



PERCONA
Performance Consulting Experts

Fighting MySQL Replication Lag

April 22, 2009

Percona Performance
Conference

Santa Clara, CA

by Peter Zaitsev, Percona Inc

What is MySQL Replication Lag?

- Slave Can't keep up with master
 - “Falling Behind”
- “Seconds_Behind_Master” gets large
- Slave has stale data
 - Application functionality may be impaired
 - Slave unavailable for immediate failback

Two Types of Lag

- **Permanent Lag**
 - Slave just falls behind and unable to keep up
 - Often it falls behind during day time and catches up during the night just before that
- **Periodic Lag**
 - Slave falls behind at certain intervals and catches up quickly
 - Often related to batch jobs/reporting etc
 - Single bad query can “clog” replication
- These types may have different solutions

Replication Limits

- Replication in MySQL is single threaded
 - Though there is work on a way to change that
- Meaning it is able to use single CPU core and single disk efficiently
- Single complex query (transaction) can clog the process
- Single lock (table or row level) pauses the whole process.

Replication Capacity

- Learn “replication capacity”
 - How much does it take from idle slave to just keep up with master traffic
- Pause replication for 1 hour and see how long it takes to catch up.
 - Make sure testing it on warm caching
- Or use Percona patches to see replication thread utilization
- I would recommend having at least 3x capacity (1 hour takes no more than 20 minutes to catch up)

Are we loading slave too much ?

- Low replication capacity
 - Means slave can handle very little extra load
- Look at reducing load if replication capacity more than 3x
- Replication optimization is a good idea anyway
- Because replication is single thread it may hard time to compete with many concurrent queries
 - You may need to restrict concurrency on the slave

Understand your replication traffic

- You do not have to guess any more
 - <http://www.percona.com/percona-lab.html>
 - **long_query_time=0**
 - **log_slow_slave_statements=1**
 - Log statements from slave thread
- Analyze the log with **mk-query-digest**
- Do the run both with idle slave and normally loaded
 - Helps to understand which queries are affected by side load
- Look at both queries causing largest impact and long queries

Optimizing Replication

- Focus on Statement Based Replication
 - It is by far more used at this point
- Goal: Reduce the side load slave has to do
 - Side load – everything but doing updates which should be done
- **INSERT ... SELECT <complex query>**
 - Complex select will be ran on each slave.
 - Copy data via application instead
- **UPDATE WHERE <complex clause>**
 - Do select + some updates by primary

Beware large transactions

- Binary log records complete transactions
 - If you have transaction containing 100000 update over few hours it will be written in binary block as single transaction
 - And no transactions will be allowed to interleave
- Transactions taking long to replicate cause replication lag spikes
- You need to “chop” transactions, not just queries.
- Instead of single **DELETE** use many **DELETE ... LIMIT 100**

Which transactions to chop ?

- How much of replication lag is tolerable ?
 - Make sure think both of application and failover
- Half this time
 - This is your max allowed time
- No transactions taking longer than 5 seconds if you want lag to be no more than 10 seconds.
- If using MyISAM on the Slave need to ensure no long read queries too

Going around application

- Run complex jobs on slaves directly
 - ALTER TABLE
 - Purging old data
- “Task System” can be handy for that
- Insert “task query” in the special table on Master
 - On the slave scripts reads table; executes it and removes.
 - Can do nice things like archiving data on some slaves while purging on others.
 - Exercise case

Replication on Steroids

- If nothing else helps consider prefetching
- **Mk-slave-prefetch** part of maatkit
- Helps most if you have extra IO capacity
 - Multiple disks are not well used by single replication thread

Hardware choices for Slave

- Fast CPU Cores
 - Single thread workload does not scale with many cores
- Good disk cache hit ratio
 - “Miss” stalls whole replication thread and nothing can be done during this stall
- Fast disks are better than more disks
 - Single thread = typically single outstanding read = no more than 1-2 disks are well used
- SSDs can be good help with slave performance



The End