MySQL Replication : advanced features in all flavours

Giuseppe Maxia

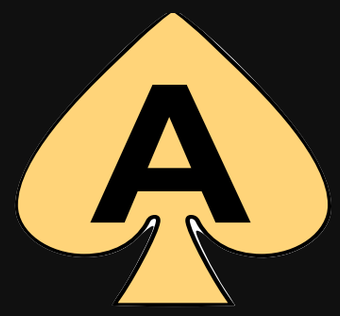
Quality Assurance Architect at VMware

@datacharmer

About me

Who's this guy?

- Giuseppe Maxia, a.k.a. "The Data Charmer"
  - QA Architect at VMware
  - 25+ years development and DB experience
  - Long timer MySQL community member.
  - Oracle ACE Director
  - Blog: http://datacharmer.blogspot.com
  - Twitter: @datacharmer



continuent by vmware

Supporting material

- MySQL Replication monitoring 101
- http://bit.ly/repl-mon-101

- Slides:
- http://bit.ly/adv-repl-all-flavors-2015

continuent by vmware

Summary

What will we see in this tutorial

- The concepts of Replication
- Why monitoring matters
- Global Transaction Identifiers
- Multi source replication
- Parallel replication

continuent by vmware

Actors

We will see practical examples with the following systems

- MySQL 5.6.24
- MySQL 5.7.7
- MariaDB 10.0.17
- Tungsten Replicator 3.0.1

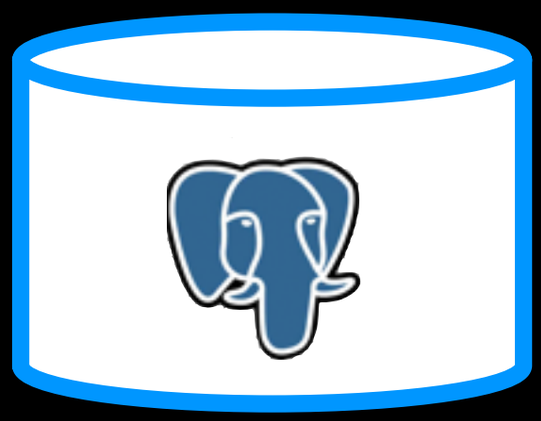
Why MySQL?

The world today is
dominated by the
web economy

Databases are the backbone of the web economy

What database for the web?

Oracle: The most **powerful** database



Postgresql: The most **advanced** open source database

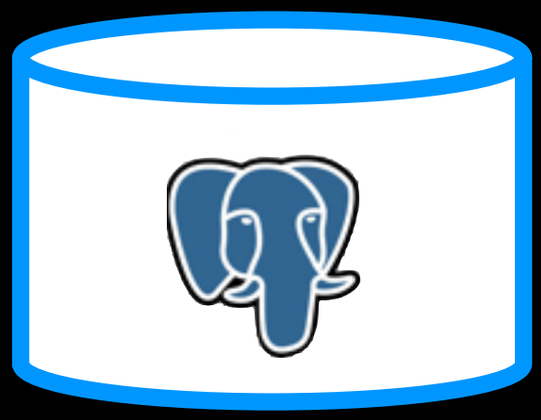
SQLite: The most **deployed** open source database

MySQL: The most **popular** open source database

Actually, **MySQL** databases are the backbone of the web economy

What database for the web?

No built-in replication



No built-in replication

No built-in replication

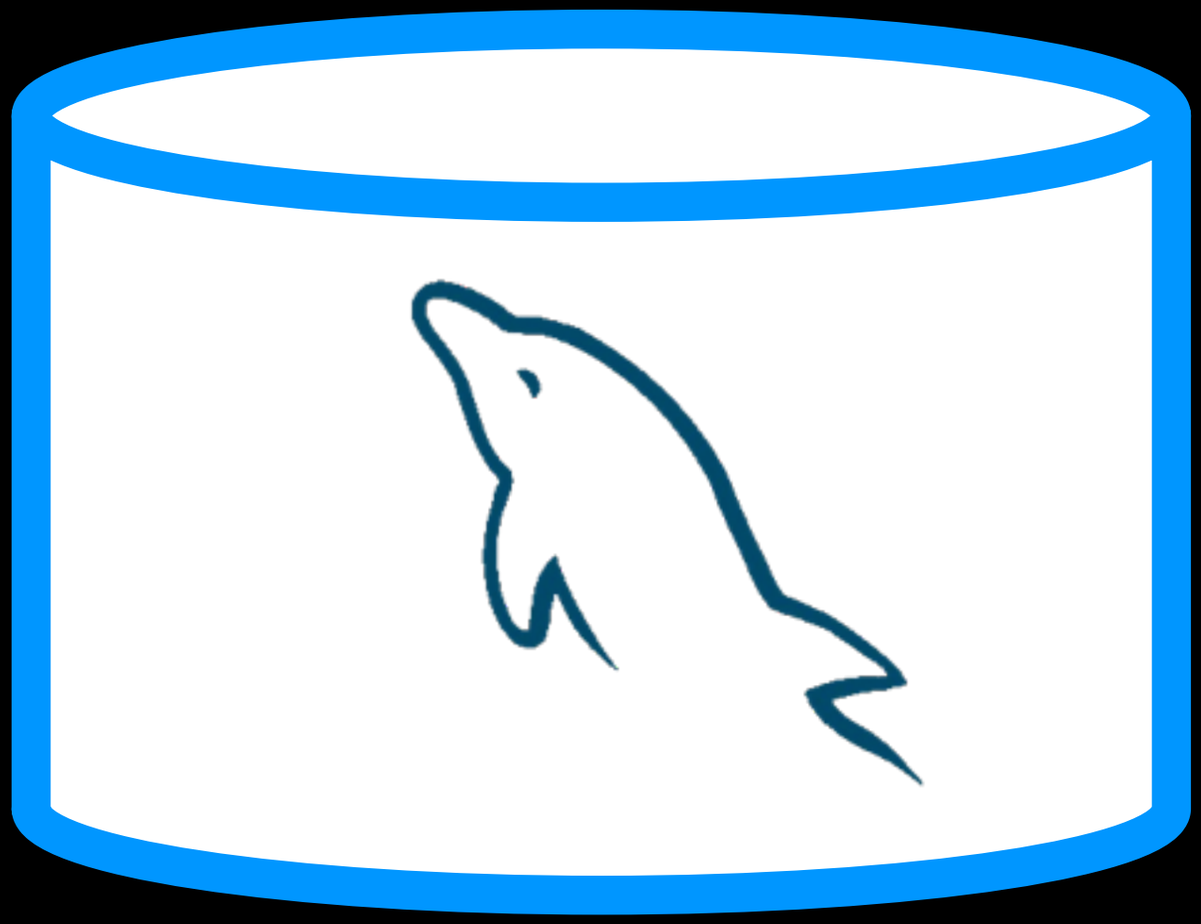
Built-in replication

More precisely, **MySQL REPLICATION** is the backbone of the web economy

MySQL

A community tool for the enterprise.
How did it happen?

WHY?

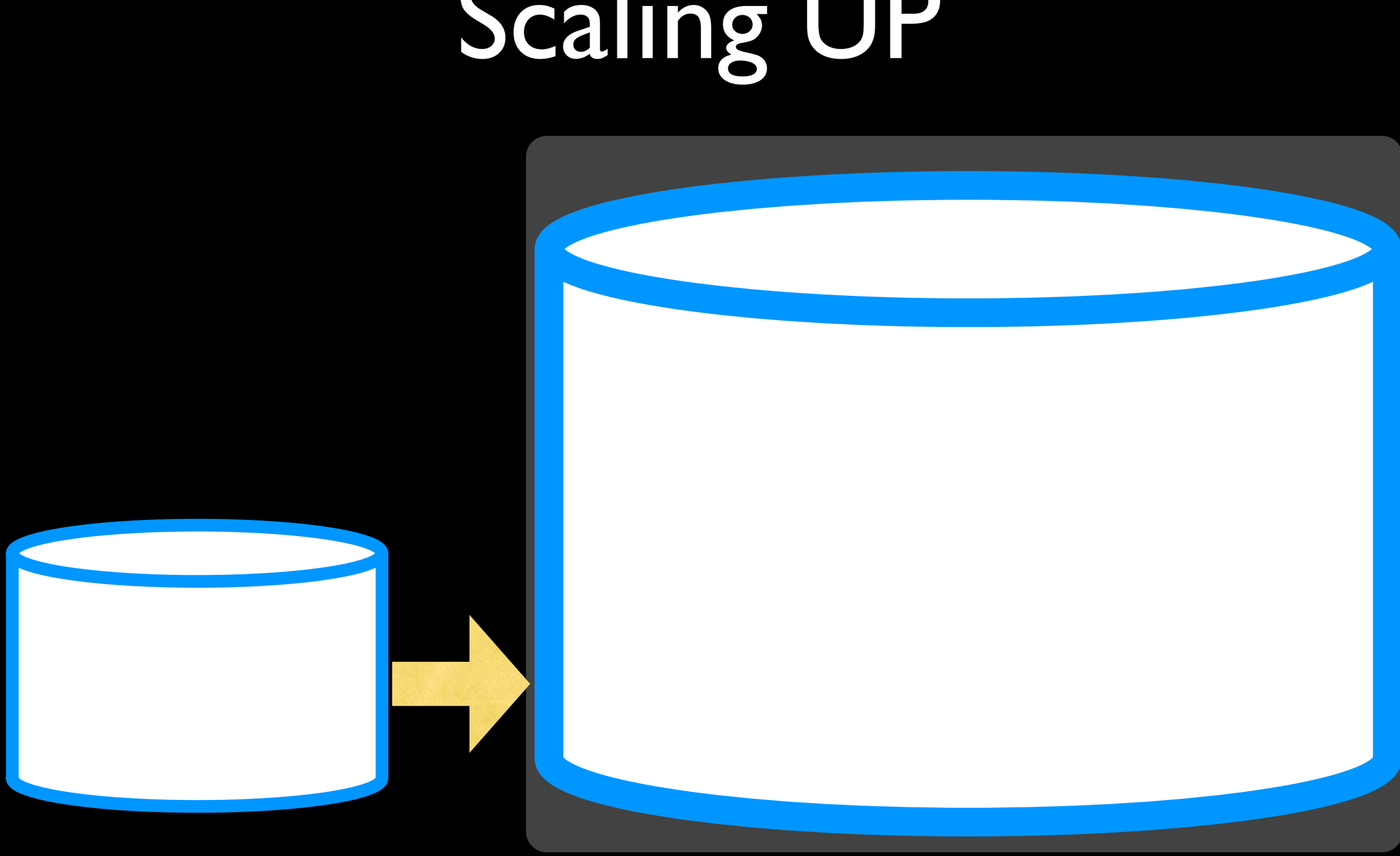


The building block of the web economy

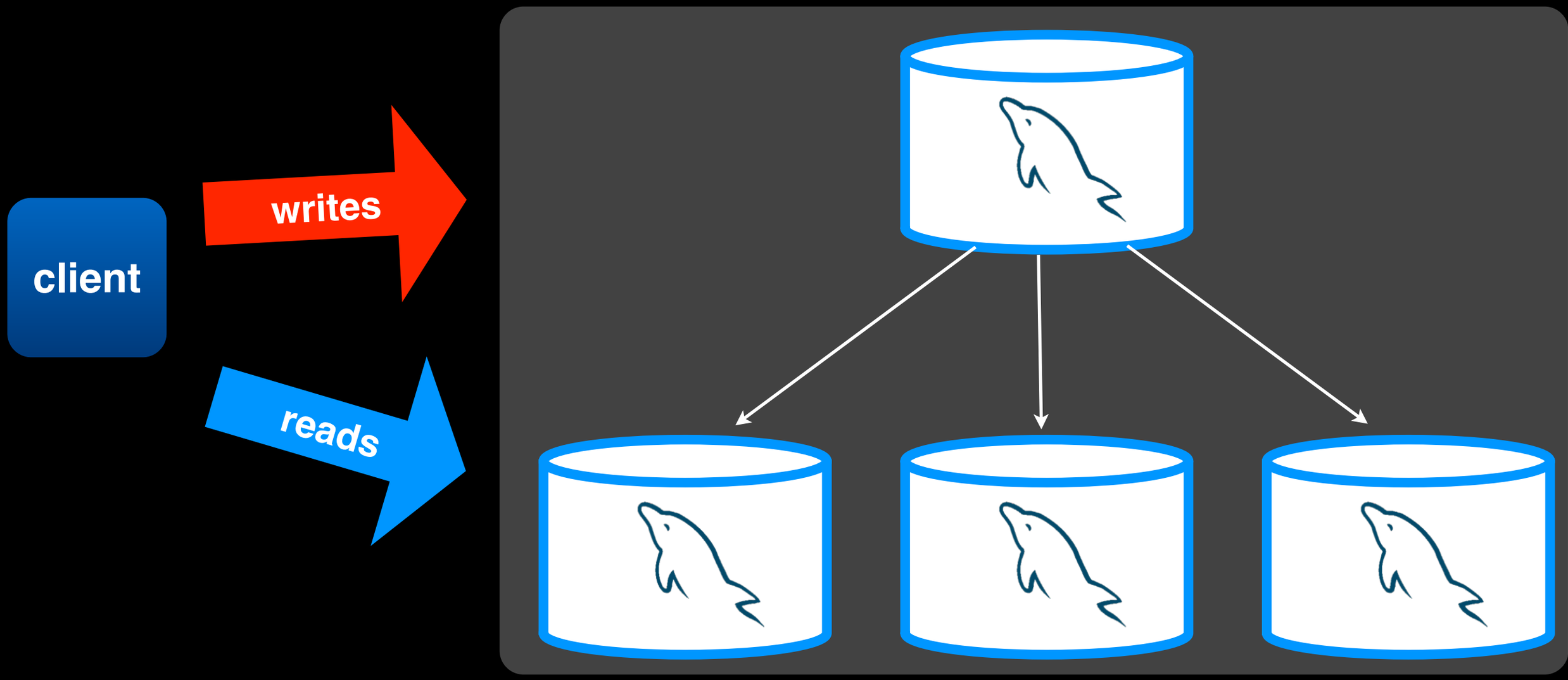
Why is MySQL popular?

- PERFORMANCE : small, agile, fast!
- RELIABILITY: keeps your data safe, runs forever
- EASE OF USE: up and running in minutes!
- SCALABLE
- FRIENDLY

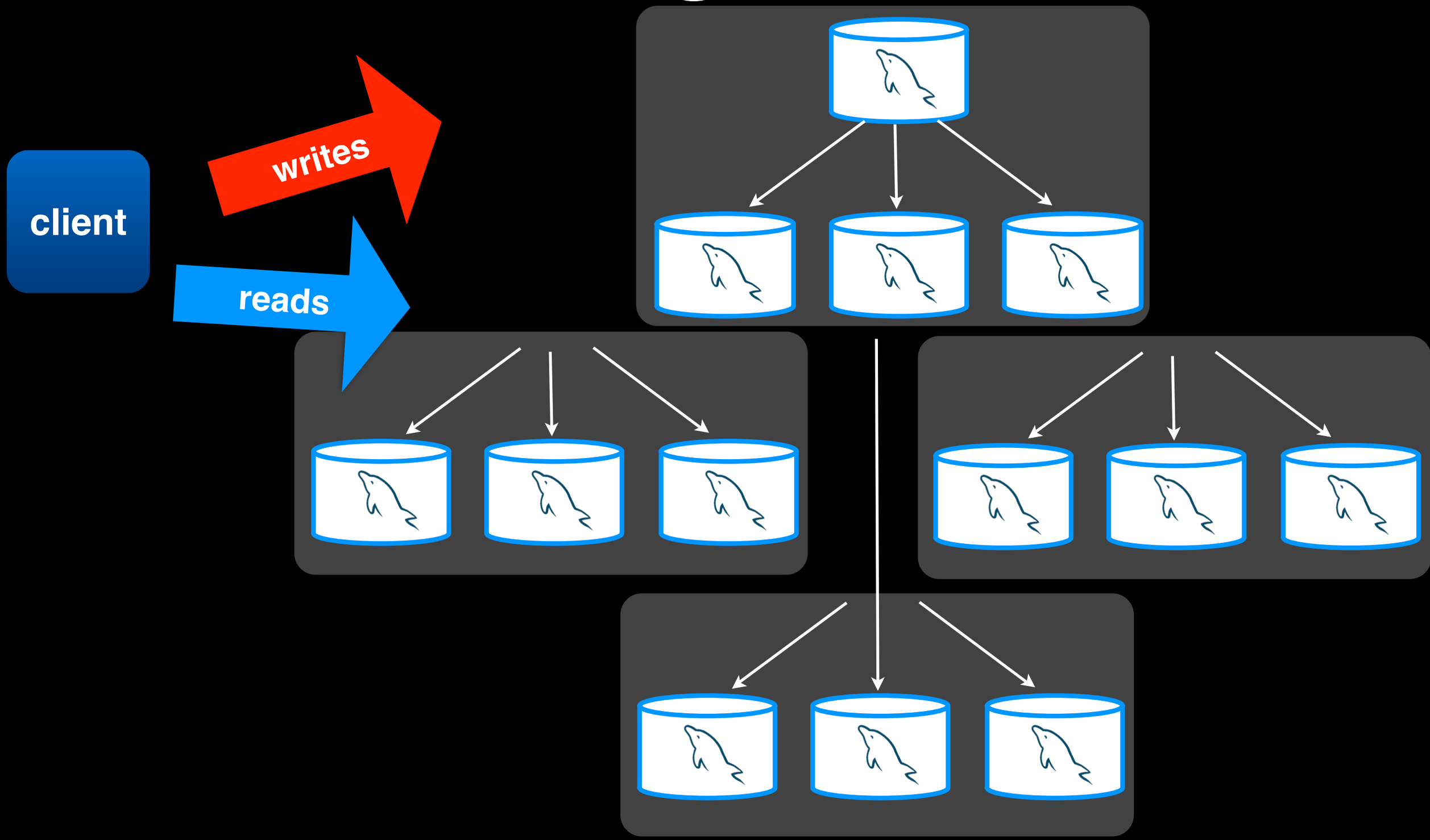
Scaling UP



Scaling OUT



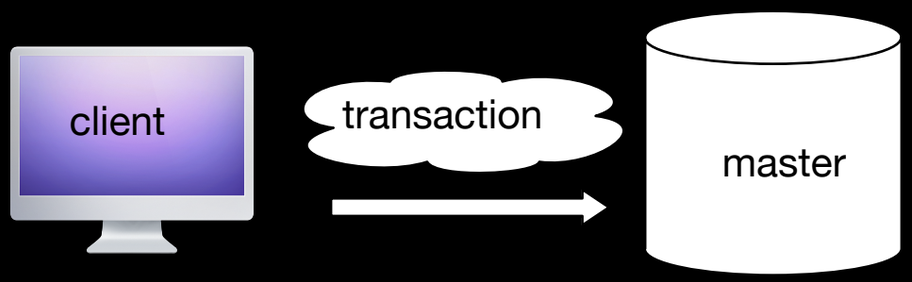
Scaling OUT

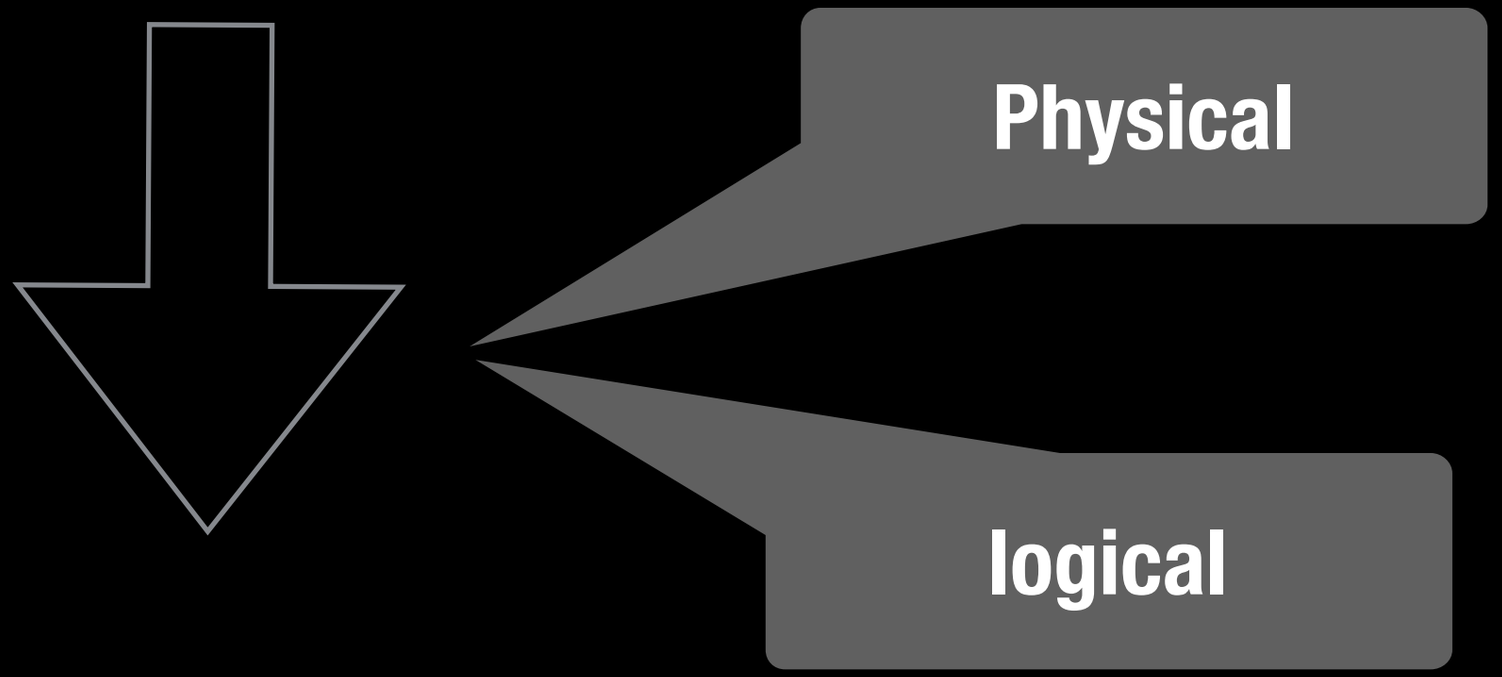


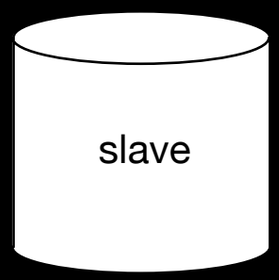
Replication in a nutshell

The concepts of replication

The basics

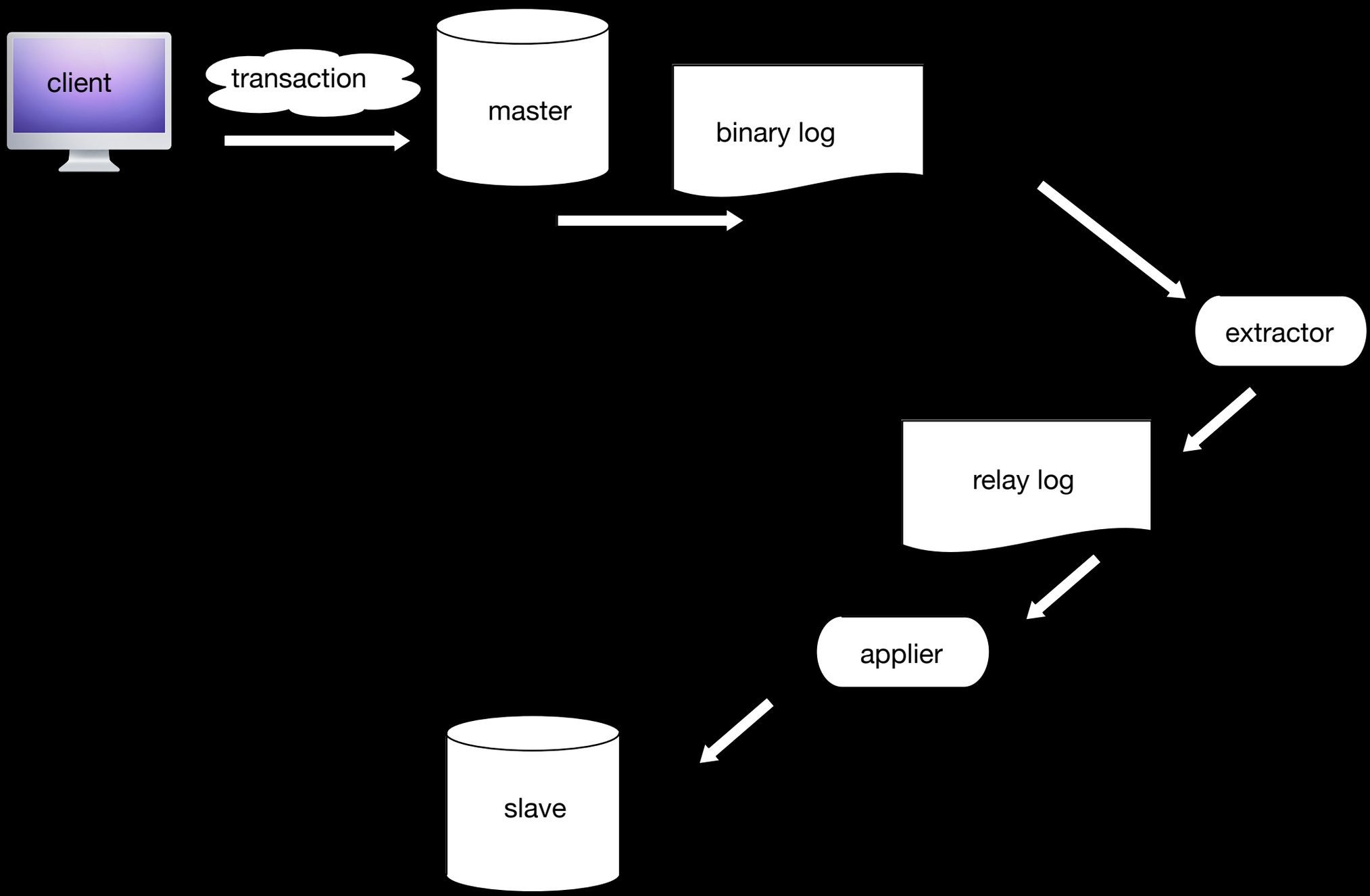






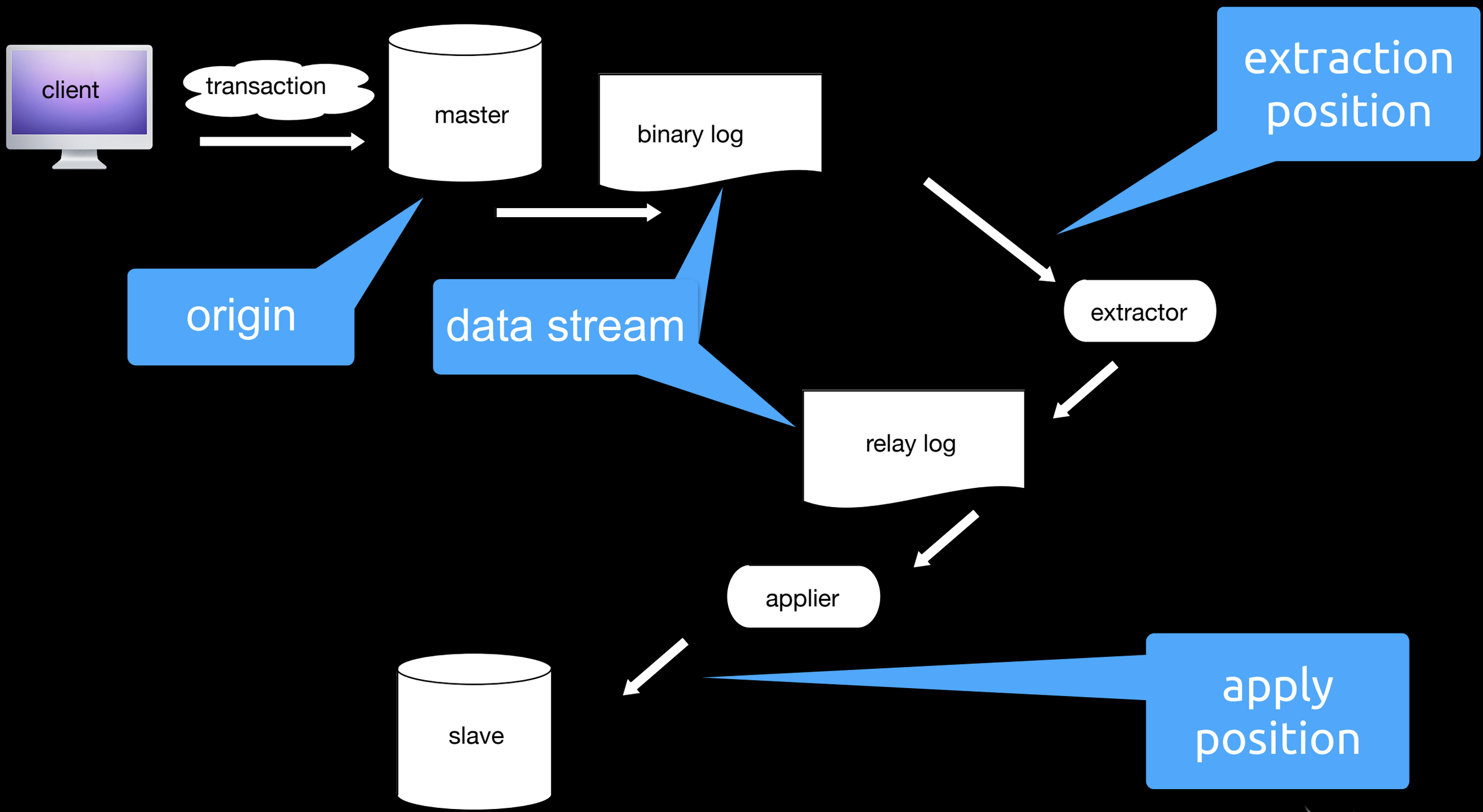
The concepts of replication

The basics



The concepts of replication

The basics



Focus on monitoring

Why we will see lots of monitoring concepts

- Monitoring tells you if replication is working properly
- It also show us how some features work
- If we can see the data moving, we understand it better

Focus on monitoring

The most important reason:

- Replication will fail, sooner or later.
- Good monitoring metadata is what can tell you what the problem is

MySQL replication monitoring

There is more than one way of knowing if replication is working

- Sentinel data : tap tap, is this thing working?
- Monitoring: are you still there?
- Latency: are you catching up?
- Status persistence: are you OK, dear?
- Completeness: did you miss anything?
- Checksum probes: have you got it all?

Sentinel data : tap tap, is this thing on?

The simplest, system-independent method of testing replication

1. Make sure the data is not in the master or in the slave
2. Write the data to the master
3. Retrieve the data in the slave
4. Modify the data in the master
5. Check the changes in the slave
6. Delete the data from the master
7. Make sure it was also deleted in the slave

Sentinel data: what is it and isn't

Caveats

- It tells you if replication **CAN** work
- It won't tell you if replication works **ALWAYS**
- It won't tell you if **all your data** is replicated

Monitoring: are you still there?

Objectives of monitoring :

1. Making sure that the slave is replicating from the intended master.
2. Checking that the slave is replicating from the right binary logs.
3. Checking that the data from the master is transferred to the slave.
4. Checking that the slave is applying data without errors.
5. Checking that the slave is keeping up with the master

How monitoring works

To monitor effectively, we need to cover all bases

- We need to know :
  - who the master is
  - what the master is doing
  - what the slave is doing
- And then compare what we have got

Who is the master?

Get information about the master, from the master server

```
mysql> show variables like 'port';
    +---------------+-------+
    | Variable_name | Value |
    +---------------+-------+
    | port          | 22786 |
    +---------------+-------+
```

Who is the master?

Get information about the master, from the master server

```
mysql> show variables like 'hostname';
    +---------------+-----------+
    | Variable_name | Value     |
    +---------------+-----------+
    | hostname      | localhost |
    +---------------+-----------+
```

What is the master doing?

Get information about the master, from the master server

```
mysql> show master status\G
*************************** 1. row ***************************
             File: mysql-bin.000003
         Position: 5149170
     Binlog_Do_DB:
 Binlog_Ignore_DB:
1 row in set (0.00 sec)
```

What is the slave doing?

This means, usually, running “SHOW SLAVE STATUS”

```
SHOW SLAVE STATUS\G
```

```
Master_Host: 127.0.0.1

Master_Port: 22786

Master_Log_File: mysql-bin.000003

Read_Master_Log_Pos: 5149170

Relay_Log_Pos: 2060153

Relay_Master_Log_File: mysql-bin.000003

Slave_IO_Running: Yes

Slave_SQL_Running: Yes

Exec_Master_Log_Pos: 2060007

Relay_Log_Space: 5149528
```

Latency

Are you catching up?

- Delta between
  - Time of commit in the master
  - Time of apply in the slave
- Can be measured with a simple *sentinel* system
  1. Insert a high res timestamp in the master
  2. Retrieve the record from the slave
  3. Measure the interval
  4. Subtract the commit time.

Status persistence

Assume your servers will fail. Prepare for it

- Problem:
  - When server crashes and resumes, replication status on file may not be in sync
- Solution:
  - Crash-safe slave tables
  - Replication status is kept in the database

Completeness: did you miss anything?

Using filters during extraction or apply can have side effects

- Data can be removed by filters
  - in the master bin log (never gets to the slaves)
  - in the slaves (apply rules)
- Data can be modified
  - for heterogeneous replication
  - for ETL tasks
- **If you have filters, your slave CAN`T become master.**

Checksum probes: have you got it all?

Checking that replicas have the right data is a sound idea

- total probes (expensive: may stop or slow down operations)
- incremental probes (take long time, but have low impact on operations)
- Many methods. A popular one is pt-table-checksum from Percona Toolkit

Global Transaction Identifiers

continuent by vmware

Transactions blues

You think you know where your transactions are … until something unexpected happens

- Problem:
  - MySQL replication identifies transactions with a combination of binary log file name and offset position;
  - When using many possible masters, file names and positions may differ.
  - Practical cases: failover, circular replication, hierarchical replication
- Solution: use a global ID, not related to the file name and position

Implementation: (1) MySQL 5.6 & 5.7

A half baked feature, which kind of works

- Made of server UUID + transaction ID
  - (e.g.: “e8679838-b832-11e3-b3fc-017f7cee3849:1”)
- Only transactional engines
- No “create table … select …” supported
- No temporary tables within transactions
- Requires **log-slave-updates** in all nodes

Implementation: (1) MySQL 5.6 & 5.7

A half baked feature, which kind of works

- The good
  - GTID are easily parseable by scripts in the binlog
  - Failover and transaction tracking are easier
- The bad
  - Not enabled by default
  - Hard to read for humans!
  - Little integration between GTID and existing software (ignored in crash-safe tables, parallel replication)
  - makes log-slave updates mandatory

GTID in MySQL 5.7.6+

Something was changed ...

- GTID can now be enabled dynamically.
- However, it requires a 9 (NINE!) steps procedure.
- http://mysqlhighavailability.com/enabling-gtids-without-downtime-in-mysql-5-7-6/

Implementation (2) MariaDB 10

A well thought feature, with some questionable choices

- Made of domain ID+server ID + number
  - e.g. (0-101-10)
- Enabled by default
- Uses a crash-safe table
- No limitations
- Lack of integration with old replication coordinates.

Implementation (3) Tungsten Replicator

The oldest implementation, well designed and integrated

- Made of service name + transaction number
  - e.g. “alpha 1”
- Integrated with crash-safe tables
- Integrated with old coordinates
- Provides extra information about each transaction
- No limitations

GTID demo

Monitoring (MySQL 5.6+ - MariaDB 10)

continuent by vmware

The new trend : using tables to monitor

(Almost) all data is now in tables

- Both MySQL and MariaDB 10 can monitor replication using tables.
- Well ... almost all

MySQL 5.6 crash-safe tables

There are tables that can replace files, and SHOW statements ... up to a point

- up to 5.5:
  - SQL in the slave
    - show slave status
  - SQL in the master
    - show master status
- 5.6 & 5.7:
  - Tables in the slave
    - slave_master_info
    - slave_relay_log_info
  - SQL in the master
    - show master status
    - select @@global.gtid_executed

MySQL tables

Very detailed, but designed in different stages

- One table replaces the file master.info
- Another replaces relay-log.info
- They were designed before introducing GTID
- There IS NO GTID in these tables
- They are not updated continuously

MySQL 5.7 additional tables

Performance Schema helps with monitoring
(NOTE: table names changed in 5.7.6)

- replication_applier_configuration
- replication_applier_status
- replication_applier_status_by_coordinator
- replication_applier_status_by_worker
- replication_connection_configuration
- replication_connection_status

MariaDB 10 crash-safe tables

A complete redesign of the monitoring system, integrated with GTID

- up to 5.5:
  - SQL in the slave
    - show slave status
  - SQL in the master
    - show master status
- 10.0
  - Table in the slave
    - gtid_slave_pos
  - SQL in the master
    - show master status
    - select @@gtid_current_pos

Monitoring demo

Multi-source replication

What is it?

The dream of every DBA is to have a group of database servers that behave like a single server

- Traditional replication allows master/slave and chain replication (a.k.a. circular or ring)
- Up to MySQL 5.6, a slave cannot have more than one master.
- Multi source is the ability of replicating from more than one master at once.
- Implemented in Tungsten Replicator (2009), MySQL 5.7 (2015), MariaDB 10 (2013).

Implementation (1) MySQL 5.7.7

An evaluation release, which might not be in the final GA

- New syntax: CHANGE MASTER TO … FOR CHANNEL “name”
- SHOW SLAVE STATUS FOR CHANNEL “name”
- START/STOP SLAVE FOR CHANNEL “name”
- Includes replication tables in performance_schema
- Requires log-slave-updates for monitoring

implementation (2) : MariaDB 10

Now GA, the multi source was well planned and executed

- New syntax “CHANGE MASTER “name” …”
- START/STOP/RESET SLAVE “name”
- SHOW SLAVE “name” STATUS
- SHOW ALL SLAVE**S** STATUS

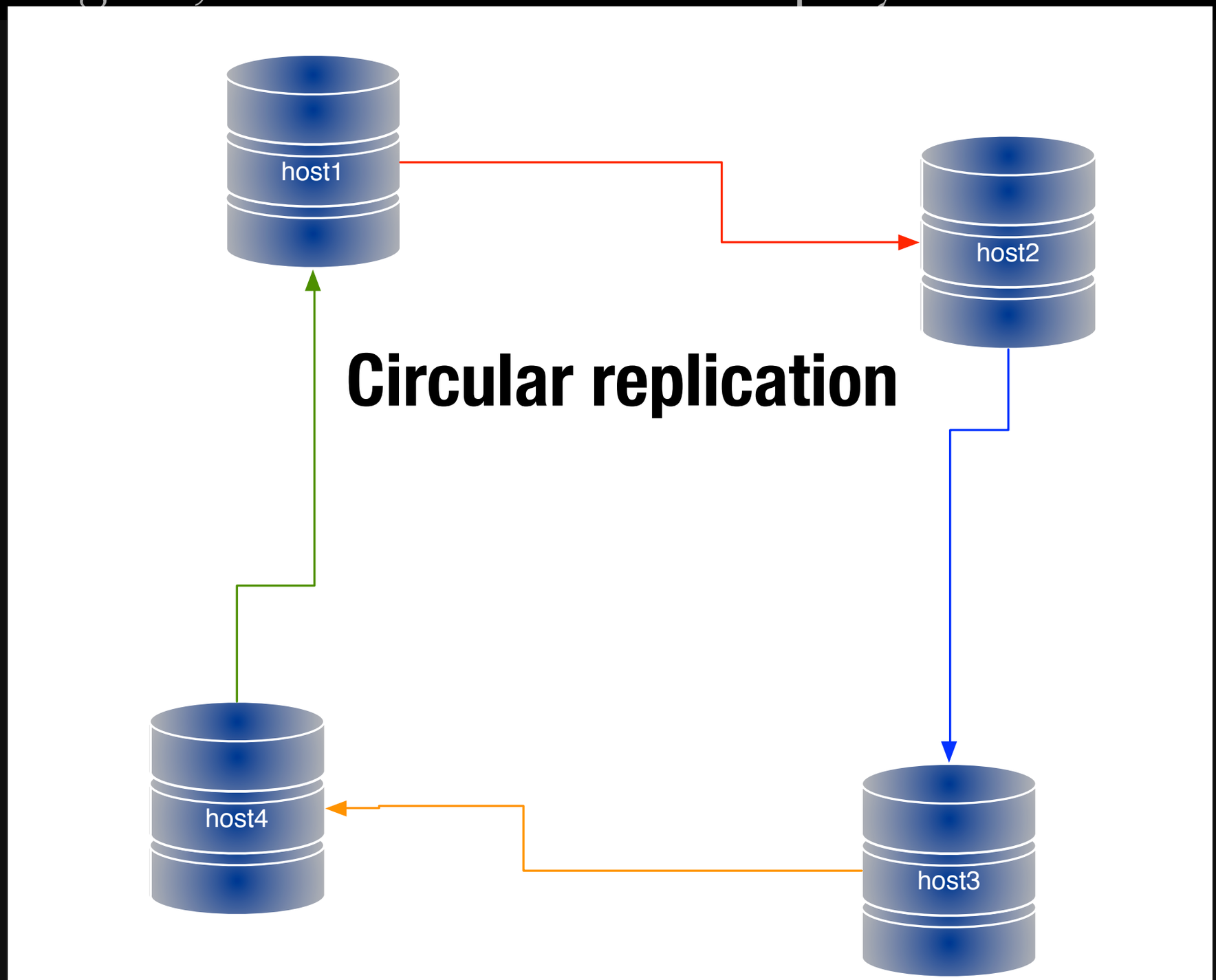
Implementation (3) Tungsten Replicator

An external tool for replication, which supports multiple topologies

- Can create pre-defined topologies, and complex ones on-the-fly
- Supports the concept of service data streams, allowing multiple streams per server
- No SQL syntax implemented. All admin work through dedicated tools.
- Full integration with GTID
- Allows both point-to-point replication and full slave replay.

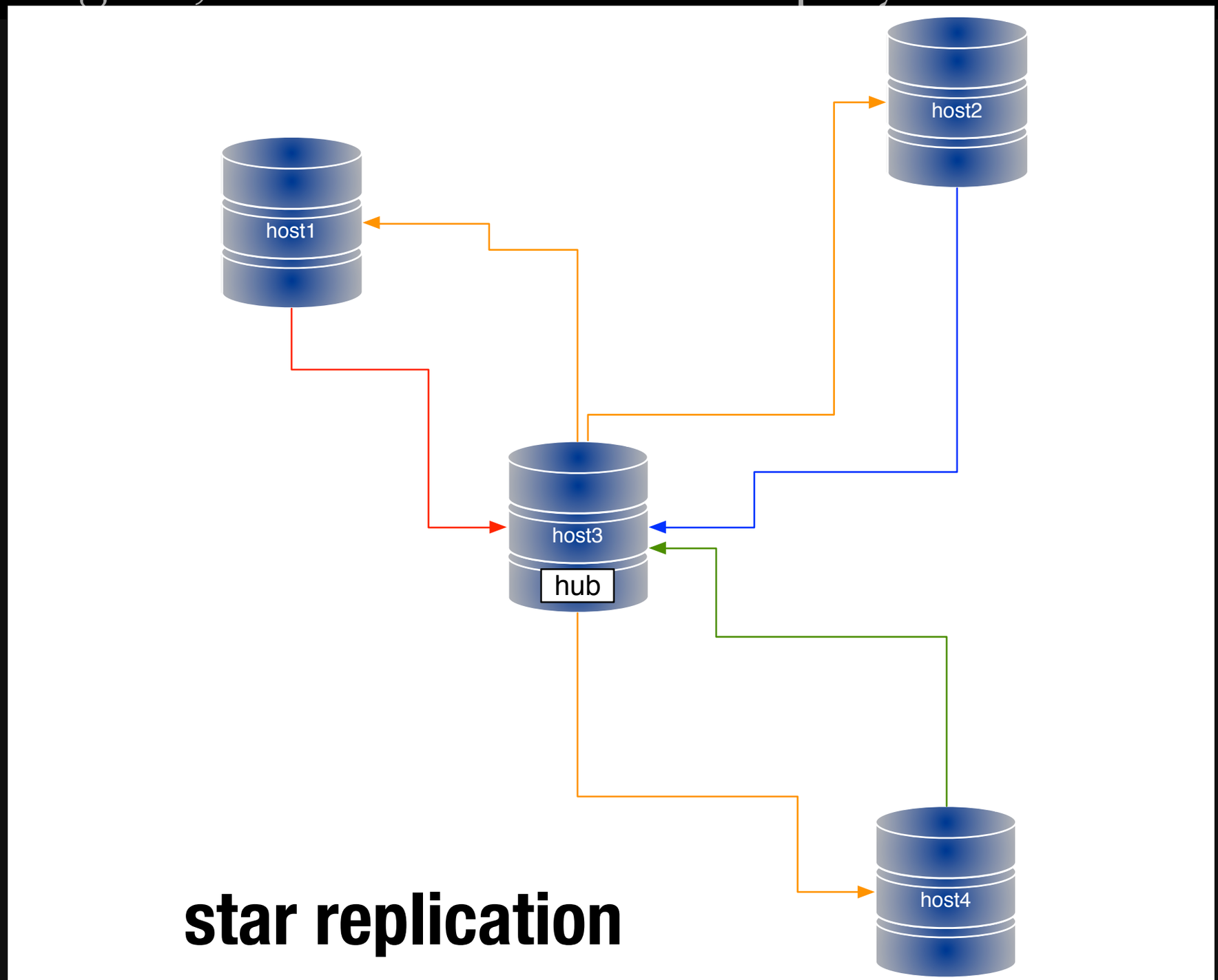
Full slave replay

When the data is applied, saved to a binary log, and then replicated again, we have a full slave replay



Full slave replay

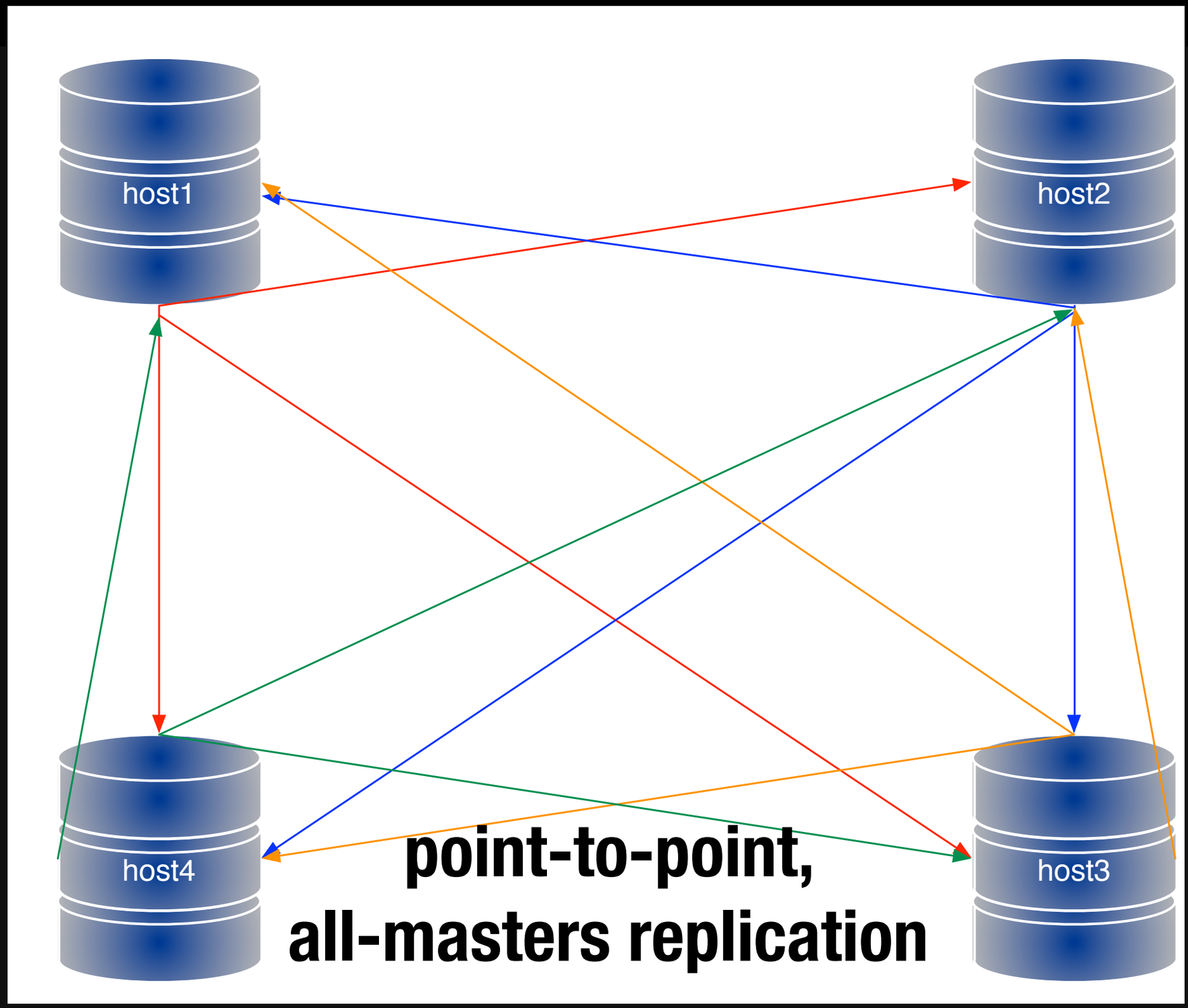
When the data is applied, saved to a binary log, and then replicated again, we have a full slave replay



star replication

Point-to-point replication

Allows data flow where the replicated data is applied only once



Multi-Source demo

Parallel replication

Parallel apply

When the slave lags, using parallel threads may speed up things

- It’s the ability of executing binary log events in parallel.
- Implemented in Tungsten Replication (2011, schema based), MySQL 5.6 (2012, schema based), MariaDB 10 (2013, boundless), MySQL 5.7 (2013, boundless)

Implementation (1) Tungsten Replicator

The granddaddy of parallel replication, happily deployed in production for years

- Based on schema boundaries.
- No risk of deadlocks.
- Can be shared by criteria other than database, but only during provisioning.
- Fully integrated in the instrumentation;
- Provides extra information for monitoring and troubleshooting

Implementation (2) MySQL 5.6

The first integrated solution for parallel replication

- Schema based, same as Tungsten.
- Requires both master and slave of the same version;
- No integration with GTID;
- No extra instrumentation.

Implementation (3) MySQL 5.7

Breaking the schema barriers

- Not schema based. Parallelism is defined by extra metadata from the master.
- Requires both master and slave of the same version;
- Uses monitoring tables in performance schema
- Limited troubleshooting info;
- With multi-source, it’s all or nothing

Implementation (4) MariaDB 10

The latest contender

- Not schema based. Uses information from the coordinator to define how to parallelise;
- Integrated with GTID;
- Little instrumentation for troubleshooting.
- You can choose to which channel to apply (set default_master_connection='x').

New development in MariaDB 10.1

A new algorithm for parallel replication

- Optimistic parallelisation
- Does not require preparation in the master
- Still in beta.

continuent by vmware

Supporting material

- MySQL Replication monitoring 101
- http://bit.ly/repl-mon-101

- Slides:
- http://bit.ly/adv-repl-all-flavors-2015

- Scripts

Useful links

- GTID in MySQL
- Performance_schema tables for replication
- GTID in MariaDB
- Multi Source in MySQL
- Multi Source in MariaDB
- Parallel Replication in MariaDB
- Parallel Replication in Tungsten

Q&A