High Availability Using MySQL in the Cloud: Today, Tomorrow and Keys to Success

Jason Stamper, Analyst, 451 Research
Michael Coburn, Senior Architect, Percona
June 10, 2015
Scaling MySQL: no longer a nice-to-have?

Jason Stamper, 451 Research
About 451 Research

- Founded in 2000
- 210+ employees, including over 100 analysts
- 1,000+ clients: Technology & Service providers, corporate advisory, finance, professional services, and IT decision makers
- 10,000+ senior IT professionals in our research community
- Over 52 million data points each quarter
The Challenge

Businesses and their users are facing what one might call a perfect storm – decision-makers need insight faster than ever, and yet IT is struggling to avoid becoming a bottleneck.
The Facts Speak for Themselves...

Recent survey by trade magazine Computer Business Review: 98% (of 200 UK CIOs) admit “significant gap” between what business expects and what IT can deliver.
So What Does the Business Want?

- **Speed**
- **Flexibility**
- **Mobility**
- **New ways of working**
- **Information, not data**
- **Scale**
- **Ease-of-use**
- **Self-service**
- **Collaboration**
What Causes IT to Become a Bottleneck?

- Governance
- Control
- Legacy
- Staff
- Security
- Budget
What Have We Learned So Far?

• So far, the emergence of so-called ‘hot’ data platform and analytics technologies have not solved the IT information bottleneck.

• Hadoop isn’t going to save the world (and neither is NoSQL).

• The ability to analyze large data sets, in real- or near real-time, is only set to grow in the era of the Internet of Things.

• IT is still critical, but it needs to enable the business to help itself. The question is how to achieve the right blend of usability, value-for-money and scalability.
A Word or Two on Hadoop Adoption

Average total storage capacity (TBs), and total storage footprint by workload illustrate the low level of adoption today.
451 Research’s View of the ‘Total Data Approach’
What is Driving the Change?

Developers
Agile
Schema-on-read
Flexible

- JSON
- REST
- Schemaless

New development approaches demand new architecture

New dev approaches enable new lightweight apps

Cloud
Distributed
Elastic
Virtual
Flexible

New app requirements demand new development approaches

Distributed architecture encourages new development approaches

Distributed architecture enables new applications

New applications require distributed architecture

Architecture

Applications
Web
Interactive
Mobile
Always-on
Social
Local
The Database Challenge

– The traditional relational database has been stretched beyond its normal capacity limits by the needs of high-volume, highly distributed or highly complex applications.

– There are workarounds – such as DIY sharding – but manual, homegrown efforts can result in database administrators being stretched beyond their available capacity in terms of managing complexity.

– **Scalability**
– **Performance**
– **Relaxed consistency**
– **Agility**
– **Intricacy**
– **Necessity**

*Increased willingness to look for emerging alternatives*
Scalability, and Other Challenges

• As usage of MySQL and MariaDB has grown, so has the usage of applications that depend on them:
  – Games; Social; Customer Facing; Web; Business apps like Ad Networks;

• This has highlighted a number of challenges
  – Scalability of master-slave architecture
  – Performance and predictability at scale
    – Lower latency; greater throughput; richer apps
    – User expectations rising
  – Manageability of increasing database/app sprawl

• External factors driving greater complexity:
  – Distributed computing architectures
  – Proliferation of cloud and elasticity requirements
  – Geo-distributed application requirements
  – Viral success means growth can come very quickly
Conclusions

• The success of MySQL and MariaDB has led to complications in terms of scalability concerns.

• Distributed computing, proliferation of cloud, and geographically distributed applications are adding to the complexity.

• Manual sharding techniques transfer the strain from the database to the database administrator.

• MySQL – and MySQL administrators – has/have never been under so much strain.

• Database scalability software enables users to move beyond the limitations and complexity of DIY sharding; precisely how data is managed with a distributed database in the cloud or on premise is key.
HA Using MySQL in the Cloud

Michael Coburn, Senior Architect
Michael.Coburn@percona.com
June 10th, 2015
1. What is High Availability
2. MySQL and Asynchronous Replication (Master/Slave)
   • Stage 1: All in One Instance
   • Stage 2: Separation and Redundancy + Distribution of Work
   • Stage 3: Multiple Geographic Zones
3. MySQL with Galera Replication - Percona XtraDB Cluster (PXC)
4. High Availability Failover Technologies
   • Load Balancers
   • VIP Failover
   • MySQL High Availability (MHA)
   • Percona Replication Manager (PRM)
   • MySQL Fabric
5. Keys to Success
High Availability

- Is the Application available to my end users?
- What does "available" mean to me? to the business? to my end users?
- CAP Theorem
  - C - Consistency
    • Your data is consistent across all nodes
  - A - Availability
    • Your system is available to handle requests in case of failure of one or several nodes
  - P - Partitioning tolerance
    • Given inter-node connection failure, each node is still available to handle requests
- Distributed systems requires you to pick two
Stage 1: All in One Instance

- Most applications start out with little budget and little traffic
- Most cost effective model is to source a single instance
- Application and database run on same instance
- Drawbacks: single point of failure (classic case)
Stage 2: Separation and Redundancy + Distribution of Work

- Application and database are competing with each other for limited resources
  - CPU, Disk, RAM
- Additional database instance provides redundancy
- Implementation of Read-Write splitting in the Application
Stage 3: Multiple Geographic Zones

- Single **active** zone which receives all traffic
- Second zone is **passive**
  - Typically used for disaster recovery, or is in **read-only** state
MySQL with Galera Replication - Percona XtraDB Cluster (PXC)

- Synchronous replication
- Multi-master replication support
- Parallel replication
- Automatic node provisioning
- Data consistency
MySQL Virtually Synchronous Replication
Load Balancers

- Designed to evaluate health of Database instances and distribute queries
- Publish small number of IP addresses to Application
- Can scale workload without changing Application by adding more nodes behind Load Balancer
- Popular Options:
  - HAProxy
  - F5 BigIP
VIP Failover

- Concept of moving a Virtual IP (VIP) across servers
- Generally a single Writer VIP, and multiple Reader VIPs
- Normally implies you don't have a Load Balancer
  - Application connects directly to Database instances
- As a Database instance switches roles (Master or Slave), the proper VIP is attached accordingly
- Depends on Network switch updating ARP caches in a timely manner
- Popular Options:
  - MHA
  - Percona Replication Manager
  - Corosync + Pacemaker with IPAddr module
  - MMM (now deprecated)
MySQL Master HA (MHA)

- Designed to automate promotion of slave to Active Master
- Minimises the complexity of re-pointing many Slaves to new Active Master
- MHA provides facility for scheduled online master switch
- Can be run in Monitoring (Automatic) or Manual modes
- Supports GTID-based replication

Manager

MySQL-MasterHA-Manager
- masterha_manager
- other helper commands

Master

MySQL-MasterHA-Node
- save_binary_logs
- apply_diff_relay_logs
- purge_relay_logs

Slave

Manager package: Can manage multiple {master, slaves} pairs
- masterha_manager: Automated master monitoring and failover command
- Other helper scripts: Manual master failover, online master switch, con checking, etc

Node package: Deploying on all MySQL servers
- save_binary_logs: Copying master’s binary logs if accessible
- apply_diff_relay_logs: Generating differential relay logs from the latest slave, and applying all differential binlog events
- purge_relay_logs: Deleting relay logs without stopping SQL thread
Percona Replication Manager (PRM)

- Based on the Corosync and Pacemaker projects with a MySQL module
- PRM behaves as a cluster and nodes vote for role distribution and elect members
- Agent makes changes to local MySQL instance only
- Can synchronise slave binary log positions then provide for new Master promotion and slave re-pointing
- Supports GTID-based replication
MySQL Fabric

- Works with asynchronous replication (Master Slave)
- Servers are organised into high-availability groups
- Fabric can monitor status of all servers in a group
- If current master fails, a new master is elected and promoted
- Requires a Fabric-aware connector, such as Connector/J or Connector/Python
  - Other languages are under development and thus varying levels of support for Fabric
- Latest GA release: MySQL Utilities 1.5.4 (2015-03-04)
Keys to Successful Deployment

1. Awareness of CAP Theorem and how it applies to your Application availability
2. Application awareness of Read-Write Splitting
3. Load Balancers and your Application
4. Usage of Multiple Geographic Zones
5. Automated vs Manual operations:
   a. Addition and removal of Database nodes from Application
   b. Promotion of slave to role of new master + re-pointing of remaining slaves to new master
   c. Failover across multiple geographic zones