EXPERIENCES USING GH-OST IN A MULTI-TIER TOPOLOGY

Ivan Groenewold
Valerie Parham-Thompson

26 April 2017

Pythian

WHY USE GH-OST?

Why not use native online schema change capabilities of MySQL/MariaDB?

- Some changes still not supported online (e.g., data change types).
- You want to be able to better view status of a change (via interactive commands).
- You want to be able to pause (throttle) a change during a busy time without impacting the master or slave.
- Note the table copy is still done, but using gh-ost gives you more flexibility to control the impact on production traffic.

INTRODUCTION TO INTERACTIVE COMMANDS

- Send commands using netcat
  - `echo 'status' | nc -U /tmp/gh-ost.test.sock`
  - `echo chunk-size=? | nc -U /tmp/gh-ost.test.sock`
  - `echo 'chunk-size=2500' | nc -U /tmp/gh-ost.test.sock`
  - `echo '[no-]throttle' | nc -U /tmp/gh-ost.test.sock`

- https://github.com/github/gh-ost/blob/master/doc/interactive-commands.md

INTERACTIVE COMMANDS: Pause a Change

Scenario: I’m running a change on a table, but it’s taking longer than expected.

Pause the change.
echo throttle | nc -U /tmp/gh-ost...sock

(Make sure binlogs don’t get purged.)

Continue the change after workday is done.
echo no-throttle | nc -U /tmp/gh-ost...sock

INTERACTIVE COMMANDS: Pause a Change

Scenario: A multisource slave is replicating from master 1 and master 2. A change is being executed on master 1, but a new one needs to be run on master 2 immediately, and we want to avoid too much load on the slave from two changes.

Pause the change running from master 1.
echo throttle | nc -U /tmp/gh-ost...sock

Run the change from master 2.

Continue the change from master 1.
echo no-throttle | nc -U /tmp/gh-ost...sock

INTERACTIVE COMMANDS: View Status

Scenario: View status while the change from master 1 is paused:

```
echo sup | nc -U /tmp/gh-ost.d2.t1.sock

Copy: 0/57064 0.0%; Applied: 9411; Backlog: 100/100; Time: 2m24s(total), 2m14s(copy);
streamer: pres_gh_multis1-bin.000006:249953149; State: migrating; ETA: N/A

echo sup | nc -U /tmp/gh-ost.d1.t1.sock

Copy: 0/461251 0.0%; Applied: 905; Backlog: 100/100; Time: 2m53s(total), 2m51s(copy);
streamer: pres_gh_multis1-bin.000006:241532683; State: throttled, commanded by user; ETA:
N/A
```

INTERACTIVE COMMANDS: Postpone Cutover

- Scenario: You want to perform cut-over on lowest traffic time
- Gh-ost won't swap tables while flag file is present
  - `--postpone-cut-over-flag-file=/tmp/file`
- Old and new tables will be kept in sync indefinitely
- When you are ready, just remove the flag file
- Can also send text command to socket
  - `echo unpostpone | nc -U /tmp/gh-ost.test.sock`

USE CASE: TUNGSTEN

BRIEF INTRO TO TUNGSTEN

Tungsten Replicator

- direct replacement for the native MySQL replication
- can replicate to other DBMS (Oracle, MongoDB, etc.)

Tungsten Manager (Enterprise edition only)

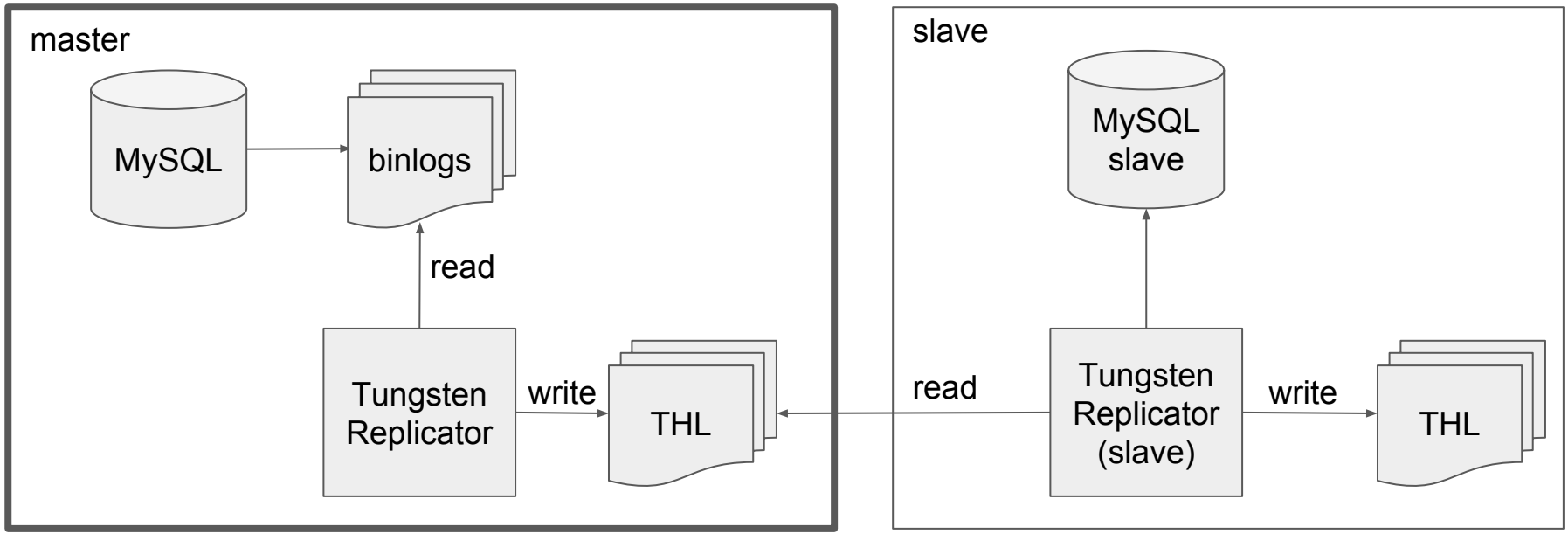
- control and information source for the status and health of the cluster
- Topology changes

Tungsten Connector (Enterprise edition only)

- routes connections from application servers
- performs r/w splitting

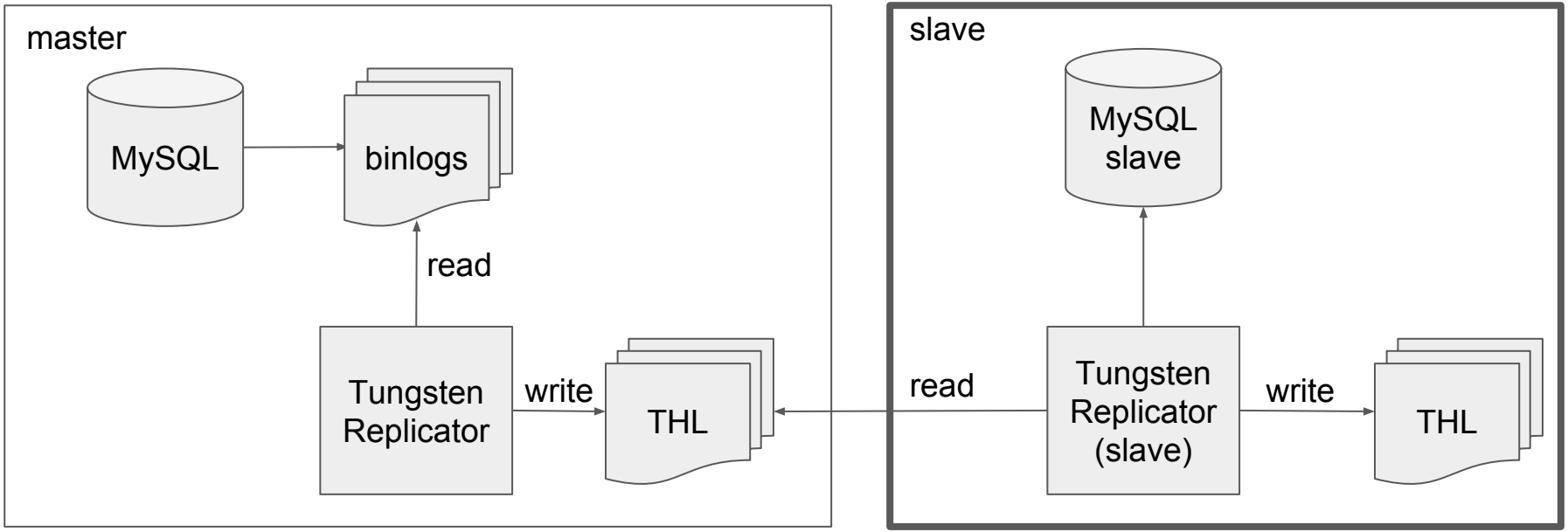
BRIEF INTRO TO TUNGSTEN - MASTER ROLE

- Replication is external to MySQL
- Replicator installed on each server
- Events are read from master's binlog, written to db-agnostic files (THL)

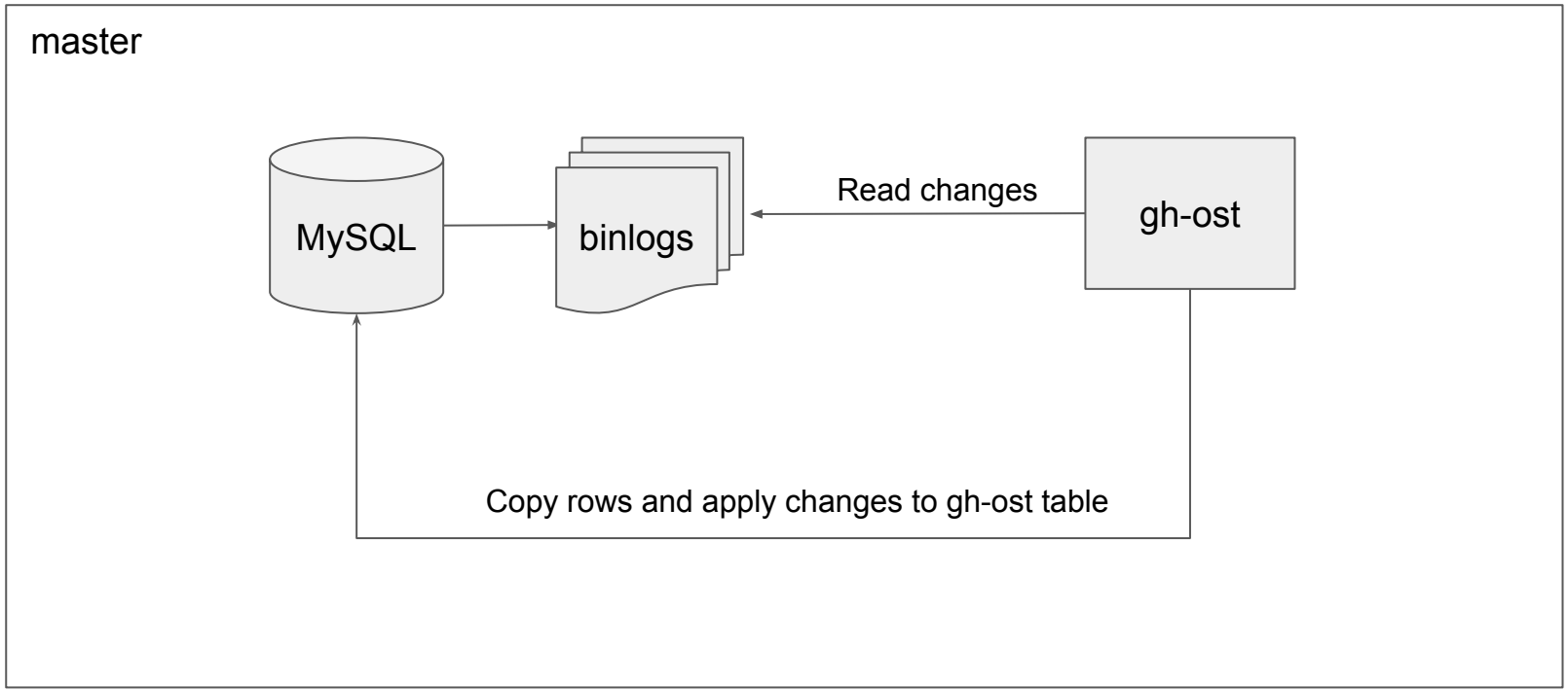


BRIEF INTRO TO TUNGSTEN - SLAVE ROLE

- Slave replicator connects to master replicator to fetch transactions
- Slave replicator applies events to slave DB
- Slave replicator stores THL locally



GH-OST AGAINST TUNGSTEN MASTER



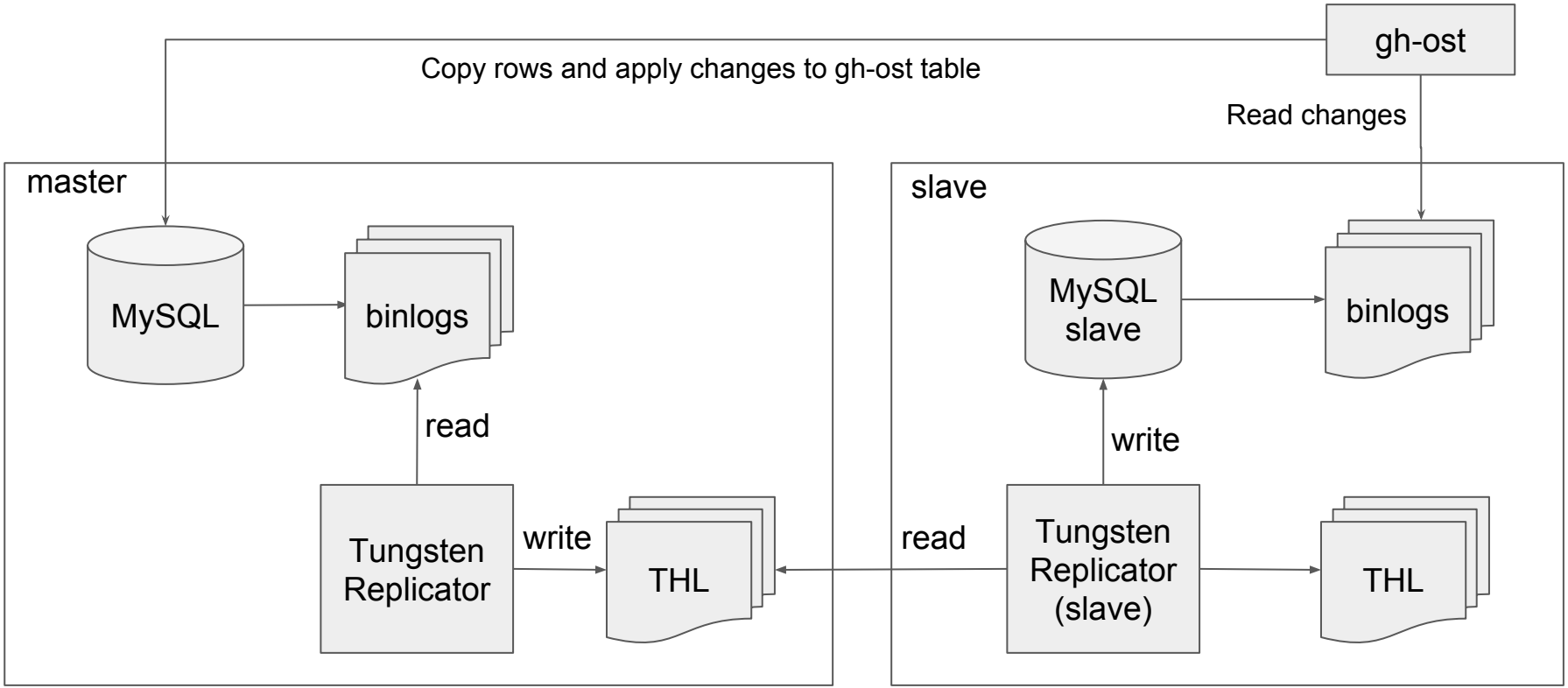
GH-OST AGAINST TUNGSTEN MASTER

- Use `--assume-master-host` to specify the actual master
- Specify `--allow-on-master`
- `--host` this is where gh-ost connects to read changes (use master)
- No need to use `--tungsten` flag
- Master needs `binlog_format=ROW`
- Be careful with extra load introduced by `--exact-rowcount`

GH-OST AGAINST TUNGSTEN MASTER

```
./gh-ost \
--assume-master-host=<master_ip> \
--allow-on-master \
--chunk-size=1000 \
--throttle-control-replicas="<slave1_ip>,<slave2_ip>" \
--host=<master_ip> \
--database="test" \
--table="test_t" \
--verbose \
--alter="add index(...)" \
--panic-flag-file=/tmp/ghost.panic.flag \
--postpone-cut-over-flag-file=/tmp/ghost.postpone.flag \
--execute
```

GH-OST USING TUNGSTEN SLAVE



GH-OST USING TUNGSTEN SLAVE

- Slave needs Tungsten `log-slave-updates` parameter, otherwise it will hang at `INFO Waiting for tables to be in place`
  - `tpm configure property=log-slave-updates=true`
- Use `--assume-master-host` to specify actual master
- Set `--host` to use the slave
- Specify `--tungsten` argument
- Slave needs `binlog_format=ROW`
- Consider using `--exact-rowcount`

GH-OST USING TUNGSTEN SLAVE

```
./gh-ost \
--assume-master-host=<master_ip> \
--tungsten \
--chunk-size=1000 \
--throttle-control-replicas="<slave2_ip>" \
--host=<slave1_ip> \
--database="test" \
--table="test_t" \
--verbose \
--alter="add index(...)" \
--panic-flag-file=/tmp/ghost.panic.flag \
--postpone-cut-over-flag-file=/tmp/ghost.postpone.flag \
--exact-rowcount \
--execute
```

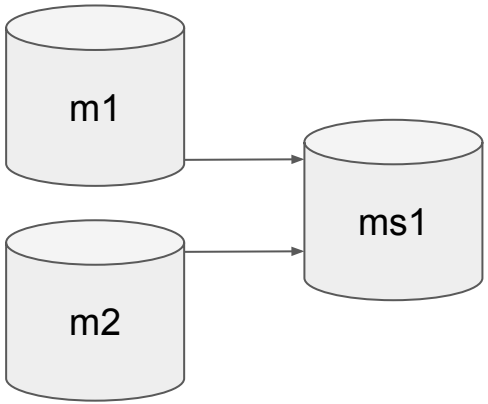
GH-OST AND TUNGSTEN: FINAL CONSIDERATIONS

- Use `binlog_format=ROW` on the host gh-ost is running
- Changes to `binlog_format` requires replicator restart
- Avoid `--switch-to-rbr` (doesn't affect Tungsten)
- Leverage `--exact-rowcount` (default is to use statistics)
- If running on a slave, configure Tungsten's `log-slave-updates`

USE CASE: MULTISOURCE REPLICATION

INTRODUCTION TO MARIADB MULTISOURCE REPLICATION

Multisource replication is commonly found in MariaDB because it has been supported in the product for a while. One slave supports multiple replication threads (commonly with filters to feed changes to different schemas).



INTRODUCTION TO MARIADB MULTISOURCE REPLICATION

Sample replication setup.

```
CHANGE MASTER 'm1' TO MASTER_HOST =
'192.168.56.10', MASTER_LOG_FILE =
'pres_gh_m1-bin.000008', MASTER_LOG_POS = 428,
MASTER_USER='repl', MASTER_PASSWORD='repl';

CHANGE MASTER 'm2' TO MASTER_HOST =
'192.168.56.11', MASTER_LOG_FILE =
'pres_gh_m2-bin.000004', MASTER_LOG_POS = 412,
MASTER_USER='repl', MASTER_PASSWORD='repl';

start slave 'm1';
start slave 'm2';
```

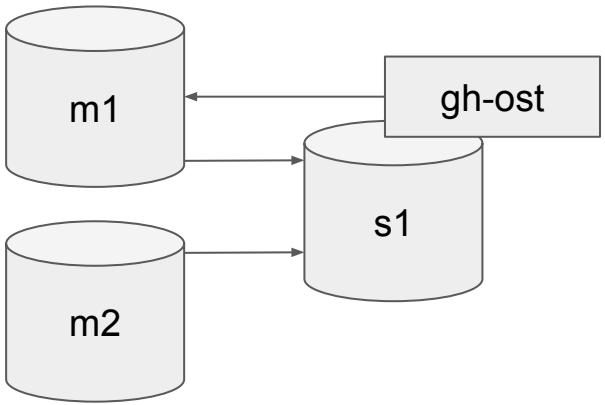
INTRODUCTION TO MARIADB MULTISOURCE REPLICATION

Sample output.

```
MariaDB [(none)]> show all slaves status\G

               Connection_name: m1
...
                  Master_Host: 192.168.56.10
                  Master_User: root
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: maria101-bin.000004
          Read_Master_Log_Pos: 2176440
               Relay_Log_File:
maria101multislave-relay-bin-d1.000007
                Relay_Log_Pos: 2176731
        Relay_Master_Log_File: maria101-bin.000004
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes

               Connection_name: m2
...
                  Master_Host: 192.168.56.11
                  Master_User: root
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File:
maria101master2-bin.000002
          Read_Master_Log_Pos: 1731
               Relay_Log_File:
maria101multislave-relay-bin-d2.000003
                Relay_Log_Pos: 2029
        Relay_Master_Log_File:
maria101master2-bin.000002
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes

2 rows in set (0.00 sec)
```

GH-OST AGAINST SLAVE IN MARIADB MULTI-SOURCE

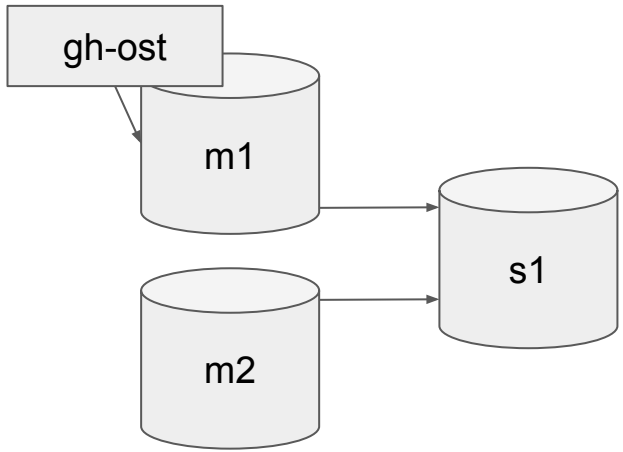
Previous version required running gh-ost on the master. See https://github.com/github/gh-ost/issues/225. Now we can use assume-master-host.




```
./gh-ost \
--max-load=Threads_running=25 \
--critical-load=Threads_running=1000 \
--chunk-size=1000 \
--throttle-control-replicas="192.168.56.12" \
--max-lag-millis=1500 \
--user="ghost" \
--password="ghost" \
--allow-master-master \
--assume-master-host=192.168.56.10 \
--database="d1" \
--table="t1" \
--verbose \
--alter="change column name name char(50)" \
--cut-over=default \
--default-retries=120 \
--panic-flag-file=/tmp/ghost.panic.flag \
--postpone-cut-over-flag-file=/tmp/ghost.postpone.flag \
--initially-drop-ghost-table \
--initially-drop-old-table \
--execute
```

GH-OST AGAINST MASTER IN MARIADB MULTI-SOURCE

This requires binlog_format=ROW on master and moves load to master as well. It's less recommended to use this method.

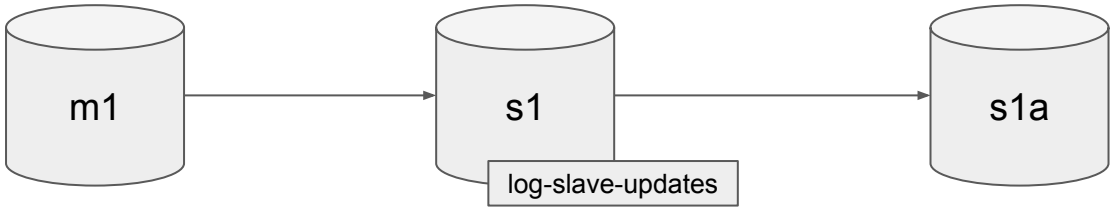



```
./gh-ost \
--max-load=Threads_running=25 \
--critical-load=Threads_running=1000 \
--chunk-size=1000 \
--throttle-control-replicas=s1 \
--max-lag-millis=1500 \
--user="ghost" \
--password="ghost" \
--host=m1 \
--database="d1" \
--table="t1" \
--verbose \
--alter="add column whatever varchar(50)" \
--switch-to-rbr \
--allow-on-master \
--cut-over=default \
--default-retries=120 \
--panic-flag-file=/tmp/ghost.panic.flag \
--postpone-cut-over-flag-file=/tmp/ghost.postpone.flag \
--initially-drop-ghost-table \
--initially-drop-old-table \
--execute
```

USE CASE: DAISY-CHAINED REPLICATION

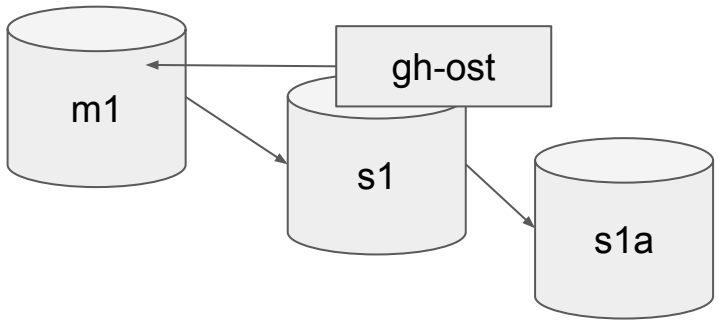
INTRODUCTION TO DAISY-CHAINED SLAVES

Daisy-chained slaves are commonly used in complex topologies to scale a large number of second-level slaves while avoiding load on the master from the replication threads. The key is to use “log-slave-updates” so that changes are passed to the second-level slaves.



GH-OST AGAINST SLAVE IN DAISY-CHAINED SLAVES

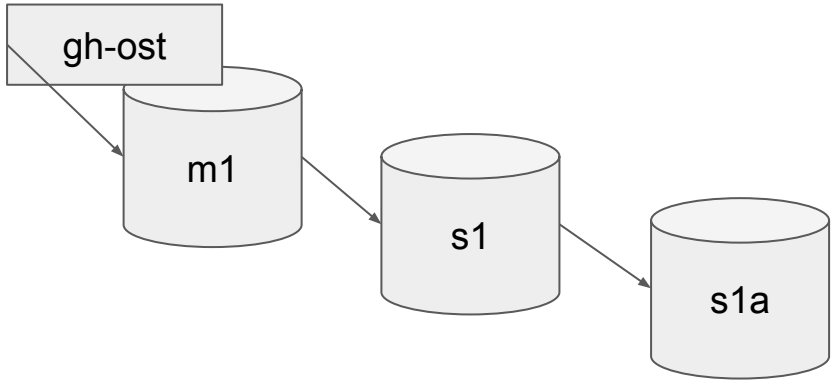
Run on intermediate slave. Because gh-ost is unwilling to set RBR when it detects a slave, set binlog_format=ROW on the intermediate slave beforehand.




```
./gh-ost \
--max-load=Threads_running=25 \
--critical-load=Threads_running=1000 \
--chunk-size=1000 \
--throttle-control-replicas=s1 \
--max-lag-millis=1500 \
--host=s1 \
--user="ghost" \
--password="ghost" \
--database="d1" \
--table="t1" \
--verbose \
--alter="change column name name varchar(50)" \
--assume-rbr \
--assume-master-host=m1 \
--cut-over=default \
--default-retries=120 \
--panic-flag-file=/tmp/ghost.panic.flag \
--postpone-cut-over-flag-file=/tmp/ghost.postpone.flag \
--initially-drop-ghost-table \
--initially-drop-old-table \
--execute
```

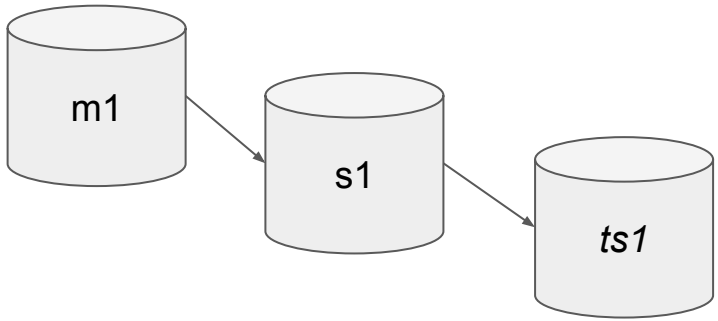
GH-OST AGAINST MASTER IN DAISY-CHAINED SLAVES

As in multisource replication, you can also run the change against the master, but it requires binlog_format=ROW on the master and incurs load on the master as well.



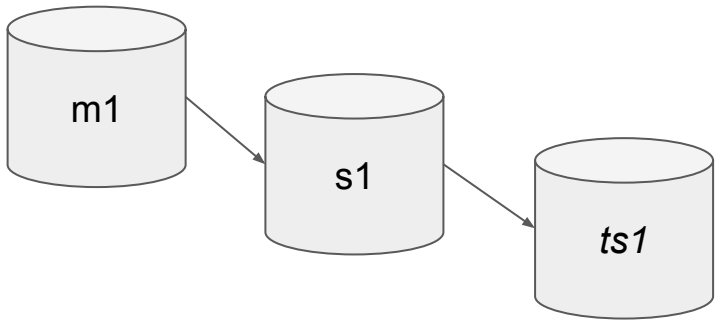
GH-OST WITH A TRANSFORMED SLAVE

Let's say we have a backend process that changes the values of these fields in d1.t1 on ts1: userid=(2*userid) and name='redacted' (for reporting reasons, or to cleanse data). After using gh-ost for the data change, the schema differences are gone. You'll have to redo the changes or pull them from the "del" table if you kept it (e.g., _t1_del).



GH-OST WITH A DIFFERENT ENGINE

Different engines are fine (e.g., TokuDB) as long as the data isn't different (e.g., blackhole).



BEST PRACTICES

SAFELY DROP THE OLD TABLE AFTER SWITCH

- Dropping a large table can cause server stalls
- Stalls are not related to gh-ost itself
  - Bug #51325 - Dropping an empty innodb table takes a long time with large buffer pool
  - Bug #64284 - Eliminate LRU scan when dropping a table
- Buffer pool pages and Adaptive Hash index for old table need to be cleared
- File removed at OS level

SAFELY DROP THE OLD TABLE AFTER SWITCH

- Control behaviour with `--ok-to-drop-table` (default: no)
- Workarounds
  - Slowly delete old data, wait some time, then drop old table
  - Create a hard link, then drop old table, remove hard link
    - http://blog.balazspocze.me/2015/01/13/how-to-drop-table-in-a-hacky-way/
- See discussion at: https://github.com/github/gh-ost/issues/307

WHY GH-OST: REVISITED

- Direct alter for change that rebuilds the table

```
MariaDB [d1]> alter table d1.t1 add column whatever
varchar(50);
Query OK, 0 rows affected (24.57 sec)
Records: 0  Duplicates: 0  Warnings: 0
max lag on replica
Seconds_Behind_Master: 24
```

WHY GH-OST: REVISITED

- pt-online-schema-change for same change

```
time pt-online-schema-change --alter "ADD COLUMN whatever
varchar(50)" D=d1,t=t1 --user ghost --password ghost
--recurse 1 --critical-load Threads_running=1000
--max-load Threads_running=25 --chunk-size 30000
--max-lag 1s --no-drop-old-table --execute
real    0m36.318s
user    0m0.169s
sys    0m0.260s
```

WHY GH-OST: REVISITED

- gh-ost for same change

```
time ./gh-ost --max-load=Threads_running=25
--critical-load=Threads_running=1000 --chunk-size=30000
--throttle-control-replicas=192.168.56.13 --max-lag-millis=1500
--host=192.168.56.13 --user="ghost" --password="ghost" --database="d1"
--table="t1" --verbose --alter="add column whatever varchar(50)"
--assume-rbr --assume-master-host=192.168.56.10 --cut-over=default
--default-retries=120 --panic-flag-file=/tmp/ghost.panic.flag
--postpone-cut-over-flag-file=/tmp/ghost.postpone.flag --execute

real    0m43.111s
```

ANTI-PATTERNS

WHEN WOULD YOU NOT USE GH-OST?

- Foreign keys / triggers
- Can’t change to RBR
- Need to change only unique key or primary key
- Newer column types in 5.7: JSON, generated columns
- Active master - active master
- https://github.com/github/gh-ost/blob/master/doc/requirements-and-limitations.md

ABOUT PYTHIAN

Pythian’s 400+ IT professionals help companies adopt and manage disruptive technologies to better compete