Performance comparisons and trade-offs for various MySQL replication schemes

Darpan Dinker  
VP Engineering

Brian O’Krafka,  
Chief Architect

Schooner Information Technology, Inc.  
http://www.schoonerinfotech.com/

Oracle, MySQL, Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.
MySQL Replication

- Replication for databases is critical
  - **High Availability**: avoid SPOF, fail-over for service continuity, DR
  - **Scale-Out**: scale reads on replicas
  - **Misc. administrative tasks**: upgrades, schema changes, backup, PITR ...

- MySQL/InnoDB with base replication has serious gaps
  - Failover is not straight-forward
  - Possibility of data loss and data inconsistency
  - Possibility of stale data read on Slaves
  - Writes on Master need to be de-rated to Slave’s single-thread applier performance (forcing unnecessary sharding)
  - Applications and deployments need to work around these issues or live with the consequences

- Several alternatives exist with different design considerations
Qualitative Comparison

MySQL-Specific Replication

1. MySQL async  
   – Available in MySQL 5.1, 5.5
2. MySQL semi-sync  
   – Available in MySQL 5.5
3. Schooner sync  
   – Available in Schooner Active Cluster

External Replication

4. GoldenGate
5. Tungsten
6. DRBD  
   – Available on Linux
• Loosely coupled master/slave relationship
  • Master does not wait for Slave
  • Slave determines how much to read and from which point in the binary log
  • Slave can be arbitrarily behind master in reading and applying changes

• Read on slave can give old data
• No checksums in binary or relay log stored on disk, data corruption possible
• Upon a Master’s failure
  • Slave may not have latest committed data resulting in data loss
  • Fail-over to a slave is stalled until all transactions in relay log have been committed – not instantaneous
#2 MySQL Semi-synchronous Replication

- Semi-coupled master/slave relationship
  - On commit, Master waits for an ACK from Slave
  - Slave logs the transaction event in relay log and ACKs (may not apply yet)
  - Slave can be arbitrarily behind master in applying changes

- Read on slave can give old data
- No checksums in binary or relay log stored on disk, data corruption possible
- Upon a Master’s failure
  - Fail-over to a slave is stalled until all transactions in relay log have been committed – not instantaneous
#3 Schooner MySQL Active Cluster (SAC): An integrated HA and replication solution for MySQL/InnoDB

- Tightly-coupled master/slave relationship
  - After commit, all Slaves guaranteed to receive and commit the change
  - Slave in lock-step with Master

- Read on slave gives latest committed data
- Checksums in log stored on disk
- Upon a Master’s failure
  - Fail-over to a slave is fully integrated and automatic
  - Application writes continue on new master instantaneously
Loosely coupled master/slave relationship
- Master does not wait for Slave
- Changes applied on slave similar to MySQL
- Slave can be arbitrarily behind master in reading and applying changes

Read on slave can give old data

Heterogeneous Database support
- Oracle, Microsoft SQL Server, IBM DB2, MySQL
#5 Tungsten Replicator

- Loosely coupled master/slave relationship
  - Master does not wait for Slave
  - Changes applied on slave similar to MySQL
  - Slave can be arbitrarily behind master in reading and applying changes

- Read on slave can give old data*
- Heterogeneous Database support
  - MySQL, PostgreSQL
- Global Tx ID, useful to point to new master upon failure
- SaaS & ISP feature: Parallel replication for multi-tenant MySQL databases

Tungsten is a registered trademark of Continuent

http://code.google.com/p/tungsten-replicator/
**#6 Linux DRBD**

- **Active-Passive mirroring at block device**
  - After each commit, the Stand-by server is guaranteed to have identical blocks on device
  - Stand-by in lock-step with Master

- **Stand-by server does not service load**
- **No data-loss**
- **Upon a Master’s failure**
  - MySQL is started on stand-by, database recovery takes ~minutes
  - Stand-by is made new Master
  - Application writes may use VIPs to write to new Master when its ready
Quantitative Comparison: Performance

- Performance comparison of:
  1. MySQL asynchronous (v5.5.8)
  2. MySQL semi-synchronous (v5.5.8)
  3. Schooner Active Cluster (SAC) synchronous replication (v3.1)

- Benchmark: DBT2 open-source transaction processing benchmark
DBT2 Benchmark

  - On-line Transaction Processing (OLTP) performance test
  - Fair-usage implementation of the TPC-C benchmark
  - Simulates a wholesale parts supplier with a database containing inventory and customer information

- Benchmark scale determined by number of warehouses
  - Results here based on a scale of 1000

- Use InnoDB storage engine with 48GB buffer pool and full consistency/durability settings

- Schooner has found that optimizing MySQL/InnoDB for DBT2 yields significant benefit for many real customer workloads
Measurement Setup

SERVERS

- IBM 3650 2RU Westmere Server
- CPU: 2x Intel Xeon X5670 processors, 6 cores/12 threads per processor, 2.93 GHz
- DRAM: 72GB
- HDD: 2x300GB 10k RPM HDD RAID-1 for logging, with LSI M5015 controller with NVRAM writeback cache
- Flash: 8x200GB OCZ MLC SSD’s, with 1 LSI 9211 controller, md RAID-0
- Network: 1x Chelseo 10Gb Ethernet NIC

CLIENT

- DBT2 client runs on master server

SWITCH

- Arista 10Gb Ethernet switch
Results: DBT2 Performance

DBT2 Throughput (kTpm)

- 2-node 5.5 async
- 2-node semi
- 2-node SAC

DBT2 Response Time (ms)

- 2-node 5.5 async
- 2-node semi
- 2-node SAC
Results: DBT2 Performance

DBT2 Throughput (kTpm)

- 5.5 Async and Semi-sync limited by serial slave applier
- SAC optimizations yield 4-5x boost in throughput

DBT2 Response Time (ms)

- SAC optimizations yields lower response times
Results: Master CPU Utilization

Master CPU Utilization (%)

- 2-node 5.5 async
- 2-node 5.5 semi
- 2-node SAC
Results: Master CPU Utilization

Higher throughput means higher CPU utilization, better system balance.
Results: Transient Behavior on Master

5.5 Async
Master Throughput vs. Time

5.5 Semi-sync
Master Throughput vs. Time

SAC
Master Throughput vs. Time

Inconsistent performance with 5.5 Async
Results: Slave Utilization

Slave CPU Utilization (%)

- 2-node 5.5 async
- 2-node 5.5 semi
- 2-node SAC
Results: Slave Utilization

Single applier in 5.5 saturates one core

Even at 5X throughput on SAC Slave, enough headroom to service Read traffic
Results: Transient Behavior on Slave

5.5 Async
Slave CPU Utilization vs. Time

5.5 Semi-sync
Slave CPU Utilization vs. Time

SAC
Slave CPU Utilization vs. Time

Fluctuations with 5.5 Async
Results: Storage IO Utilization

Master Storage IOPS Utilization (%)

- 2-node 5.5 async
- 2-node 5.5 semi
- 2-node SAC
Results: Storage IO Utilization

Master Storage IOPS Utilization (%)

Higher throughput means higher storage bandwidth, better system balance
Results: Network Utilization

Replication Network Utilization (%)

<table>
<thead>
<tr>
<th>2-node async</th>
<th>2-node semi</th>
<th>2-node SAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>40</td>
</tr>
</tbody>
</table>

Replication Network Bandwidth (MB/s)

<table>
<thead>
<tr>
<th>2-node async</th>
<th>2-node semi</th>
<th>2-node SAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>10</td>
<td>50</td>
</tr>
</tbody>
</table>
Results: Network Utilization

Higher throughput means higher replication bandwidth, better system balance.
Results: Summary

• Sustainable replication performance of 5.5.8 Async and Semi-sync is severely limited

• Deeply integrated synchronous replication in SAC yield 4-5X boost in sustainable replication performance

• High performance and fully synchronous replication are not mutually exclusive
Conclusion

• Asynchronous replication is de-facto standard for 10+ years, issues with consistency, data-loss and service continuity

• Recently released semi-synchronous replication mitigates the need to DRBD to avoid data-loss on failover, however consistency and service continuity issues exist

• Schooner synchronous provides consistent reads on Slaves instantaneous failover with zero data-loss

• For database interoperability, GoldenGate and Tungsten are a good choice

• Performance comparisons between MySQL 5.5.8 async/semi-sync and Schooner Active Cluster using DBT2 show:
  • 5.5.8 async/semi-sync is severely limited by serial slave applier
  • Parallel slave applicers plus Schooner core optimizations in SAC yield 4-5x increase in throughput at lower response times, while maintaining read consistency across the cluster.
Backup Slides
## Reference: MySQL Replication Solutions Compared

<table>
<thead>
<tr>
<th></th>
<th>Schooner Active Cluster 3.1</th>
<th>MySQL 5.5 (semi-sync)</th>
<th>MySQL 5.1/5.5 (async)</th>
<th>Tungsten</th>
<th>Linux Heartbeat + DRBD</th>
<th>GoldenGate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eliminates Slave Lag</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Slave Consistent w/ Master</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Write scalability</td>
<td>High</td>
<td>Moderate</td>
<td>Moderate</td>
<td>Low</td>
<td>Low</td>
<td>Moderate</td>
</tr>
<tr>
<td>Read scalability</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>N/A</td>
<td>High</td>
</tr>
<tr>
<td>Data loss Probability on Failover</td>
<td>Zero</td>
<td>Zero</td>
<td>High</td>
<td>High</td>
<td>Zero</td>
<td>High</td>
</tr>
<tr>
<td>Planned Failover time (stop mysql)</td>
<td>&lt;2sec</td>
<td>Slave-lag-catch-up</td>
<td>Slave-lag-catch-up</td>
<td>Slave-lag-catch-up</td>
<td>InnoDB recovery time</td>
<td>Slave-lag-catch-up</td>
</tr>
<tr>
<td>Automated Unplanned Failover time</td>
<td>&lt;8sec</td>
<td>&quot;~8sec + slave-lag-catch-up&quot;</td>
<td>&quot;~8sec + slave-lag-catch-up&quot;</td>
<td>&quot;~8sec + slave-lag-catch-up&quot;</td>
<td>&quot;~8sec + InnoDB recovery time&quot;</td>
<td>&quot;~8sec + slave-lag-catch-up&quot;</td>
</tr>
<tr>
<td>Manual Unplanned Failover time</td>
<td>N/A</td>
<td>Minutes-Hours</td>
<td>Minutes-Hours</td>
<td>Minutes-Hours</td>
<td>Minutes-Hours</td>
<td>Minutes-Hours</td>
</tr>
<tr>
<td>Checksum in replication logs</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>?</td>
</tr>
<tr>
<td>Replication Solution</td>
<td>Built-in</td>
<td>Built-in</td>
<td>Built-in</td>
<td>External</td>
<td>External</td>
<td>External</td>
</tr>
<tr>
<td>Replicate to Non-MySQL Databases</td>
<td>N^</td>
<td>N^</td>
<td>N^</td>
<td>Y</td>
<td>N^</td>
<td>Y</td>
</tr>
<tr>
<td>Point-in Time Recovery (PITR)</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Automated DB Instantiation</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Incremental data movement</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Rolling upgrade</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Automated Rolling migration</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>?</td>
<td>N</td>
<td>?</td>
</tr>
</tbody>
</table>

- Using Multi-master for MySQL (MMM), Flipper, etc. assuming 8sec for heartbeat retries
- Manual response to an alert and running trouble-shooting and replication commands/scripts
- Possible to leverage complementing solution like GoldenGate