Binlog Servers at Booking.com

Jean-François Gagné
jeanfrancois DOT gagne AT booking.com

Presented at Oracle Open World 2015

Booking.com

Booking.com’

- Based in Amsterdam since 1996
- Online Hotel and Accommodation Agent:
  - 170 offices worldwide
  - +819.000 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

Binlog Server: Session Summary

1. Replication and the Binlog Server
2. Extreme Read Scaling
3. Remote Site Replication (and Disaster Recovery)
4. Easy High Availability
5. Other Use-Cases
   (Crash Safety, Parallel Replication and Backups)
6. Binlog Servers at Booking.com
7. New master without touching slaves

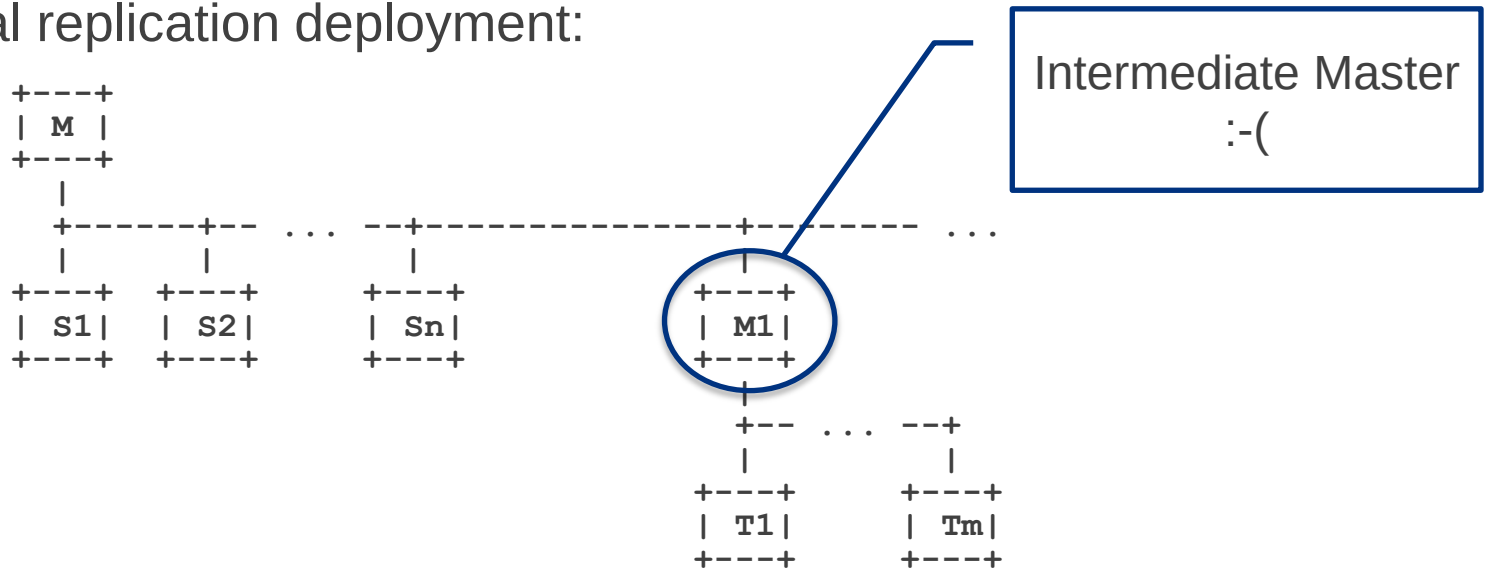
Binlog Server: Replication

- 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)
- Replication is:
  - Asynchronous → lag
  - Single threaded (in MySQL 5.6) → slower than the master

Binlog Server: Booking.com”

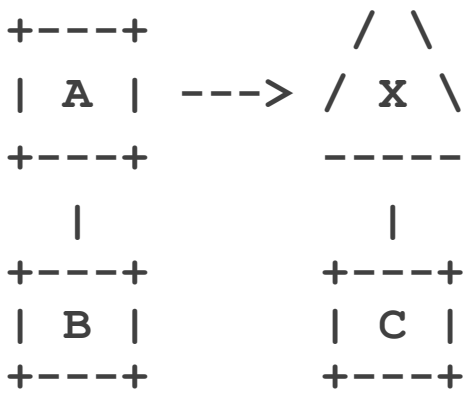
- Typical replication deployment:



  - Si and Tj are for read scaling
  - Mi are the DR master

Binlog Server: What

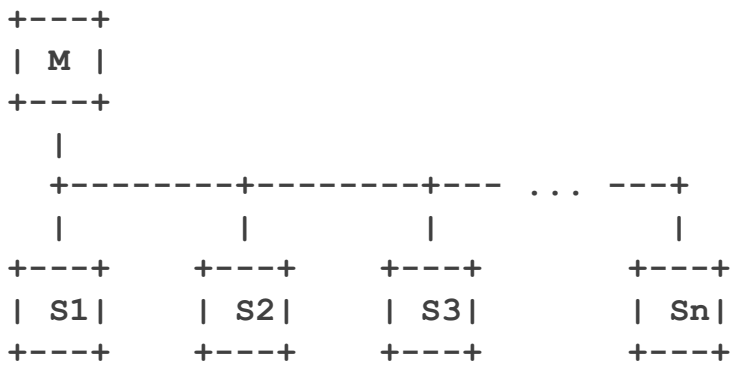
- Binlog Server (*BLS*): is a daemon that:
  - Downloads the binary logs from the master
  - Saves them identically as on the master
  - Serves them to slaves



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

Binlog Server: Read Scaling

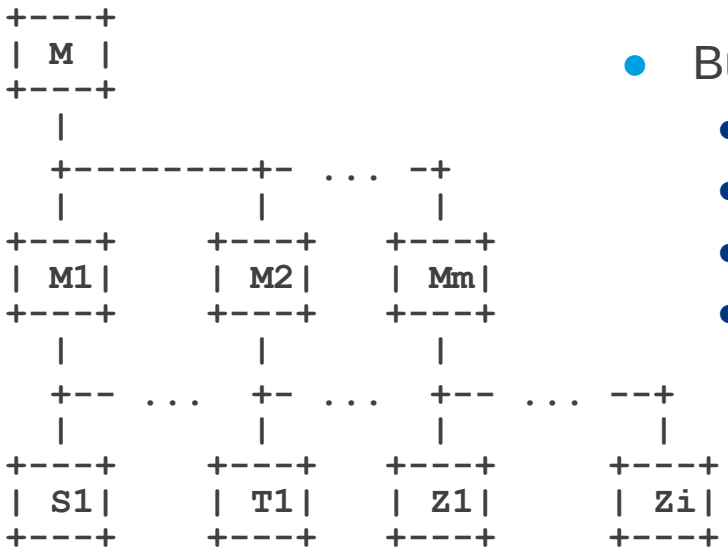
- Typical replication topology for read scaling:



- When there are too many slaves, the network of M is overloaded:
  - 100 slaves x 1Mbit/s: very close to 1Gbit/s
  - OSC or purging data in RBR becomes hard
  - ➢ Slave lag or unreachable master for writes

Binlog Server: Read Scaling’

- Typical solution: fan-out with Intermediary Masters (*IM*):



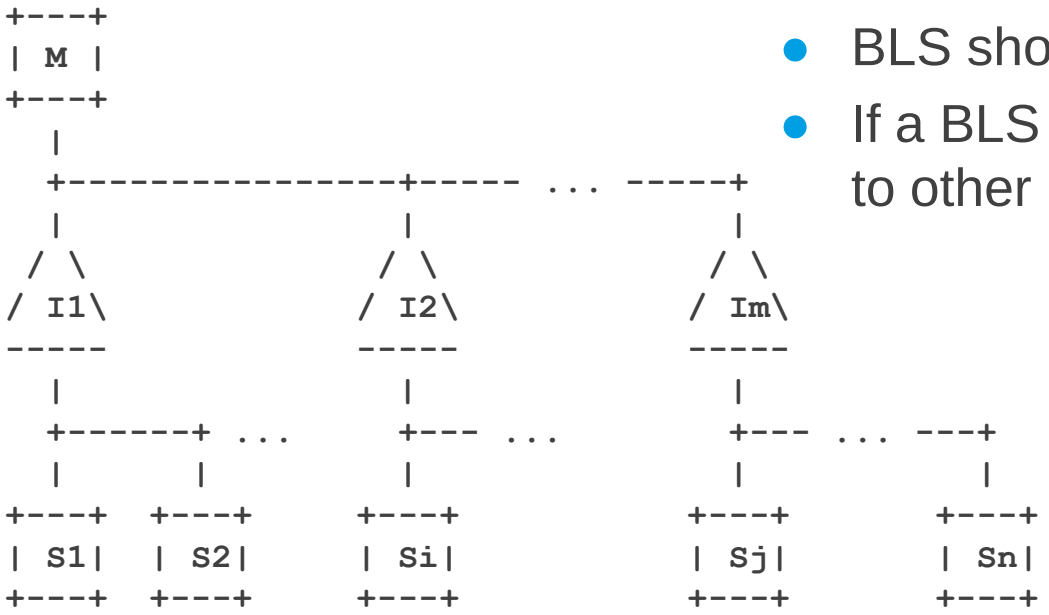
- But Intermediate Masters bring problems:
  - log-slave-updates → IM are slower than slaves
  - Lag of an IM → all its slaves are lagging
  - Rogue transaction on IM → infection of all its slave
  - Failure of an IM → all its slaves stop replicating (and action must be taken fast)

Binlog Server: Read Scaling”

- Solving IM problems with shared disk:
  - Filers (expensive) or DRBD (doubling the number of servers)
  - sync_binlog = 1 + trx_commit = 1 → slower replication → lag
  - After a crash of an Intermediate Master:
    - we need InnoDB recovery → replication on slaves stalled → lag
    - and the cache is cold → replication will be slow → lag
- Solving IM problems with GTIDs:
  - They allow slave repointing at the cost of added complexity :-|
  - But they do not completely solve the lag problem :-(
  - And we cannot migrate online with MySQL 5.6 :-(:-(

Binlog Server: Read Scaling’’’

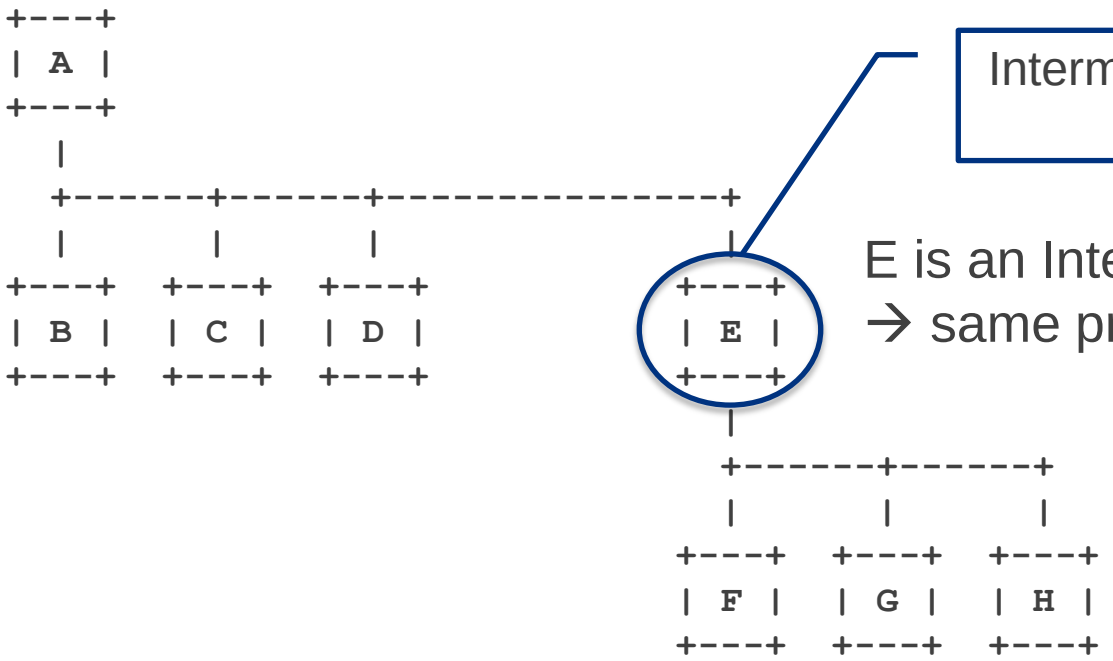
- New Solution: replace IM by Binlog Servers



- BLS should not lag
- If a BLS fails, repoint its slaves to other BLSs (easy by design)

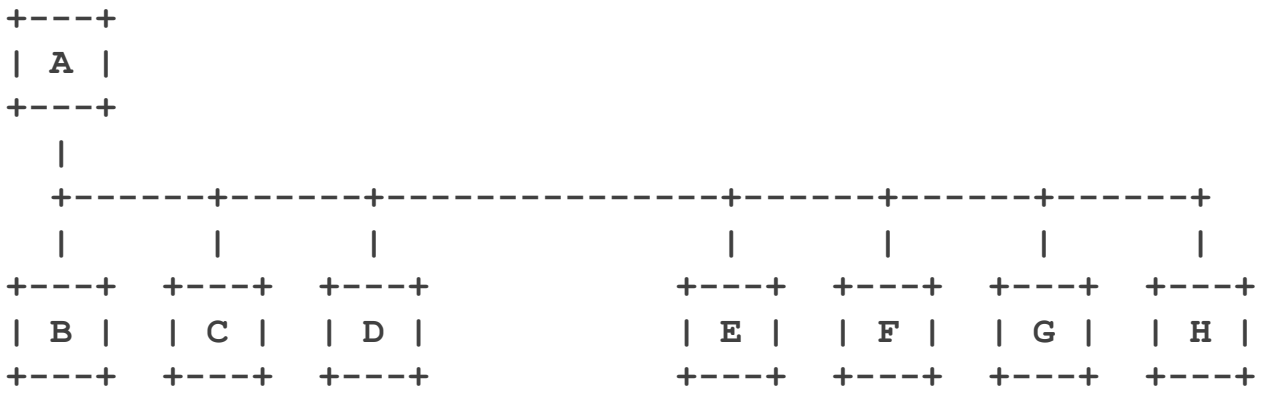
Binlog Server: Remote Site

- Typical deployment for remote site:



Binlog Server: Remote Site’

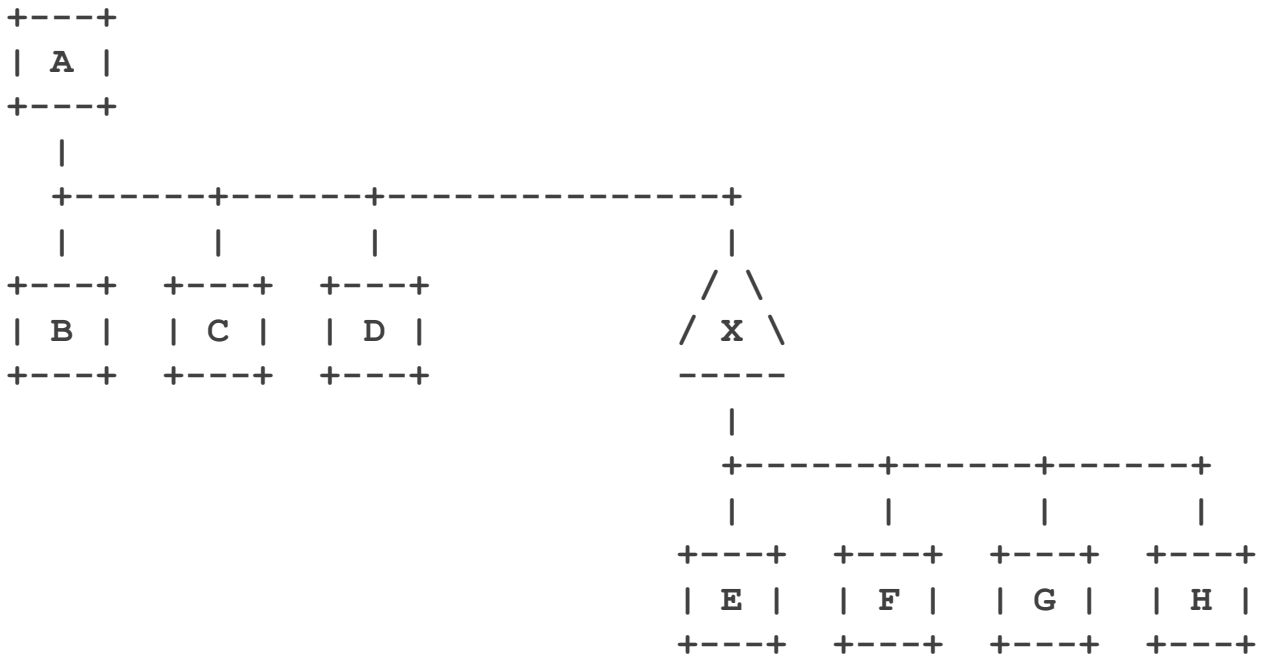
- Ideally, we would like this:



- No lag and no Single Point of Failure (*SPOF*)
- But no master on remote site for writes (easy solvable problem)
- And expensive in WAN bandwidth (harder problem to solve)

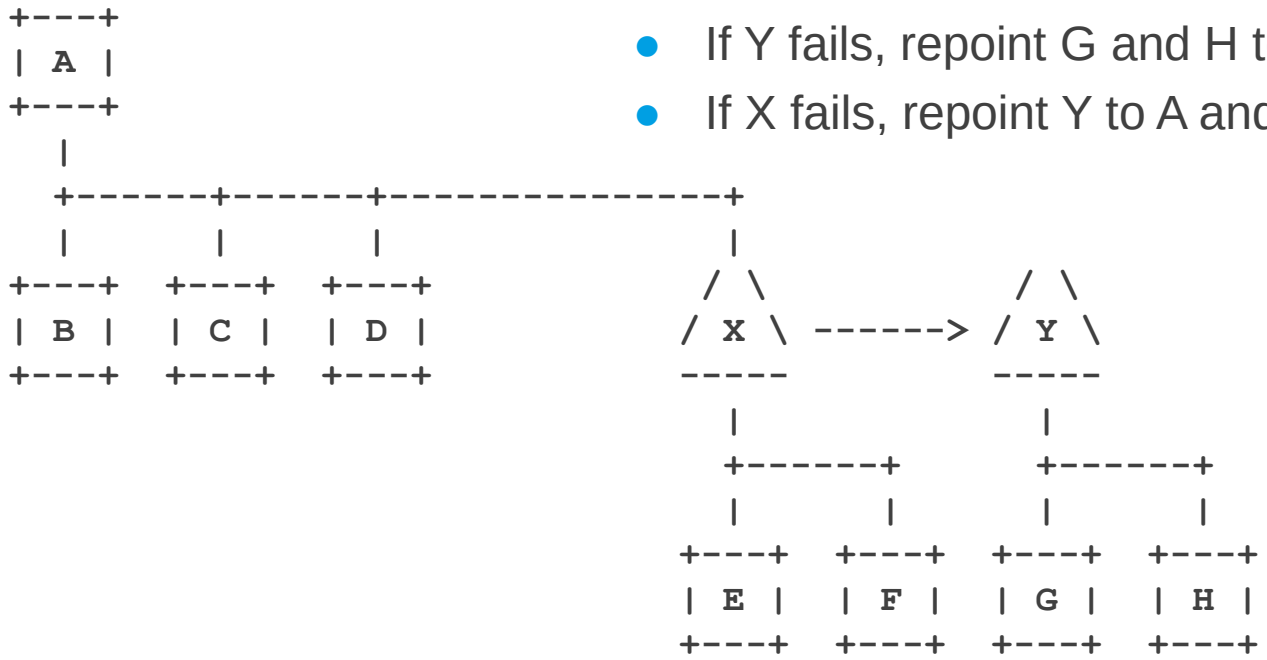
Binlog Server: Remote Site”

- New solution: a Binlog Server on the remote site:



Binlog Server: Remote Site”’

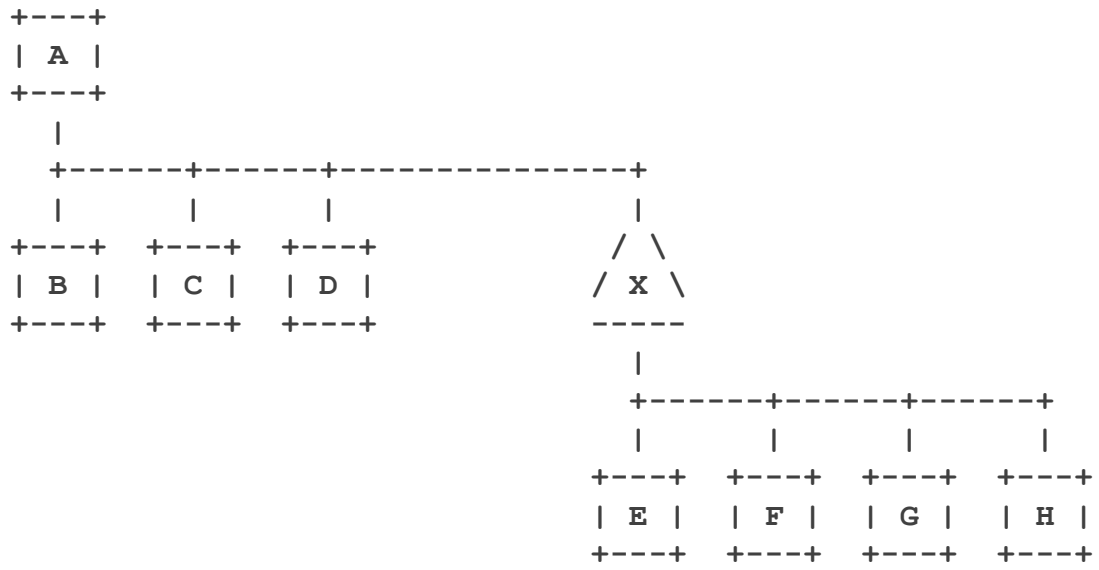
- Or deploy 2 Binlog Servers to get better resilience:



- If Y fails, repoint G and H to X,
- If X fails, repoint Y to A and E and F to Y

Binlog Server: Remote Site”’ ’

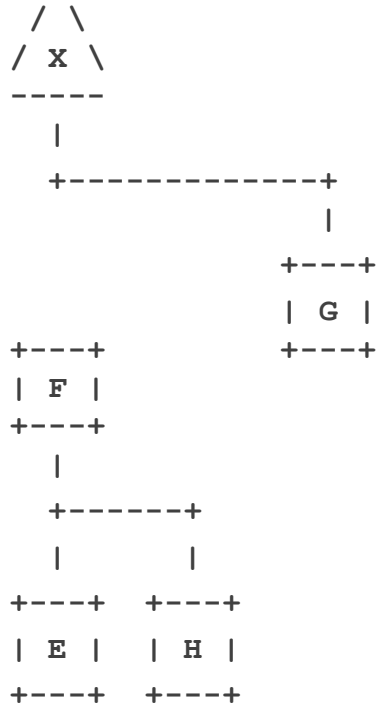
- Interesting property: if A fails, E, F, G & H converge to a common state



- New master promotion is easy on remote site

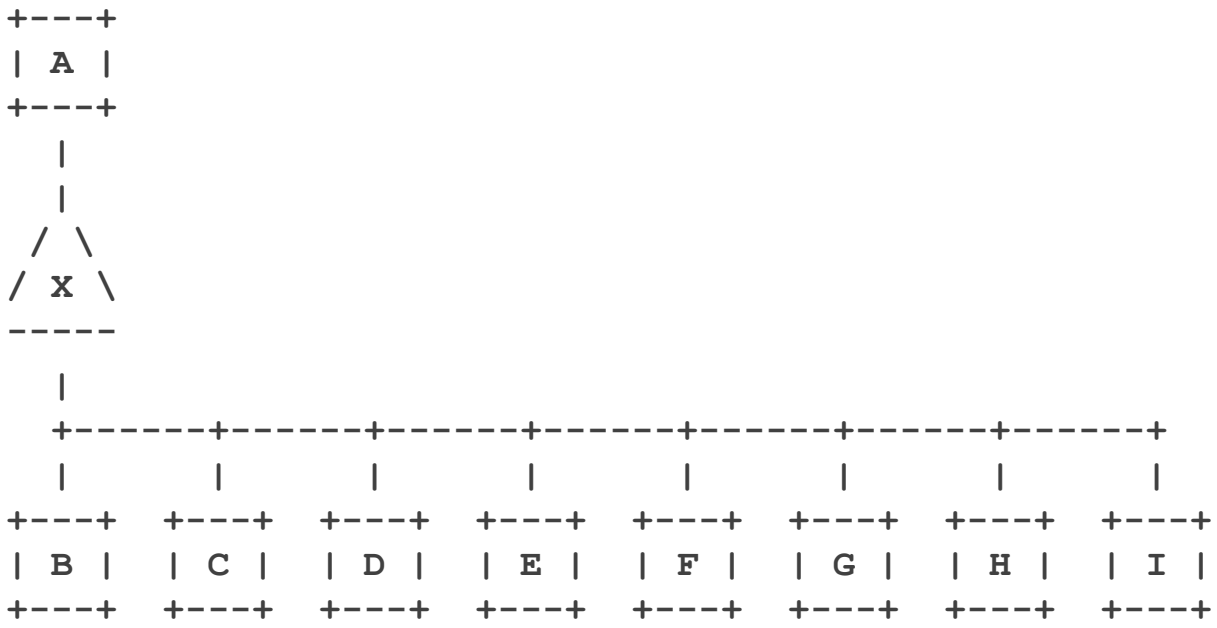
Binlog Server: Remote Site’’’ ’’

- Step by step master promotion:
  1. The 1st slave that is up to date can be the new master
  2. “SHOW MASTER STATUS” or “RESET MASTER”, and “RESET SLAVE ALL” on the new master
  3. Writes can be pointed to the new master
  4. Once a slave is up to date, repoint it to the new master at the position of step # 2
  5. Keep delayed/lagging slaves under X until up to date
  6. Once no slaves is left under X, recycle it as a Binlog Server for the new master



Binlog Server: High Availability

- This property can be used for high availability:

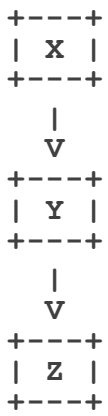


Binlog Server: Other Use-Cases

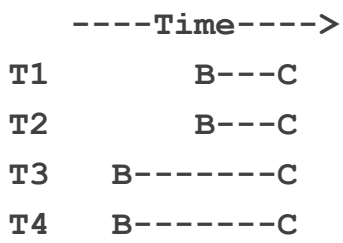
- Better Crash-Safe Replication
  - http://blog.booking.com/better_crash_safe_replication_for_mysql.html
- Better Parallel Replication in MySQL 5.7 (LOGICAL_CLOCK)
  - http://blog.booking.com/better_parallel_replication_for_mysql.html
- Easier Point in Time Recovery
  - http://jfg-mysql.blogspot.com/2015/10/binlog-servers-for-backups-and-point-in-time-recovery.html

Binlog Server: Better // Replication

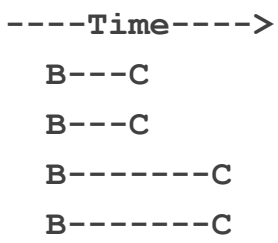
- Four transactions on X, Y and Z:



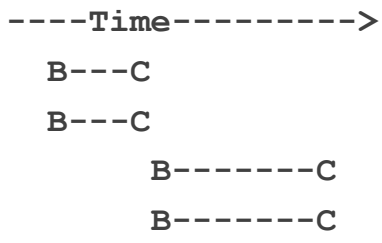
On X:



On Y:



On Z:



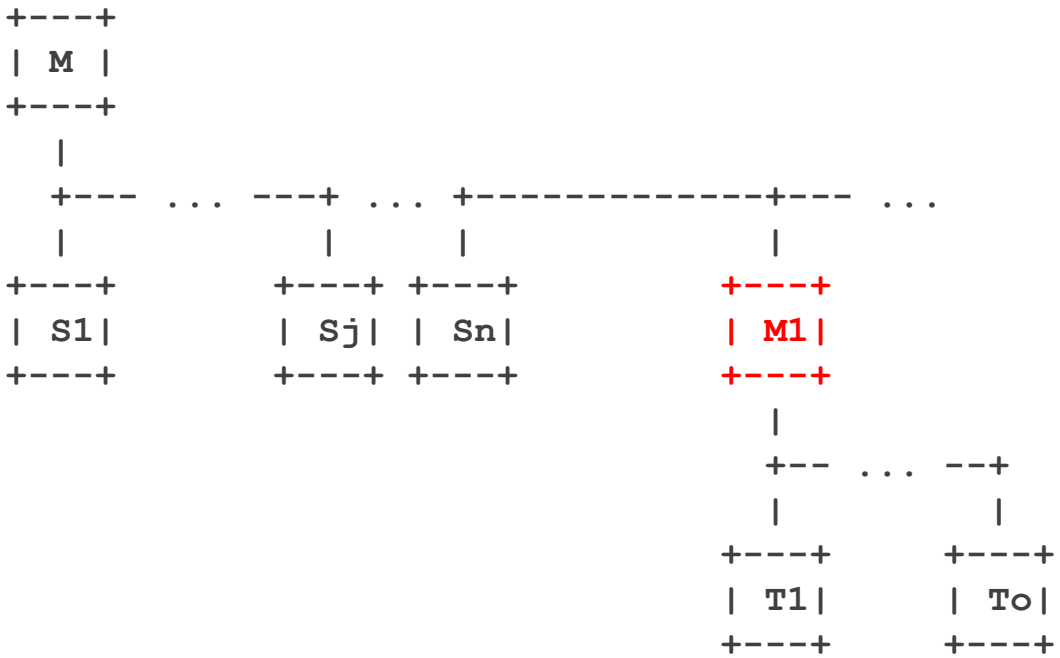
- IM might stall the parallel replication pipeline
- To benefit from parallel replication, IM must disappear
- The Binlog Server allows exactly that

Binlog Server: Point in Time Recovery

- Implementing Point in Time Recovery means:
  - to regularly take a backup of the database,
  - and to save the binary logs of that database.
- Executing Point in Time Recovery means:
  - Restoring the backup,
  - Applying the binary logs.

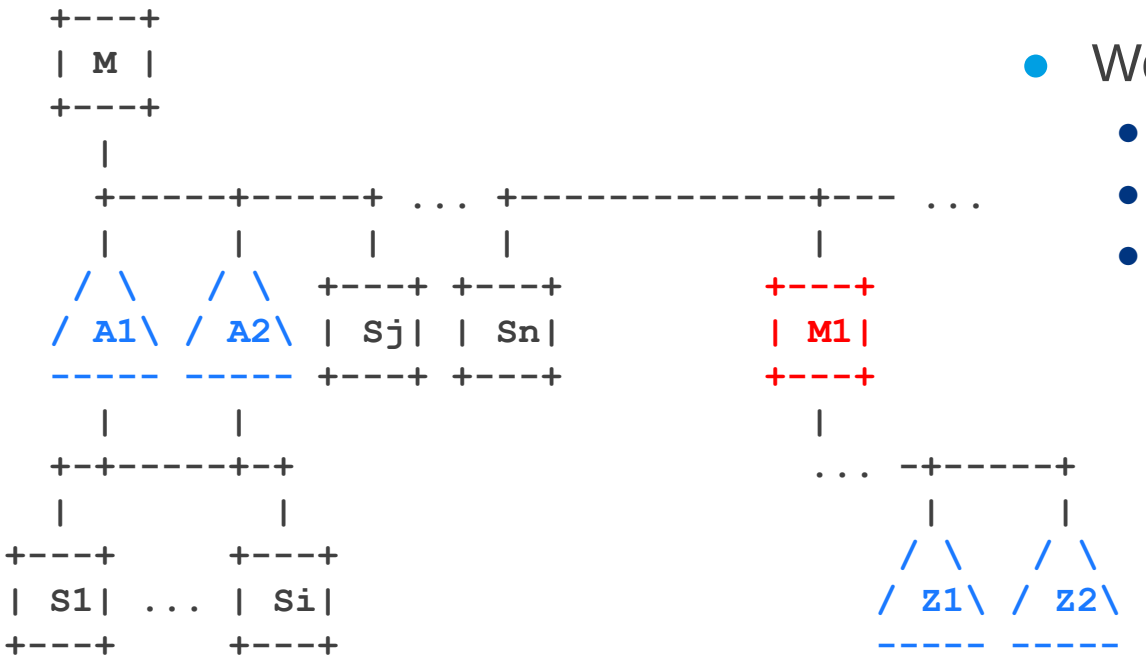
BLS@Booking.com

- Reminder: typical deployment at Booking.com:



BLS@Booking.com’

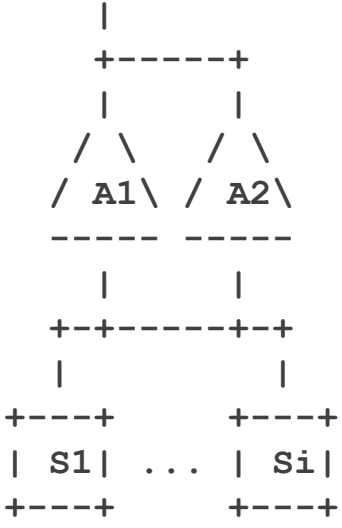
- We are deploying *Binlog Server Clusters* to offload masters:



- We have in production:
  - >40 Binlog Servers
  - >20 BLS Clusters
  - >650 slaves replicating from Binlog Servers

BLS@Booking.com”

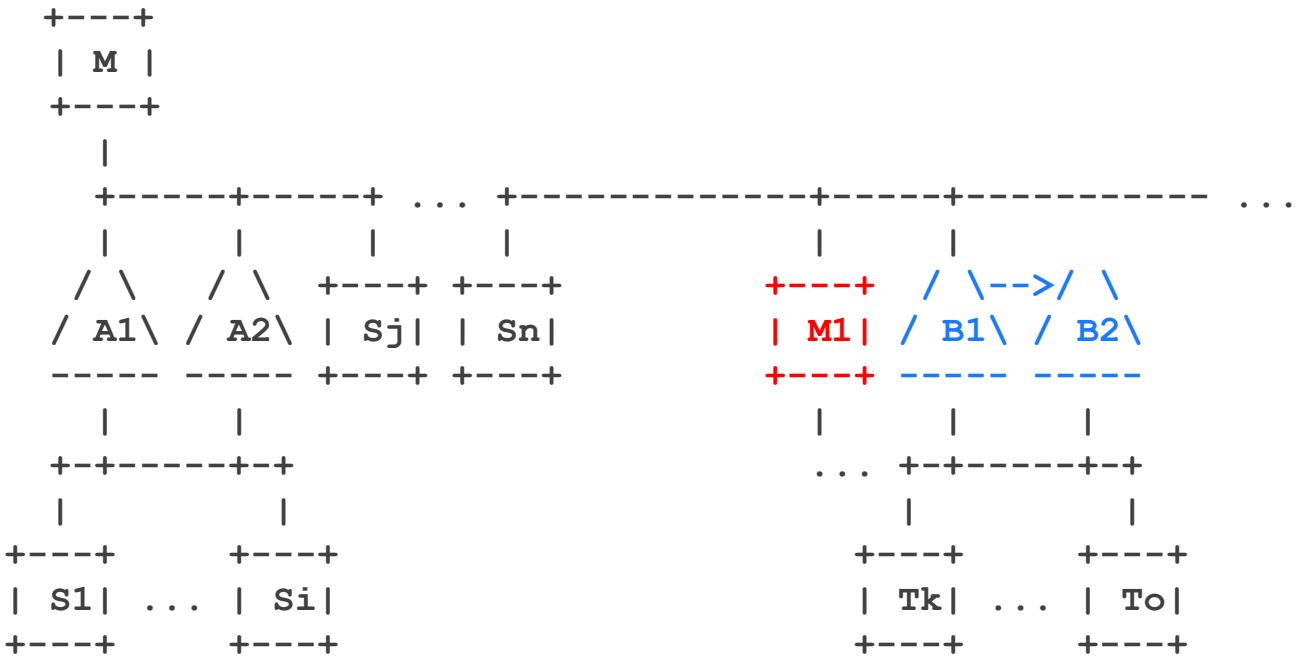
- What is a Binlog Server Cluster ?
  - At least 2 Binlog Servers
  - Replicating from the same master
  - With independent failure mode (not same switch/rack/…)
  - With a *Service* DNS entry resolving to all IP addresses



- Failure of a BLS transparent to slaves
  - Thanks to DNS, the slaves connected to a failing Binlog Server reconnect to the others
  - Easy maintenance/upgrade of a Binlog Server

BLS@Booking.com”’

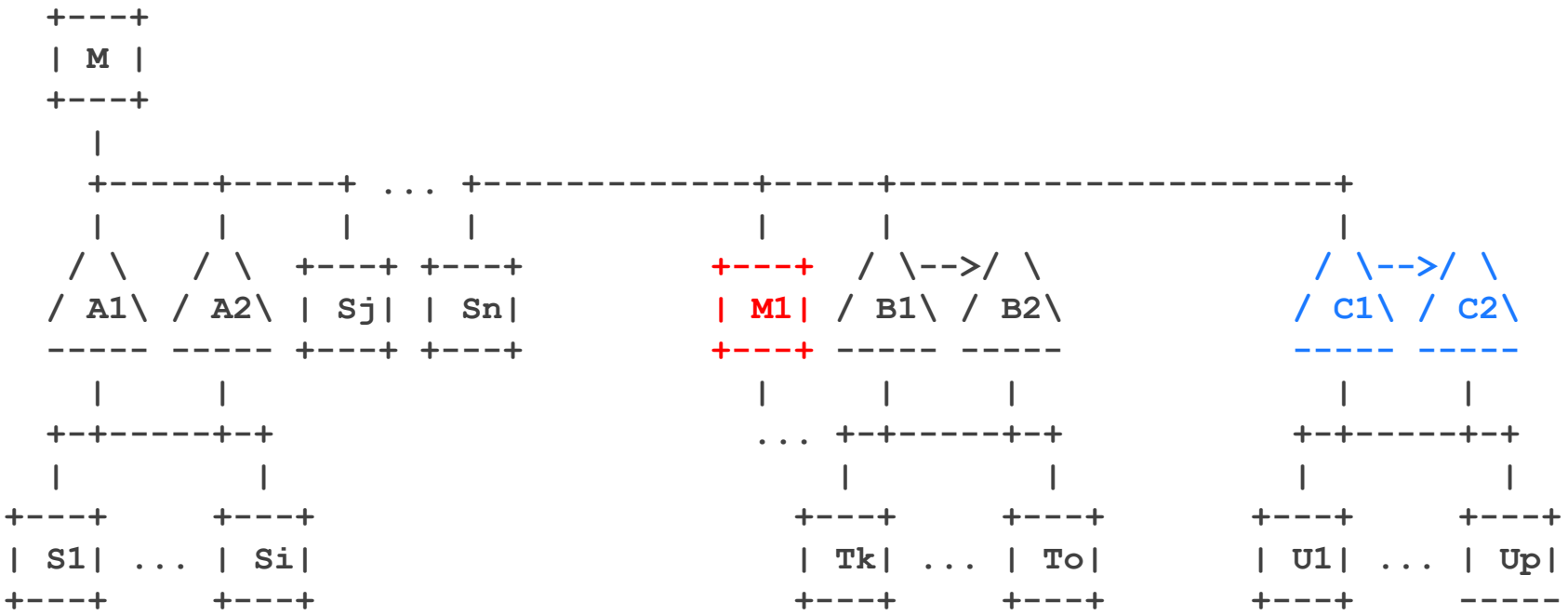
- We are deploying BLS side-by-side with IM to reduce delay:



Booking.com

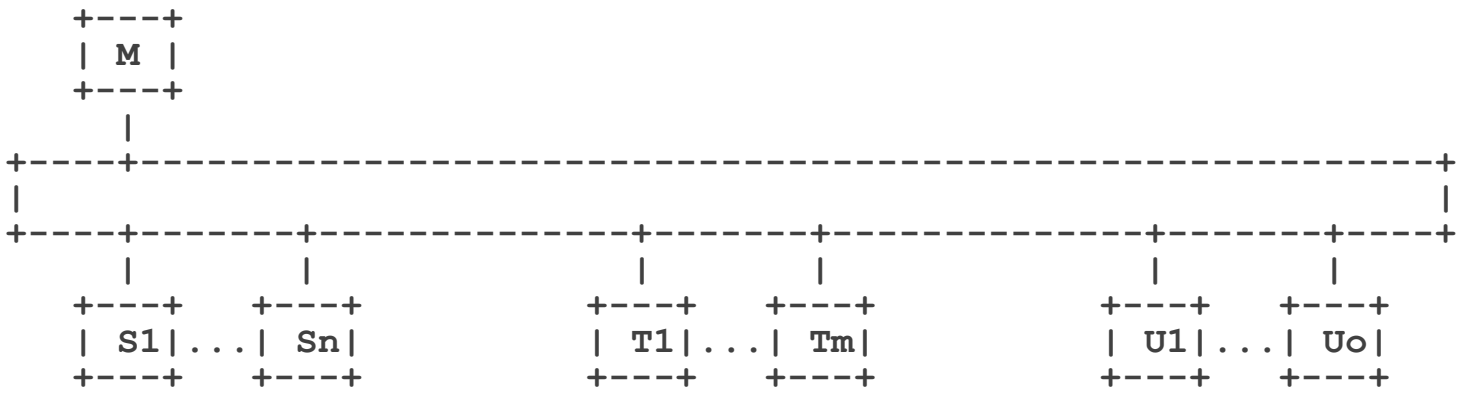
BLS@Booking.com”’ ’

- We are deploying a new Data Center without IM:



HA with Binlog Servers

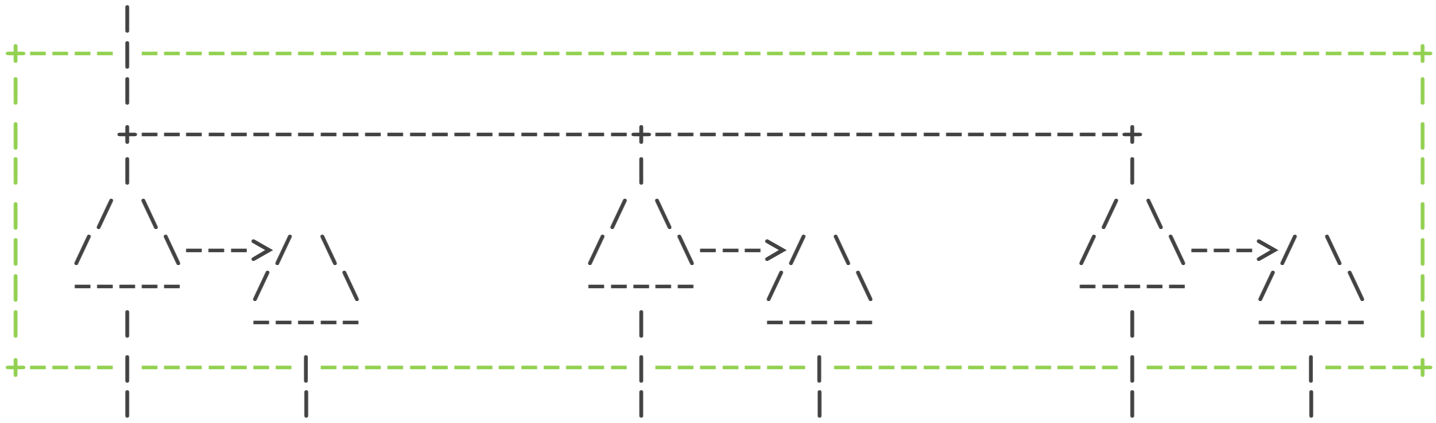
- Distributed Binlog Serving Service (*DBSS*):



- Properties:
  - A single Binlog Server failure does not disrupt the service (resilience)
  - Minimise inter Data Center bandwidth requirements
  - **Allows to promote a new master without touching any slave**

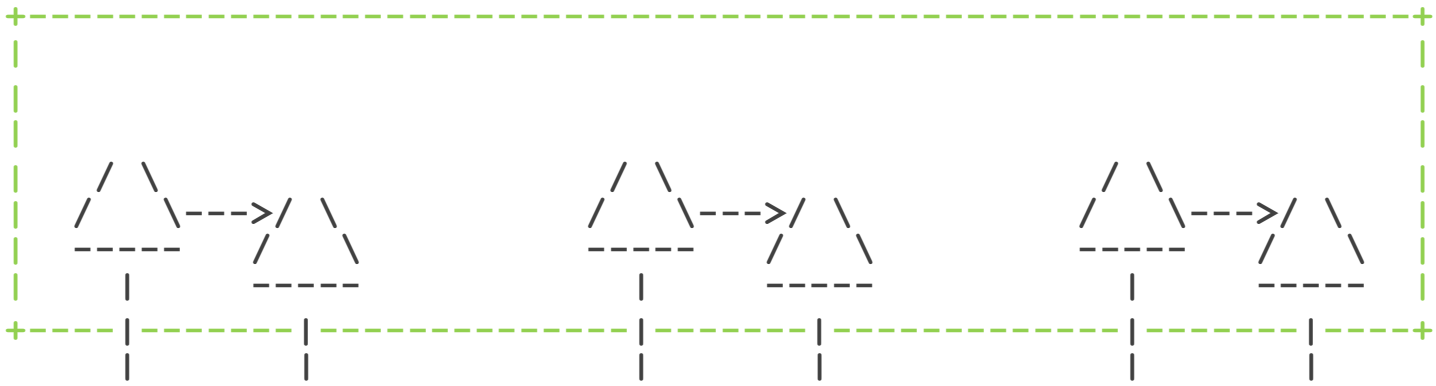
HA with Binlog Servers'

- Zoom in DBSS:



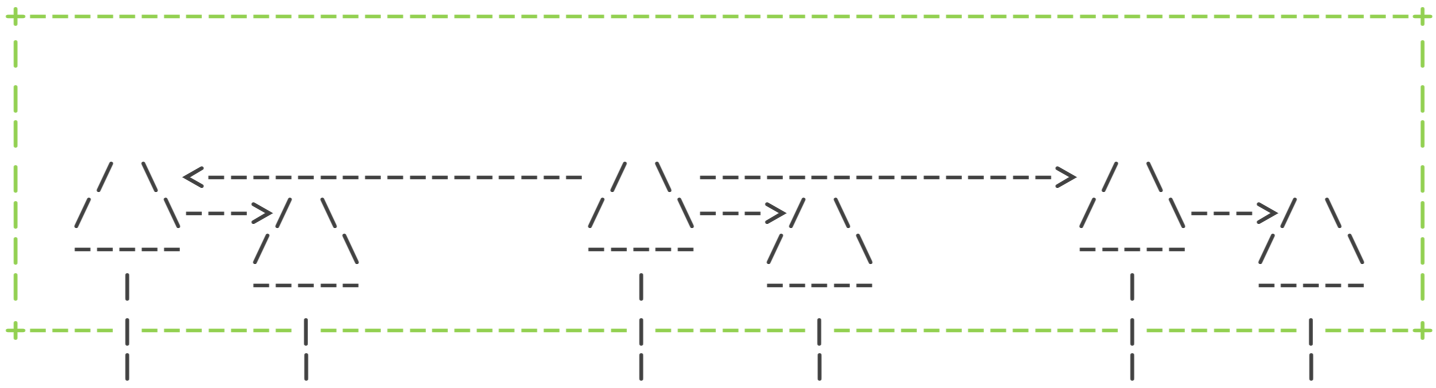
HA with Binlog Servers”

- Crash of the master:



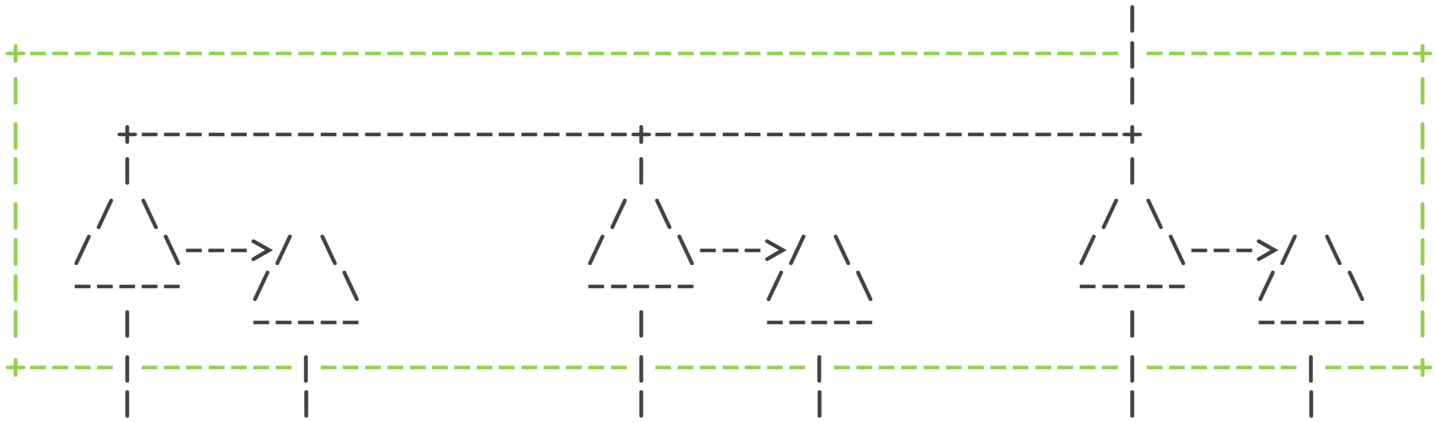
HA with Binlog Servers”

- Crash of the master:
  - Step # 1: level the Binlog Servers (the slaves will follow)



HA with Binlog Servers’’’

- Crash of the master:
  - Step # 2: promote a slave as the new master (there is a trick)



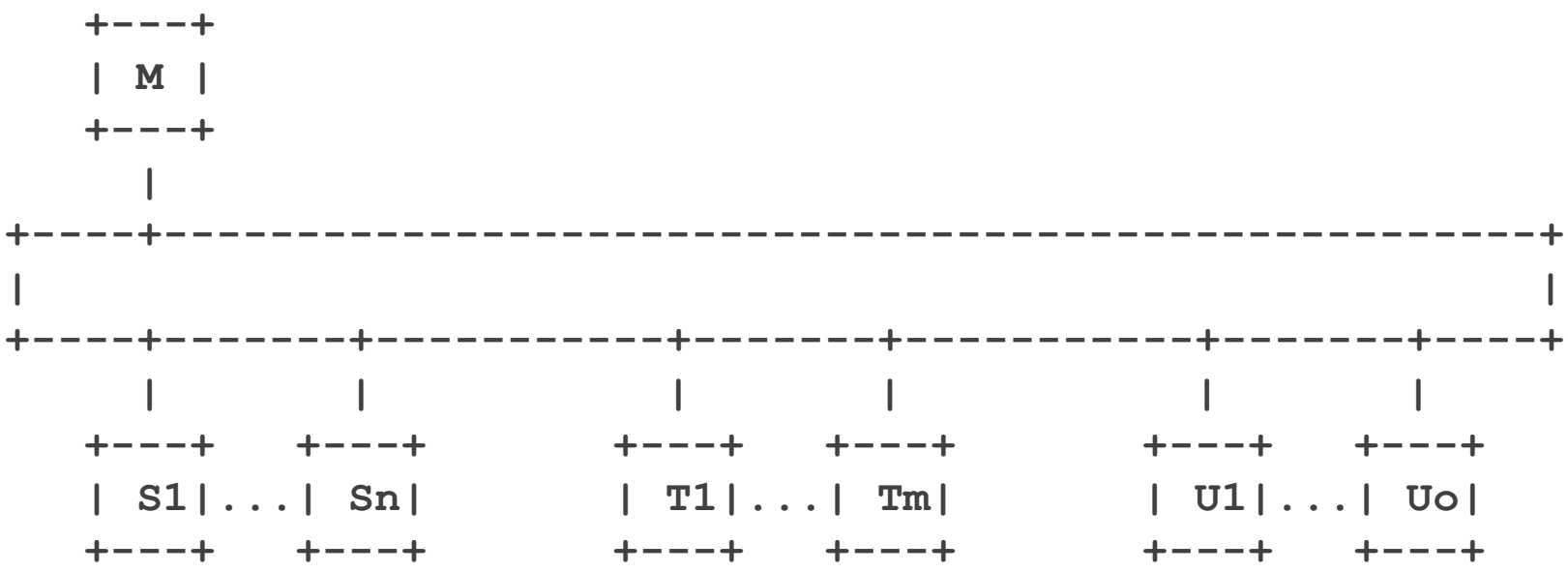
HA with Binlog Servers”’ ’

- Crash of the master - the trick:
  - Needs the same binary log filename on master and slaves

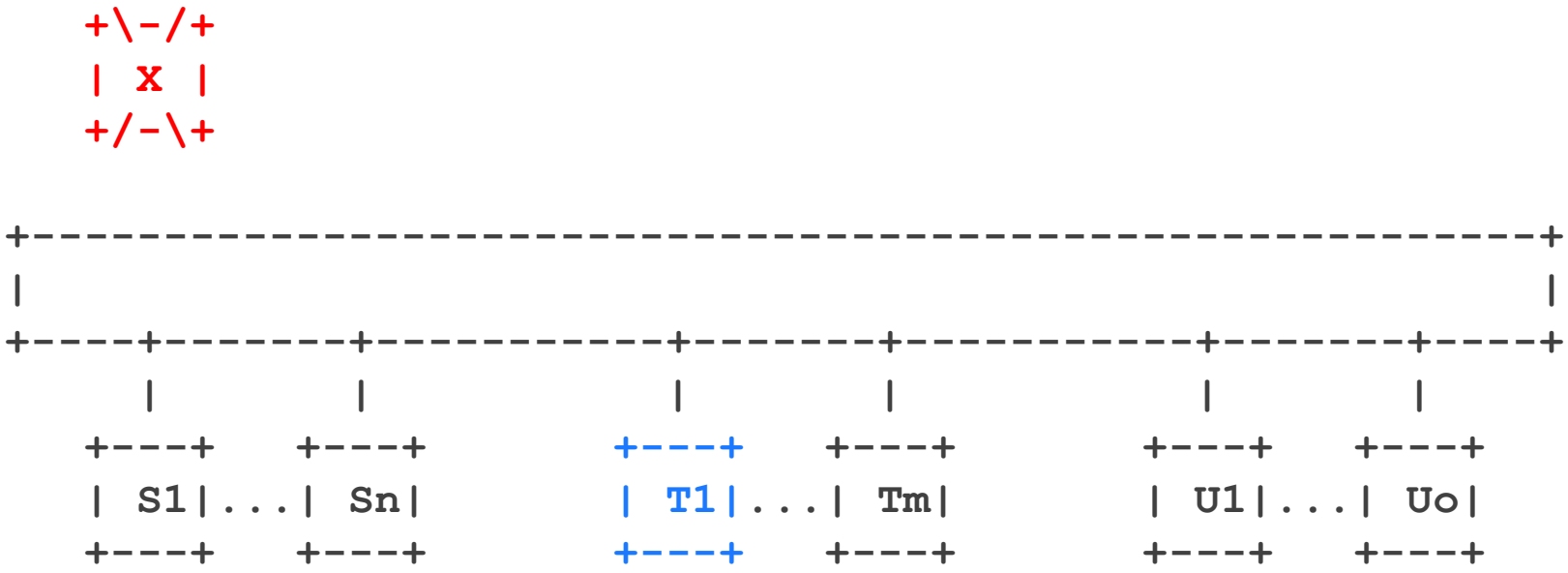
  1. “FLUSH BINARY LOGS” on candidate master
     until its binary log filename follows the one available on the BLSs
  2. On the new master:
     - “PURGE BINARY LOGS TO ‘*<latest binary log file>*'”
     - “RESET SLAVE ALL”
  3. Point the writes to the new master
  4. Make the Binlog Servers replicate from the new master

- From the point of view of the Binlog Server, the master only rebooted with a new ServerID and a new UUID.

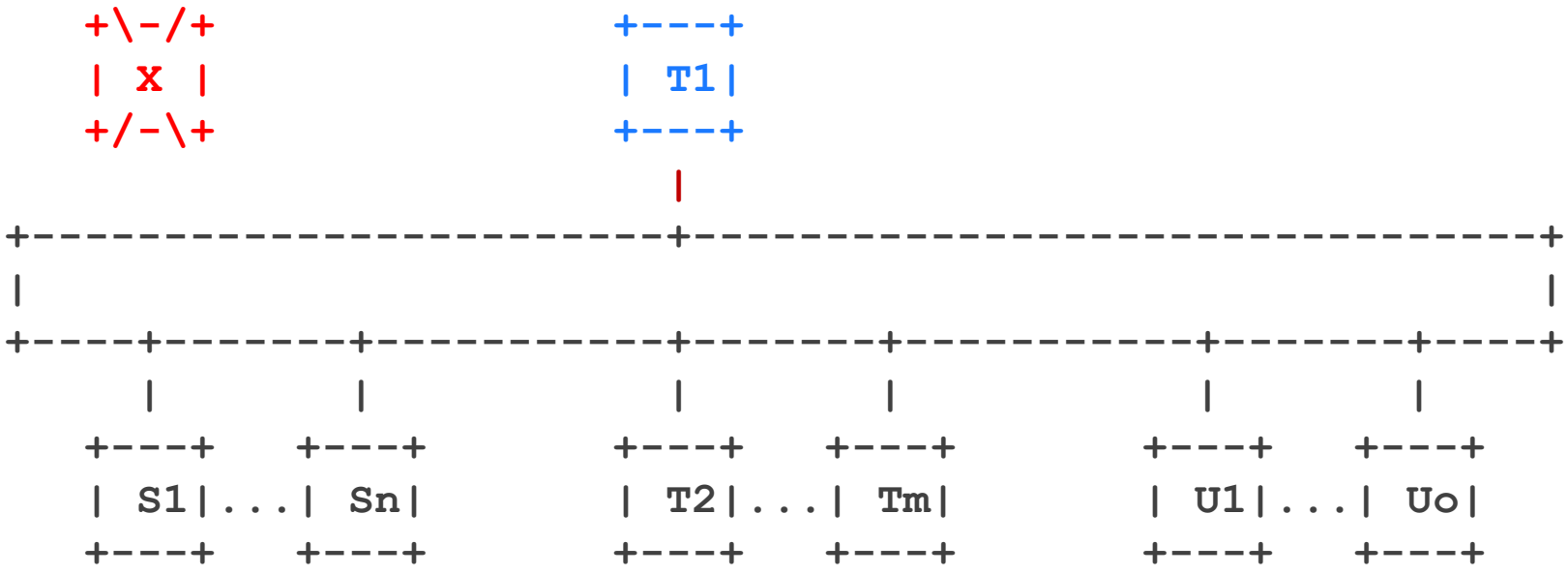
New Master wo Touching Slaves



New Master wo Touching Slaves



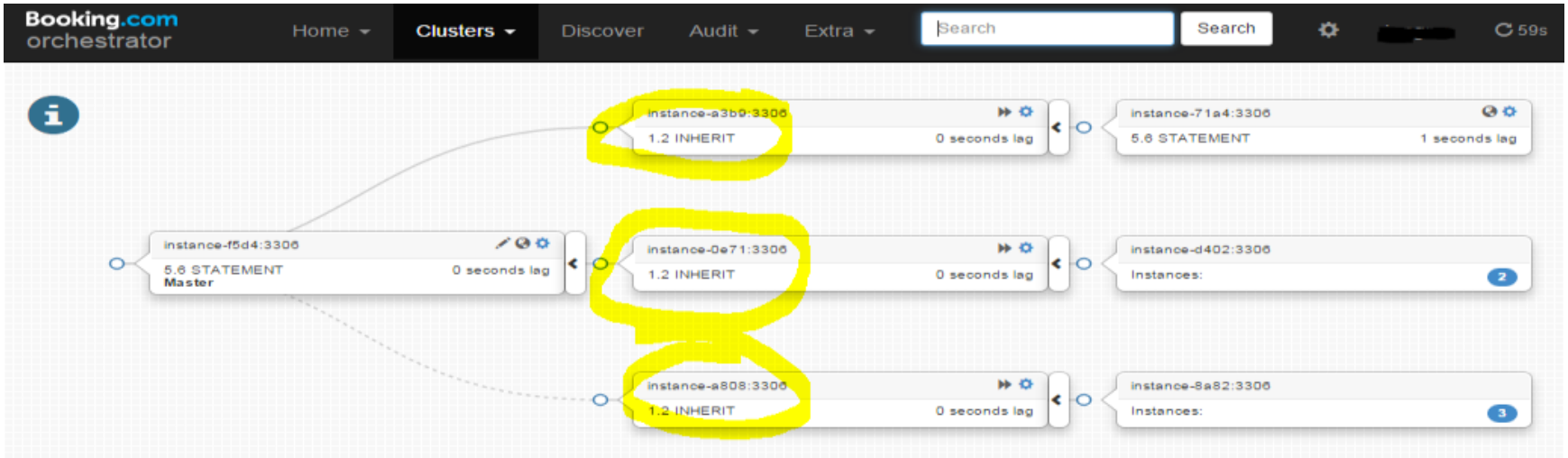
New Master wo Touching Slaves



New Master wo Touching Slaves'

- "FLUSH BINARY LOGS" in a loop is ugly (but it works)
- A "RESET MASTER at/to 'binlog.00xxxx'" would be much nicer:
  - https://bugs.mysql.com/bug.php?id=77438

Binlog Servers with Orchestrator



- Orchestrator is the tool we use for managing Binlog Servers
  - https://github.com/orchestrator/orchestrator

Binlog Server: Links

- 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
- **HOWTO Install and Configure Binlog Servers**: http://jfg-mysql.blogspot.com/2015/04/maxscale-binlog-server-howto-install-and-configure.html
- http://blog.booking.com/better_crash_safe_replication_for_mysql.html
- http://blog.booking.com/better_parallel_replication_for_mysql.html (http://blog.booking.com/evaluating_mysql_parallel_replication_2-slave_group_commit.html)
- http://jfg-mysql.blogspot.nl/2015/10/binlog-servers-for-backups-and-point-in-time-recovery.html
- **Note: the Binlog Servers concept should work with any version of MySQL (5.7, 5.6, 5.5 and 5.1)**

Questions

Jean-François Gagné
jeanfrancois DOT gagne AT booking.com