Booking.com

Riding the Binlog: an in Deep Dissection of the Replication Stream

Jean-François Gagné
jeanfrancois DOT gagne AT booking.com

Presented at Percona Live Amsterdam 2015

Booking.com

Booking.com’

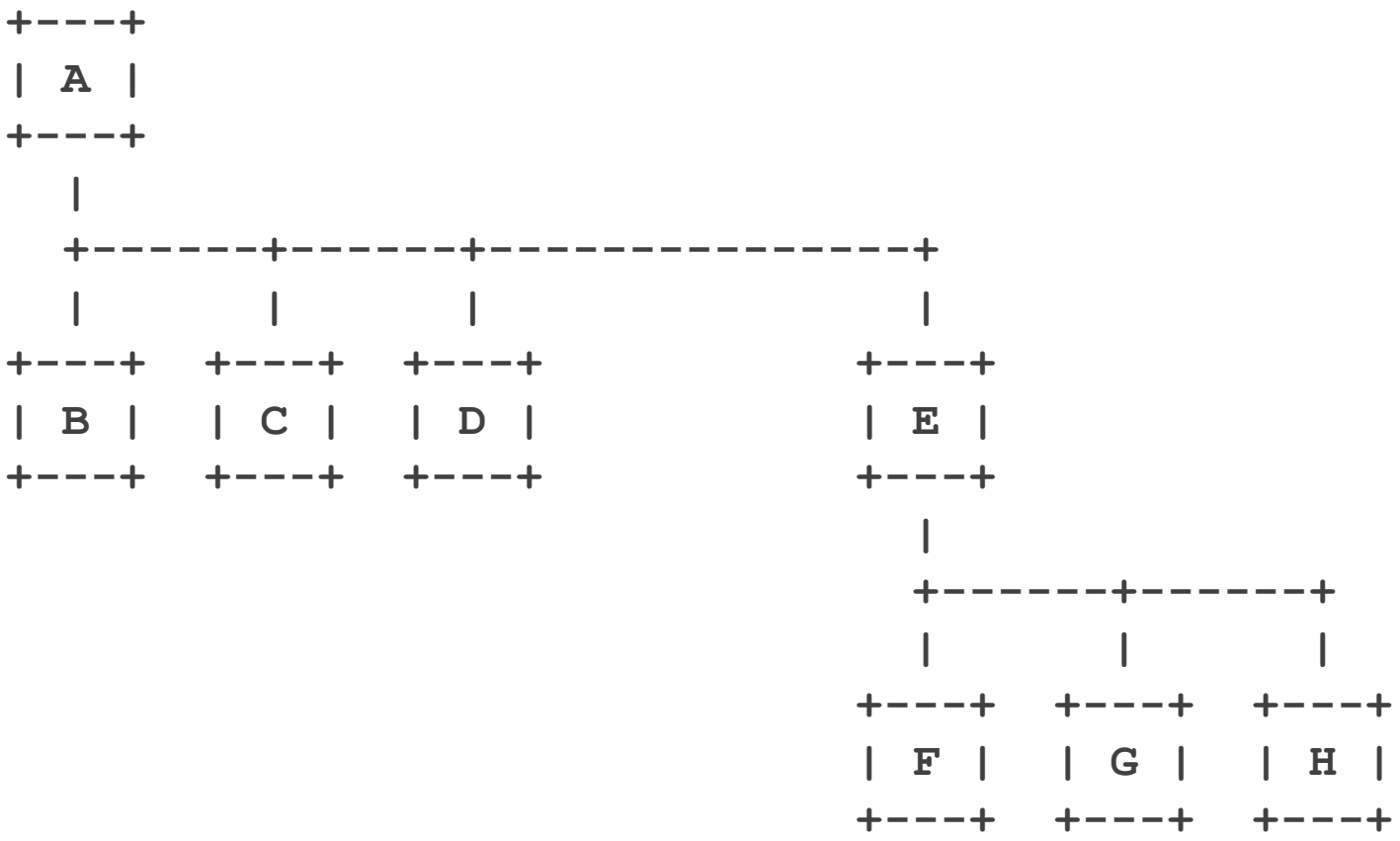
- Based in Amsterdam since 1996
- Online Hotel and Accommodation Agent:
  - 170 offices worldwide
  - +795.000 properties in 221 countries
  - 42 languages (website and customer service)
- Part of the Priceline Group

- And we use MySQL:
  - Thousands (1000s) of servers, ~85% replicating
  - >110 masters: ~25 >50 slaves & ~8 >100 slaves

Riding the Binlog: Replication

- One master / one or more slaves
- The master records all writes in a journal: *the binary logs*
- Each slave:
  - Downloads the journal and saves it locally (IO thread): *relay logs*
  - Executes the relay logs on the local database (SQL thread)
  - Could produce binary logs to be itself a master (log-slave-updates)
- Understood replication features:
  - Asynchronous
  - Single threaded

Riding the Binlog: Replication'



Booking.com

Riding the Binlog: Replication”

- More understood binlog/replication features:
  - Master side binary log filtering
    - Binlog_Do_DB / Binlog_Ignore_DB
      (avoid those in favor of slave side filtering)
    - sql_log_bin
  - Slave side filtering:
    - Replicate_Do_DB / Replicate_Ignore_DB
    - Replicate_Do_Table / Replicate_Ignore_Table
    - Replicate_Wild_Do_Table / Replicate_Wild_Ignore_Table
  - Circular replication
- New innovations in replication land:
  - Global Transaction Identifiers (GTIDs)
  - Multi-Source and Multi-Threaded Replication (many types)
  - Binlog Servers

Riding: the Binary Logs (on master)

```
> ls -l # Dots (.) in the file sizes are added for readability.
[...]
-rw-rw---- 1 mysql mysql  537.362.886 Sep 13 10:19 binlog.000878
-rw-rw---- 1 mysql mysql  537.341.120 Sep 13 10:22 binlog.000879
-rw-rw---- 1 mysql mysql 6431.760.382 Sep 13 11:13 binlog.000880
-rw-rw---- 1 mysql mysql  537.394.773 Sep 13 11:26 binlog.000881
-rw-rw---- 1 mysql mysql  537.288.989 Sep 13 11:29 binlog.000882
-rw-rw---- 1 mysql mysql  111.419.262 Sep 13 11:30 binlog.000883
-rw-rw---- 1 mysql mysql   10.411.193 Sep 13 11:30 binlog.000884
-rw-rw---- 1 mysql mysql  537.271.616 Sep 13 11:34 binlog.000885
-rw-rw---- 1 mysql mysql  537.290.232 Sep 13 11:38 binlog.000886
[...]
```

- Let's use `mysqlbinlog` to see what is in those files.

Riding: the Binary Logs'

```
mysql plams <<< "CREATE TABLE test1 (\
 my_pk BIGINT AUTO_INCREMENT,\
 my_bigint BIGINT DEFAULT NULL,\
 my_datetime DATETIME DEFAULT NULL,\
 PRIMARY KEY(my_pk));"

# at 217
#150919 12:04:27 server id 1 end_log_pos 428 CRC32 0xebef2e40\
Query   thread_id=3100  exec_time=0     error_code=0
use `plams`
SET TIMESTAMP=1442657067
CREATE TABLE test1 ( my_pk BIGINT AUTO_INCREMENT, my_bigint
BIGINT DEFAULT NULL, my_datetime DATETIME DEFAULT NULL,
PRIMARY KEY(my_pk))
```

Riding: the Binary Logs”

```
mysql plams <<< "INSERT INTO test1(my_pk) VALUES (1);"

# (Omitting “SET TIMESTAMP=...”.)

# at 428
#150919 12:04:30 server id 1  end_log_pos 509 ... Query ...
BEGIN
# at 509
#150919 12:04:30 server id 1  end_log_pos 620 ... Query ...
INSERT INTO test1(my_pk) VALUES (1)
# at 620
#150919 12:04:30 server id 1  end_log_pos 651 ... Xid = ...
COMMIT
```

Riding: the Binary Logs'''

```
# PK (auto_increment) not specified.
mysql plams <<< "INSERT INTO test1(my_bigint, my_datetime) \
VALUES (floor(1024*1024*rand()), now());"

# (Omitting BEGIN/COMMIT.)

# at 740 / # at 772 / # at 811
#150919 12:04:38 server id 1  end_log_pos 772 ... Intvar
SET INSERT_ID=2
#150919 12:04:38 server id 1  end_log_pos 811 ... Rand
SET @@RAND_SEED1=80977572, @@RAND_SEED2=154388004
#150919 12:04:38 server id 1  end_log_pos 976 ... Query ...
SET TIMESTAMP=1442657078
INSERT INTO test1(my_bigint, my_datetime) VALUES
(floor(1024*1024*rand()), now())
```

Riding: the Binary Logs’’’ ’

```
mysql plams <<< "
BEGIN;
INSERT INTO test1(my_datetime) VALUES (now());
SELECT sleep(5);
INSERT INTO test1(my_datetime) VALUES (now());
COMMIT;" &

mysql plams <<< "
SELECT sleep(1);
INSERT INTO test1(my_datetime) VALUES (now());"
```

Riding: the Binary Logs”’ ”

```
# (Omitting BEGIN/COMMIT/“SET TIMESTAMP=...”.)

# at 1096 / # at 1128 / SET INSERT_ID=4
#150919 12:05:00 server id 1  end_log_pos 1257 ... Query ...
INSERT INTO test1(my_datetime) VALUES (now())

# at 1377 / # at 1409 / SET INSERT_ID=3
#150919 12:04:58 server id 1  end_log_pos 1538 ... Query ...
INSERT INTO test1(my_datetime) VALUES (now())
# at 1538 / # at 1570 / SET INSERT_ID=5
#150919 12:05:03 server id 1  end_log_pos 1699 ... Query ...
INSERT INTO test1(my_datetime) VALUES (now())
```

Riding: the Binary Logs”’ ”’

```
mysql plams <<< "SET binlog_format = ROW;
INSERT INTO test1(my_datetime) VALUES (now());"

# (Omitting BEGIN/COMMIT.)

# at 1811
#... Table_map: `plams`.`test1` mapped to number 143
# at 1863
#... Write_rows: table id 143 flags: STMT_END_F

BINLOG '
VTP9VRMBAAAANAAAAEcHAAAAAI8AAAAAAAEABXBsYW1zAAV0ZXN0MQADCAgSAQAGVr3KSw==
VTP9VR4BAAAAMQAAAHgHAAAAAI8AAAAAAAEAAgAD//oGAAAAAAAAAJmXJsFJmHAAYw==
```

Riding: the Binary Logs”’ ”’

```
# mysqlbinlog -v
# (Omitting BEGIN/COMMIT.)

# at 1811
#... Table_map: `plams`.`test1` mapped to number 143
# at 1863
#... Write_rows: table id 143 flags: STMT_END_F

BINLOG '
VTP9VRMBAAAANAAAAEcHAAAAAI8AAAAAAAEABXBsYW1zAAV0ZXN0MQADCAgSAQAGVr3KSw==
VTP9VR4BAAAAMQAAAHgHAAAAAI8AAAAAAAEAAgAD//oGAAAAAAAAAJmXJsFJmHAAYw==

### INSERT INTO `plams`.`test1`
### SET
###   @1=6
###   @2=NULL
###   @3='2015-09-19 12:05:09'
```

Riding: the Binary Logs”’ ”’

```
# mysqlbinlog -vv
# (Omitting BEGIN/COMMIT.)

# at 1811
#... Table_map: `plams`.`test1` mapped to number 143
# at 1863
#... Write_rows: table id 143 flags: STMT_END_F

BINLOG '
VTP9VRMBAAAANAAAAEcHAAAAAI8AAAAAAAEABXBsYW1zAAV0ZXN0MQADCAgSAQAGVr3KSw==
VTP9VR4BAAAAMQAAAHgHAAAAAI8AAAAAAAEAAgAD//oGAAAAAAAAAJmXJsFJmHAAYw==

### INSERT INTO `plams`.`test1`
### SET
###   @1=6 /* LONGINT meta=0 nullable=0 is_null=0 */
###   @2=NULL /* LONGINT meta=0 nullable=1 is_null=1 */
###   @3='2015-09-19 12:05:09' /* DATETIME(0) meta=0 nullable=1 is_null=0 */
```

Riding: the Binary Logs’’’ ’’’ ’

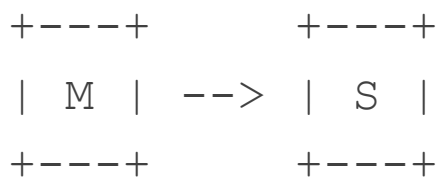
- Binary Logs: Journal for MySQL Replication
  - Files with a basename and 6 digit index
  - Stream of transactions in commit order
  - A transaction is never split in 2 binlog files
  - But a transaction is split in many events
- Want to know more:
  - Read the documentation
  - Or source dive: `sql/log_event.{h,cc}`

Riding: some Event Types

sql/log_event.h

```
enum Log_event_type {
  UNKNOWN_EVENT= 0,
  QUERY_EVENT= 2,
  STOP_EVENT= 3,
  ROTATE_EVENT= 4,
  INTVAR_EVENT= 5,
  LOAD_EVENT= 6,
  CREATE_FILE_EVENT= 8,
  APPEND_BLOCK_EVENT= 9,
  EXEC_LOAD_EVENT= 10,
  DELETE_FILE_EVENT= 11,
  NEW_LOAD_EVENT= 12,
  RAND_EVENT= 13,
  USER_VAR_EVENT= 14,
  FORMAT_DESCRIPTION_EVENT= 15,
  XID_EVENT= 16,
  TABLE_MAP_EVENT = 19,
  INCIDENT_EVENT= 26,
  HEARTBEAT_LOG_EVENT= 27,
  IGNORABLE_LOG_EVENT= 28,
  ROWS_QUERY_LOG_EVENT= 29,
  WRITE_ROWS_EVENT = 30,
  UPDATE_ROWS_EVENT = 31,
  DELETE_ROWS_EVENT = 32,
  GTID_LOG_EVENT= 33,
  ANONYMOUS_GTID_LOG_EVENT= 34,
  PREVIOUS_GTIDS_LOG_EVENT= 35,
```

Riding: the Binlogs on Slaves



- Depends on log-slave-updates (lsu):
  - If lsu disabled: no trx from the master in the binlogs of the slave
  - else: SQL Thread logs its transactions
- binlogs on a slave != binlogs on the master:
  - FLUSH BINARY LOGS
  - Restart of mysqld on the master or slave
  - RESET MASTER on slave
  - Different binlog filenames on master and slave
  - …

Riding: the Binlogs on Slaves'

- Binlogs on the master:

```
-rw-rw---- 1 mysql mysql 40720631 Sep 19 13:16 binlog.000001
-rw-rw---- 1 mysql mysql 16777476 Sep 19 13:17 binlog.000002
-rw-rw---- 1 mysql mysql 16777476 Sep 19 13:18 binlog.000003
-rw-rw---- 1 mysql mysql 10124412 Sep 19 13:18 binlog.000004
-rw-rw---- 1 mysql mysql  8388820 Sep 19 13:18 binlog.000005
-rw-rw---- 1 mysql mysql  8388820 Sep 19 13:19 binlog.000006
-rw-rw---- 1 mysql mysql  6964108 Sep 19 13:19 binlog.000007
[...]
```

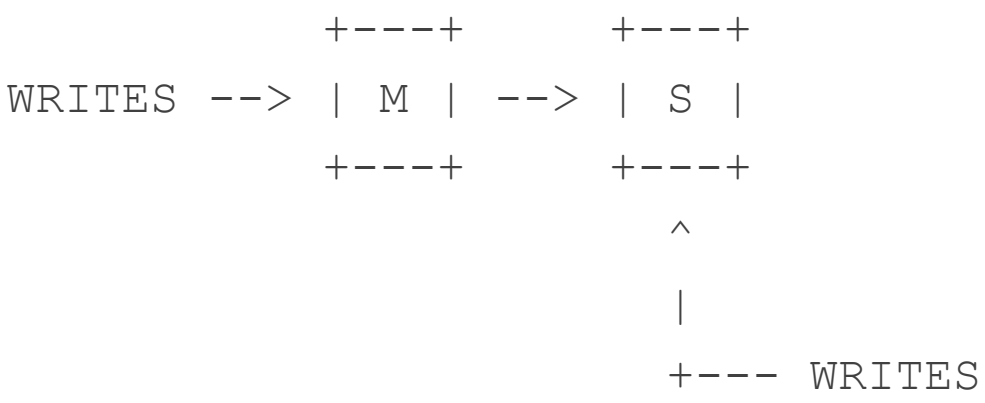
- And on a slave:

```
-rw-rw---- 1 mysql mysql 67109007 Sep 19 13:21 binlog.000001
-rw-rw---- 1 mysql mysql 67108948 Sep 19 13:22 binlog.000002
-rw-rw---- 1 mysql mysql 52518911 Sep 19 13:22 binlog.000003
-rw-rw---- 1 mysql mysql 45279097 Sep 19 13:30 binlog.000004
```

➢ The content of the binary logs on slaves is (hopefully) the same as on the master, but the transport is different.

Riding: the Binlogs on Slaves”

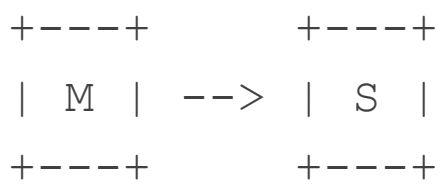
- Depends on slave-local transactions (and lsu):
  - If lsu disabled: the slave behaves like a master
    - The slave has its own transaction/binlog stream
    - This stream is independent from the master stream (lsu disabled)
  - else: local trx UNION master trx (multiplexing)
    - The slave transaction stream merges with the master stream
    - Those might be independent, or might depend on each other



Riding: the Binlogs on Slaves”’

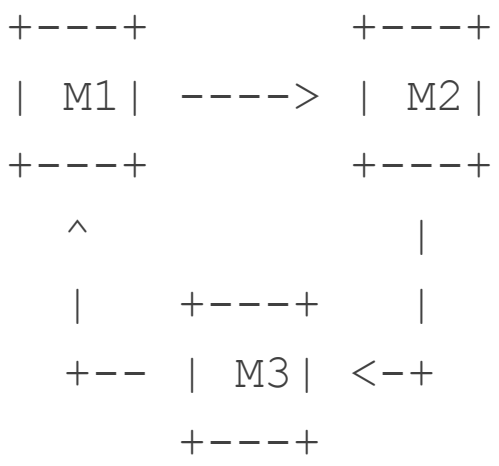
- If lsu enabled, also depends on replication filters:
  - If no filter: binlogs on slave contain local and master trx
  - else: depends of MySQL 5.6 GTIDs:
    - Without 5.6 GTIDs, filtered transaction are skipped (exclusion)
    - With 5.6 GTIDs: an empty transaction is logged (mutation)

  (Transactions can be filtered fully or partially : also mutation)



Riding: the Binlogs on Slaves”’ ’

- Circular-replication (needs log-slave-updates):
  - Combination of slave local transactions and filters
  - local trx UNION (master trx SUB “same server_id” trx) (multiplexing with exclusion)

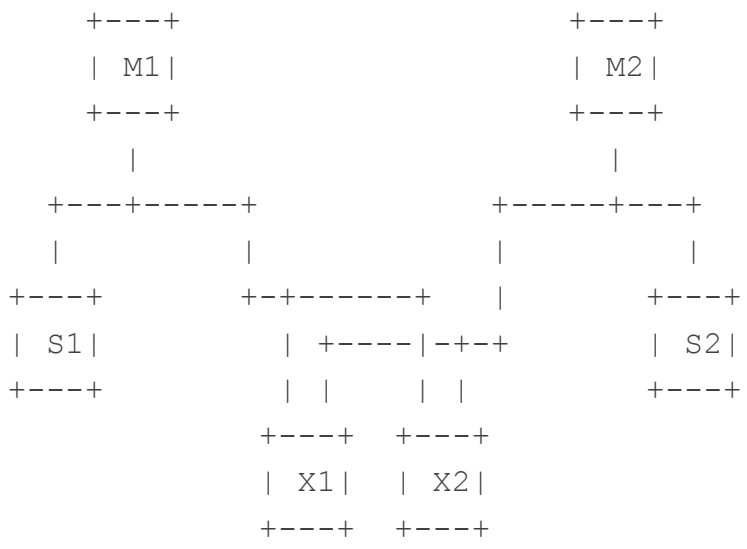


Riding: the Binlogs on Slaves”’ ”

- Depends on Multi-Source Replication:
  - If lsu disabled: only slave local transactions
  - else: local trx UNION M1 trx UNION M2 trx … (multiplexing)

(Identically configured slave might multiplex in different ways)

(Filters can be added to multi-source → mutation)



Booking.com

Riding: the Binlogs on Slaves”’ ”’

- Depends on Multi-Threaded Replication (MySQL 5.6 schema based):
  - If lsu disabled: only slave local transactions
  - else: reordering of transactions (morphing)
- On the master, some transactions in 3 schemas (A, B, C):
  - Order in the Binary Logs: `A1, A2, B1, B2, C1, A3, C2, B3, C3`
- On the slave, transactions in different schema run in //: “`A1,A2,A3`” in // with “`B1,B2,B3`” in // with “`C1,C2,C3`”
  - One possible order: `A1, B1, C1, A2, B2, C2, A3, B3, C3`
  - Another: `A1, C1, A2, C2, A3, C3, B1, B2, B3` (if `B1` is big)
  - Many others…

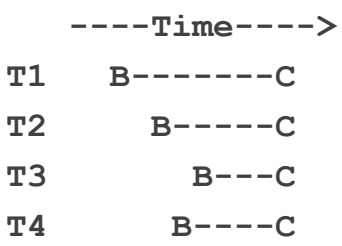
Riding: the Binlogs on Slaves’’’ ’’’ ’

- Depends on MySQL 5.7 Logical Clock Parallel Replication:
  - If lsu disabled: only slave local transactions
  - else if slave_preserve_commit_order = OFF (default): reordering of transactions (morphing)
  - else (slave_preserve_commit_order = ON) modification of parallelism information (still morphing)

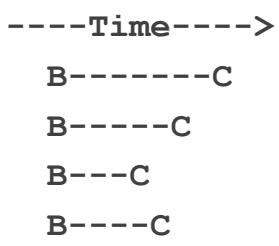
Riding: the Binlogs on Slaves”’ ”’ ”

- Four transactions on X and Y:

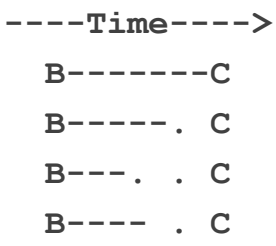
On X:



On Y:



On Y (with slave_preserve_commit_order):



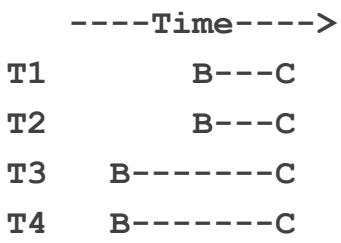
- Transaction order in the binary logs of Y: (without slave_preserve_commit_order)
  - `T3, T4, T2, T1`

Booking.com

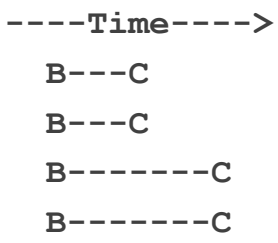
Riding: the Binlogs on Slaves”’ ”’ ”’

- Four other transactions on X and Y:

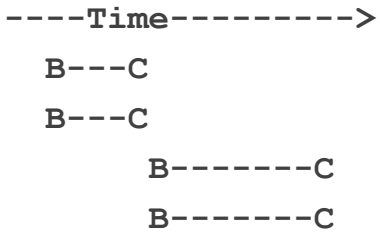
On X:



On Y:



On Z:



- IM (Y) might stall the parallel replication pipeline
- Replicating with IM will slow down parallel replication

Booking.com

Riding: the Binlogs on Slaves’’’ ’’’ ’’’ ’

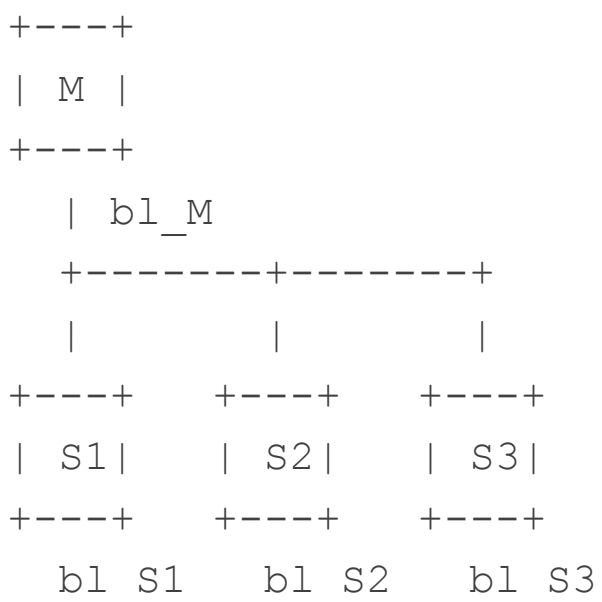
- Let’s try to summarise:
  - Multiplexing:
    - Local transactions + log-slave-updates
    - Circular Replication (needs log-slave-updates)
    - Multi-Source + log-slave-updates
  - Morphing:
    - All types of Multi-Threaded replication + log-slave-updates
  - Mutation:
    - Filters + MySQL 5.6 GTIDs + log-slave-updates
- See the pattern…
  - log-slave-updates

Riding the Binlog: why lsu ?

- Lsu is needed for:
  - master promotion with GTIDs
  - fan-out
  - circular replication
  - external trigger
  - converting SBR to RBR
  - new version/feature testing
  - multiplexing with a side effect

Riding the Binlog: lsu with GTIDs

- After the failure of M, new master promotion needs levelling the remaining slaves



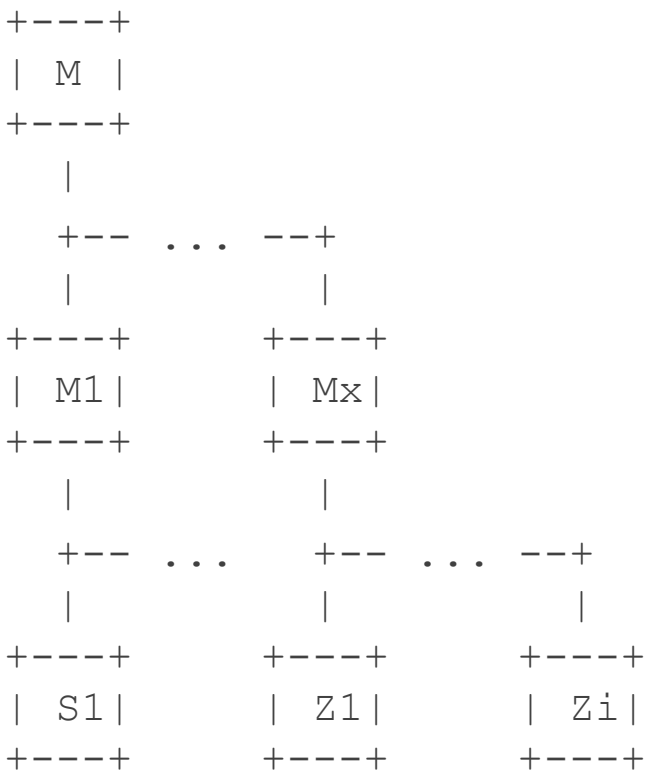
- If S2 is the most up to date slave:
  - bl_S2 will be used to level S1 and S3
- Ok if bl_S2 is a different transport of bl_M
- Ok-ish if bl_S2 is a simple morphing of bl_M
- NOT-Ok if bl_S2 is a mutation of bl_M

➢ If you must replicate using lsu put Intermediate Master between the master and mutation (filtering) slave (or you might loose those slave during promotion)

➢ Or use Binlog Servers

Riding the Binlog: Isu for Fan-Out

- Remote site or too many slaves: needs a fan-out solution

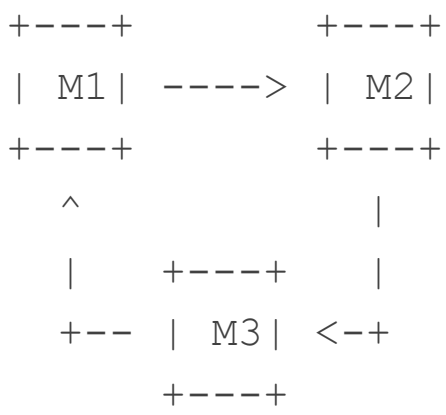


- Intermediate Masters slow down replication (including parallel replication)
- Their failure needs to be managed
- Rogue transactions need to be managed
- And we still have the risks/problems associated with morphing and mutations

➢ IMs come with many drawbacks
➢ Avoiding them might be better
➢ Binlog Servers allow that

Riding the Binlog: Circular Replication

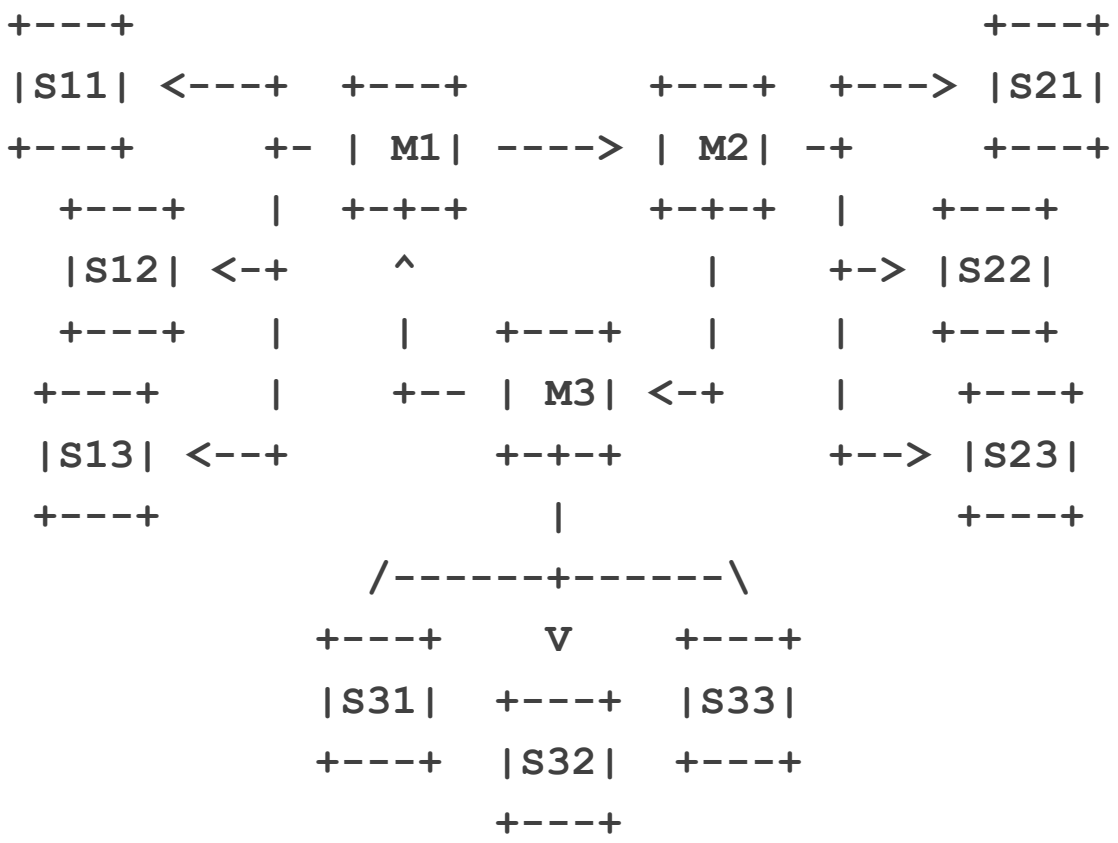
- Circular replication is a form of multiplexing



- But Multi-Source replication is also multiplexing
- ➢ Replace Circular replication by Multi-Source
  - High Availability is tricky
  - But it was not simple either with Circular Replication
  - And it is very simple with Binlog Servers

Riding the Binlog: Circular Replication'

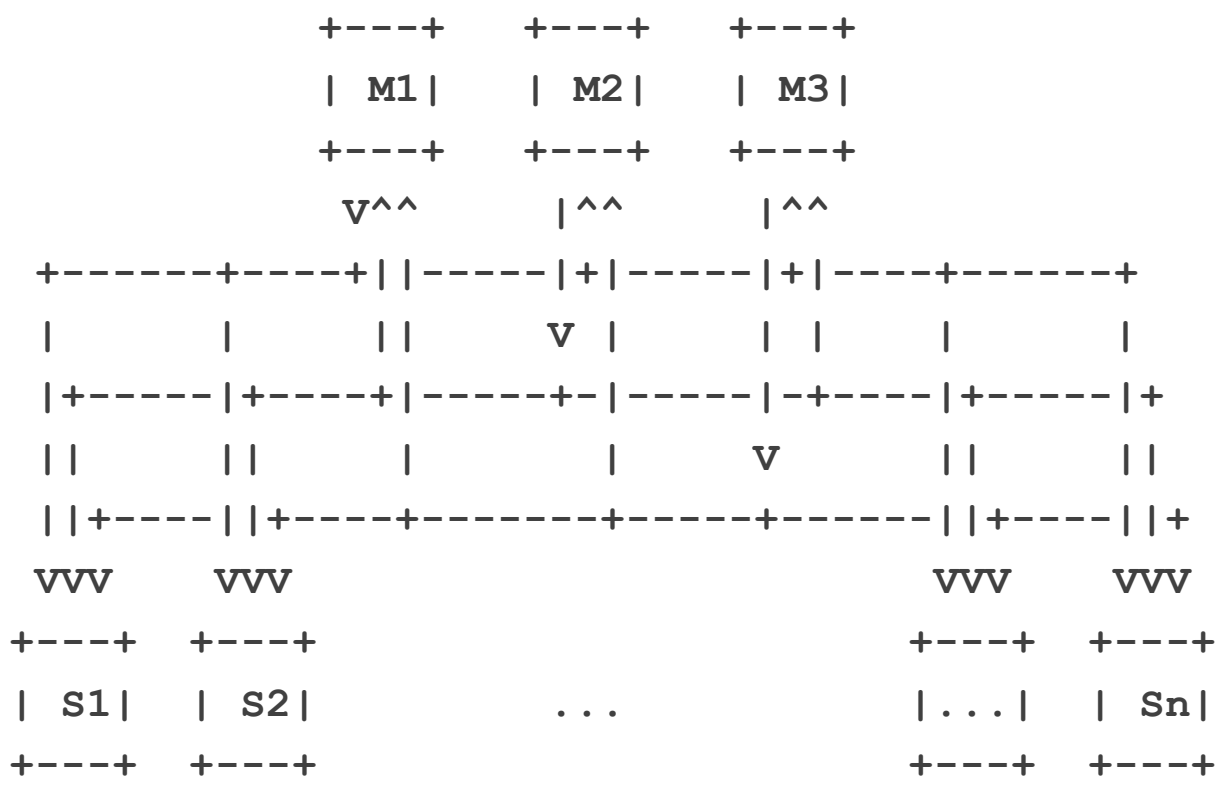
- Replace the following:



Booking.com

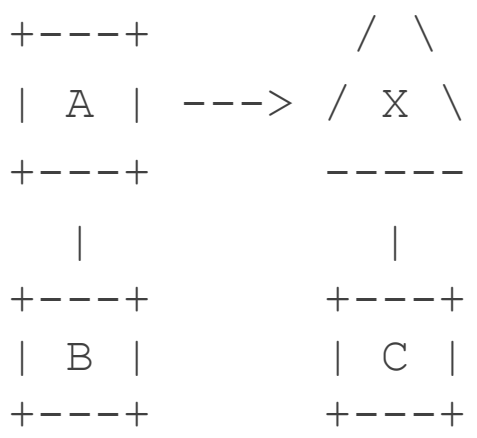
Riding the Binlog: Circular Replication”

- By this:



Riding the Binlog: Binlog Server

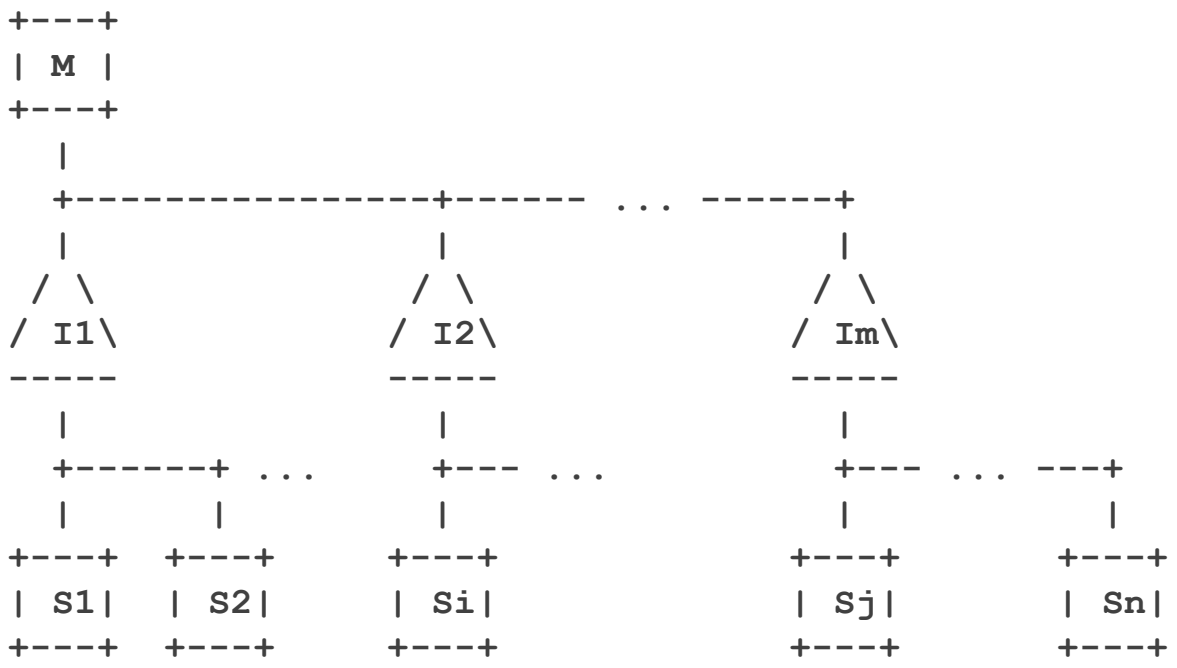
- Binlog Server (*BLS*): is a daemon that:
  - Downloads the binary logs from the master
  - Saves them identically as on the master
  - Serves them to slaves



  - A or X are the same for B and C:
    - By design, the binary logs served by A and X are the same

Riding the Binlog: BLS – Fan-out

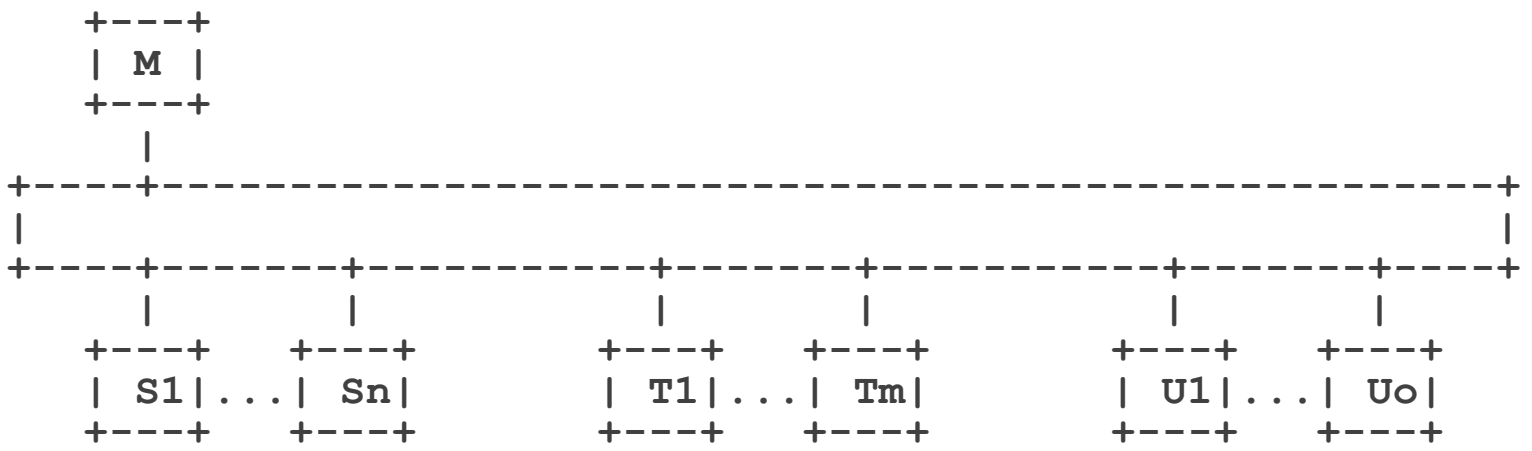
- Replace Intermediate Master by Binlog Servers:



- If BLS fails, repoint slaves to other BLSs (easy by design)

Riding the Binlog: BLS – HA

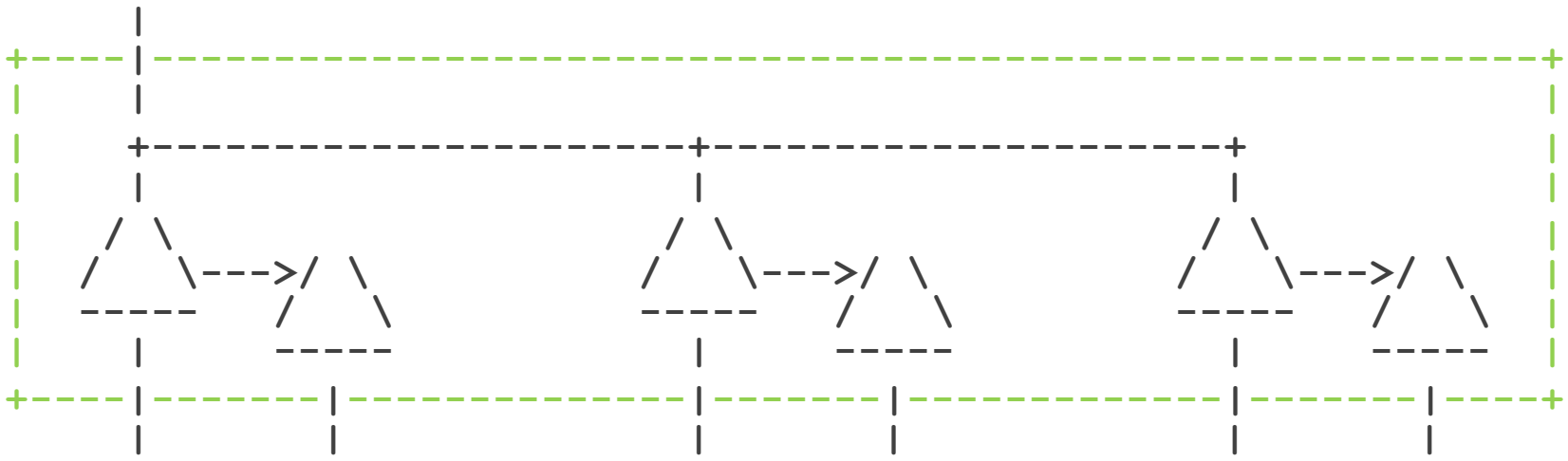
- Distributed Binlog Serving Service (*DBSS*):



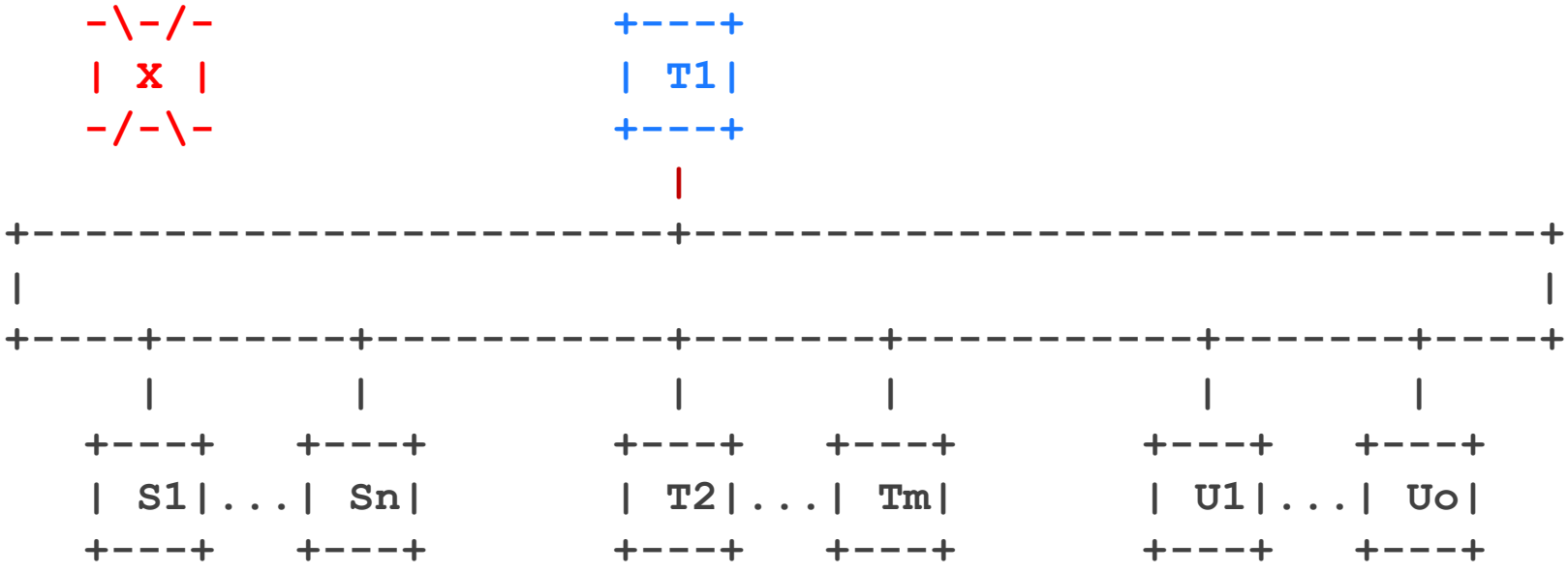
Booking.com

Riding the Binlog: BLS – HA’

- Zoom in DBSS:



Riding the Binlog: BLS – HA”



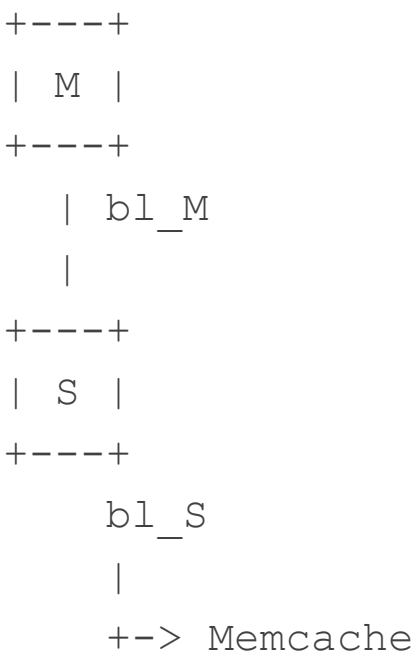
Booking.com

Riding the Binlog: why lsu ?’

- Lsu is needed for:
  - master promotion with GTIDs: use with care or Binlog Servers
  - ~~fan-out~~: Binlog Servers
  - ~~circular replication~~: replaced by Multi-Source + Binlog Servers
  - external trigger
  - converting SBR to RBR
  - new version/feature testing
  - multiplexing with a side effect

Riding the Binlog: external trigger

- If we need to trigger an event after commit on a slave, log-slave-updates is a good solution



- An example is cache invalidation
- But morphing and mutation is still there
- Remember above when tempted to use the slave binlogs for other use-cases (HA)

- Cache invalidation does not need full binlogs
- ➢ A custom binary log format dedicated to that might be useful

Riding the Binlog: SBR to RBR

- When a master is SBR and you need Row-Based Events
  - log-slave-updates is currently your only solution
- But consuming Row-Based Events on a slave introduces morphing, mutation and a single point of failure
- Also, migrating from SBR to RBR is complex
- Wouldn't it be great if:
  - the master logs in both SBR and RBR format,
  - And each slave chooses to follow the SBR or RBR stream.
- I suggest a new binary log format: multiplexed
  - (with the corresponding configuration on slaves)

Riding the Binlog: why lsu ?”

- Lsu is needed for:
  - master promotion with GTIDs: use with care or Binlog Servers
  - ~~fan-out~~: Binlog Server
  - ~~circular replication~~: replaced by Multi-Source + Binlog Servers
  - external trigger (cache invalidation): OK
  - ~~converting SBR to RBR~~: new binary log format
  - new version/feature testing
  - multiplexing with a side effect

Riding the Binlog: why lsu ?”

- Lsu is needed for:
  - master promotion with GTIDs: use with care or Binlog Servers
  - ~~fan-out~~: Binlog Server
  - ~~circular replication~~: replaced by Multi-Source + Binlog Servers
  - external trigger (cache invalidation): OK
  - ~~converting SBR to RBR~~: new binary log format
  - new version/feature testing: ok, not really production
  - multiplexing with a side effect

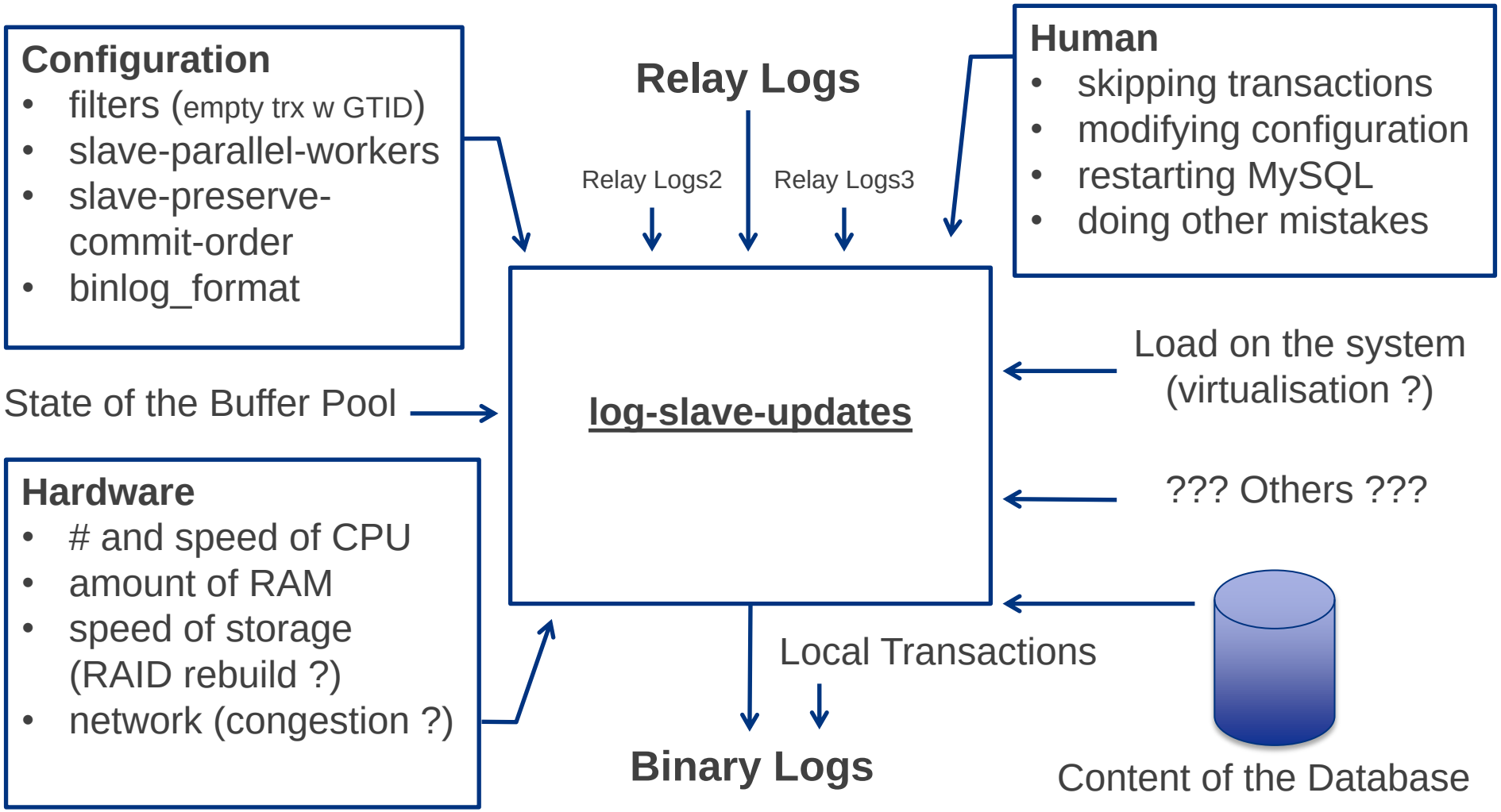
Riding the Binlog: why lsu ?”’

- Lsu is needed for:
  - master promotion with GTIDs: use with care or Binlog Servers
  - ~~fan-out~~: Binlog Server
  - ~~circular replication~~: replaced by Multi-Source + Binlog Servers
  - external trigger (cache invalidation): OK
  - ~~converting SBR to RBR~~: new binary log format
  - new version/feature testing: ok, not really production
  - multiplexing with a side effect:
    very complex use-case, not covered in the talk

Riding the Binlog: Conclusion

- The Binary logs are a stream of transactions
- Enabling log-slave-updates makes this stream evolves
- Those evolutions are complex
- Tips to manage those evolutions in combination with HA:
  - A filtering slave should replicate through an intermediate master
  - A slave with local transaction should also replicate through an IM
  - Avoid using log-slave-updates for replication (use Binlog Servers)
- Learn more tomorrow: Binlog Server at Booking.com
  - Wed. 23 September, 2:10PM - 3:00PM @ Matterhorn 1
  - https://www.percona.com/live/europe-amsterdam-2015/sessions/binlog-servers-bookingcom

Riding: the Binlog Function on Slaves



Riding: the Binlog Function on Slaves

Without log-slave-updates

Local Transactions

Binary Logs

Riding the Binlog: Links

Binlog Servers:

- http://blog.booking.com/mysql_slave_scaling_and_more.html
- http://blog.booking.com/abstracting_binlog_servers_and_mysql_master_promotion_wo_reconfiguring_slaves.html

IM Morphing Loosing Parallelism Information:

- http://blog.booking.com/better_parallel_replication_for_mysql.html
- (http://blog.booking.com/evaluating_mysql_parallel_replication_2-slave_group_commit.html)

Others

- https://www.percona.com/blog/2009/05/14/why-mysqls-binlog-do-db-option-is-dangerous/

Thanks

Jean-François Gagné
jeanfrancois DOT gagne AT booking.com