ORACLE®

MySQL Fabric

MySQL™

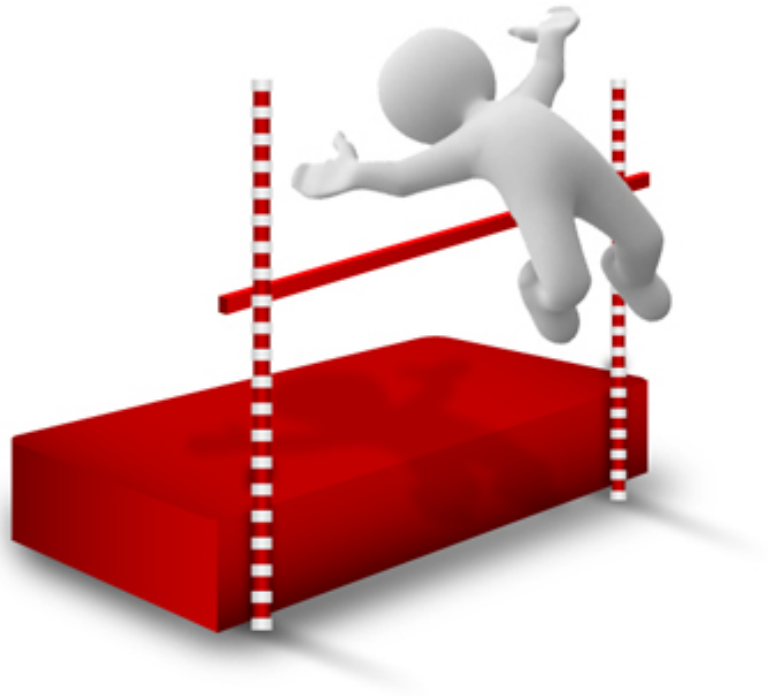
Safe Harbour Statement

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.

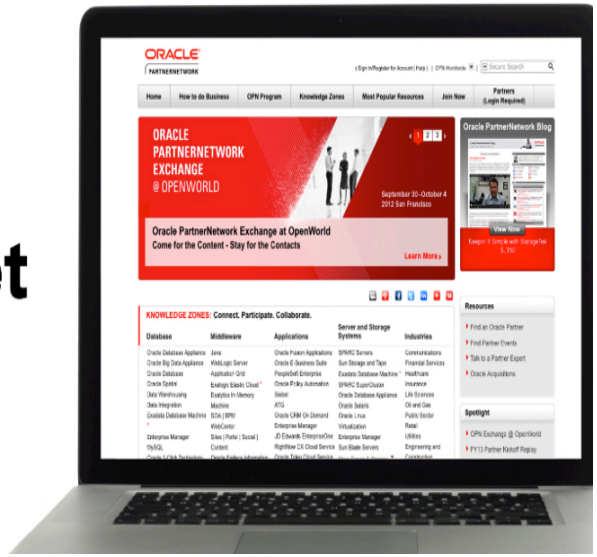
Raising The Bar

Again and Again, Evolving MySQL for You



2.3B Internet Users

Source: IDC



1.1B Global

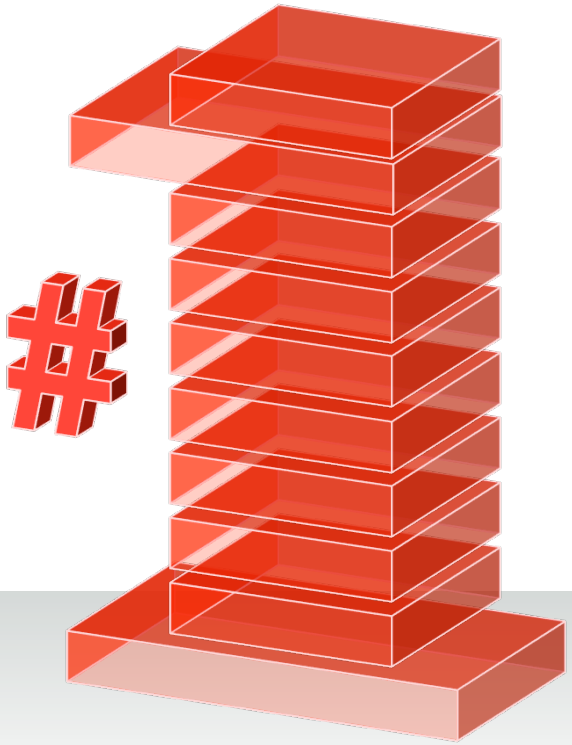
3G mobile subscribers

Source: Mobithink

3B to 50B Devices

Source: Ericsson

BEST IN CLASS
COMPONENTS



MySQL: Next Generation Web Applications
On-Premises, in the Cloud, Distributed Applications

ORACLE

MySQL@Oracle: 4 Years of MySQL Innovation

MySQL Fabric 1.4

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

MySQL Cluster 7.1

Oracle Certifications

ORACLE

Program Agenda

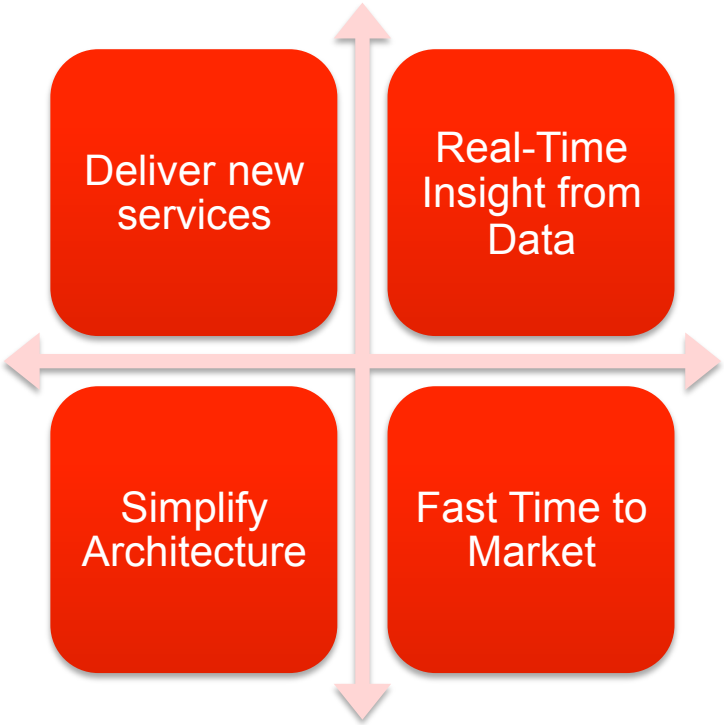
- Requirements for Next Gen Services
- Simple, transparent High Availability
- Delivering SQL & ACID at scale
- Where MySQL Fabric fits with Oracle's other MySQL solutions
- Getting started

Program Agenda

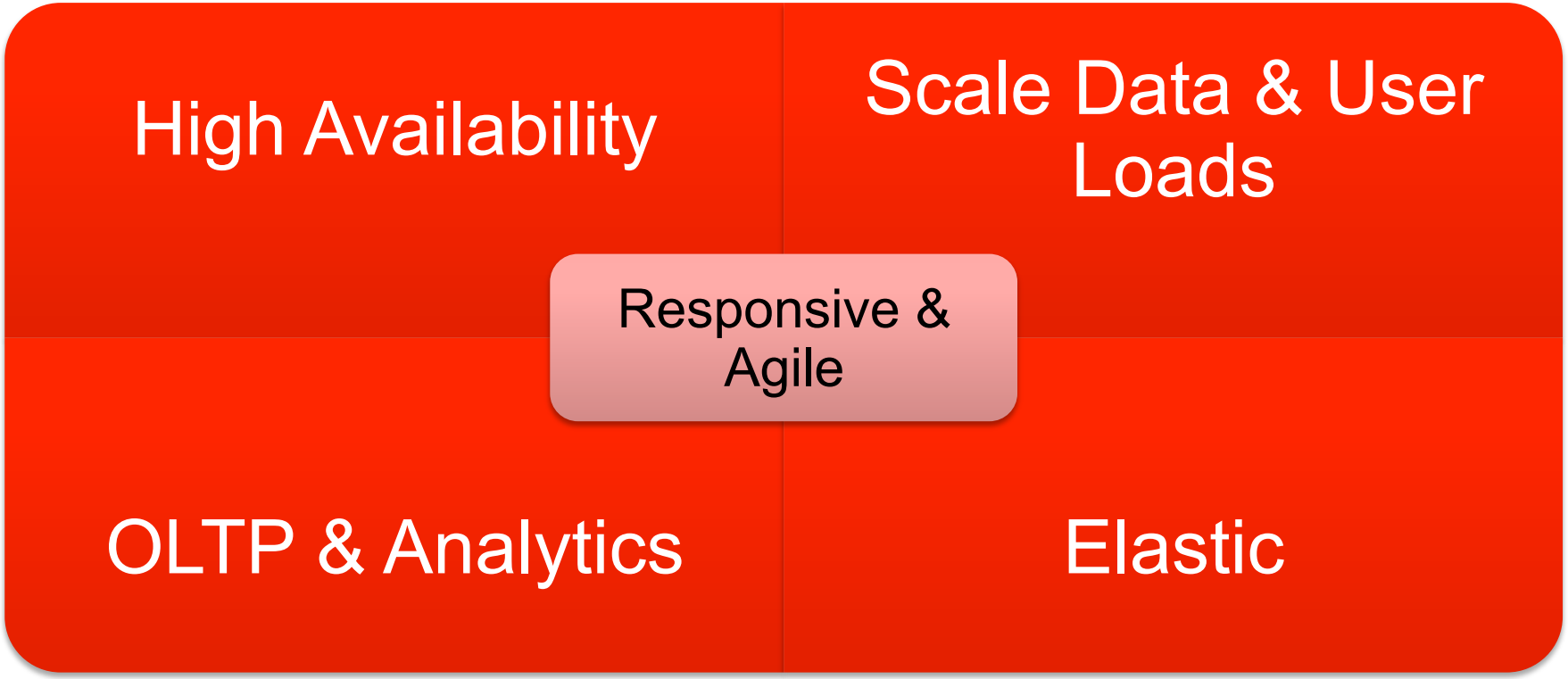
- Requirements for Next Gen Services
- Simple, transparent High Availability
- Delivering SQL & ACID at scale
- Where MySQL Fabric fits with Oracle's other MySQL solutions
- Getting started

Business Priorities for IT

Focus on driving the business rather than on infrastructure



Application Requirements



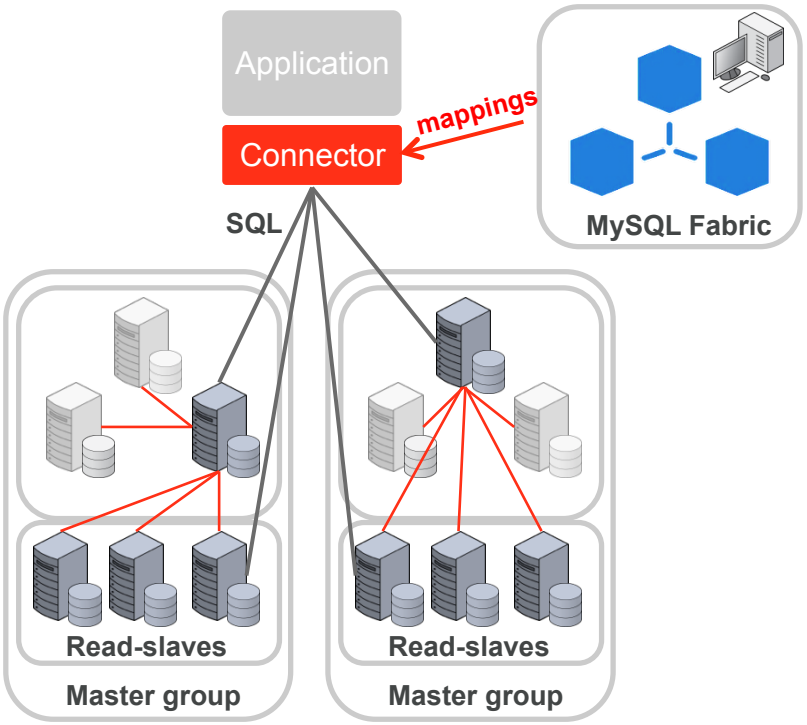
MySQL Fabric

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

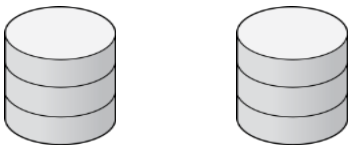


- High Availability:
  - Server monitoring with auto-promotion and transparent application failover
- Fabric-aware connectors rather than proxy: Python, Java & PHP
- Optionally scale-out through sharding
  - Application provides shard key
  - Range or Hash
  - Tools for resharding
  - Global updates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

Program Agenda

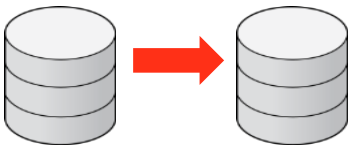
- Requirements for Next Gen Services
- Simple, transparent High Availability
- Delivering SQL & ACID at scale
- Where MySQL Fabric fits with Oracle's other MySQL solutions
- Getting started

Layers of HA



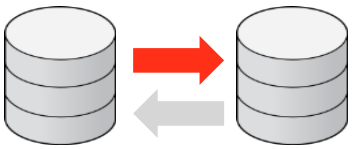
Data Redundancy

Layers of HA



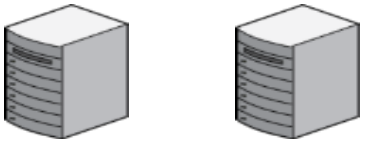
Data Redundancy

Layers of HA

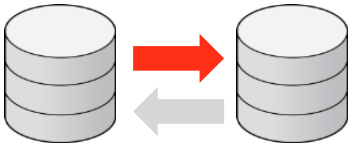


Data Redundancy

Layers of HA

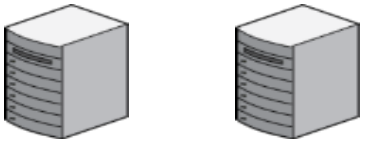


Redundant App Servers



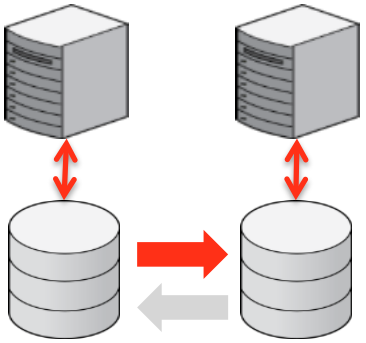
Data Redundancy

Layers of HA



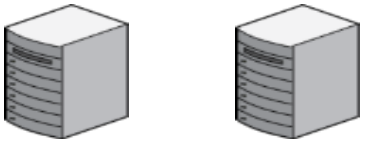
Redundant App Servers

Redundant Access to Data



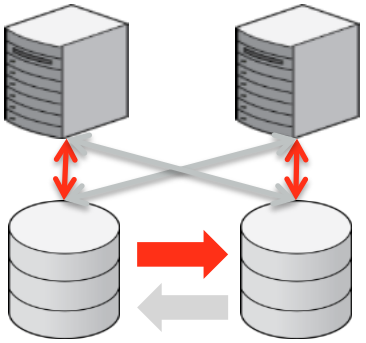
Data Redundancy

Layers of HA



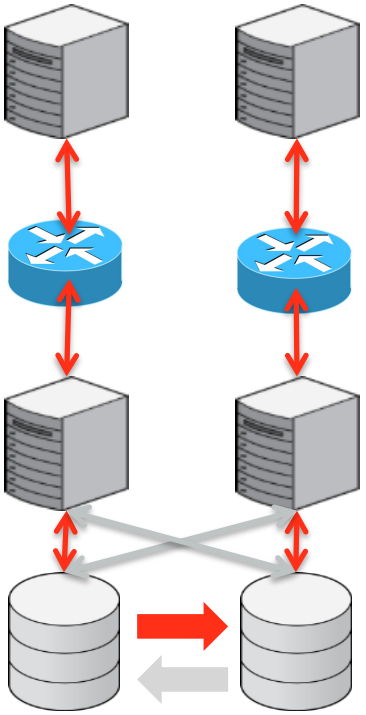
Redundant App Servers

Redundant Access to Data



Data Redundancy

Layers of HA



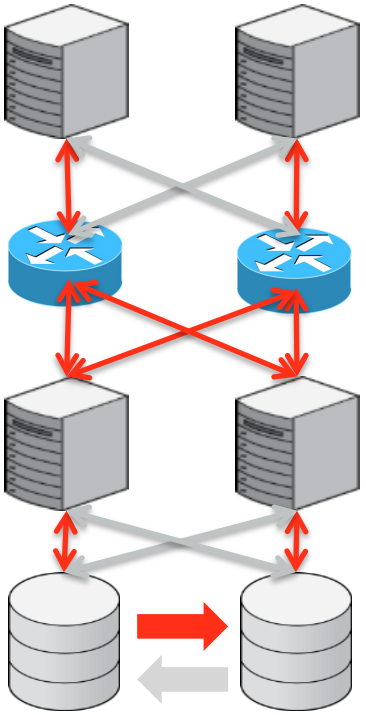
Redundant App Servers

Routing to the Data

Redundant Access to Data

Data Redundancy

Layers of HA



Redundant App Servers

Routing to the Data

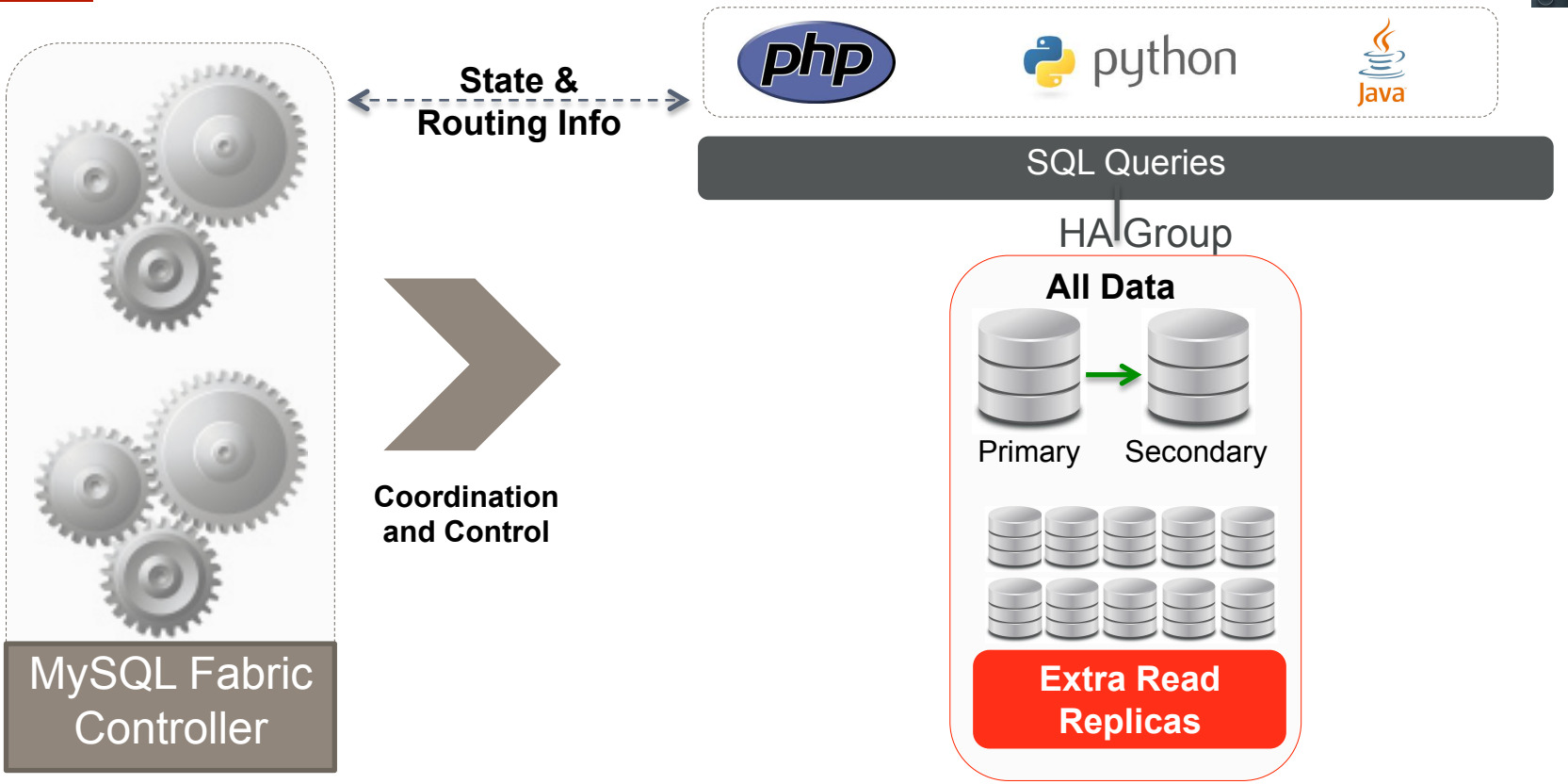
Redundant Access to Data

Data Redundancy

Asynchronous vs. Synchronous Replication

- Asynchronous (MySQL Default)
  - **In parallel:** Master **acks to app** and **sends** transaction to slave
    - Fast
    - Risk of lost changes if master dies
- Synchronous (Only available with MySQL Cluster)
  - **Serially:** Master **waits** for change to be applied on all slaves before **ack to app**
    - Higher latency
    - If Active/Active, best suited to small transactions
    - Loss-less
- Semi-Synchronous (MySQL 5.5+ - Enhanced in MySQL 5.7)
  - **Serially:** Master **waits** for change to be **received** by slave then **In parallel** **ack to app** and **apply** changes on slave
    - Intermediate latency
    - Lossless

MySQL Fabric Framework (HA)



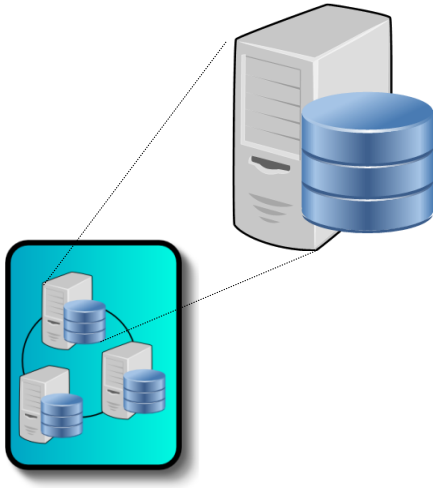
MySQL Replication & MySQL Fabric HA

& 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
mysqlfabric group add my_group server2.example.com
```


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
```


Program Agenda

- Requirements for Next Gen Services
- Simple, transparent High Availability
- Delivering SQL & ACID at scale
- Where MySQL Fabric fits with Oracle's other MySQL solutions
- Getting started

The Path to Scalability

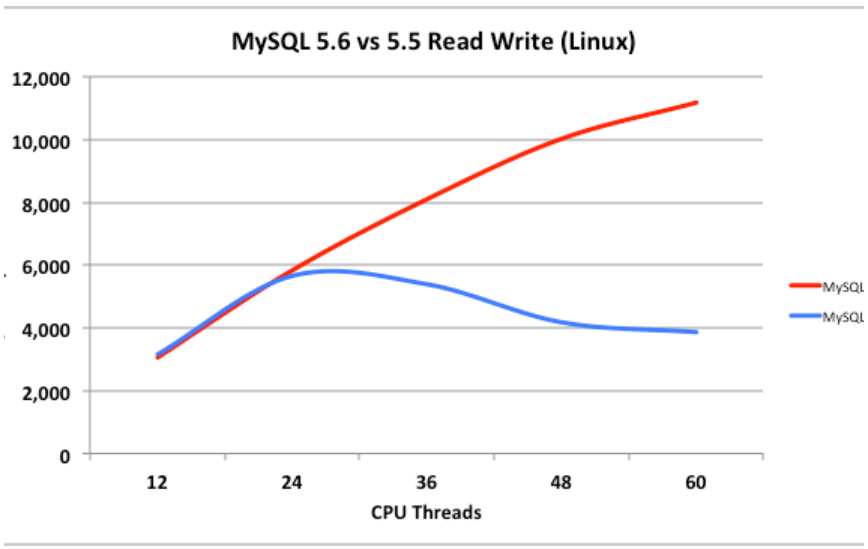
Scaling-Up can take you a long way

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

- 10s - 100GBs RAM
- SSDs

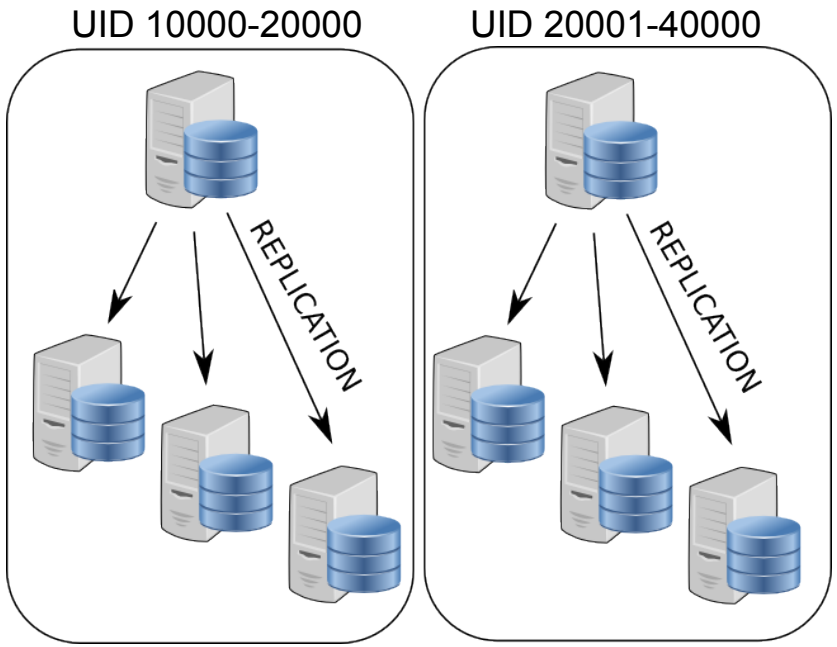
Scale across cores within a single instance

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

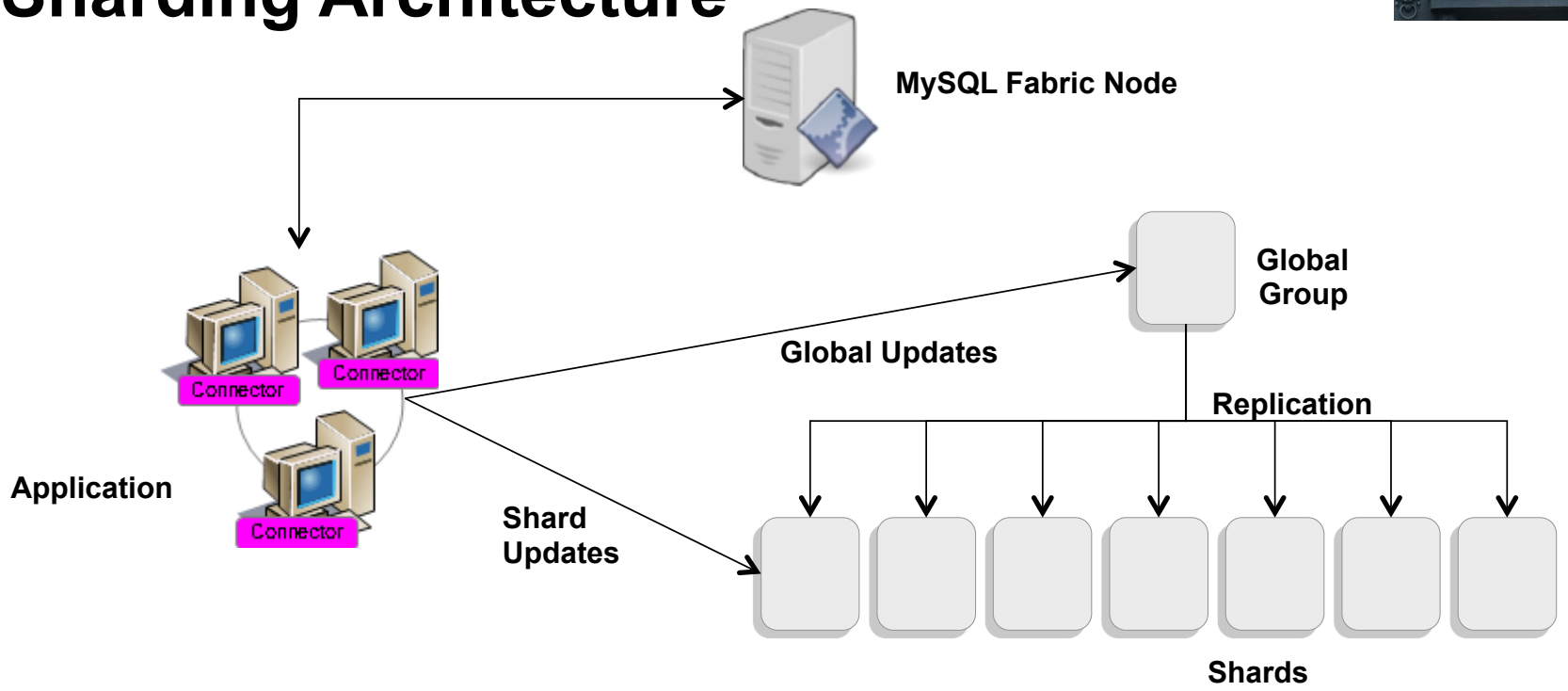


MySQL Fabric Features

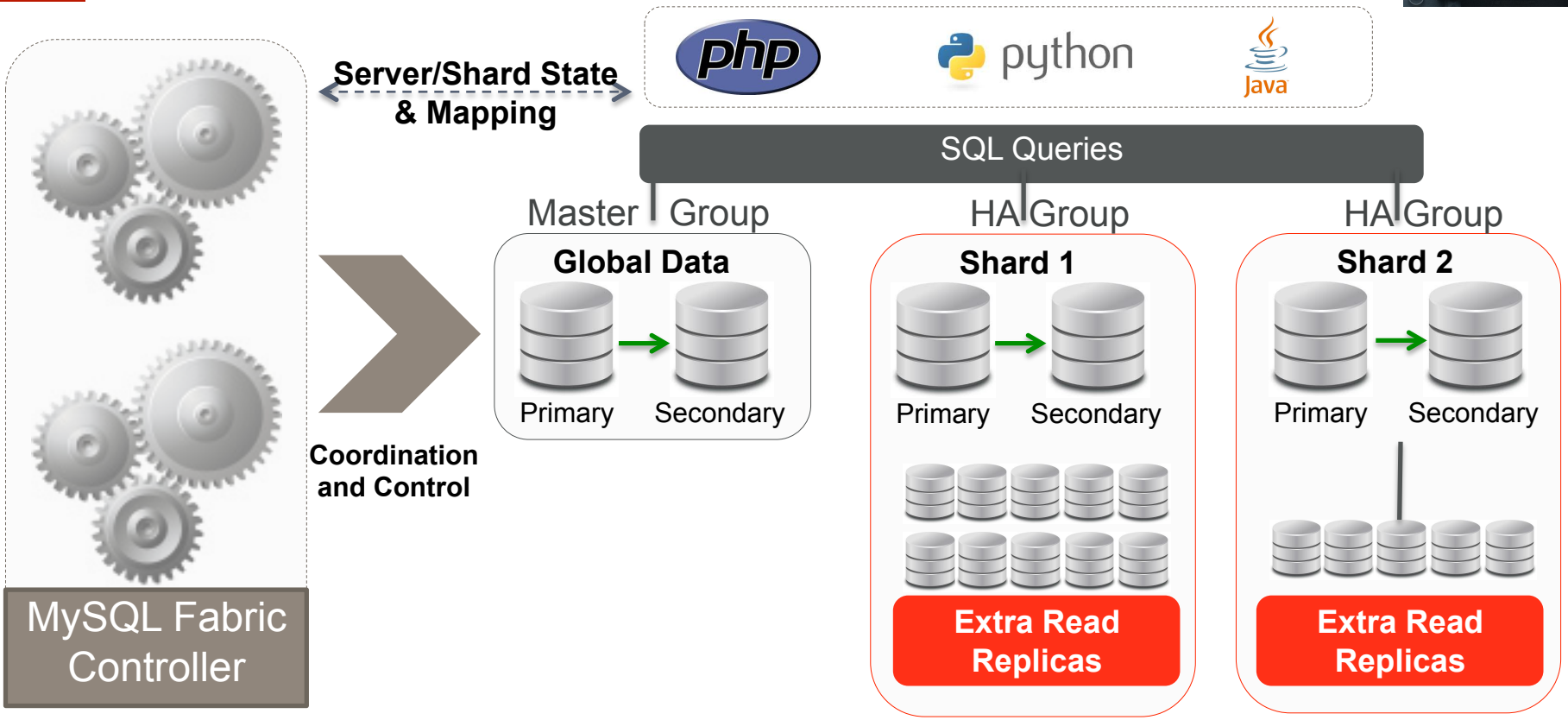
- Connector API Extensions
  - Support Transactions
  - Support full SQL
- Decision logic in connector
  - Reducing network load
- Shard Multiple Tables
  - Using same key
- Global Updates
  - Global tables
  - Schema updates

- Sharding Functions
  - Range
  - (Consistent) Hash
- Shard Operations
  - Shard move
  - Shard split

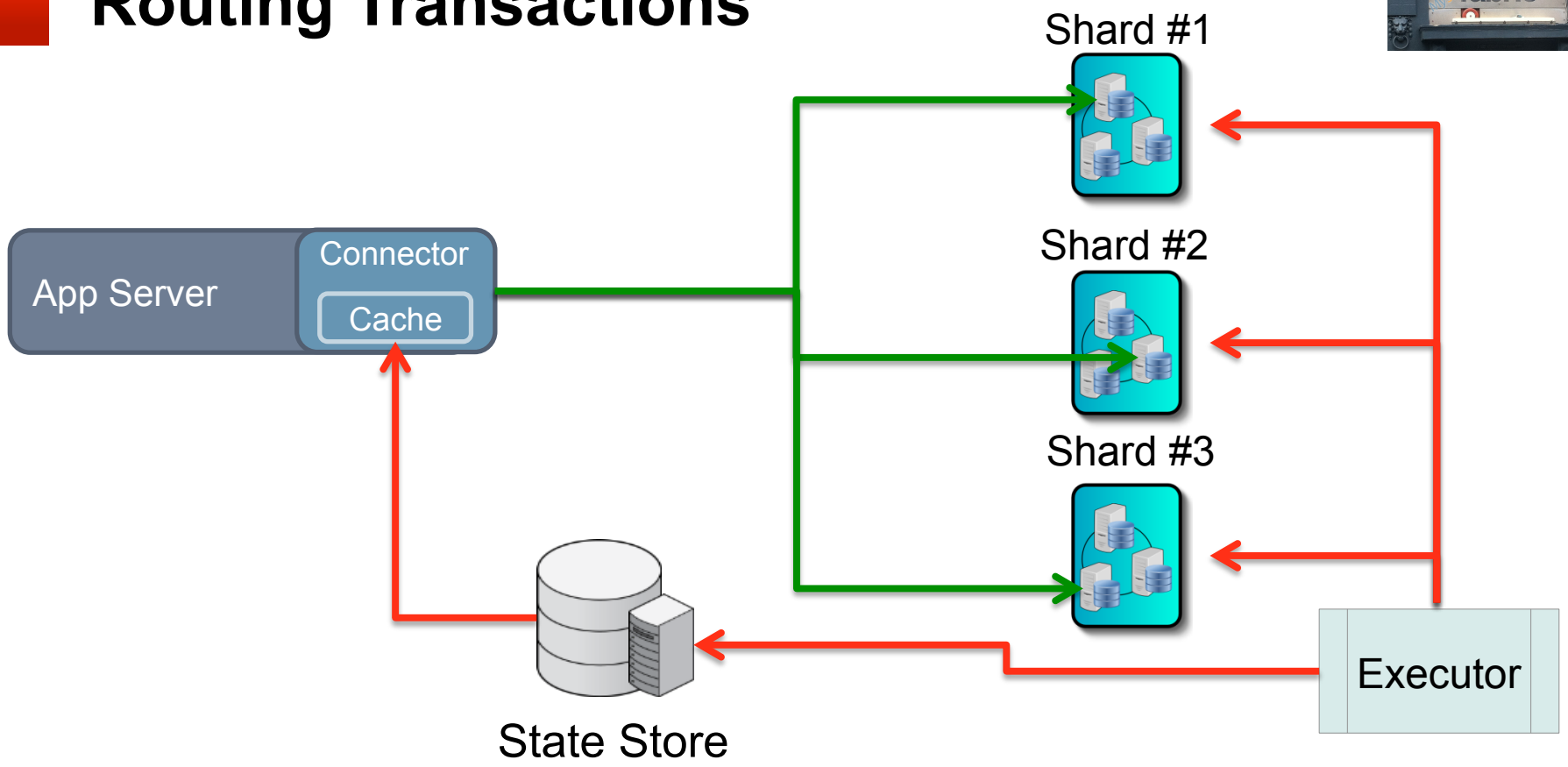
Sharding Architecture



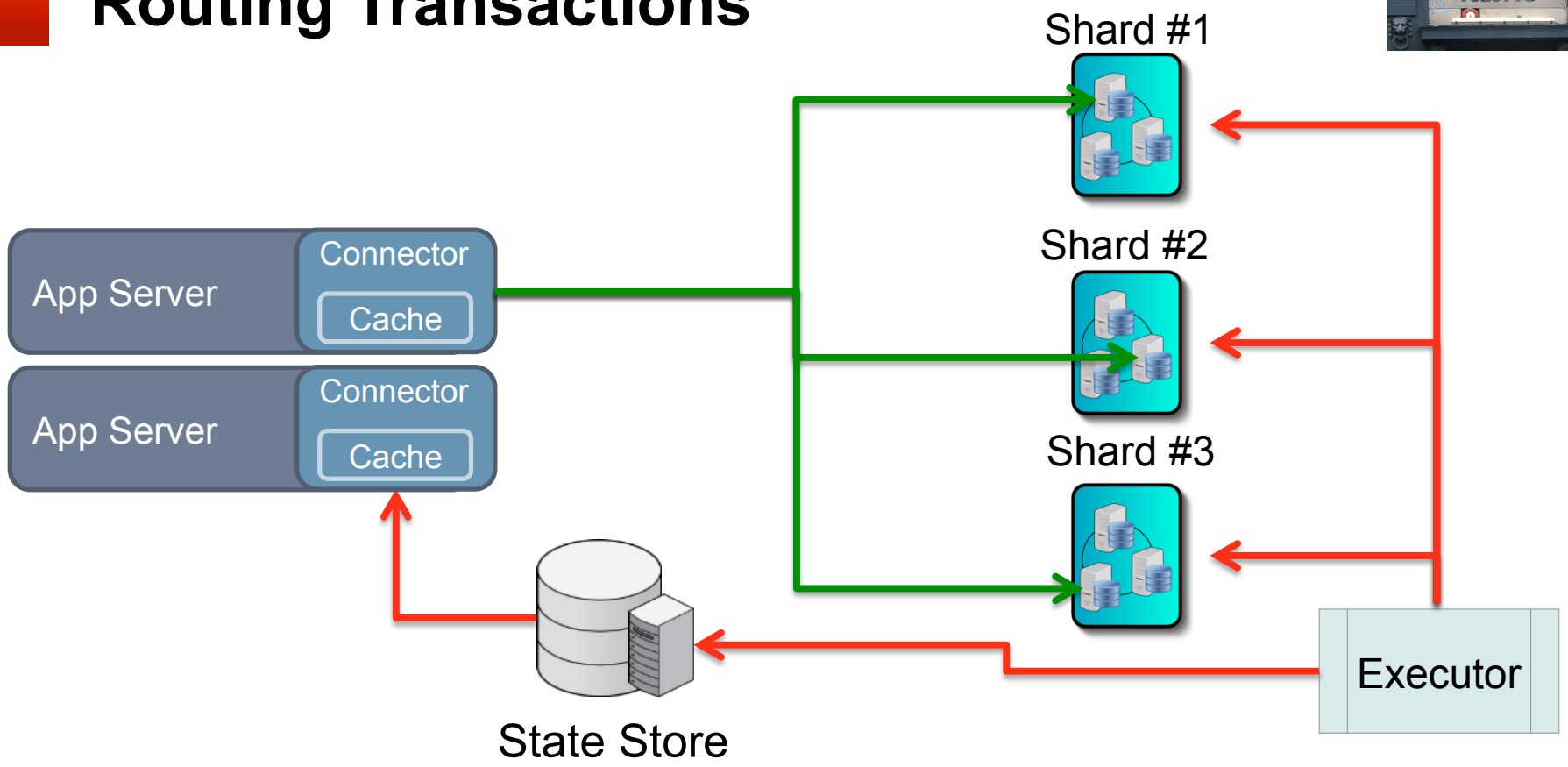
MySQL Fabric (HA + Sharding)



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 (`id=5`) from existing group to another (`my_group.8`)

```
mysqlfabric sharding move 5 my_group.8
```

- Splitting a shard (`id=5`) into two parts with new half stored in group `my_group.6`

```
mysqlfabric sharding split 5 my_group.6
```


Connector API: Shard Specific Query

- Indicate tables to be used in query
  - **Property:** tables
  - Fabric will compute map
- Provide sharding key
  - **Property:** key
  - Fabric will compute shard

```
conn.set_property(tables=["test.subscribers"], key=sub_no, \
        mode=fabric.MODE_READWRITE)
cur = conn.cursor()
cur.execute(
    "INSERT INTO subscribers VALUES (%s, %s, %s)",
    (sub_no, first_name, last_name))
```


Connector API: Shard Specific Query

- Provide tables in query
  - **Property:** tables
  - Fabric will compute map
- Provide sharding key
  - **Property:** key
  - Fabric will compute shard
- Joins within the shard (or with global tables) supported

```
conn.set_property(tables=["test.subscribers"], key=sub_no, \
        mode=fabric.MODE_READONLY)
cur = conn.cursor()
cur.execute(
    "SELECT first_name, last_name FROM subscribers "
    "WHERE sub_no = %s", (sub_no, )
    )
for row in cur:
    print row
```

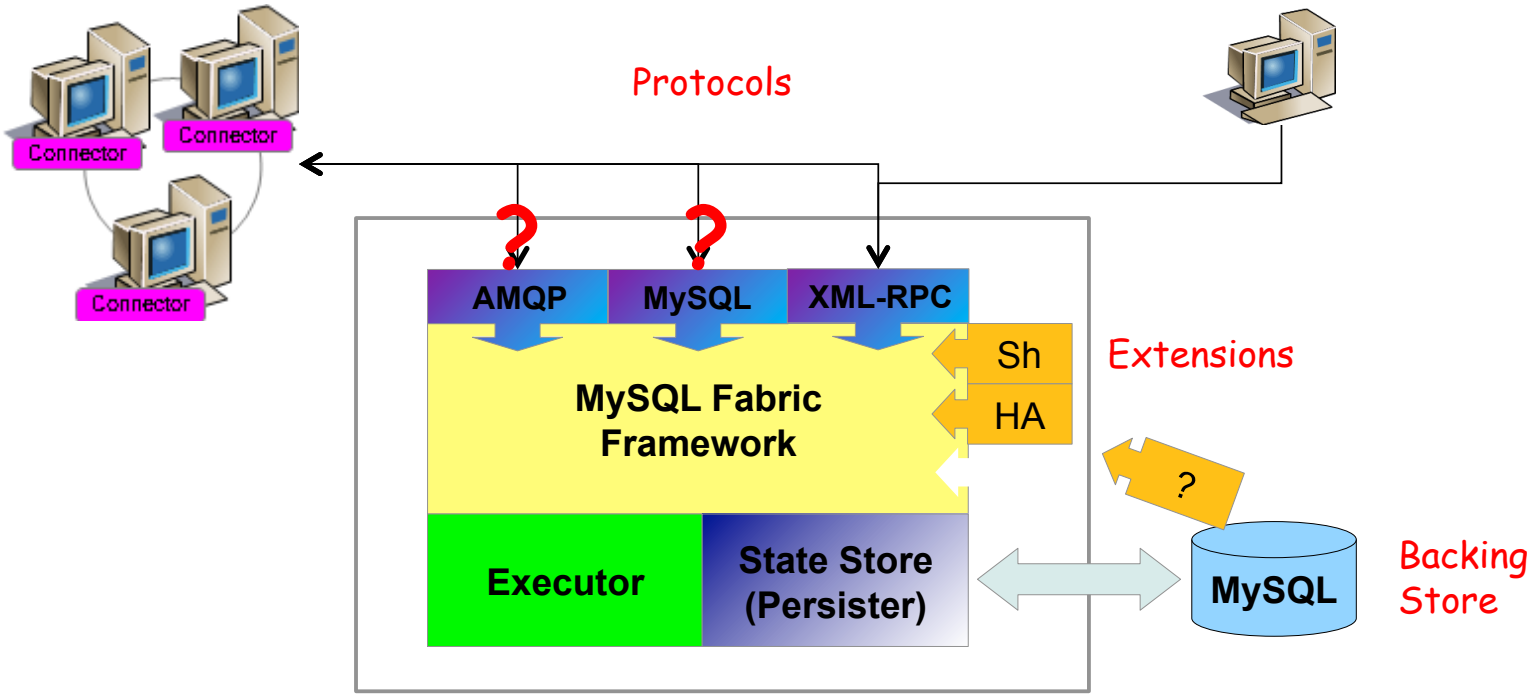

Connector API: Global Update

- 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=['test.subscribers'], scope='GLOBAL')
cur = conn.cursor()
cur.execute("ALTER TABLE test.subscirbers 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

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 completely transparent to application (must provide shard key – column from application schema)
- No cross-shard joins or other queries
- Management is through CLI or XML/RPC API
  - No GUI

Program Agenda

- Requirements for Next Gen Services
- Simple, transparent High Availability
- Delivering SQL & ACID at scale
- Where MySQL Fabric fits with Oracle's other MySQL solutions
- Getting started

Oracle MySQL HA & Scaling Solutions

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

Program Agenda

- Requirements for Next Gen Services
- Simple, transparent High Availability
- Delivering SQL & ACID at scale
- Where MySQL Fabric fits with Oracle's other MySQL solutions
- Getting started

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

