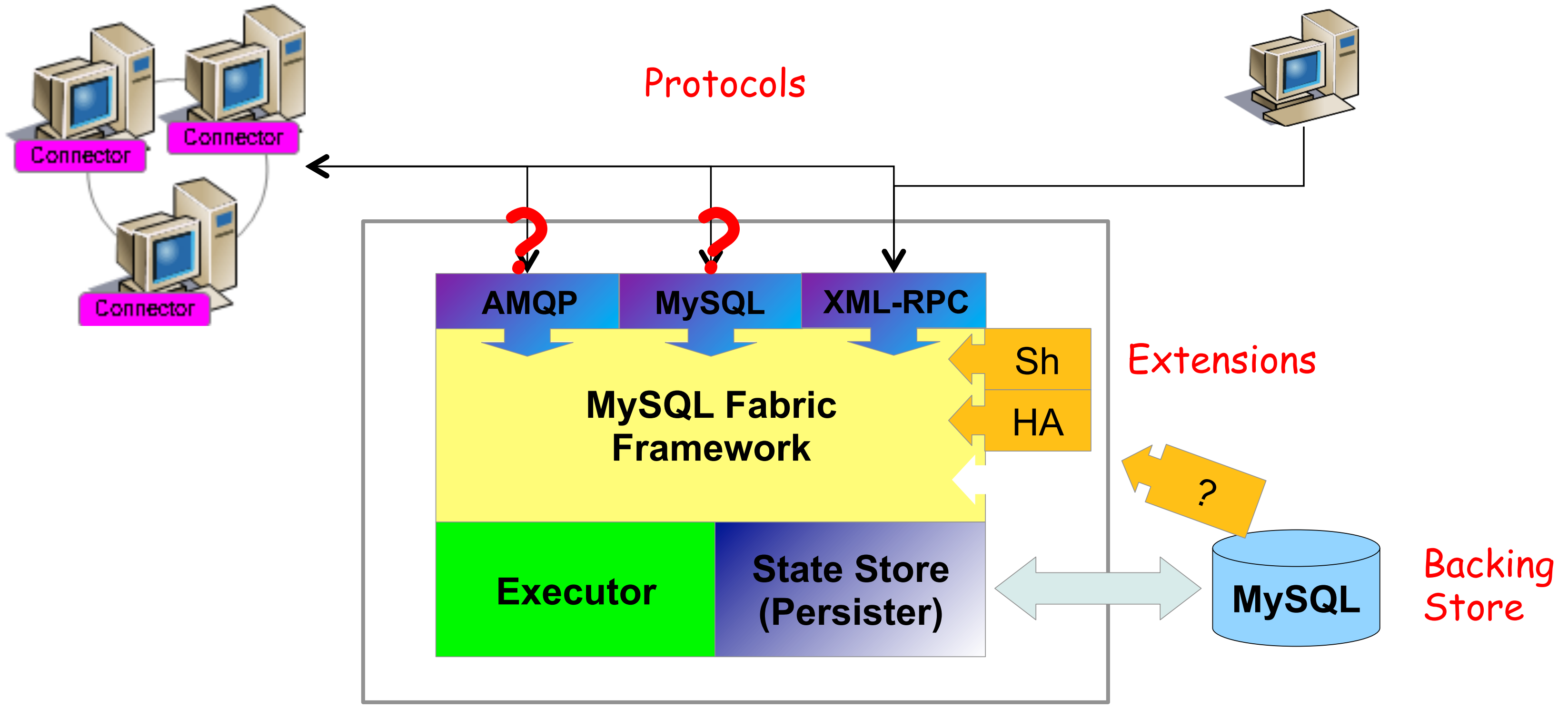
Connector API: Global Update

- 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

```
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: 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

ORACLE

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

	MySQL Replication	MySQL Fabric	Oracle VM Template	Solaris Cluster	Windows Cluster	DRBD	MySQL Cluster
App Auto-Failover	✖	✔	✔	✔	✔	✔	✔
Data Layer Auto-Failover	✖	✔	✔	✔	✔	✔	✔
Zero Data Loss	MySQL 5.7	✔	✔	✔	✔	✔	✔
Platform Support	All	All	Linux	Solaris	Windows	Linux	All
Clustering Mode	**Master + Slaves**	**Master + Slaves**	**Active/ Passive**	**Active/ Passive**	**Active/ Passive**	**Active/ Passive**	**Multi-Master**
Failover Time	**N/A**	**Secs**	**Secs +**	**Secs +**	**Secs +**	**Secs +**	**< 1 Sec**
Scale-out	Reads	✔	✖	✖	✖	✖	✔
Cross-shard operations	**N/A**	✖	**N/A**	**N/A**	**N/A**	**N/A**	✔
Transparent routing	✖	For HA	✔	✔	✔	✔	✔
Shared Nothing	✔	✔	✖	✖	✖	✔	✔
Storage Engine	InnoDB+	InnoDB+	InnoDB+	InnoDB+	InnoDB+	InnoDB+	NDB
Single Vendor Support	✔	✔	✔	✔	✖	✔	✔

MySQL Fabric Resources

- Download and try
  http://dev.mysql.com/downloads/fabric/
- Documentation
  http://dev.mysql.com/doc/mysql-utilities/1.4/en/fabric.html
- Forum (MySQL Fabric, Sharding, HA, Utilities)
  http://forums.mysql.com/list.php?144
- Tutorial: MySQL Fabric - adding High Availability and Scaling to MySQL
  http://www.clusterdb.com/mysql-fabric/mysql-fabric-adding-high-availability-and-scaling-to-mysql
- White Paper: MySQL Fabric - A Guide to Managing MySQL High Availability and Scaling Out
  http://www.mysql.com/why-mysql/white-papers/mysql-fabric-product-guide
- Webinar Replays
  http://www.mysql.com/news-and-events/on-demand-webinars/#en-20-41

ORACLE®