Online Schema Changes for Maximizing Uptime

David Turner - Dropbox
Ben Black - Tango

About us

- Dropbox
  - Dropbox is a free service that lets you bring your photos, docs, and videos anywhere and share them easily. Never email yourself a file again!
- Tango
  - Tango is the only mobile application in the world that combines cross-platform video communication, social discovery, and content on one integrated-mobile platform to connect you with the world around you in a meaningful way.

Where we’re headed

- Schema changes and Defrags: including best practices
- MySQL online DDL
- Percona online schema change

Best Practices

- Backups
- Benchmark
- Schema Repository
- Deployment Strategy
- Monitor

Backups

- You will make a mistake
- Probability increase with number of hosts
- Automate recovery and confirm. At least run quarterly recoveries.

Family Feud - Benchmarking

- Why should you benchmark?

Benchmark

- Expected time to completion
  - Hardware
- Is it better to create sk after loading table?
  - Fast index creation (note: drop and add)
- It’s fun

Schema Repository

- Inconsistency can break promotions
  - missing index
  - missing column
- Automate schema consistency verification

Deployment Strategy

- Wrapper
  - Enforce process
    - Naming conventions
    - SQL_LOG_BIN = 0
    - RBR and slave_type_conversions(no signage..)
    - Performance gains with server variables (restart db)
    - rename before drop
  - Logging
  - Confirm slave updated (lag.., exclusive locks)

Monitor

- Progress
  - Masters or slaves may block on ddl
  - Impact on applications (query latency)
  - Show processlist
  - Show engine innodb status
  - select * from information_schema.global_temporary_tables
  - ls *tablename*
- Errant transactions on gtid enabled clusters

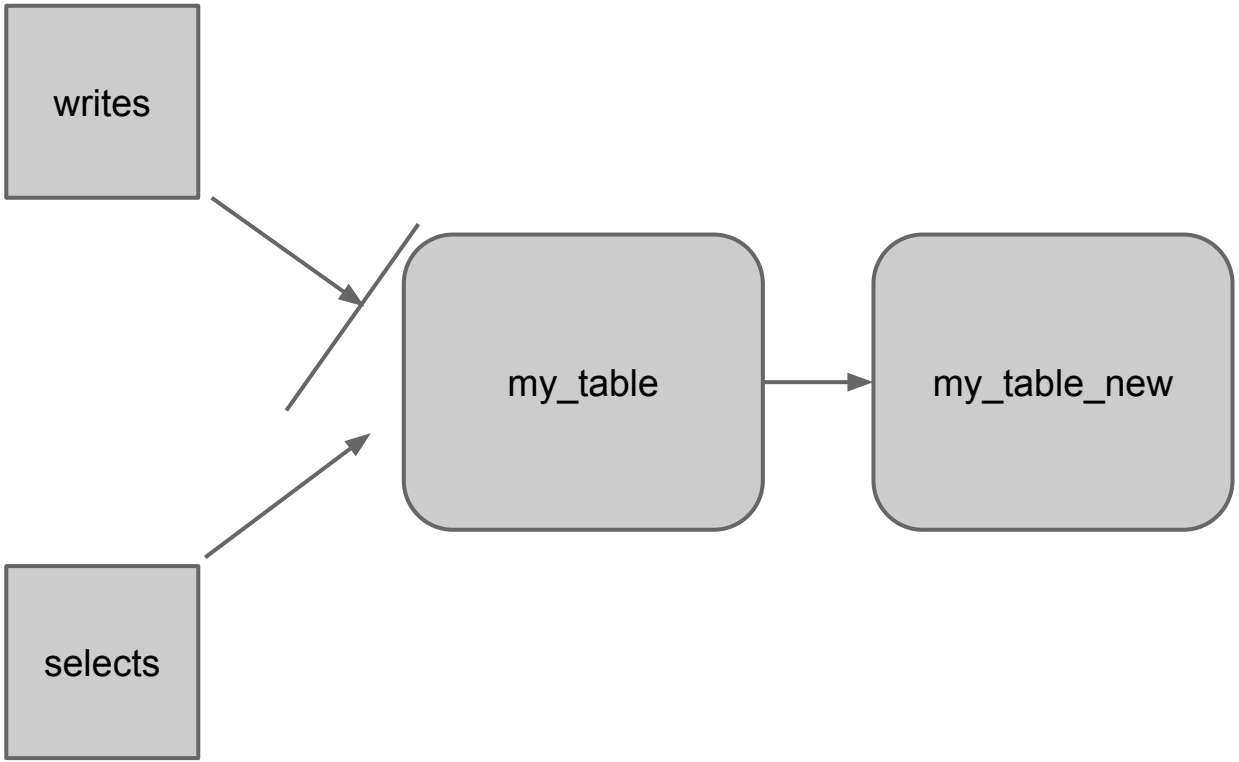
Verify data

- comprehensive slave scans
- pt-table-checksum

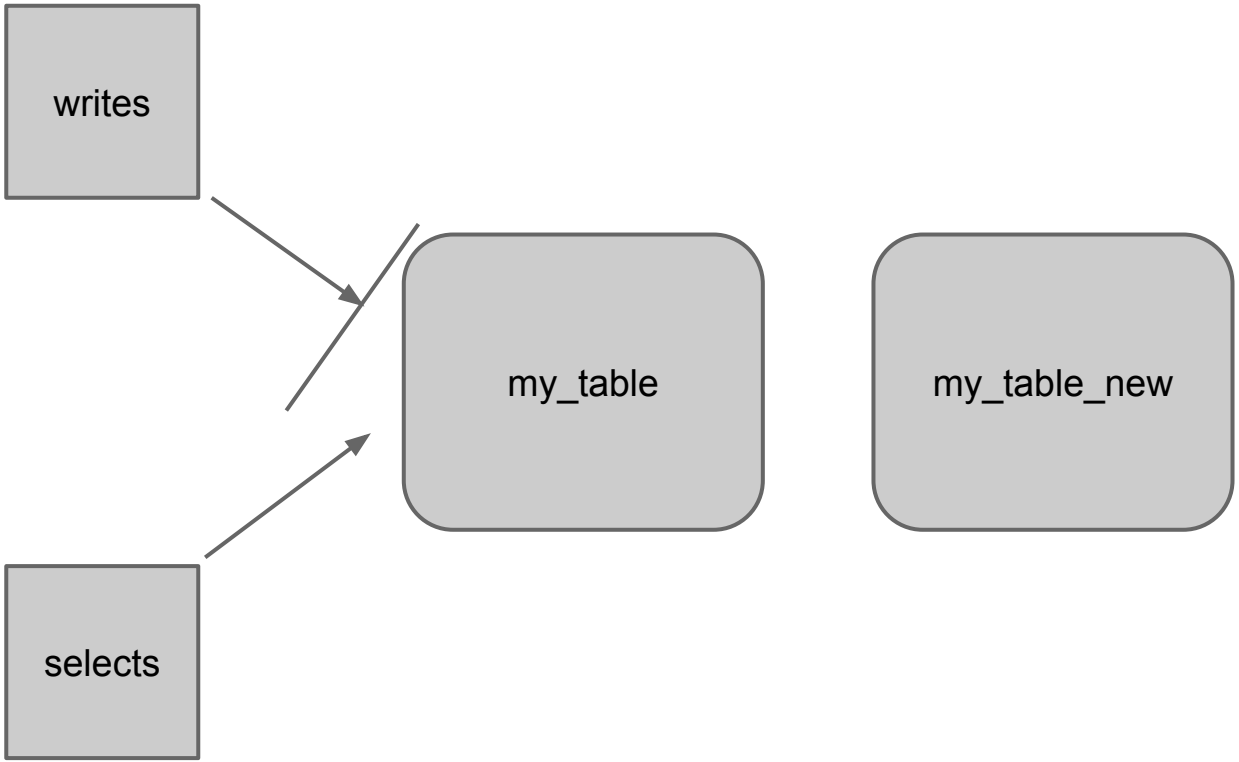
Old school schema changes and defrags

- Alter tables and reclaim space
- Blocking alters
- Customer workarounds

Blocking alters



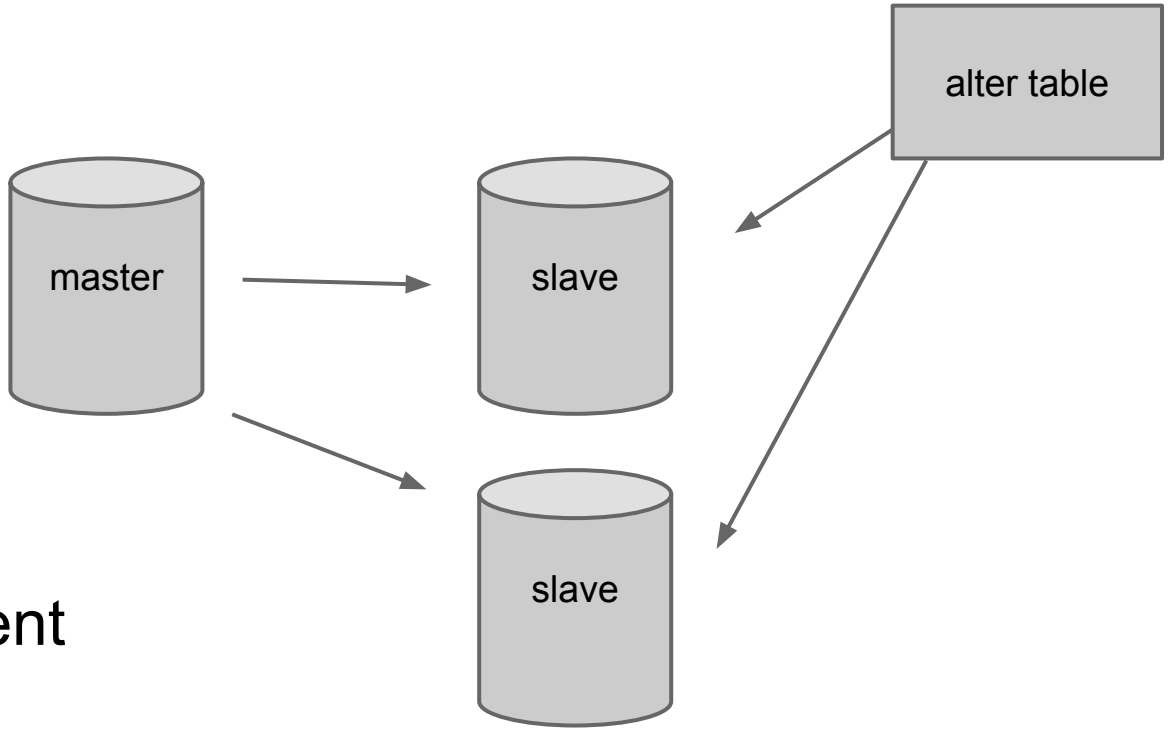
Blocking alters



Problems with blocking alters

- Unable to write to the table
- The larger the table the longer the writes are blocked.

Customer workarounds



- Slave Lag
- Less efficient

Additional customer workarounds

- Openark
- Facebook
- Percona

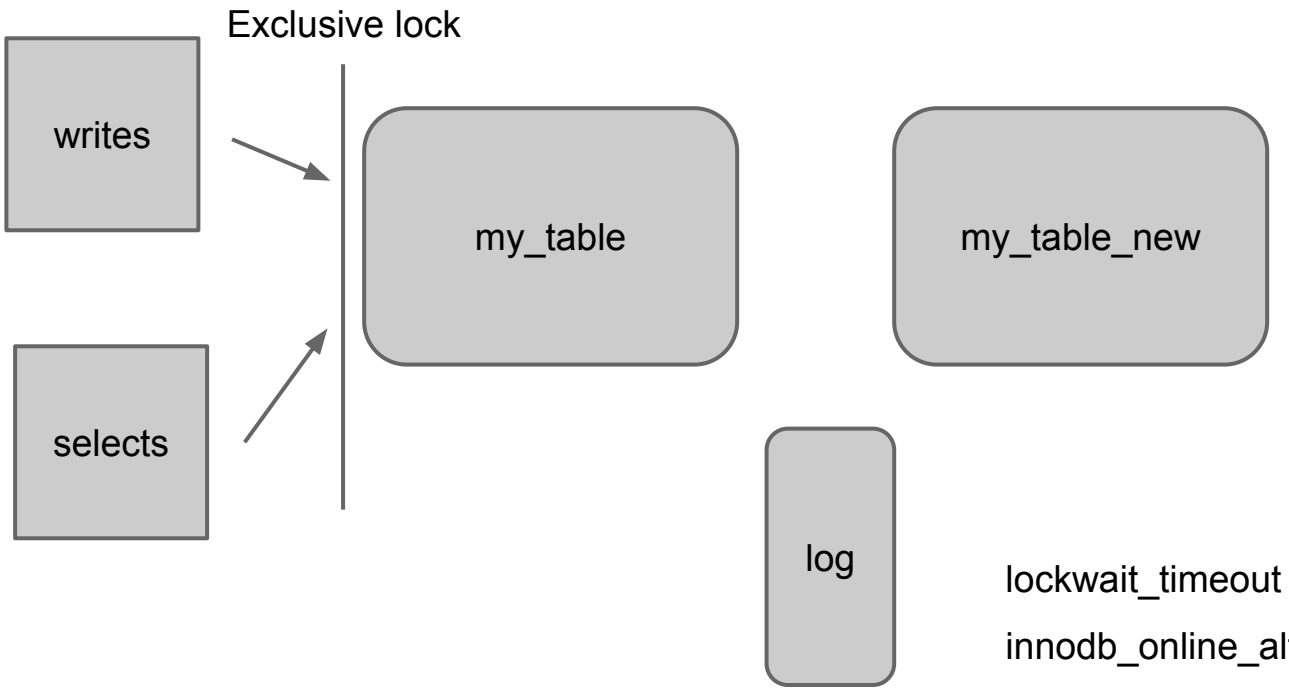
MySQL fast index creation

- Innodb 5.1 Plugin or MySQL 5.5
- No more rebuilds for secondary indexes
- Drop and add indexes faster
- Faster table loads(sk last)
- Better secondary indexes*
- Still blocking writes

MySQL 5.6 Innodb online DDL: are we there yet?

- More in place operations
- Online alters for operations
- Slaves lag for length of operations
- No throttle option

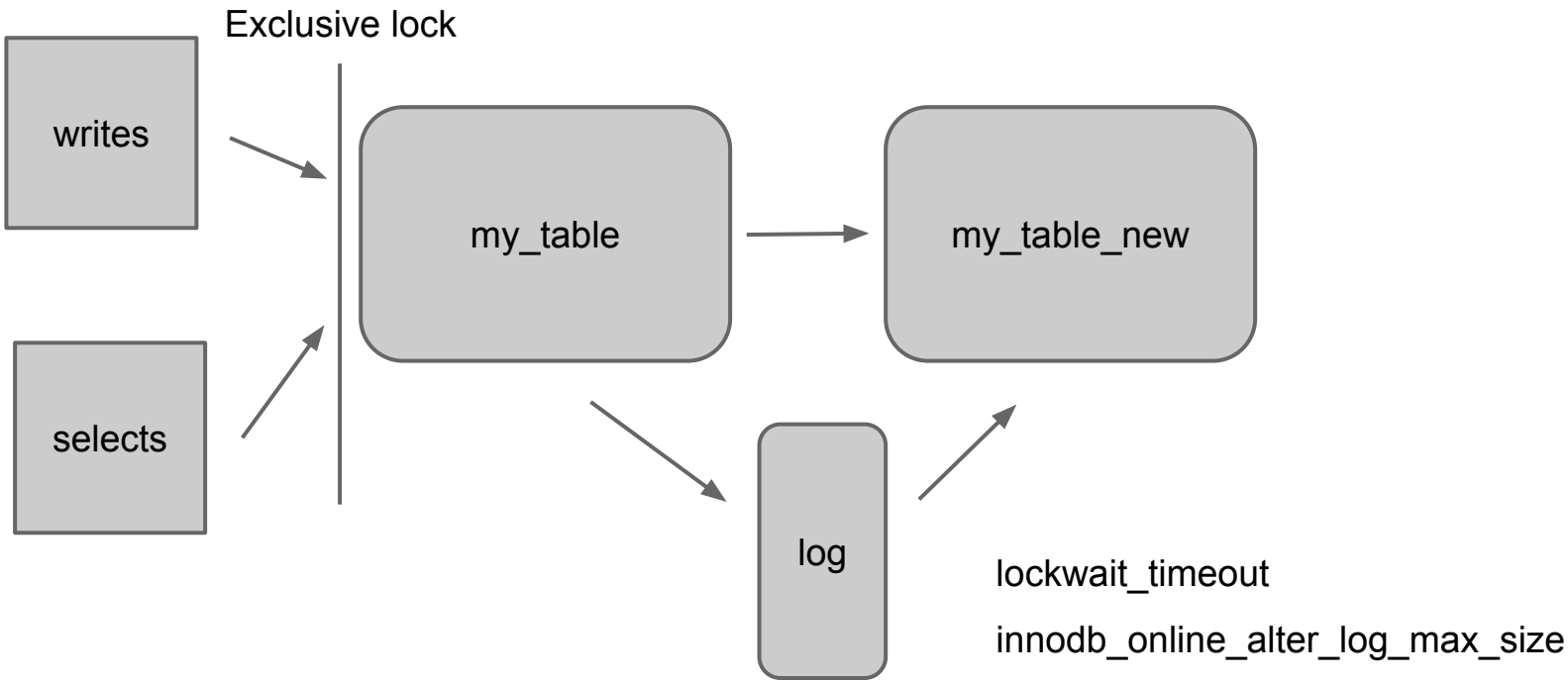
MySQL Online DDL



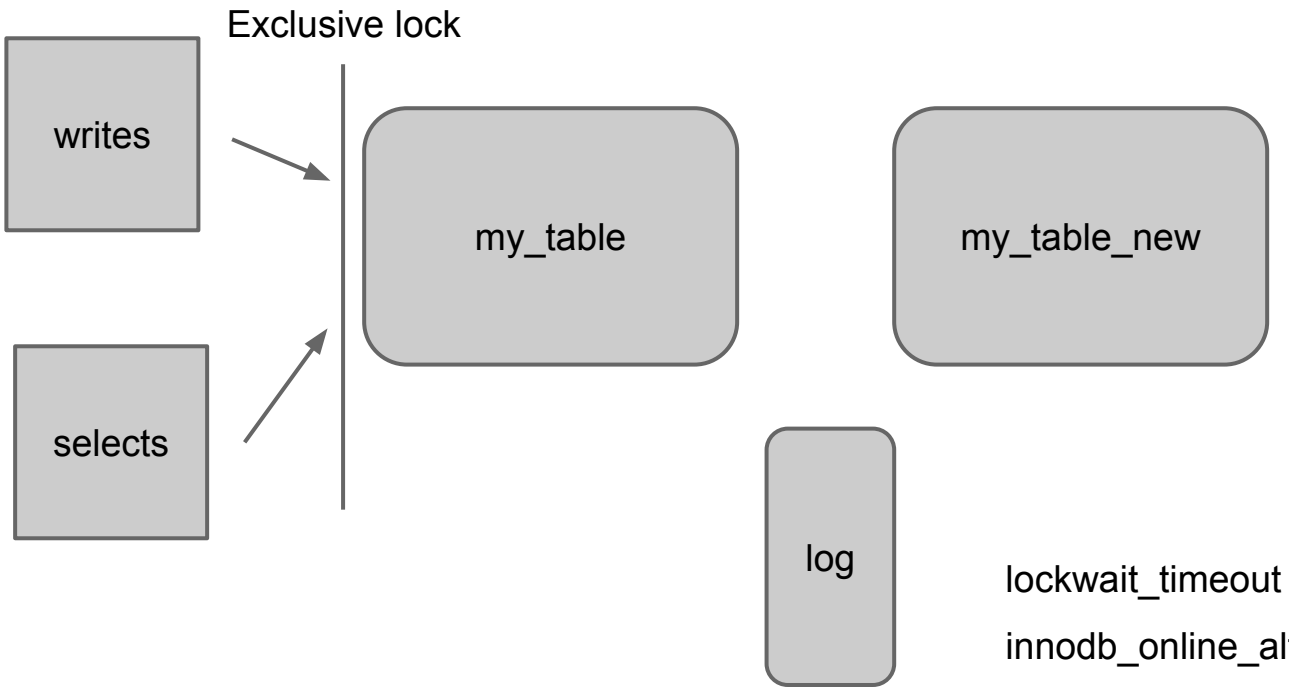
lockwait_timeout

innodb_online_alter_log_max_size

MySQL Online DDL



MySQL Online DDL



lockwait_timeout

innodb_online_alter_log_max_size

Why MySQL Online DDL

- Int to bigint
- Alters performed on slaves
  - Zero tolerance for data loss
    - Recoveries can suck
    - Table checksums
  - RBR and triggers
  - Less io than other tools
  - Avoid additional load on master
    - no throttling parameters
  - All IOD operations block the sql_thread on the slaves
  - 5.6.17 defrags and algorithm=inplace

Alter online lock modes

- Exclusive
  - alter table my_tbl engine=innodb, lock=exclusive;
- Shared
  - alter table my_tbl engine=innodb, lock=shared;
- Default
  - Know your stuff and test
- None
  - alter table my_tbl add index sk1(pid), lock=none;

Alter online algorithms

- Inplace
  - alter table my_tbl add column, algorithm=inplace, lock=none;
- Copy
  - alter table my_tbl engine=innodb, algorithm=copy, lock=shared;
  - all algorithm=copy allow only shared or exclusive lock.

Family Feud - Operations

- show process list contains only sleeping connections. Am I clear for alters?

More examples

```
# Defragging a table requires a copy and  does not allow concurrent dml prior to 5.6.17.
alter table ${table} engine=innodb, algorithm=copy, lock=shared;

# Index ops
alter table ${table} add index sk1 (zone), algorithm=inplace, lock=none;
alter table ${table} drop index sk1, algorithm=inplace, lock=none;

# Column ops
alter table ${table} add column c3 int, add column c1 int, add column c4 int, algorithm=inplace, lock=none;
alter table ${table} drop column c4,algorithm=inplace, lock=none;
alter table ${table} modify c1 int default 0, algorithm=inplace, lock=none;
alter table ${table} change c3 c2 int, algorithm=inplace, lock=none;
alter table ${table} modify column c2 int after c1, algorithm=inplace, lock=none;
alter table ${table} modify column c2 int default null, algorithm=inplace, lock=none;
alter table ${table} modify c2 bigint, algorithm=copy, lock=shared;
alter table ${table} drop column c1, drop column c2 , algorithm=inplace, lock=none;
```

Innodb online DDL: additional info

- Foreign keys
  - Create fk with foreign_key_checks=0
  - Child table restrictions
    - Alter on child blocks concurrent dml
    - Alter on child could be blocked by transaction on the parent*
    - Alter on completion on parent could be blocked by dml on same parent if it causes dml on child.
- Partitioning
- Auto increment - alter instantaneous, add requires lock=shared
- A table of info: http://dev.mysql.com/doc/refman/5.6/en/innodb-create-index-overview.html
- Full text indexes still require shared lock

Innodb online DDL variables

- innodb_online_alter_log_max_size
- lock_wait_timeout
- innodb_sort_buffer_size
- others(benchmark)

Innodb Crash Recovery

- Secondary indexes
- Clustered indexes(don’t crash!)

Summary

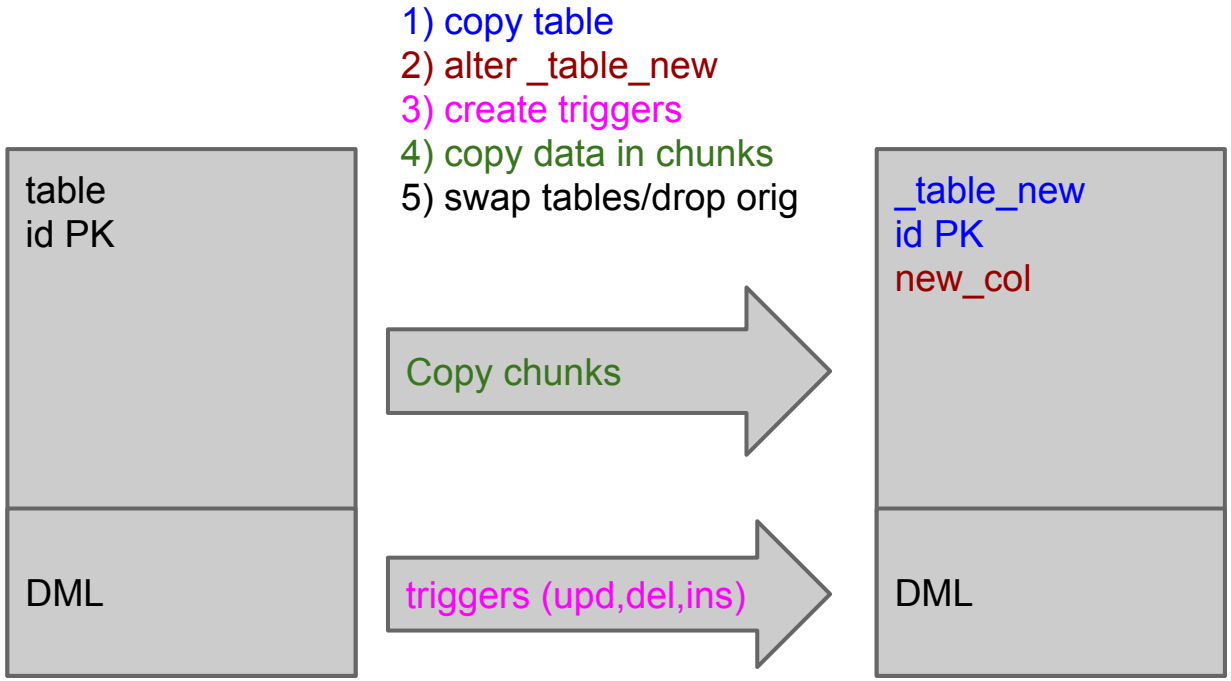
- Backup, recover, and benchmark
- Upgrade to 5.6.17

You want to do what to that 300G table?

- App down for Maintenance/blocking DDL
- Disable writes/read-only for altered table
- Alter slave and swap masters
- MySQL Online DDL and serialized replication/lag

OSC Tools or how I learned to love the alter...

- copy original table structure to new table
- alter new table
- create triggers to copy dml from original table to new table
- copy data in original table to new table in chunks
- swap table names and drop original table


1) copy table
2) alter _table_new
3) create triggers
4) copy data in chunks
5) swap tables/drop orig
table
id PK
DML
Copy chunks
triggers (upd,del,ins)
_table_new
id PK
new_col
DML

pt-online-schema-change

Percona toolkit is your friend.

Open source and you can see the tables, sql, triggers and what it is doing.

Pay attention to version you are using!!!

FK issues, log_bin with 2.0

Whoomp! There it is!

time ./pt-online-schema-change --dry-run --user=bblack --ask-pass --max-lag=2 --check-interval=1 --progress=time,10 --max-load Threads_running:50 --critical-load Threads_running:6000 --alter "DROP COLUMN col4" --host=127.0.0.1 D=my_db, t=my_table --alter-foreign-keys-method=drop_swap --recursion-method dsn=D=percona,t=dsns --no-check-replication-filters --set-vars innodb_lock_wait_timeout=10 --chunk-time=.2

screen - in case you lose connection

time - test when possible for timing

It's all ball bearings nowadays

--progress time,10 (default 30 seconds)

--max-lag and --check-interval (both default 1s)

--recursion-method (how to find slaves)

show processlist

show hosts

dsn table (great for ignoring some slaves)

Other params

--chunk-time=.2 (default .5)

--max-load and --critical-load (Threads_running=25,50)

--alter-foreign-keys-method=drop_swap

--no-check-replication-filters

--no-drop-old-table (for RDS or EXT)

What could possibly go wrong?

- PK/UK required
- FK names will change on altered table
- FK's reference table to alter

--alter-foreign-keys-method drop_swap

sets foreign_key_checks=0

drops original table (hardlink!!! - EXT)

renames new table

- NOTIFICATION EMAILS (don’t kick off on Friday and go home)

What could possibly go wrong?

- non xfs table drops (create the hardlink!)
  - Can even cause innodb to crash mysqld
- PKs with gaps greatly affect time estimates
- Largest table to alter vs free disk space
- Disk space (2x free needed for RBR)
- Global mutexes (table drops can take several seconds)
- table metadata locks (triggers)
- No triggers allowed on the table to alter

NOOOOOO!!

- Running PT-OSC against a slave with RBR
  Replication started erroring after pt-osc

why???

- And how about syncing a table using pt-osc with RBR?

Trust but verify…

show create table altered_table\G

show create table _altered_table_new\G

show table status like ‘%altered_table%’;
ls -alh /data/schema/*altered_table*.ibd

Set it and forget it???

- watch
  - w
  - df -h
  - ls -alh ibd files
  - mysql -e"show processlist;"|egrep -v "Sleep|repl"
  - slave lag
- How is it affecting the application? external monitoring/app response time

I think I’m paranoid

```
watch -n 1 'w|head -1;echo "";mysql -e"show
engine innodb status\G"|grep -i history;echo "";
df -h|grep local;echo "";mysql -e"show
processlist;"|wc -l;echo "";mysql -e"show
processlist;"|grep -v Sleep|grep -v repl'
```

The kids aren’t alright

[01:11 root@host ~]$ cat /home/bblack/chech_repl.sh
echo "master001" `mysql -hmaster001 -e"show slave status\G"|grep Seconds`
echo "slave001" `mysql -hslave001 -e"show slave status\G"|grep Seconds`

[01:14 bblack@host ~]$ watch ./chech_repl.sh
master001 Seconds_Behind_Master: 0
slave001 Seconds_Behind_Master: 0

Slaves on non-3306? Use either “hosts” recursion method with report-host set
OR - set up an ssh tunnel on 3306 that redirects

Are we there yet?

pt-online-schema-change prints progress to std err

mysql> show table status like ‘%table_name%’;

ls -alh /data/schema/*table_name*.ibd

mysql> select max(id) from table_name;
mysql> show processlist; -- to find id of current chunk

Sneaky little hobbitses. Wicked, tricksy

To do a noop and optimize the table

--alter "engine=INNODB"

100-200MB cutoff for blocking alters

Overhead for adding indexes

Never do pt-osc for dropping indexes

We’re Hiring!

- Dropbox - see us at our booth
  - dturner@dropbox.com
- Tango
  - bblack@tango.me