O'REILLY

MySQL

Conference & Expo

Advance Replication Monitoring

Gerardo "Gerry" Narvaja
@seattlegaucho

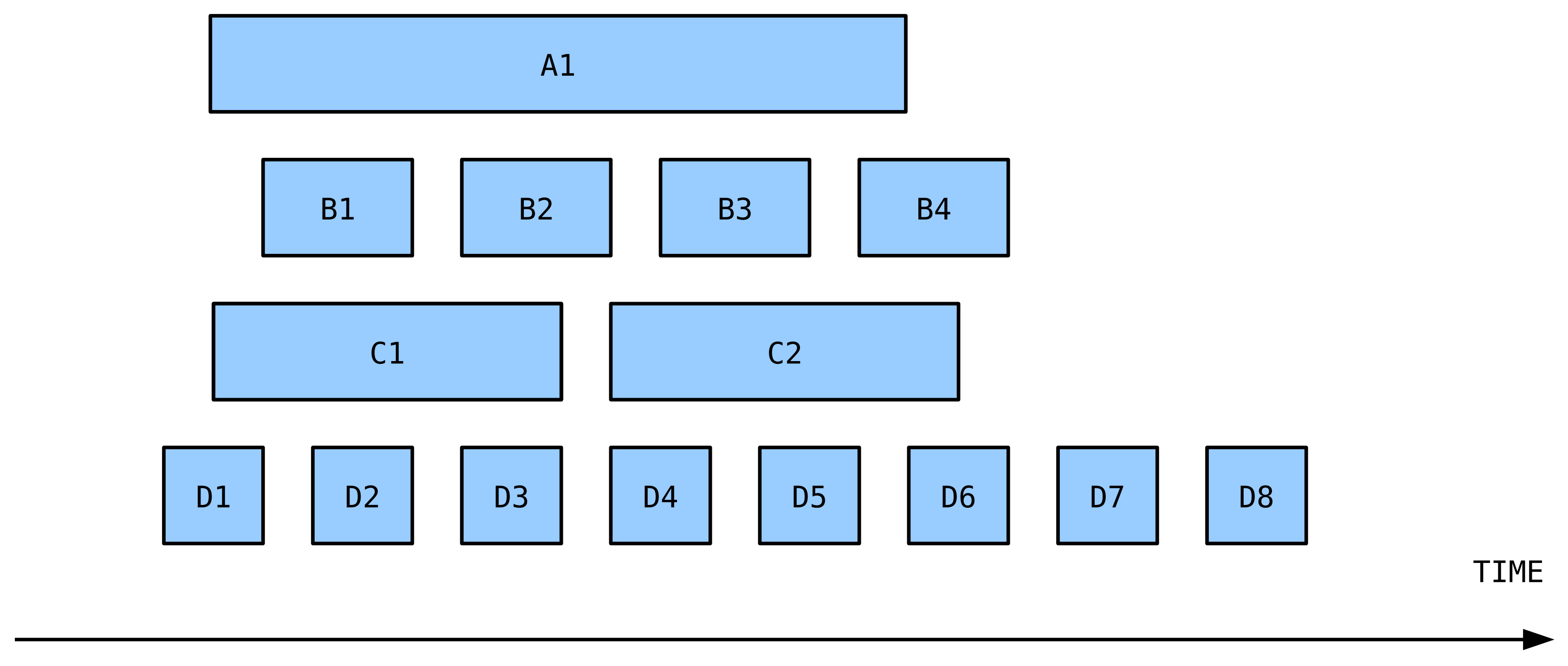
Agenda

O'REILLY MySQL Conference & Expo

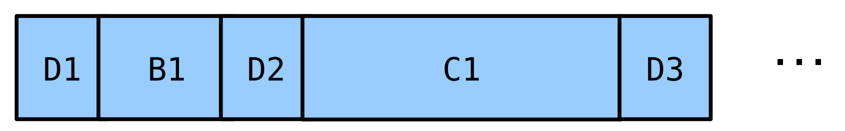
- Short Introduction
  - Make sure we all speak the same language
- Scenarios
  - What can go wrong and why it may be OK
- What To Look For / At
  - What the variables mean
  - Some *pretty pictures*
- Conclusion

Introduction

- What happens in the master …



- … in the slave it becomes …



- Replication is **single-threaded**
  - IO Thread + **SQL Thread**
  - No contention in the slave, it should run faster

Most Basic Monitoring

- SHOW SLAVE STATUS
  - IO Thread
    - Usually flags communication issues
  - SQL Thread
    - Usually flags data related issues
- Application code
  - Maatkit: mk-heartbeat
    - Simple monitoring can be implemented at the shell
  - Implement your own heartbeat table
    - Can be used to measure quality of data on the slaves
- If you don't have this basic monitoring in place, is like taking backups and not testing restores.

Replication Status

O'REILLY MySQL Conference & Expo

- SHOW SLAVE STATUS\G

```
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 10.55.197.108
                  Master_User: repl
                  Master_Port: 3306
                Connect_Retry: 60
              Master_Log_File: mysql-bin.000447
          Read_Master_Log_Pos: 673847271
               Relay_Log_File: relay-bin.005771
                Relay_Log_Pos: 673847416
        Relay_Master_Log_File: mysql-bin.000447
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
              Replicate_Do_DB:
          Replicate_Ignore_DB:
           Replicate_Do_Table:
       Replicate_Ignore_Table: mysql.user,mysql.columns_priv,mysql.tables_priv,mysql.db,mysql.procs_priv,mysql.host
      Replicate_Wild_Do_Table:
  Replicate_Wild_Ignore_Table:
                   Last_Errno: 0
                   Last_Error:
                 Skip_Counter: 0
          Exec_Master_Log_Pos: 673847271
              Relay_Log_Space: 673847506
              Until_Condition: None
               Until_Log_File:
                Until_Log_Pos: 0
           Master_SSL_Allowed: No
           Master_SSL_CA_File:
           Master_SSL_CA_Path:
              Master_SSL_Cert:
            Master_SSL_Cipher:
               Master_SSL_Key:
        Seconds_Behind_Master: 0
Master_SSL_Verify_Server_Cert: No
                Last_IO_Errno: 0
                Last_IO_Error:
               Last_SQL_Errno: 0
               Last_SQL_Error:
```

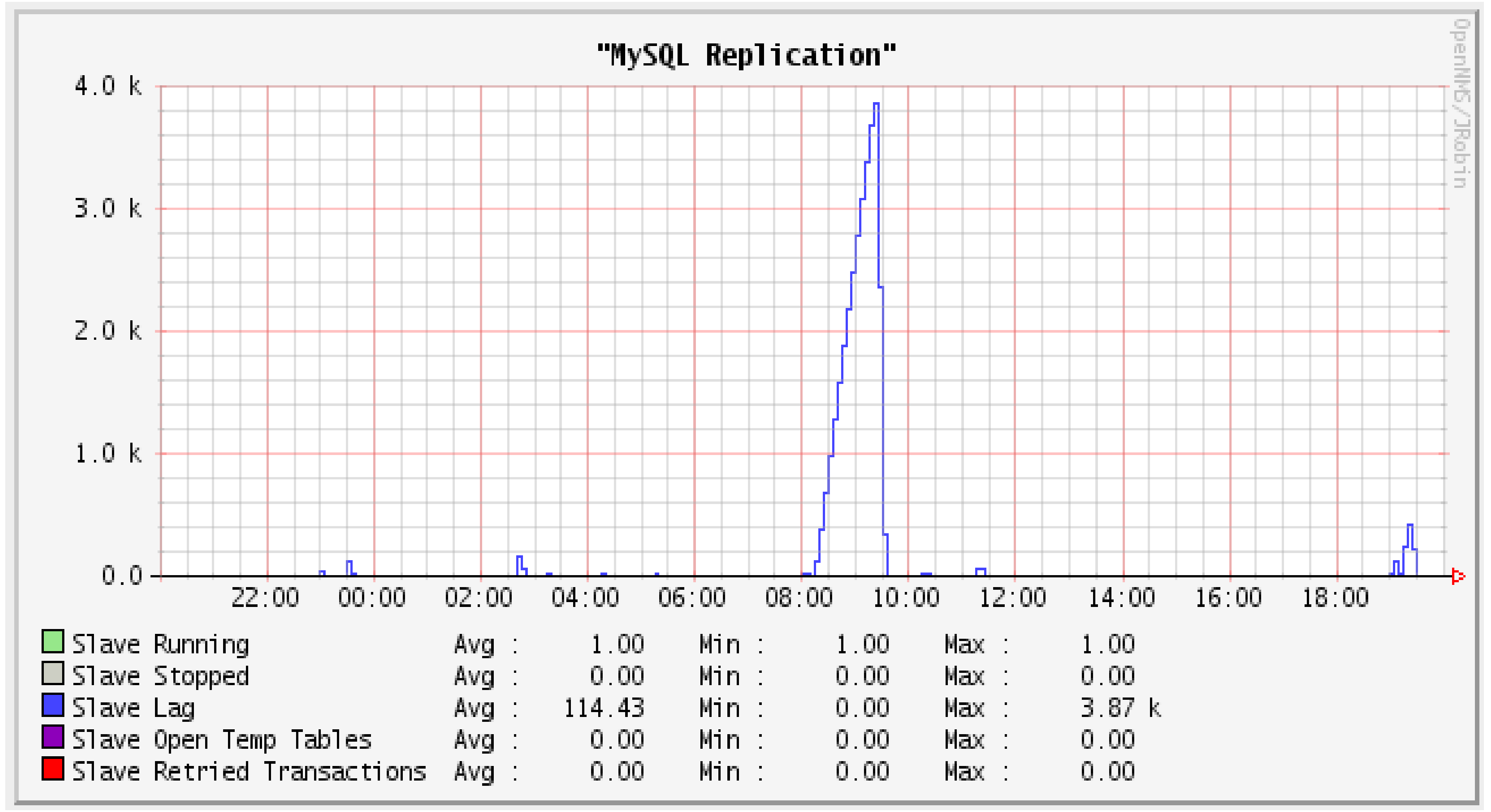
IO thread health status

SQL thread health status

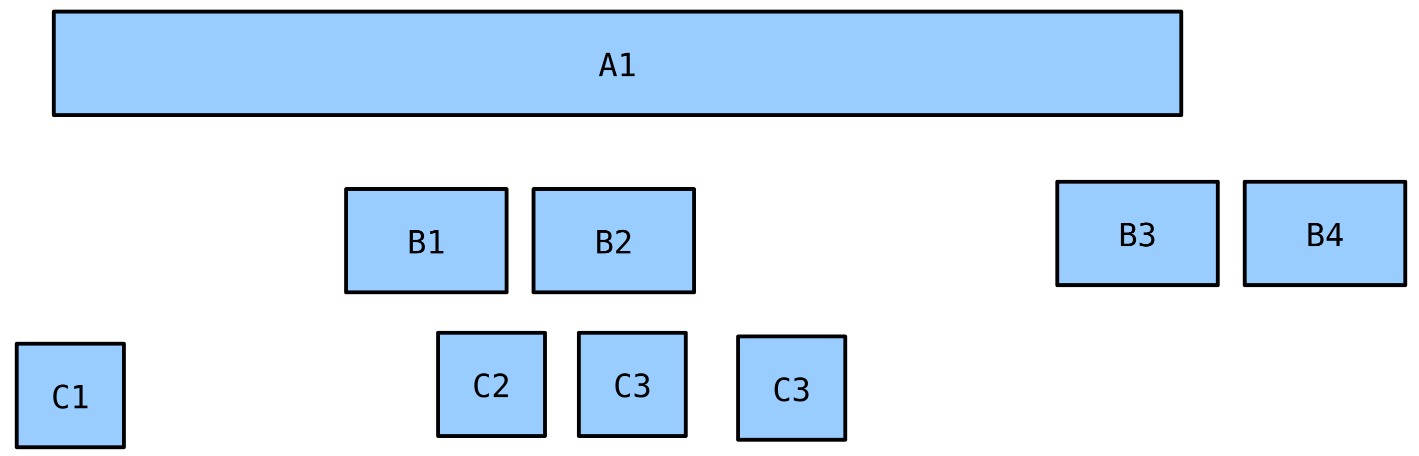
General health status

Seconds Behind Master

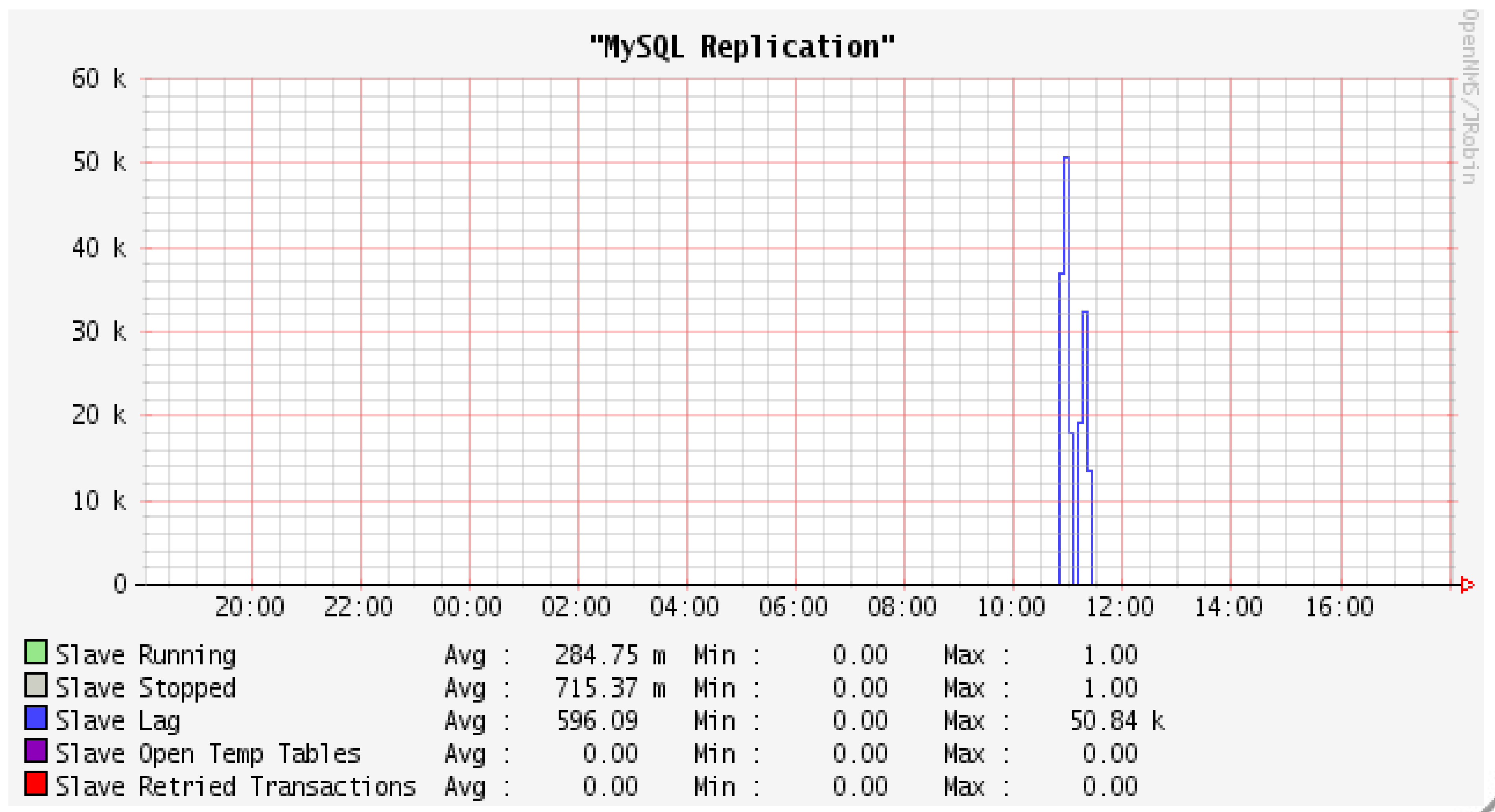
- What happens when storing BLOBs and loading them in batches



```
🟩 Slave Running             Avg :    1.00   Min :    1.00   Max :   1.00
▨ Slave Stopped              Avg :    0.00   Min :    0.00   Max :   0.00
🟦 Slave Lag                 Avg :  114.43   Min :    0.00   Max :   3.87 k
🟪 Slave Open Temp Tables    Avg :    0.00   Min :    0.00   Max :   0.00
🟥 Slave Retried Transactions Avg :   0.00   Min :    0.00   Max :   0.00
```



- SBC is based on the timestamp for the transaction
  - You can get *crazy* values based on the actual traffic
  - Is this a bad situation?
  - How do *master_log_file* and *read_master_log_pos* look like?



```
🟩 Slave Running             Avg :  284.75 m  Min :   0.00   Max :   1.00
▨ Slave Stopped              Avg :  715.37 m  Min :   0.00   Max :   1.00
🟦 Slave Lag                 Avg :  596.09    Min :   0.00   Max :  50.84 k
🟪 Slave Open Temp Tables    Avg :    0.00    Min :   0.00   Max :   0.00
🟥 Slave Retried Transactions Avg :   0.00    Min :   0.00   Max :   0.00
```

Bytes Behind Master

- Not provided directly
  - On the master: SHOW MASTER STATUS, SHOW BINARY LOGS

```
show master status; show binary logs;
+------------------+-----------+--------------+------------------+
| File             | Position  | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+-----------+--------------+------------------+
| mysql-bin.009734 | 153545495 |              |                  |
+------------------+-----------+--------------+------------------+

+------------------+------------+
| Log_name         | File_size  |
+------------------+------------+
   ....
| mysql-bin.009730 | 1073764076 |
| mysql-bin.009731 | 1073772807 |
| mysql-bin.009732 | 1073761932 |
| mysql-bin.009733 | 1073756776 |
| mysql-bin.009734 |  153545495 |
+------------------+------------+
```

  - On the slave: SHOW SLAVE STATUS
- Challenges
  - Not easy way to get information from the master, but only need past files info
  - Master position is a moving target
  - ROW vs STATEMENT vs MIXED replication
  - Example: Data purges → DELETE … FROM table WHERE ...

Replication Capacity Index

- Based on Estimating Replication Capacity blog by Percona
  - Estimate the capacity of the slave to keep up with the master load
- Some bash scripts and real data

  -
    ```
    #!/bin/bash
    # Test RCI (Replication Capacity Index)
    echo "$(date +%Y%m%d-%H%M%S) - Starting test"
    mysql -e "stop slave"
    sleep 600
    mysql -e "start slave"
    ```

  -
    ```
    while true; do
    echo $(date +%Y%m%d-%H%M%S) - `mysql -e "show slave status\G" | grep -i seconds` >> test.log
    sleep 10
    done
    ```

RCI (cont)

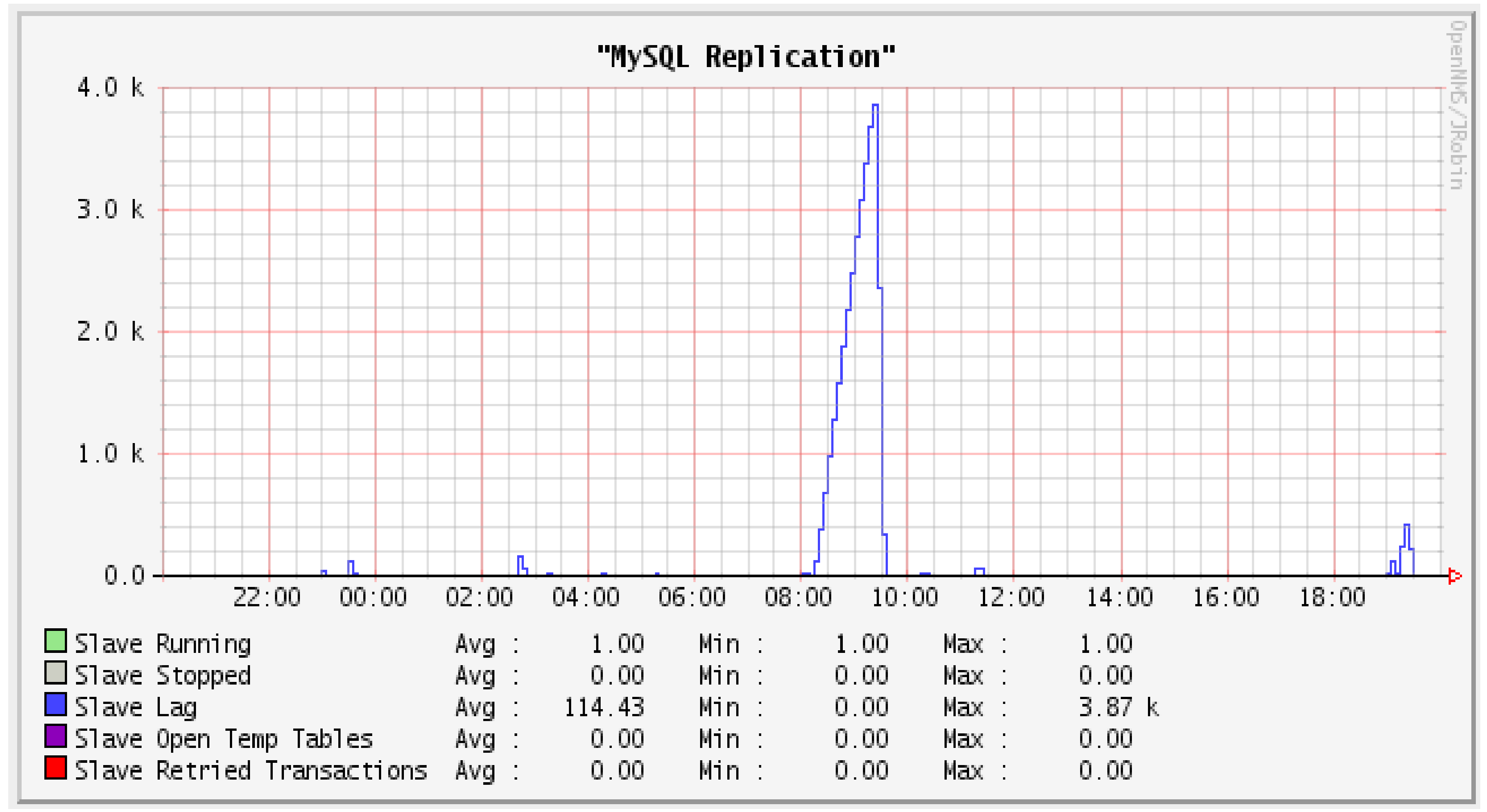
- (CONT.)

  -
    ```
    20100729-205134 - Seconds_Behind_Master: 0
    20100729-205140 - Starting test    --> Initial timestamp
    20100729-205144 - Seconds_Behind_Master: NULL
    …
    20100729-210134 - Seconds_Behind_Master: NULL
    20100729-210144 - Seconds_Behind_Master: 161
    20100729-210154 - Seconds_Behind_Master: 0 --> Last timestamp
    ```

	Pause	Start TS	1st TS	SBM	2nd TS	Diff 1	Diff 2	RCI
044	00:10:00	20:51:40	21:01:44	161	21:01:54	00:10:04	00:10:14	43.9
045	00:10:00	17:32:13	17:42:17	320	17:42:27	00:10:04	00:10:14	43.9
005	00:10:00	15:37:12	15:47:21	441	15:47:41	00:10:09	00:10:29	21.7
001	00:10:00	18:54:28	19:04:33	520	19:04:53	00:10:05	00:10:25	25.0
002	00:10:00	18:02:32	18:12:39	389	18:12:49	00:10:07	00:10:17	36.3

RCI (cont)

- Revisiting the replication delay chart
  - Lt: Time while replication falls behind
  - Rt: Time it takes for replication to catch up
  - RCI = Rt/Lt



```
🟩 Slave Running              Avg :    1.00   Min :    1.00   Max :    1.00
⬜ Slave Stopped              Avg :    0.00   Min :    0.00   Max :    0.00
🟦 Slave Lag                  Avg :  114.43   Min :    0.00   Max :    3.87 k
🟪 Slave Open Temp Tables     Avg :    0.00   Min :    0.00   Max :    0.00
🟥 Slave Retried Transactions Avg :    0.00   Min :    0.00   Max :    0.00
```

Replication Heartbeat

- Using Maatkit's mk-heartbeat
  - Run on the active master with -update option
  - Run on the slaves with -monitor or –check option
  - Output similar to Linux' *uptime*

```
mk-heartbeat --monitor --host localhost --database maatkit
  18s [  2.85s,  0.57s,  0.19s ]
  19s [  3.17s,  0.63s,  0.21s ]
  20s [  3.50s,  0.70s,  0.23s ]
  18s [  3.80s,  0.76s,  0.25s ]
  16s [  4.07s,  0.81s,  0.27s ]
```

- Issues
  - Highly sensitive to clocks in the master and slave(s) being in sync
  - It has to run on the active master in master-to-master setups
  - Better than seconds behind master

How To Monitor?

- There is no silver bullet
  - Avoid noise alerts
- Know your monitoring system
  - Tools: OpenNMS (SNMP), MONyog, MySQL Enterprise, home grown
  - Don't rely on just one
- Alarms
  - Thresholds and hysteresis
  - Number of incidents until it alarms
  - Sampling intervals
- Know your load
  - Low / High traffic? Bursts?
  - Small / big transactions? Concurrency?
- Replication type
  - Row / Statement / Mixed

O'REILLY MySQL Conference & Expo

Thank you very much