Keeping the slave's buffer pool warm for failover with Percona Playback

Peter Boros
Consultant @ Percona

FOSDEM 2013

First of all, thanks to...

- Kyle Oppenheim (Groupon)

  Director of Engineering

  engineering.groupon.com
- Fernando Ipar (Percona)

  Senior consultant

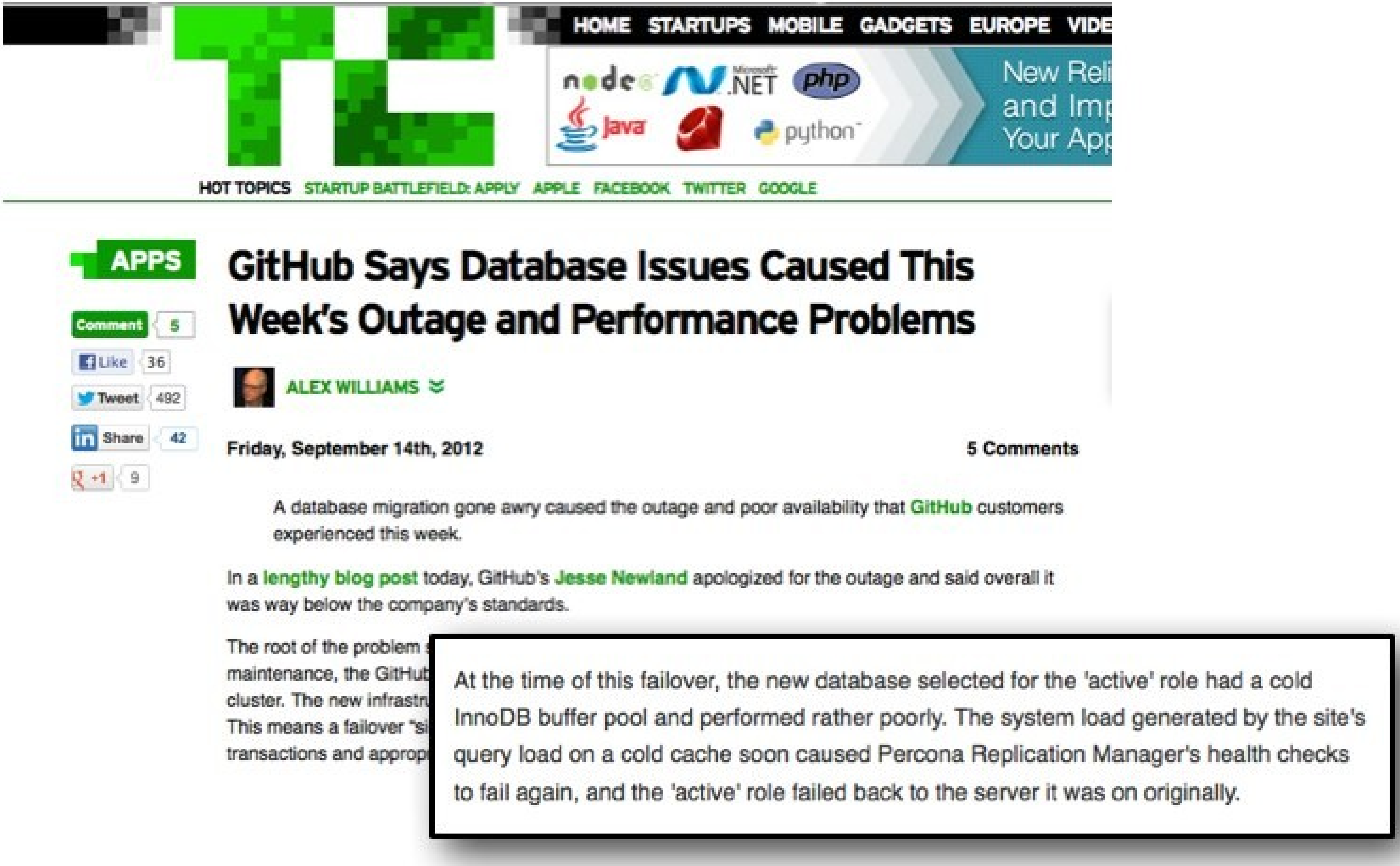
  mysqlperformanceblog.com
- Vladislav Lesin (Percona)
  - Software engineer

The issue

- After a failover, the standby host can have cold caches, which results in excessive use of IO

  http://techcrunch.com/2012/09/14/github-explains-this-weeks-outage-and-poor-performance/

  https://github.com/blog/1261-github-availability-this-week


HOME STARTUPS MOBILE GADGETS EUROPE VIDE
New Reli
and Imp
Your App
HOT TOPICS STARTUP BATTLEFIELD: APPLY APPLE FACEBOOK TWITTER GOOGLE
APPS
Comment 5
Like 36
Tweet 492
Share 42
+1 9

GitHub Says Database Issues Caused This Week's Outage and Performance Problems

ALEX WILLIAMS

Friday, September 14th, 2012

5 Comments

A database migration gone awry caused the outage and poor availability that GitHub customers experienced this week.

In a lengthy blog post today, GitHub's Jesse Newland apologized for the outage and said overall it was way below the company's standards.

The root of the problem s
maintenance, the GitHub
cluster. The new infrastru
This means a failover "si
transactions and approp

At the time of this failover, the new database selected for the 'active' role had a cold InnoDB buffer pool and performed rather poorly. The system load generated by the site's query load on a cold cache soon caused Percona Replication Manager's health checks to fail again, and the 'active' role failed back to the server it was on originally.

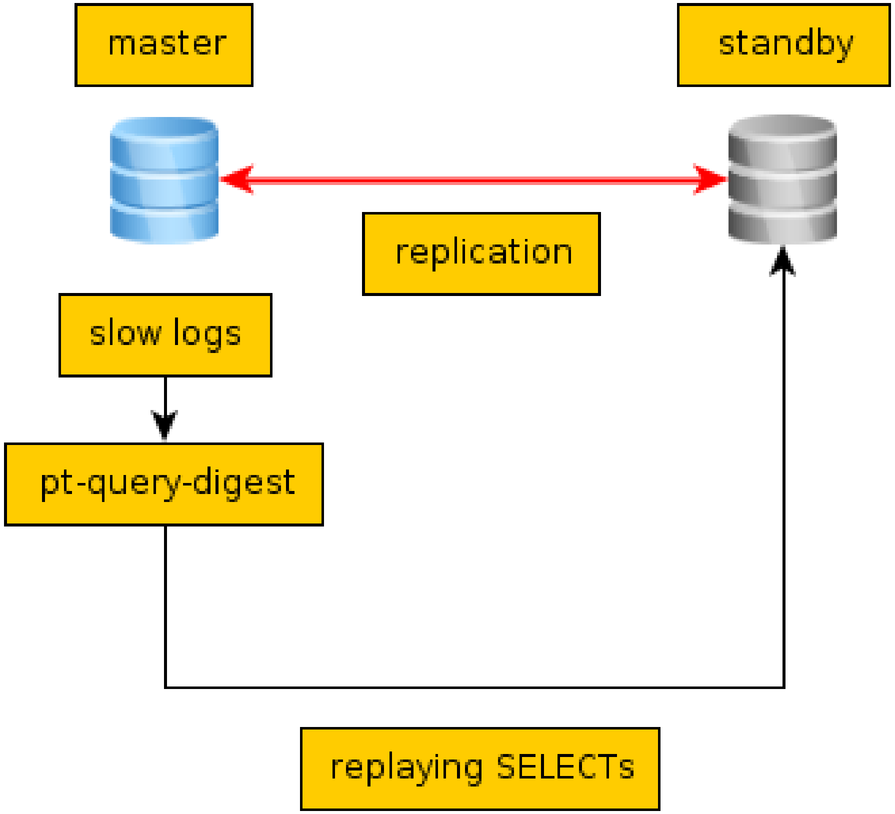
Original problem @ Groupon

- After a failover, the former standby host is heavily IO bound for several minutes (can be in the 10 minute range).
- Replication helps warm the buffer pool via writes, but it's not enough. Reads are required.
  - The reads from the production workload are warm up the buffer pool actually.

Take #1

- Simple script with pt-query-digest
  - Filters the SELECT queries
  - Executes it on the standby host
- Issues
  - Runs on the production master
  - Single Threaded
  - SELECT can also write, which would lead to inconsistencies

Take #1 architecture



master

standby

replication

slow logs

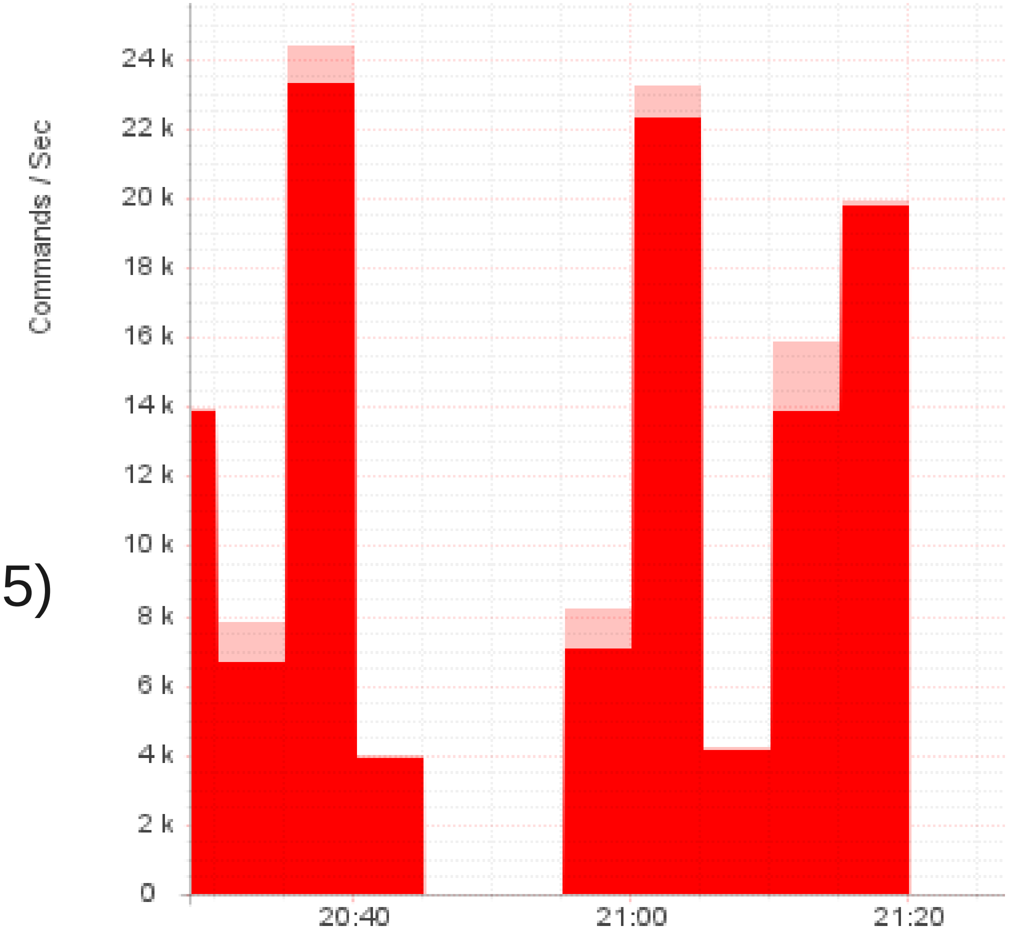
pt-query-digest

replaying SELECTs

Original workload

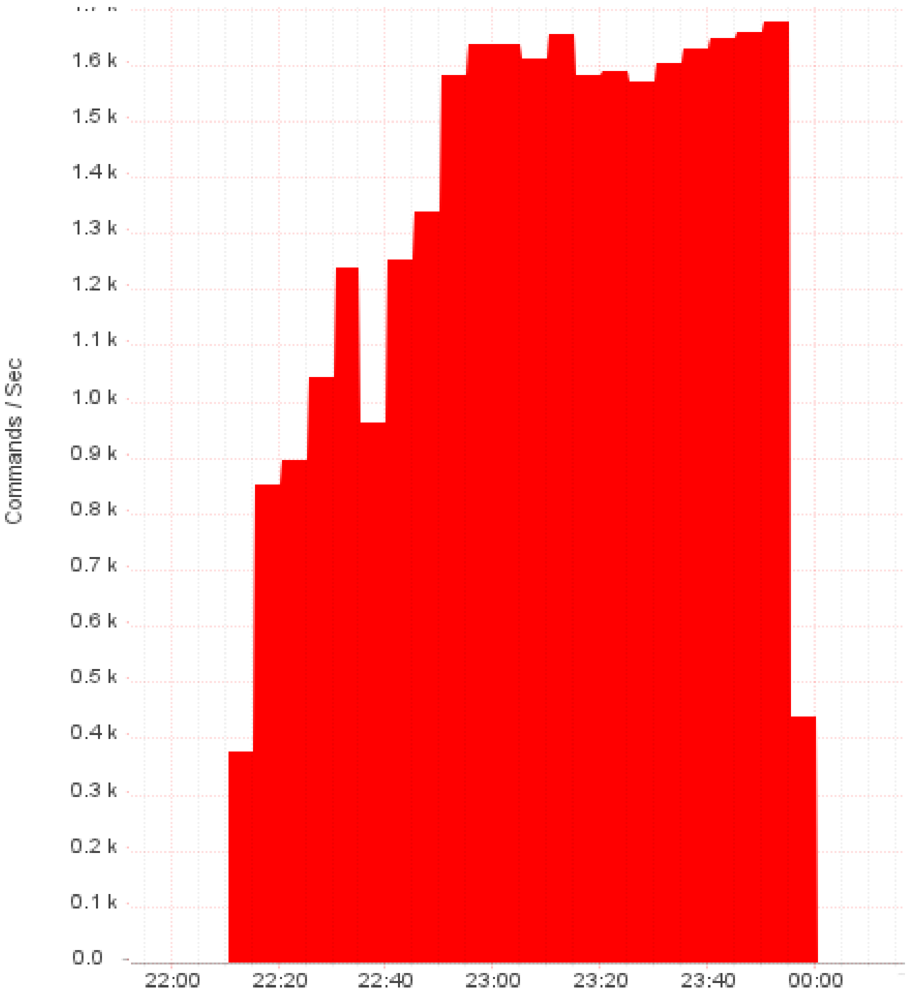
- ~20k QPS peak
- Execution took 25 minutes

(workload begins at 20:55)



Workoad played back

- ~1.7k QPS peak
- Execution took almost 2 hours



Possible Solution: rate limiting

- Do not play back every statement
  - Use rate limited slow log
    - log_slow_rate_type=query
    - log_slow_rate_limit={2..100}
      - 2 -> 50% of the statements
      - 100 -> 1% of the statements
- The warmup tool still runs on the active host

Possible Solution: Percona playback

- Reproduces a workload based on slow log
- Whenever it encouters a new thread id in slow log, a new connection is opened
- Queries executed on that connection will be executed in the opened connection
- This enables parallel replay, the degree of parallelism will be same as production workload

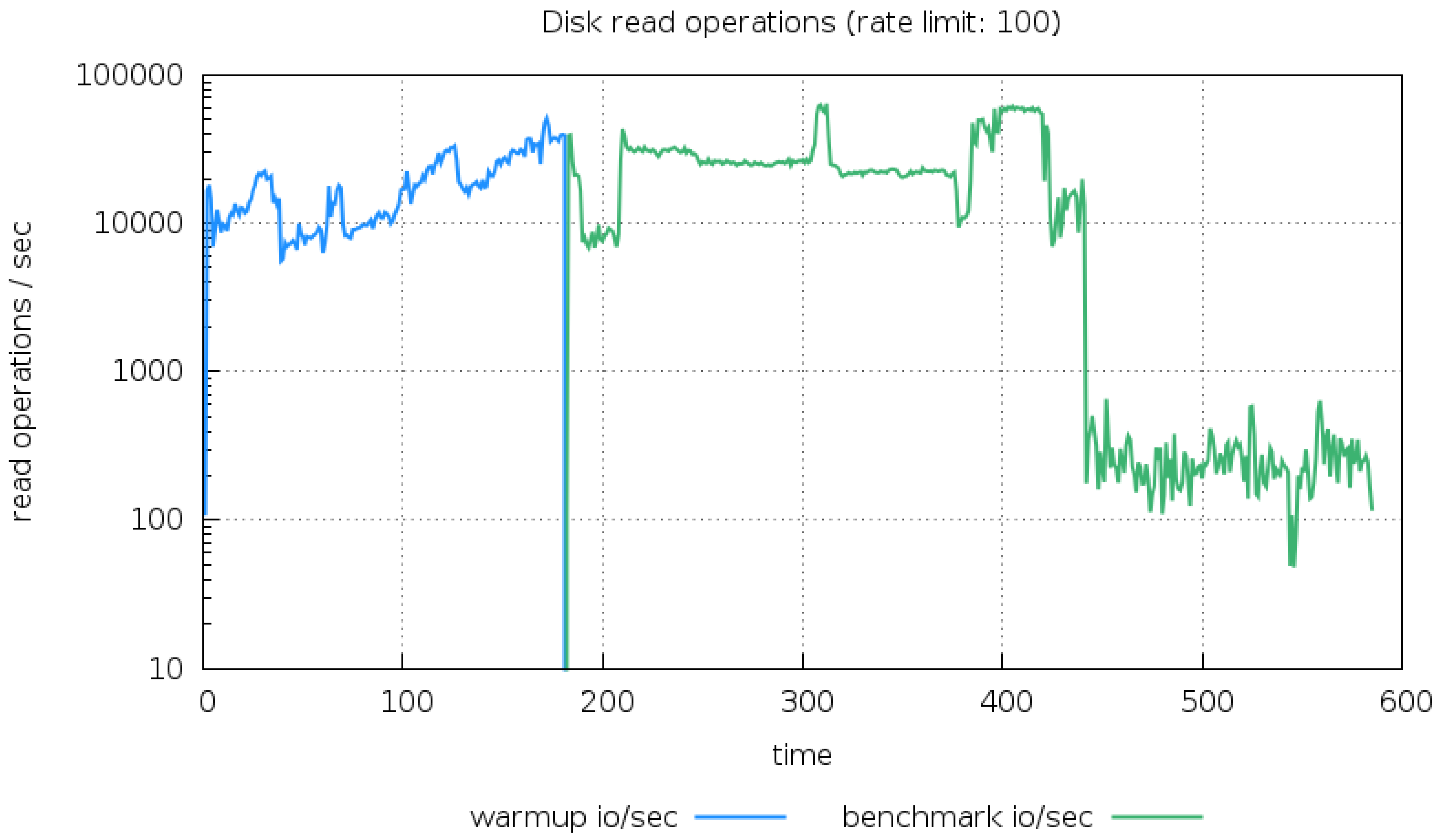
Benchmark

- A few hours of slow log was captured, and they were splitted into 38 chunks, with roughly 0.5M events in each.
- For one measurement 1 or 2 chunks were used.

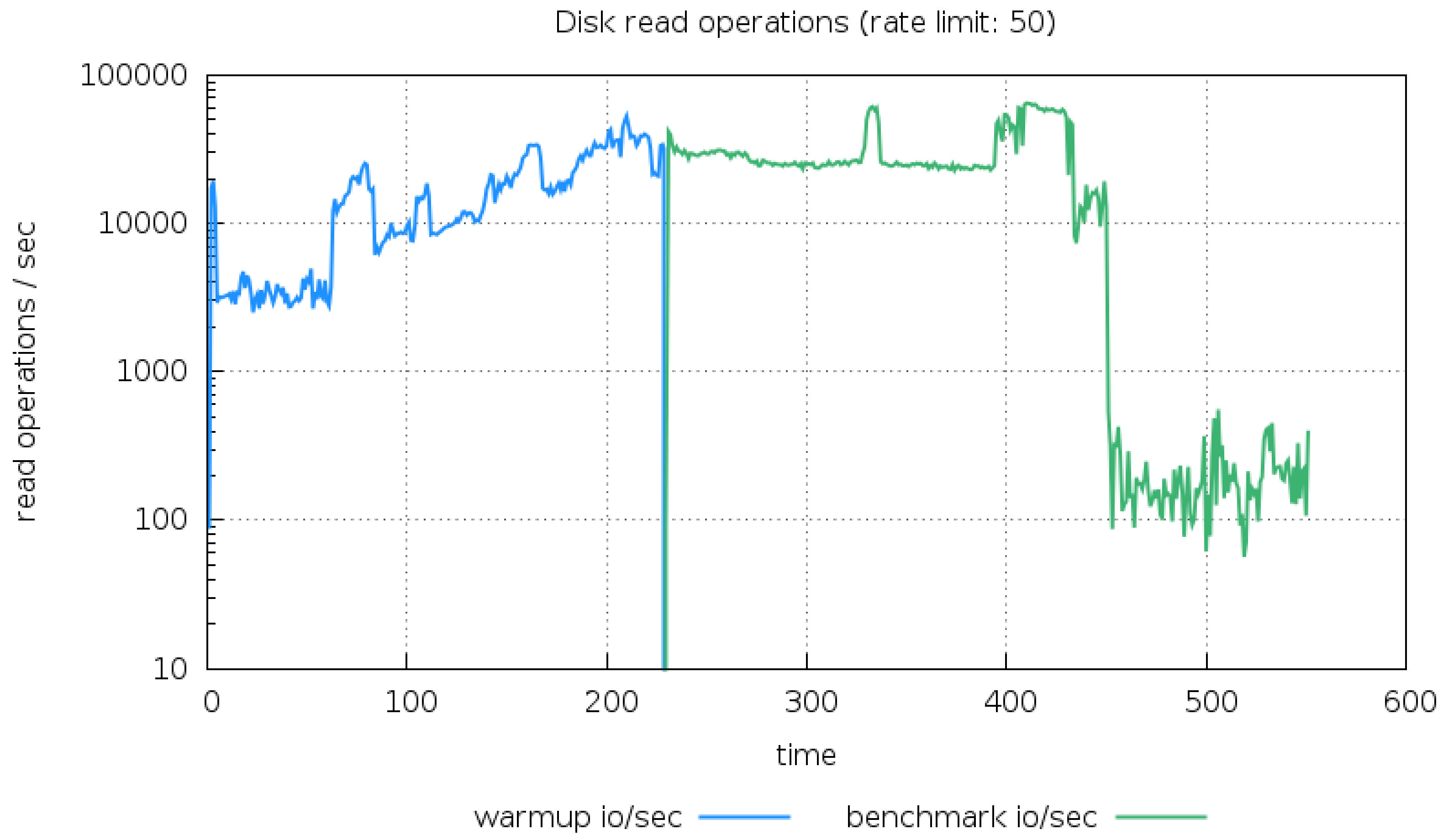
Rate limiting benchmark

- Rate limiting chunk 1, playing back chunk 2.
- Rate limiting chunk 2, playing back chunk 4.
- Normally the previous chunk warms up the buffer pool for the next chunk.
- Inconsistent results in terms of rate limit, and it is also dependent on which chunk I used.
- The solution can work, but when it warms up the slave is heavily workload dependent.

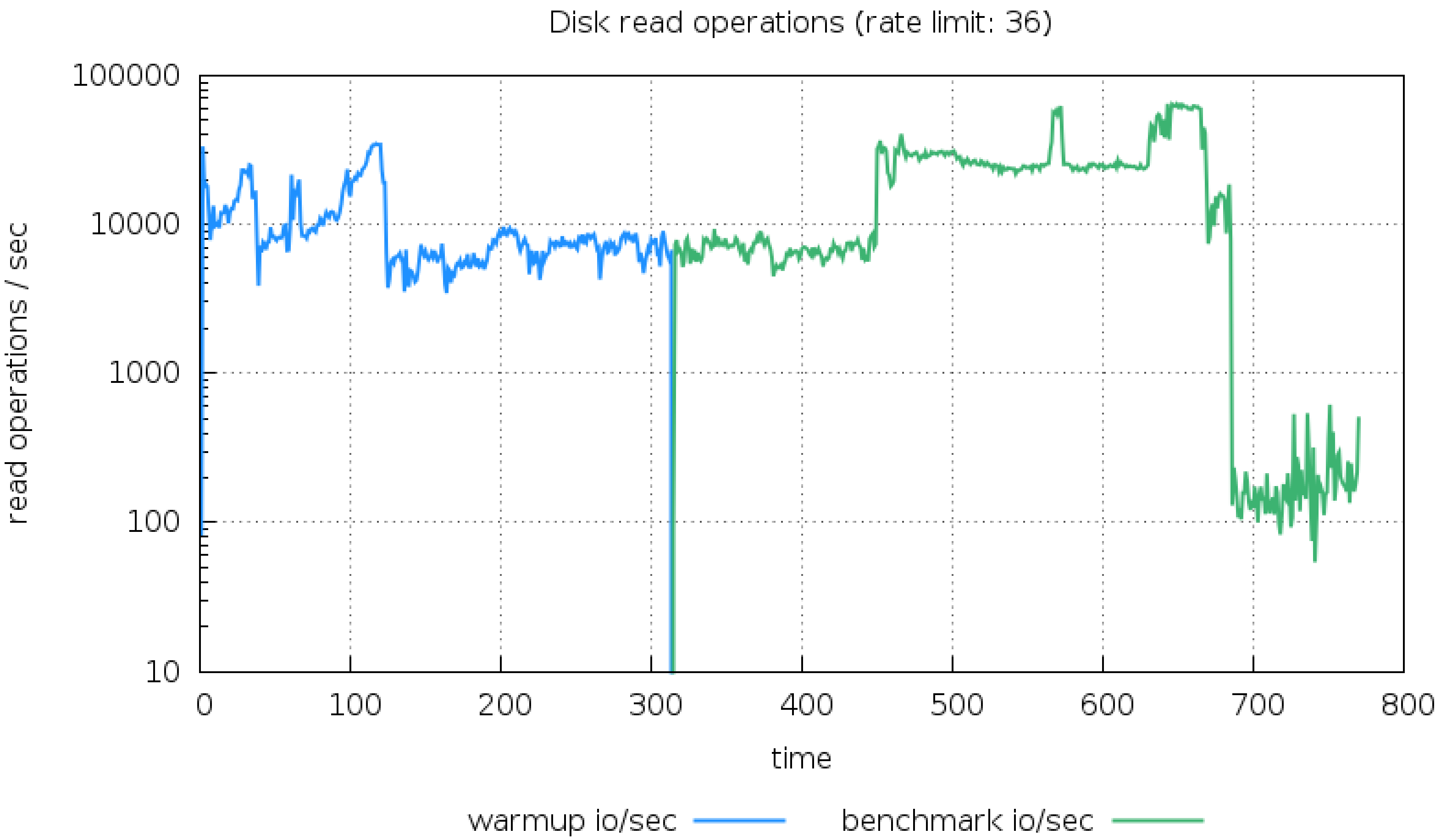
Possible Solution: rate limiting



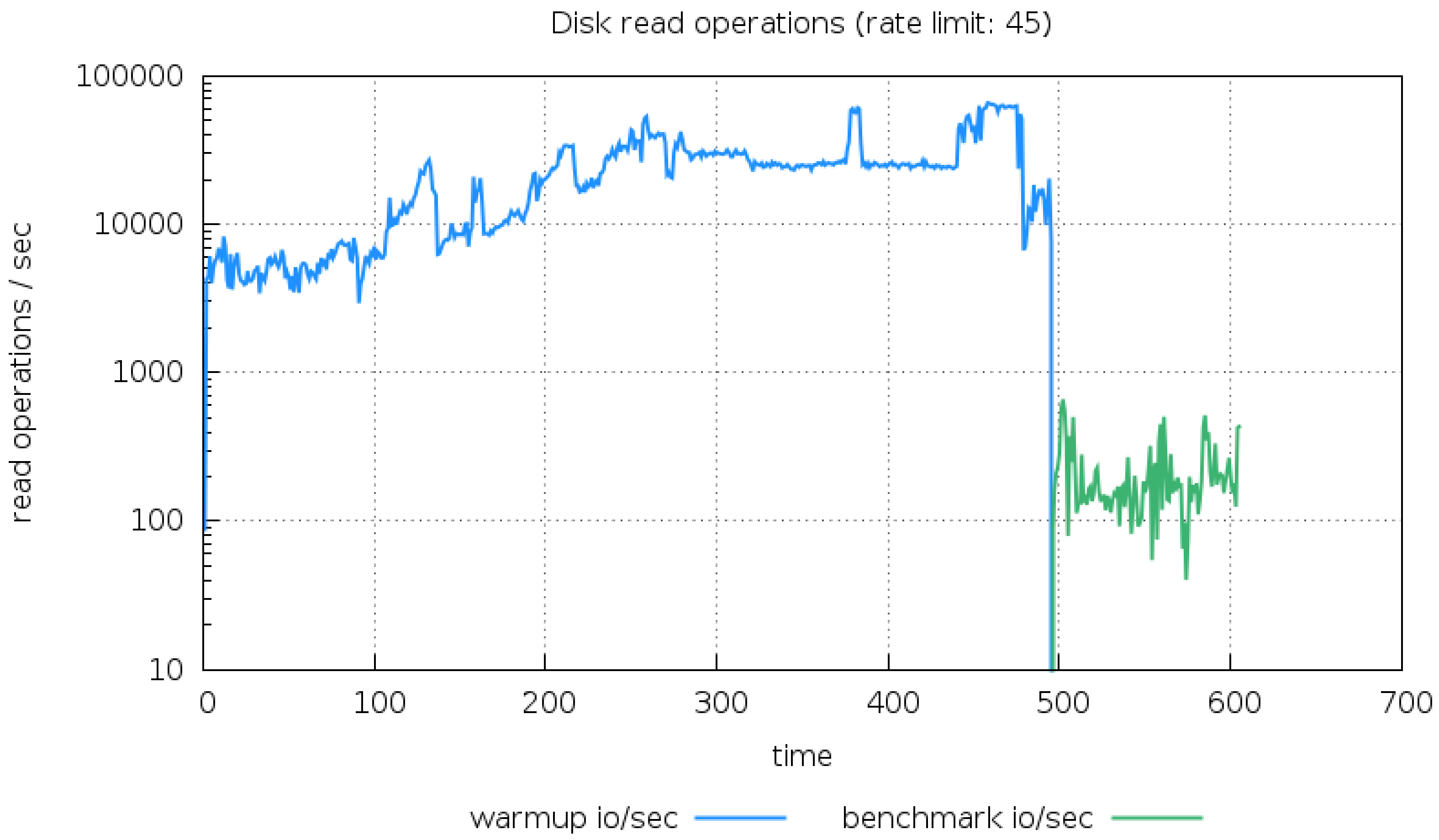
Possible Solution: rate limiting



Possible Solution: rate limiting



Possible Solution: rate limiting



Possible Solution: rate limiting

- The rate_limit=45 case looks better than 36
- Too dependent on the workload, we got inconsistent results. Sometimes every 50th query is enough, sometimes even using every second statement has a negative impact on performance.

Possible Solution: parallel playback

- Play back with the original parallelism
  - Percona playback is required
- Rate limiting is not needed
  - Can be used to handle smaller slow logs
- Need to handle and rotate out huge slow log continuously

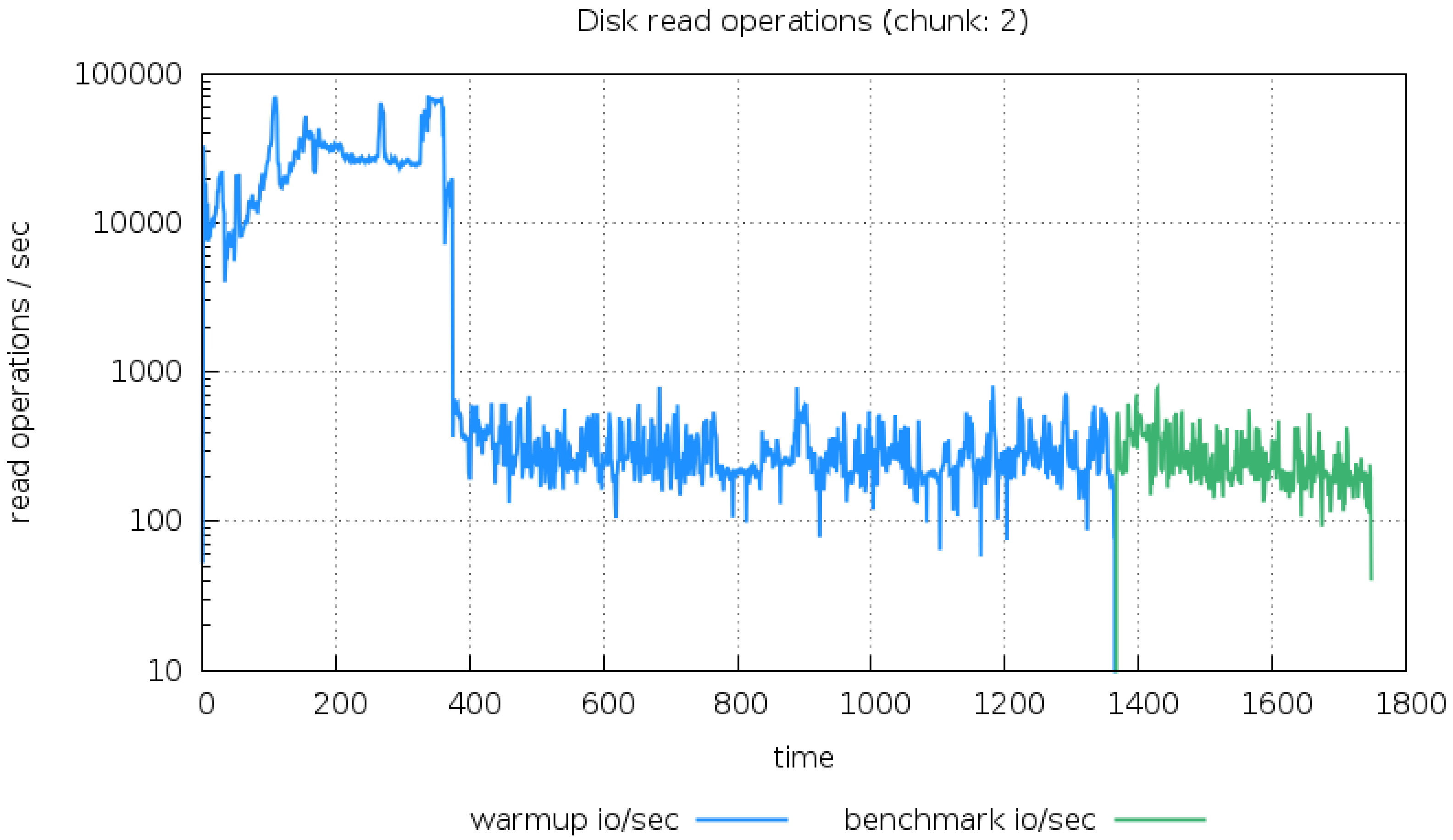
Which one is the winner?

- Sampled slow log can be efficient, most likely multiple queries in the workload are touching the same page.
- What is the difference between using a sampled slow log and a full slow log?
- With sampling, it will take more time for the slave to be failover ready.
- We chose playback

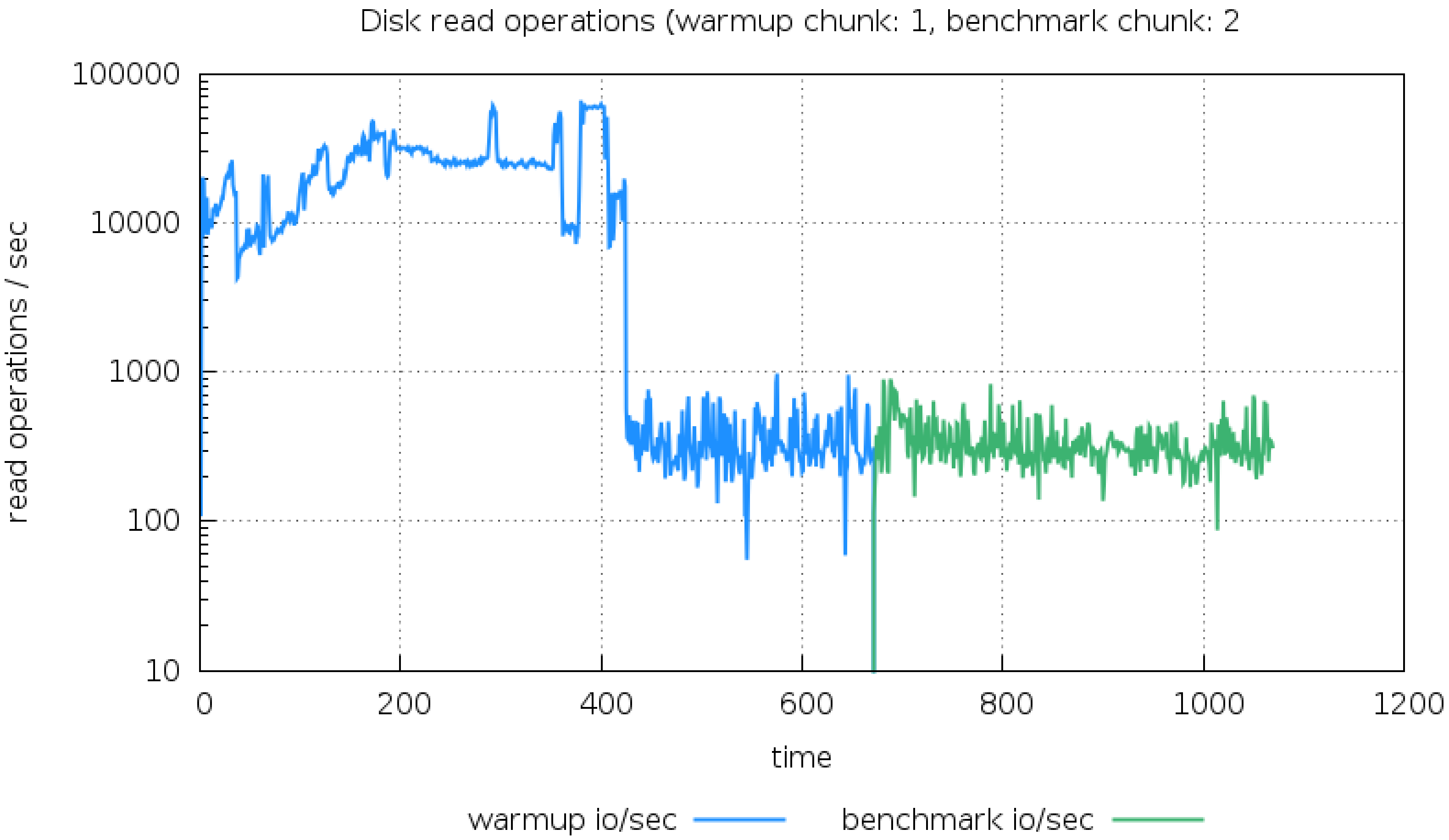
Benchmark

- Control measurement: pre-warm the database with the first file and play back the first file.
- Measurement: pre-warm the database with the first file and then play back the second file (scenario, which happens in production).

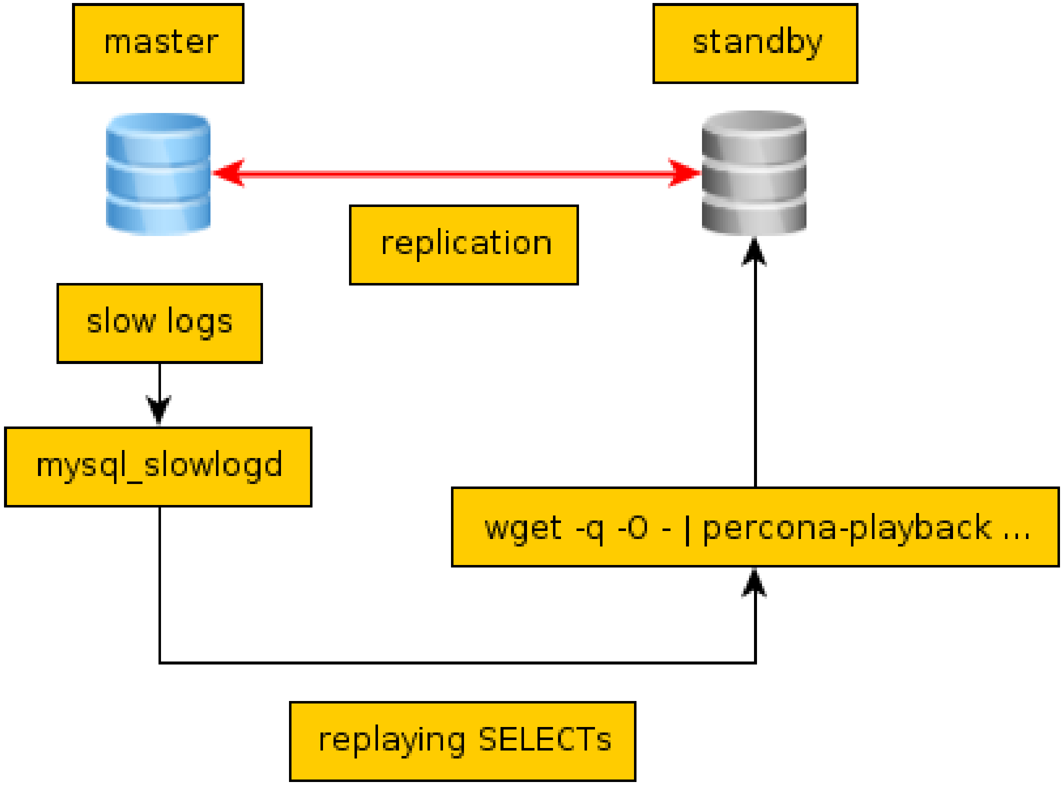
Results: chunk 2 warmed up with itself



Results: chunk 2 warmed up with chunk 1



Playback architecture



New playback features
(only available in trunk right NOW())

- Stream the slow logs to the standby as fast as possible
  - Playback from standard input
- Make playback read only
  - Use session_init_query, so we can use innodb_fake_changes
- Handle not gracefully closed connections
  - Thread pool for playback

mysql_slowlogd

- The other end of the stream on the master
    - Serves the slow log on HTTP
    - It looks for the beginning of the previous slow log event at connect time
        - It serves only full slow log events
- Mechanism is similar to xtail
    - Handles log rotations
- Groupon plans to open source it at github.com/groupon

Rotating slow log

- Don't use the default log rotation with copytruncate, all threads will be stuck in logging slow query state
- Use FLUSH SLOW LOGS and filesystem operations in pre and postrotate to do this efficiently
- On ext3, this issue is much more visible.

Handling failover

- Harness script, which does checks every minute -> if the application user is connected, then machine is active.
- There will be some time after failover (< 1 min), while playback will be running on active node.
    - This is not an issue, because data will stop flowing from the former active node (not using log_slow_slave_statements)

Q&A

Thank you