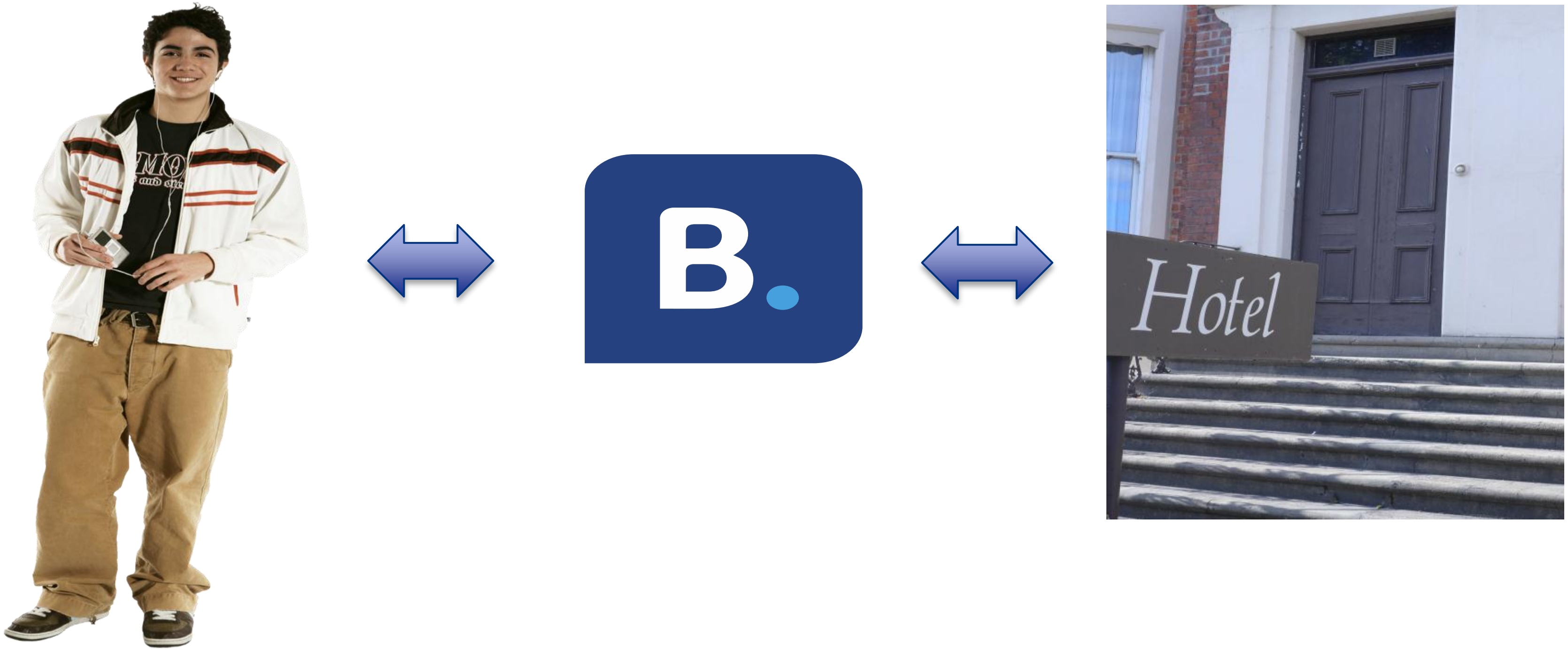
Booking.com

Binlog Servers (and MySQL) at Booking.com

Jean-François Gagné
jeanfrancois DOT gagne AT booking.com

Presented at Percona Live Santa Clara 2015

Booking.com

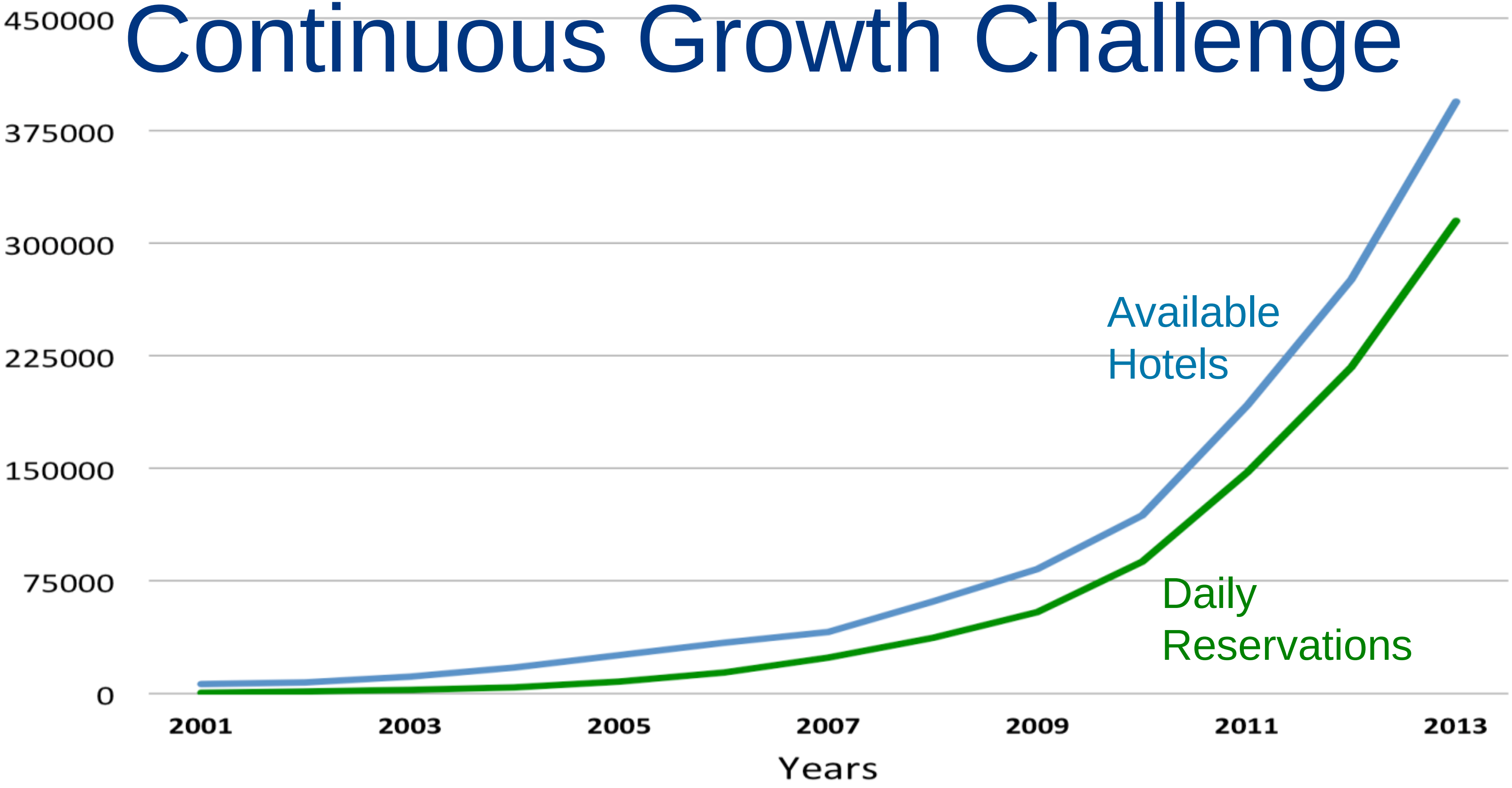


Booking.com’

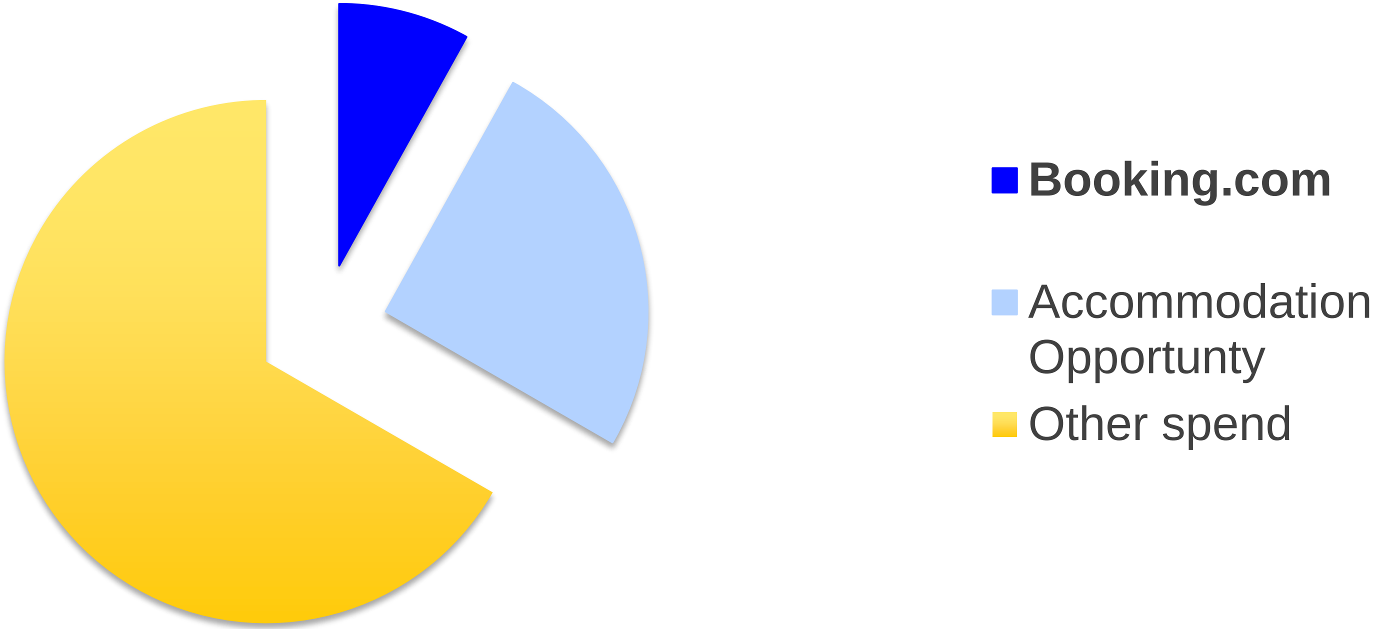
- Based in Amsterdam since 1996
- Online Hotel and Accommodation Agent:
  - 135 offices worldwide
  - +619.000 properties in 211 countries
  - 42 languages (website and customer service)
- Part of the Priceline Group

- And we are using MySQL:
  - >3300 servers, ~85% replicating
  - ~110 masters: ~30 >25 slaves, ~14 >50 slaves & ~4 >100 slaves

Continuous Growth Challenge



Travel business opportunity



Binlog Server: Session Summary

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

Binlog Server: Guiding Principles

- GTIDs are not needed for managing replication hierarchy
- Nor are Intermediate Masters
  → log-slave-updates can be avoided (for replication)

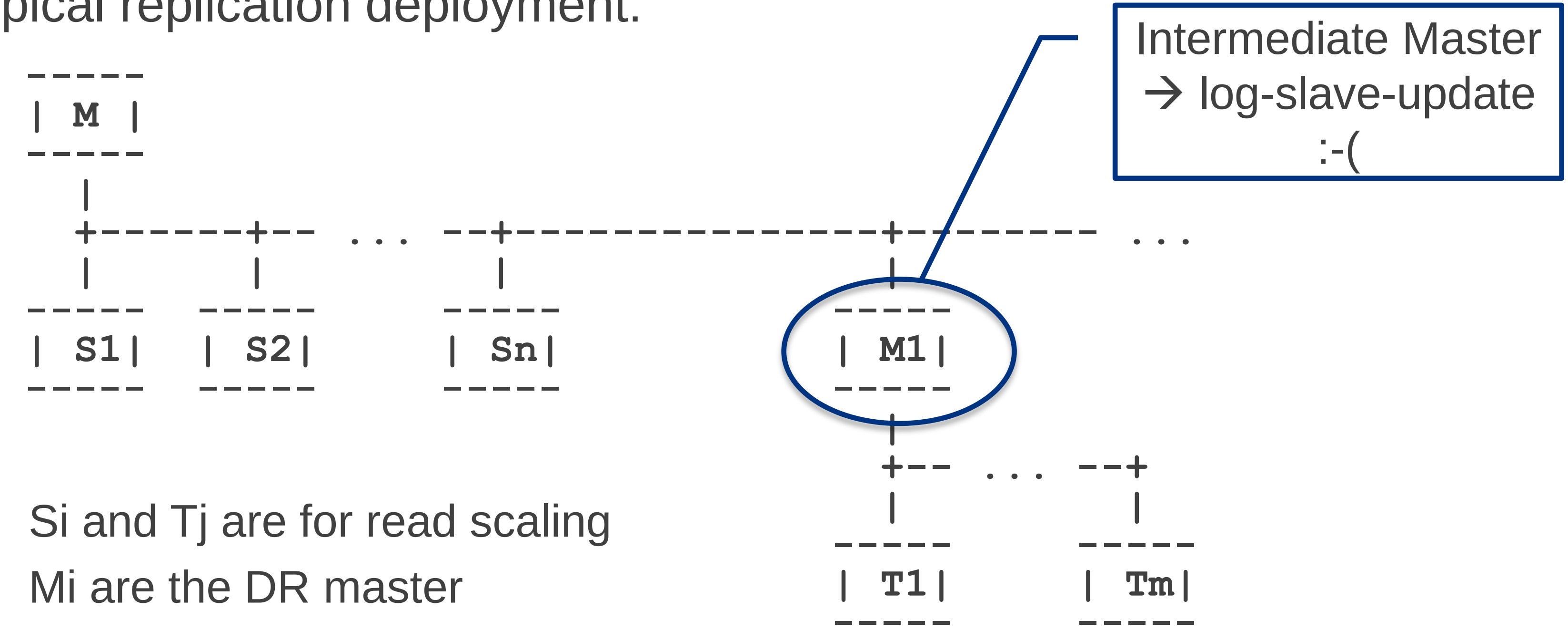
Binlog Server: Replication

- One master / one or more slaves

- The master records all its 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:



```
-----
| M |
-----
  |
  +-------+-- ... --+-----------------+---------  ...
  |       |         |                 |
-----   -----     -----             -----
| S1|   | S2|     | Sn|             | M1|
-----   -----     -----             -----
                                      |
                                      +-- ... --+
                                      |         |
                                    -----     -----
                                    | T1|     | Tm|
                                    -----     -----
```

Intermediate Master
→ log-slave-update
:-(

  - 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 in the same structure as on the master
  - Serves them to slaves

```
-----        / \
| A | ---> / X \
-----      -----
  |          |
-----      -----
| B |      | C |
-----      -----
```

  - 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:

```
-----
| M |
-----
  |
  +------+------+-- ... --+
  |      |      |         |
-----  -----  -----     -----
| S1|  | S2|  | S3|     | Sn|
-----  -----  -----     -----
```

- When there are too many slaves,
  the network interface 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*):

```
-----
| M |
-----
  |
  +----------+--- ... ---+
  |          |           |
-----      -----       -----
| M1|      | M2|       | Mm|
-----      -----       -----
  |          |           |
  +--- ...   +--- ...    +--- ... ---+
  |          |           |           |
-----      -----       -----       -----
| S1|      | T1|       |...|       | Zi|
-----      -----       -----       -----
```

- 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 an 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 Intermediate Master problems with shared disk:
  - Filers (expensive) or DRBD (doubling the number of servers)
  - HA needs sync_binlog=1 + trx_commit=1 → replication is slower → 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 Intermediate Master 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 Intermediate Masters by Binlog Servers

```
-----
| M |
-----
  |
  +----------------+------ ... ------+
  |                |                 |
 / \              / \               / \
/ I1\            / I2\             / Im\
-----            -----             -----
  |                |                 |
  +------+ ...     +--- ...          +--- ... ---+
  |      |         |                 |           |
-----  -----     -----             -----       -----
| S1|  | S2|     | Si|             | Sj|       | Sn|
-----  -----     -----             -----       -----
```

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

Binlog Server: Remote Site

- Typical deployment for remote site:

```
-----
| A |
-----
  |
  +------+------+---------------+
  |      |      |               |
-----  -----  -----           -----
| B |  | C |  | D |           | E |
-----  -----  -----           -----
                                |
                                +------+------+
                                |      |      |
                              -----  -----  -----
                              | F |  | G |  | H |
                              -----  -----  -----
```

Yuck !

- E is an Intermediate Master → same problems as read scaling

Binlog Server: Remote Site’

- Ideally, we would like this:

```
-----
| A |
-----
  |
  +-------+-------+-----------------+-------+-------+-------+
  |       |       |                 |       |       |       |
-----   -----   -----           -----   -----   -----   -----
| B |   | C |   | D |           | E |   | F |   | G |   | H |
-----   -----   -----           -----   -----   -----   -----
```

- 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:

```
-----
| A |
-----
  |
  +------+------+---------------+
  |      |      |               |
-----  -----  -----            / \
| B |  | C |  | D |           / X \
-----  -----  -----           -----
                                |
                                +------+------+------+
                                |      |      |      |
                              -----  -----  -----  -----
                              | E |  | F |  | G |  | H |
                              -----  -----  -----  -----
```

Binlog Server: Remote Site’’’

- Or deploy 2 Binlog Servers to get better resilience:

```
-----
| A |
-----
  |
  +------+------+---------------+
  |      |      |               |
-----  -----  -----            / \           / \
| B |  | C |  | D |           / X \ ------> / Y \
-----  -----  -----           -----         -----
                                |             |
                                +------+      +------+
                                |      |      |      |
                              -----  -----  -----  -----
                              | E |  | F |  | G |  | H |
                              -----  -----  -----  -----
```

- 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

```
-----
| A |
-----
  |
  +------+------+---------------+
  |      |      |               |
-----  -----  -----            / \
| B |  | C |  | D |           / X \
-----  -----  -----           -----
                                |
                                +------+------+------+
                                |      |      |      |
                              -----  -----  -----  -----
                              | E |  | F |  | G |  | H |
                              -----  -----  -----  -----
```

- 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

```
 / \
/ X \
-----
  |
  +--------------+
                 |
               -----
               | G |
-----          -----
| F |
-----
  |
  +-------+
  |       |
-----   -----
| E |   | H |
-----   -----
```

Binlog Server: High Availability

- This property can be used for high availability:

```
-----
| A |
-----
  |
 / \
/ X \
-----
  |
  +------+------+------+------+------+------+------+
  |      |      |      |      |      |      |      |
-----  -----  -----  -----  -----  -----  -----  -----
| B |  | C |  | D |  | E |  | F |  | G |  | H |  | I |
-----  -----  -----  -----  -----  -----  -----  -----
```

Binlog Server: Other Use-Cases

- Better Crash-Safe Replication
  - http://blog.booking.com/better_crash_safe_replication_for_mysql.html

- And Better Parallel Replication
  - http://blog.booking.com/better_parallel_replication_for_mysql.html

BLS@Booking.com

- Reminder: typical replication deployment at Booking.com:

```
-----
| M |
-----
  |
  +--- ... ---+ ... +-------------+--- ...
  |           |     |             |
-----       ----- -----         -----
| S1|       | Sj| | Sn|         | M1|
-----       ----- -----         -----
                                  |
                                  +-- ... --+
                                  |         |
                                -----     -----
                                | T1|     | To|
                                -----     -----
```

BLS@Booking.com’

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

```
  -----
  | M |
  -----
    |
    +-----+-----+ ... +-------------+--- ...
    |     |     |     |             |
   / \   / \  ----- -----         -----
  / A1\ / A2\ | Sj| | Sn|         | M1|
  ----- ----- ----- -----         -----
    |     |                         |
  +-+-----+-+                       ... -+-----+
  |         |                            |     |
-----     -----                         / \   / \
| S1| ... | Si|                        / Z1\ / Z2\
-----     -----                        ----- -----
```

- We have in production:
  - 18 Binlog Servers
  - 8 clusters (two with 3 BLSs)
  - ~400 slaves replicating from Binlog Servers

(those number are increasing)

BLS@Booking.com”

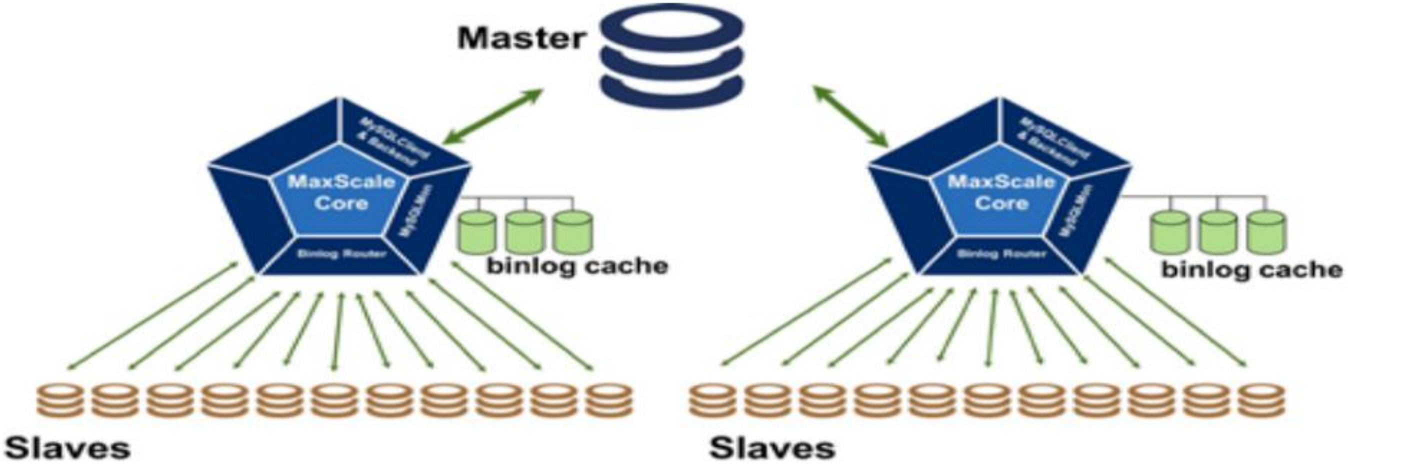
- What is a Binlog Server Cluster ?
  - At least 2 Binlog Servers
  - Replicating from the same master
  - With independent failure mode (not on the same switch/rack/…)
  - With a *Service* DNS entry resolving to all IP addresses of the BLSs

```
    |
    +-----+
    |     |
   / \   / \
  / A1\ / A2\
  ----- -----
    |     |
  +-+-----+-+
  |         |
-----     -----
| S1| ... | Si|
-----     -----
```

➢ A failure of a Binlog Server is transparent to slaves
  - Thanks to DNS, the slaves connected to a failing Binlog Server reconnect to the other still working Binlog Server(s)
  - ➢ Easy maintenance/upgrade of a Binlog Server

MariaDB MaxScale Binlog Router

- High Performance Parallel Replication with Bin Log Router:
  […] provides efficient replication from master to slave without the need for intermediate masters. This plugin receives binlog records from a master and makes them available immediately to slaves […]. You can use MaxScale's binlog server for replication to slaves across one or more datacenters and deliver a high performance user experience.



Booking.com

BLS@Booking.com’’’

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

```
  -----
  | M |
  -----
    |
    +-----+-----+ ... +-------------+-----+----------- ...
    |     |     |     |             |     |
   / \   / \  ----- -----         -----  / \-->/ \
  / A1\ / A2\ | Sj| | Sn|         | M1| / B1\ / B2\
  ----- ----- ----- -----         ----- ----- -----
    |     |                         |     |     |
  +-+-----+-+                       ... +-+-----+-+
  |         |                           |         |
-----     -----                       -----     -----
| S1| ... | Si|                       | Tk| ... | To|
-----     -----                       -----     -----
```

BLS@Booking.com””

- We are planning to deploy our next Data Center without IM:

```
  -----
  | M |
  -----
    |
    +-----+-----+ ... +-------------+-----+-----------  ...  -----+
    |     |     |     |             |     |                     |
   / \   / \  ----- -----         -----  / \-->/ \             / \-->/ \
  / A1\ / A2\ | Sj| | Sn|         | M1| / B1\ / B2\           / C1\ / C2\
  ----- ----- ----- -----         ----- ----- -----           ----- -----
    |     |                         |     |     |               |     |
  +-+-----+-+                       ...  +-+-----+-+           +-+-----+-+
  |         |                           |         |           |         |
-----     -----                       -----     -----       -----     -----
| S1| ... | Si|                       | Tk| ... | To|       | U1| ... | Up|
-----     -----                       -----     -----       -----     -----
```

Future of Binlog Servers

- Distributed Binlog Serving Service (*DBSS*):

```
   -----
   | M |
   -----
     |
+----+----------- ... --------------- ... ----------------+
|                                                         |
+----+-------+--- ... ---+-------+--- ... ---+-------+----+
     |       |           |       |           |       |
   -----   -----       -----   -----       -----   -----
   | S1|...| Sn|       | T1|...| Tm|       | U1|...| Uo|
   -----   -----       -----   -----       -----   -----
```

- 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**

Future of Binlog Servers’

- Zoom in DBSS:

```
     |
+----|------------- ... ----------------- ... -----------------+
|    |                                                         |
|    +------------- ... ----+------------ ... ----+            |
|    |                      |                     |            |
|   / \                    / \                   / \           |
|  /   \--->/ \           /   \--->/ \          /   \--->/ \   |
|  -----   /   \          -----   /   \         -----   /   \  |
|    |      -----            |      -----         |      -----  |
+----|-------|----- ... ----|-------|---- ... ----|-------|----+
     |       |              |       |             |       |
```

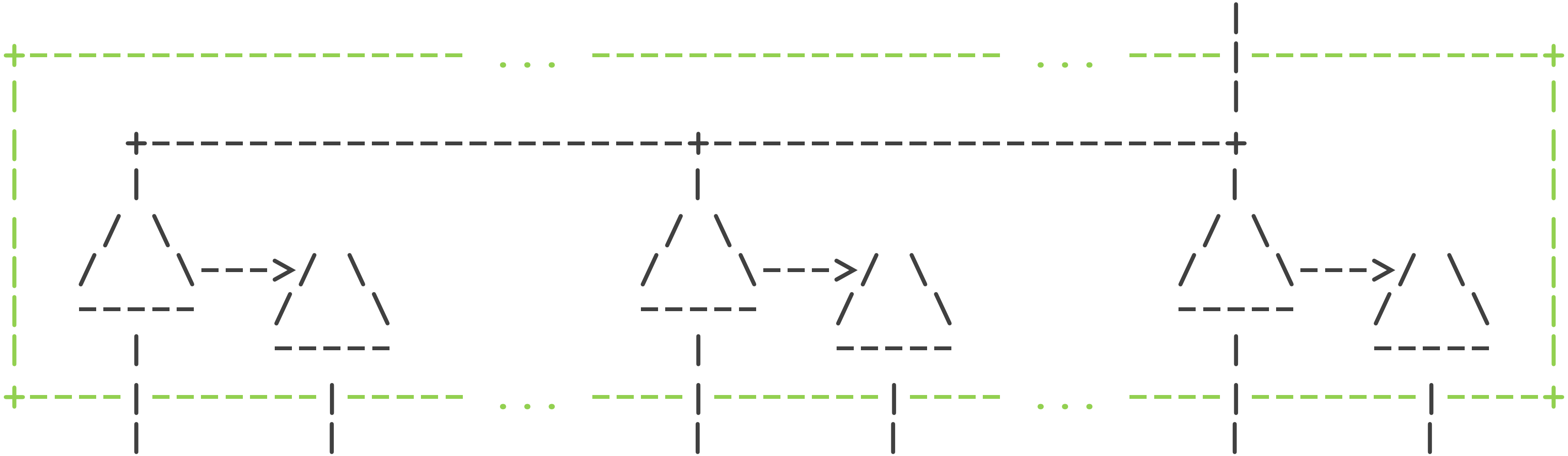
Future of Binlog Servers”

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

```
+----------------------- ... ---------------------- ... -----------------------+
|                                                                              |
|                                                                              |
|                                                                              |
|    / \<-------------------------/ \------------------------>/ \              |
|   /   \--->/ \                 /   \--->/ \                /   \--->/ \      |
|   -----   /   \                -----   /   \               -----   /   \     |
|     |     -----                  |     -----                 |     -----     |
+-----|-------|--------- ... ------|-------|------- ... -------|-------|-------+
      |       |                    |       |                   |       |
```

Future of Binlog Servers’’’

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



Future of 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 Binlog Servers
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

```
   -\-/-               -----
   | x |               | T1|
   -/-\-               -----
                         |
+----+----------- ... ---+----------- ... ----------------+
|                                                         |
+----+-------+--- ... ----------+--- ... ---+-------+----+
     |       |                   |           |       |
   -----   -----               -----       -----   -----
   | S1|...| Sn|            ...| Tm|       | U1|...| Uo|
   -----   -----               -----       -----   -----
```

Binlog Server: Links

- http://blog.booking.com/mysql_slave_scaling_and_more.html
- MaxScale High Performance Binlog Relay: https://mariadb.com/products/mariadb-maxscale
- **HOWTO Install and Configure a MaxScale Binlog Server**: 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)
- The MaxScale mailing list (to ask questions): maxscale@googlegroups.com
- **Note: the Binlog Servers concept should work with any version of MySQL (5.7, 5.6, 5.5 and 5.1) or MariaDB (10.1, 10.0, 5.5. 5.3, 5.2 and 5.1)**

Questions

Jean-François Gagné

jeanfrancois DOT gagne AT booking.com

(maxscale AT googlegroups.com)