Not Your Grandpa’s Replication

The New Wave of MySQL Replication and How It Helps Your Applications

Robert Hodges - Continuent, Inc.
Jay Pipes - Rackspace, Inc.
Agenda

/ About Us
/ Replication Problems New and Old
/ Old and New Replication Contenders
/ Questions
About Us

/ Jay Pipes -- Drizzle code monkey and man of Rackspace
  • Drizzle replication designer and chief implementer

/ Robert Hodges -- Tungsten chief propeller-head (and CTO of Continuent)
  • Tungsten Replicator for MySQL & PostgreSQL, backups, distributed management, etc.

/ Continuent: Cross-platform database clustering and replication

/ Rackspace: Hosting, Fanatical Support, etc.

MySQL Conference 2010
In Days of Old Life Was Simple
MySQL Replication Addressed Problems

/ Switch to new database after crash
/ Scale website performance on read-only copies
/ Perform schema upgrades and system maintenance with minimal downtime
/ Keep appliance and embedded DBMS available
/ Allow updates across sites

/ But times have changed!
Replication Meets Industrial Data Farms
New Replication Challenges Emerge

/ Big data -- Too big to back up or move
  * Intrusion detection systems generate burst updates of 100K/sec

/ Multi-tenant applications
  * SaaS / ISP want to backup/restore/migrate/manage tenants

/ New hardware - Multi-core, large memory, SSD
  * Sites like craigslist.org want multiple cores to reduce slave latency

/ Complex topologies
  * Market automation apps shard data across dozens of servers with complex data flows

/ Scalable operation across sites
  * Merchant systems and on-line testing update multiple locations

MySQL replication does not handle any/all of these problems especially well
And Some Problems Never Go Away

Education is required. People don't want to hear this. But from my experience a lot of problems are caused by SQL app developers.

Mark Callaghan
Replicate Statements or Rows?

// SQL updates can be represented in two different ways
// Statements -- What the client said
// Row updates -- What the client actually did

**Statement Replication**
Replicate changes as SQL statements

**Row Replication**
Replicate changes other than DDL as row updates

DDL, only way some DBMS can log changes/replicate

Flexible, fewer weird exceptions
Physical vs. Logical Replication

- Databases can update either at disk or logical level, hence two replication approaches.

- Log records -- Databases apply them automatically during recovery.

- SQL statements -- Clients send SQL to make changes.

<table>
<thead>
<tr>
<th>Physical Replication</th>
<th>Logical Replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replicate log records/events to create bit-for-bit copy</td>
<td>Replicate SQL to create equivalent data</td>
</tr>
</tbody>
</table>

- Transparent, high performance, hard to cross architectures and versions, may limit slaves.

- Flexible, fewer/different restrictions, allow schema differences, can manage upgrade.
Asynchronous vs. Synchronous

- Replicating is like buying a car--there are lots of ways to pay for it
  - $0 down - Pay later; hope nothing goes wrong
  - Down payment - Pay some so less goes wrong later
  - Cash - Pay up front and it’s yours forever

<table>
<thead>
<tr>
<th>Asynchronous Replication</th>
<th>Semi-Synchronous Replication</th>
<th>Synchronous Replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commit now, replicate later</td>
<td>Replicate to at least one other node</td>
<td>Replicate fully to all other node</td>
</tr>
</tbody>
</table>

- Lose data but robust against network failure
- Trade-off data loss vs. partition handling
- Network fails --> you fail

MySQL Conference 2010

[MySQL Conference 2010 Logo]
### Multi-Master or Master/Slave or…?

OK, now it gets confusing! Should I…

- Update one database and let it serialize all changes?
- Update any database with global update ordering?
- Update any database and replicate without global ordering?

<table>
<thead>
<tr>
<th>Master/Slave</th>
<th>Multi-Master</th>
<th>Master-Master</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single master serializes and replicates</td>
<td>Multiple masters with global serialization</td>
<td>Multiple masters with <strong>no</strong> global serialization</td>
</tr>
</tbody>
</table>

Fast serialization, SPOF, no split brain | Good scaling but **really** hard to implement | Convenient for WAN but hard for applications |

MySQL Conference 2010
Current Contenders
MySQL Native Replication: The Default

/ High-performance, built-in replication used by just about everyone

/ Key Characteristics
  • Logical - Replicates statements and/or rows
  • Asynchronous - Applications do not wait
  • Log-based - Based on MySQL binlog with a variety of options/tricks

/ Fastest and most mature replication for MySQL
MySQL Replication Architecture

Master
- Dump thread
- binlogs

Slave
- I/O thread
- SQL thread
- relay logs

:3306
MySQL Master Master Replication

- Handles maintenance very well (painless resync, application upgrades, cross architecture/version)
- Tools like Flipper, MMM, and Heartbeat support it very well
MySQL Replication Features

- It replicates *everything*
- Very mature and fast enough for most uses
- Row-based replication added in 5.1
  - Removes corner cases
- Features for many use cases:
  - Relay logs
  - replicate-ignore-db/replicate-do-db/etc.
  - Black hole replication
  - Bi-directional replication
- Lots of tool support: Maatkit, MMM, Heartbeat, mysqlbinlog
Development Still Advancing

/ MySQL 5.5
  • Semi-synchronous replication
  • Slave fsync tuning
  • Automatic relay log recovery
  • Replication heartbeats
  • SHOW RELAY LOGS command

/ Plus regular bug fixes (397 since 2009 UC according to Lars)

/ Plus MariaDB is getting into the act!
  • (We’ll have more news in the next talk)
MySQL Replication: What’s Not to Like?

/ Data protection still weak
  • No checksums on data
  • 2PC issues between log and stores
  • No global transaction IDs

/ Difficult to manage as topologies scale

/ Broken slaves a common problem

/ Fully pluggable interfaces still a long way off
Tungsten: Complete Master/Slave Clusters

/ Build complete data services using copies of MySQL databases
/ Think of Tungsten as a data service appliance
/ Key Characteristics
  • Logical - Replicates statements and/or rows
  • Asynchronous - Applications do not wait
  • Log-based - Reads MySQL binlogs directly or via client protocol to master
/ Features for SaaS, ISP and large enterprises
Tungsten Replication Pipelines

Tungsten Replicator Process

Pipeline

Stage

Extractor → Filters → Applier

Stage

Extractor → Filters → Applier

Transaction History Log

Slave DBMS

binlogs

MySQL Conference 2010
Tungsten Features

/ Unaltered MySQL 5.0/5.1 databases
/ Very flexible pipelines and extensions
/ Global transaction IDs, crash-safe slaves, heartbeats, consistency checks, checksums
/ Autonomic failover and management
/ Seamless failover/app scaling
/ Rapid new feature additions
New SaaS-Oriented Features

/ Tungsten 2.0 adds for SaaS/ISP usage
  • Parallel replication based on shards
  • Fast event logging
  • Low-latency WAN replication
  • Multi-master replication

/ PostgreSQL 8 warm standby support and adding features to manage PostgreSQL 9

/ Drizzle support as soon as we get customers
Tungsten Trade-Offs

/ **Strengths**
  - Complete solution for managing data, not just replication
  - Features for SaaS / multi-tenant apps
  - External replication is very flexible
  - Improving very rapidly

/ **Why Not Use It?**
  - Pretty new (< 2 years)
  - You may want to use other tools like Heartbeat
  - Built-in replication is simpler (well sometimes)
  - Needs more features for big data
Drizzle: MySQL is Not in Kansas Anymore

- Completely redesigned, pluggable replication for Drizzle DBMS

Key Characteristics

- **Physical** - Replicates change set messages including DDL
- **Asynchronous or Synchronous** - Depends on implementation
- **Row-based** - Packages row changes into Transaction GPB messages and sends replication streams

- Drizzle team supplies reference implementation of replication
Rethinking Replication Design

Drizzle's replication system looks nothing like MySQL
Drizzle is entirely row-based (yes even DDL)
Forget the terms master, slave, and binlog
We use the terms publisher, subscriber, replicator and applier

We have a transaction log, but it is not required for replication
Drizzle's transaction log is a module
The transaction log module has example implementations of an applier
Clients

Listener Plugin (Protocol)

Scheduler Plugin

Authentication Plugin

Query Cache Plugin

Logging Plugin (Pre)

Authorization Plugin

Parser

Optimizer

Executor

Plugin Registration

Replication Services

Transaction Services

Metadata Services

Replication Services

Scheduler Plugin

Encryption

Diagnostic

User-Defined Function Plugins

Dictionary Plugin

Registration

Metadata Services

Pluggable Storage Engine API

MyISAM

InnoDB

MEMORY

Archive

PBXT

MySQL Conference 2010
Client issues DML that modifies data

TransactionServices
- calls commitTransaction()

TransactionServices
- constructs Transaction message object

ReplicationServices
- pushes Transaction message out to all replication streams

plugin::StorageEngine
- makes changes to data store

plugin::TransactionReplicator
- calls replicate()

plugin::TransactionApplier
- calls apply()
What is a *Replication Stream*?

A replication stream is the pair of a replicator and an applier.

Each applier must be matched with a replicator:
- Can be done via command-line arguments
- Can be hard-coded

To see the replication streams that are active, you can query `DATA_DICTIONARY.REPLICATION_STREAMS`:

```sql
drizzle> select * from data_dictionary.replication_streams;
+-------------------+-------------------------+
| REPLICATOR        | APPLIER                 |
|--------------------+-------------------------|
| default_replicator | transaction_log_applier |
+-------------------+-------------------------+
1 row in set (0 sec)
```
Drizzle Replication Add-ons: Rabbit MQ

Developed by Marcus Eriksson
http://developian.com

Can replicate externally or internally
External by reading the Drizzle transaction log and sending logs to RabbitMQ
  Multi-threaded applier constructs SQL statements from transaction messages in log files on replica
Internal via a C++ plugin
  /plugin/rabbitmq/
  Implements plugin::TransactionApplier
  Sends transaction message to RabbitMQ
Galera: Synchronous Multi-Master Is Back!

/ In-core multi-master replication using certification to ensure global serialization

/ Key Characteristics
  • Logical replication - Replicates change sets
  • Synchronous - All databases in sync at all times
  • Multi-master - Connect to any node to update

/ So...
  • Good update performance
  • Write scaling!
  • Excellent read scaling (linear)
  • No master SPOF
Galera Architecture - Virtual Synchrony

```
INSERT INTO FOO VALUES(1, 24, 'hello');
```
Galera Benefits and Drawbacks

- Completely eliminating latency between copies lets you scale horizontally with very effective HA
- “Virtual synchrony” approach scales writes!
- Subject to aborts on hot spots
  - “Birthday problem” data sets or queue-like structures
- DDL changes typically lock the cluster
- WAN operation is “interesting” (Find out more @3pm)
- Very sensitive to group communication performance
- Nobody has ever gotten the certification approach to work! This is vaporware!! Are you kidding me??
Powered By Multi-Master (But Not Galera)
PBXT: Skip Binlog Bureaucracy!

/ Replicate directly between PBXT storage plugins

/ Key Characteristics
  • Physical - Very efficient row replication
  • Asynchronous - Applications do not wait
  • Log-based - Replicate on PBXT journal

/ First step in attacking big data--very fast replication for moving quantities of data
PBXT Replication Architecture

MySQL Master
- Index
- Transaction Log
- Data Log
- Handle Data
- Writer Thread
- Replication Thread

MySQL Slave
- Slave Thread
- Replication Stream
- ADD_ROW

MySQL Conference 2010
DBT2 Performance Results

Replication Overhead

- No Replication
- No Sync Binlog: 44%
- Sync Binlog: 10X
- Engine Replication: 9%

Transaction per second
PBXT Round up

// Very simple set-up for now (server restart to change slave configuration)
// Journals are purged after they reach all slaves
// Initial performance testing looks very promising
// This is alpha code but…
// Why didn’t anyone do this before??
PostgreSQL: What Are the Neighbors Doing?

/ PostgreSQL 8 provides “warm standby” log shipping
  • No slave queries
  • Unavoidable data loss gap

/ PostgreSQL 9 introduces log streaming and “hot” standby slave reads

/ Key characteristics of PG 9 replication
  • Physical - Bit-for-bit copy of entire database
  • Asynchronous - Applications do not wait (for now)
  • Log-based - Based on PG write-ahead log
PostgreSQL 8.4 Warm Standby

Master PostgreSQL

- WAL Files
- pg_xlogs Directory

Standby PostgreSQL

- Archived WAL Files
- Archive Directory
- pg_xlogs Directory
- Continuous recovery

rsync to standby

pg_standby
Hot Standby and Log Streaming

MySQL Conference 2010
PostgreSQL Roundup

/ Log streaming/hot standby layers on top of existing features that are very robust

/ MVCC is a problem with full physical replication
  • New updates can clear records used by slave queries

/ Configuration files simple but overall set up and replication management are tricky

/ Still no way to read PostgreSQL logs directly

/ But expect log streaming to kill a lot of interest in trigger-based replication using SLONY, Londiste, or Bucardo

/ PG 9 is due out this summer
Summary
Final Words

MySQL Replication is continuing to evolve and set the standard

New replication mechanisms are evolving quickly to attack new problems

Educate yourself and build something new!
Our Thanks To…

/ Mark Callaghan
/ Seppo Jaakola
/ Mats Kindahl
/ Paul McCullagh
/ Alexey Yurchenko
/ Jeremy Zawodny
Information/Contact

Robert Hodges
http://www.continuent.com
robert.hodges at continuent.com

Jay Pipes
http://www.rackspace.com
jaypipes at gmail.com
Extra Slides
Google Semi-Synchronous Replication

- Quorum algorithm -- Commits block until at least one slave responds affirmatively
- Protects data but avoids system freeze if a slave is unavailable
- Released as patch to MySQL; not widely available yet