OKC MySQL Users Group

OKC MySQL

- Discuss topics about MySQL and related open source RDBMS
- Discuss complementary topics (big data, NoSQL, etc)
- Help to grow the local ecosystem through meetups and events

Sponsored By...

MySQL Architecture

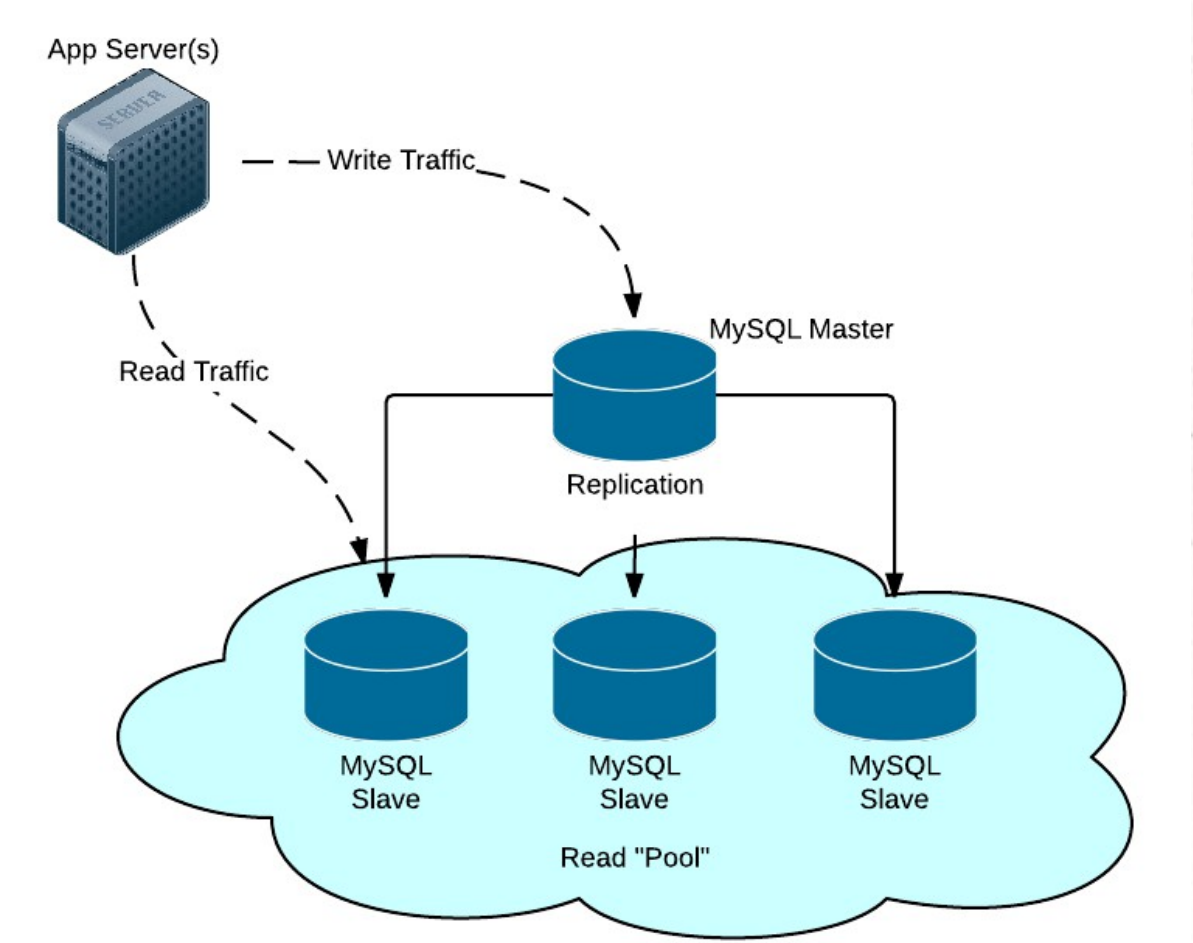
“With great power comes great responsibility”
- Voltaire (or Spiderman if you prefer)

Architecture Intro

- MySQL is relatively flexible in terms of deployment architecture
- Basis is asynchronous replication (i.e. delayed)
- Some key terms
    - Master – the server where a write originates
    - Slave – the server where the change is copied (also called a replica)
- Server can have one master* and multiple slaves

* Multi-sourced replication is coming in 5.7, but not GA yet

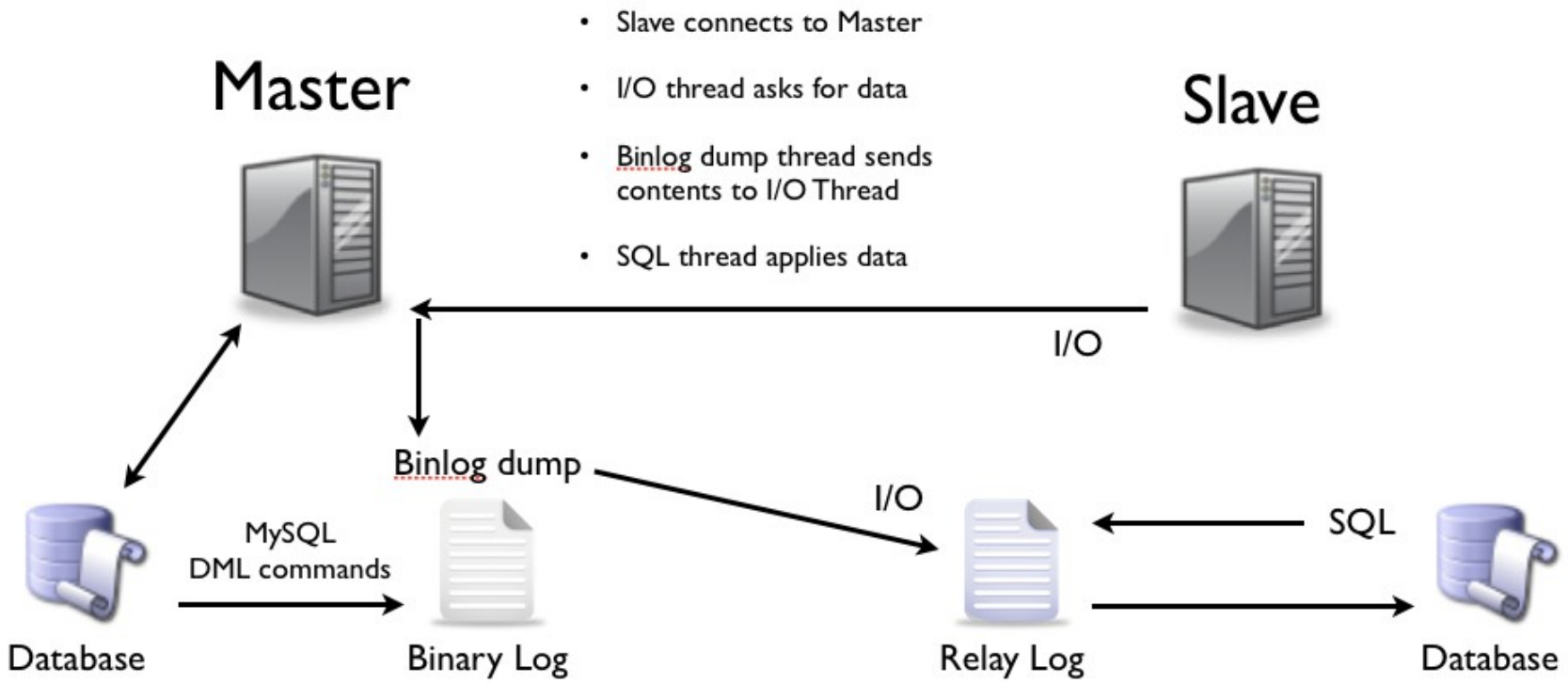
Sample Production Architecture



How it works (high level)?

- Each server has a unique server-id
- There is a feature called **the binlog**
    - After each write, the statement is written to the binlog with the server-id
    - Note that what is written is configurable,but we'll touch on that later
- Slave servers connect to the master and it simply “dumps” the binlog contents to the slaves
- Slaves replay the events from servers with different server-ids to mirror the master
- For those wanting more details, here you go...

Detailed Replication



http://scriptingmysql.wordpress.com/2013/02/21/mysql-replication-creating-a-new-masterslave-topology-with-or-without-virtual-machines/

How it looks

On the master:

```
+----+------+---------------------+------+-------------+--------+-----------------------------------------------------------------+
| Id | User | Host                | db   | Command     | Time   | State                                                           |
+----+------+---------------------+------+-------------+--------+-----------------------------------------------------------------+
| 11 | repl | 192.168.57.121:45742 | NULL | Binlog Dump | 104837 | Master has sent all binlog to slave; waiting for binlog to be updated |
| 15 | repl | 192.168.57.122:44645 | NULL | Binlog Dump | 104817 | Master has sent all binlog to slave; waiting for binlog to be updated |
| 17 | root | localhost           | NULL | Query       |      0 | init                                                            |
+----+------+---------------------+------+-------------+--------+-----------------------------------------------------------------+
3 rows in set (0.00 sec)
```

On the slave:

```
+----+-------------+-----------+------+---------+--------+-----------------------------------------------------------------+
| Id | User        | Host      | db   | Command | Time   | State                                                           |
+----+-------------+-----------+------+---------+--------+-----------------------------------------------------------------+
| 19 | system user |           | NULL | Connect | 104598 | Waiting for master to send event                                |
| 20 | system user |           | NULL | Connect | 104509 | Slave has read all relay log; waiting for the slave I/O thread to update it |
| 21 | root        | localhost | NULL | Query   |      0 | init                                                            |
+----+-------------+-----------+------+---------+--------+-----------------------------------------------------------------+
3 rows in set (0.00 sec)
```

What is replication good for?

- Scaling reads (very common use case)
- Dedicated backup machines (take backups from slave server)
- Datacenter / Geographic redundancy
- Disaster recovery

Concerns with replication

- Slaves out of sync with masters
  - Non-deterministic statements
  - Users accidentally writing to slave (i.e. human error)
- Replication delay
  - There is inherent delay as the process is asynchronous
  - This can be compounded
- Underpowered slaves
  - Masters are multi-threaded, slaves are single threaded
  - This means slaves often need to be **more** powerful than masters

More concerns

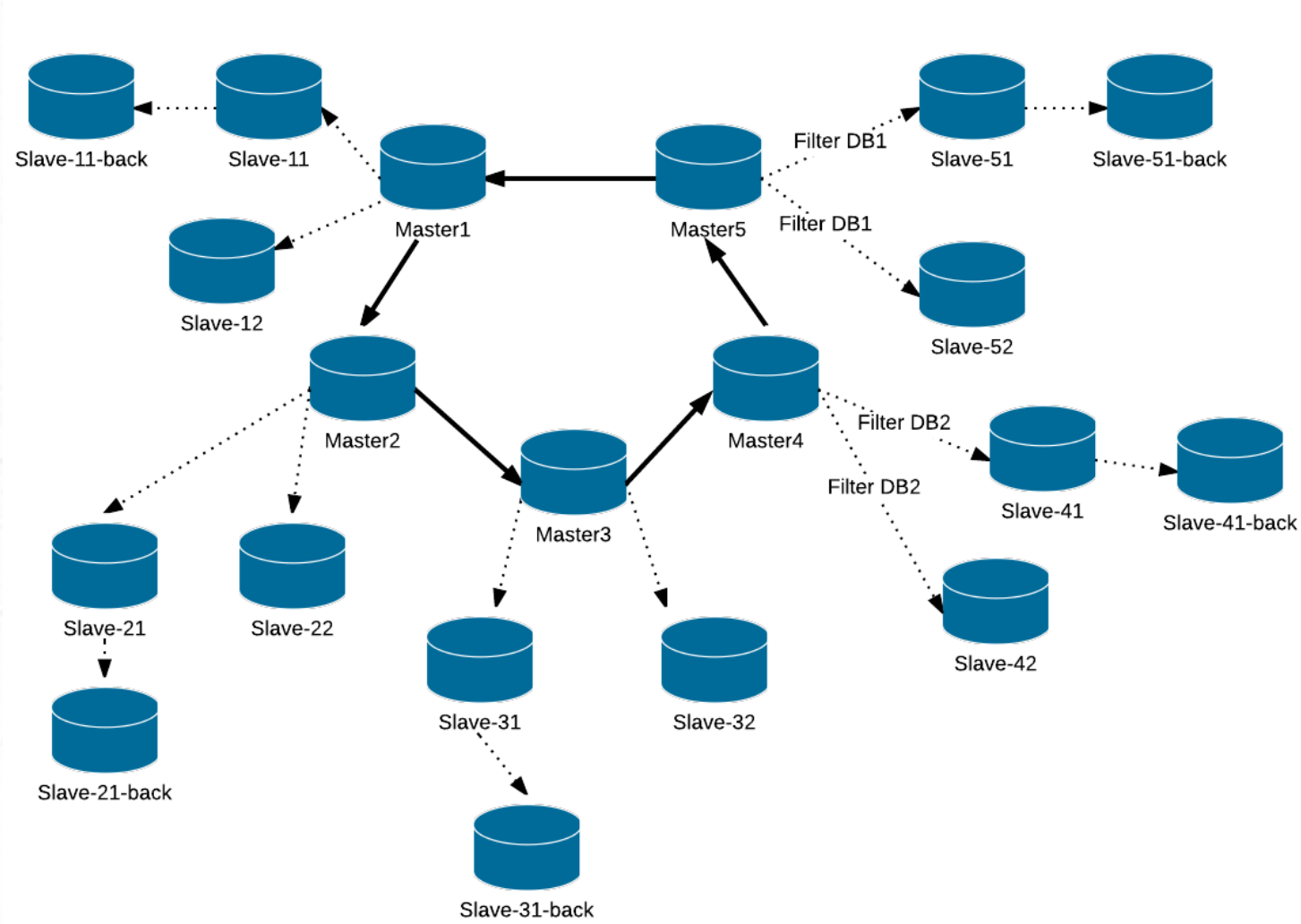
- The biggest issue that I see is due to one common theme:

Poorly designed architecture using features of replication that exist but are either misunderstood and/or used improperly.

Some advanced features

- i.e. the ones that get people into trouble...
- Circular replication
  - Very important feature for one particular use case, very bad feature for the one I see far too often
  - **You should only ever use 2 nodes in circular replication (known as master/master)**
- Replication filters
  - Great in theory, I can't stand when they are used in practice
  - It is a best practice to consider a slave an exact clone of a master
- Auto-increment Offsets
  - Great failsafe, but lead to bad practice of actively writing to multiple masters

I can, therefore I should...



Disclaimer: I have actually seen something like this in production – actively writing to all 5 masters!

Common Misconceptions

- “If I have more masters, I can write more and faster”
- “I'll just use this old server I have lying around as a slave, but my master is where I need to put all my money”
- “I'm **never** going to need this schema on this slave, so I might as well filter it out”

And my personal favorite...

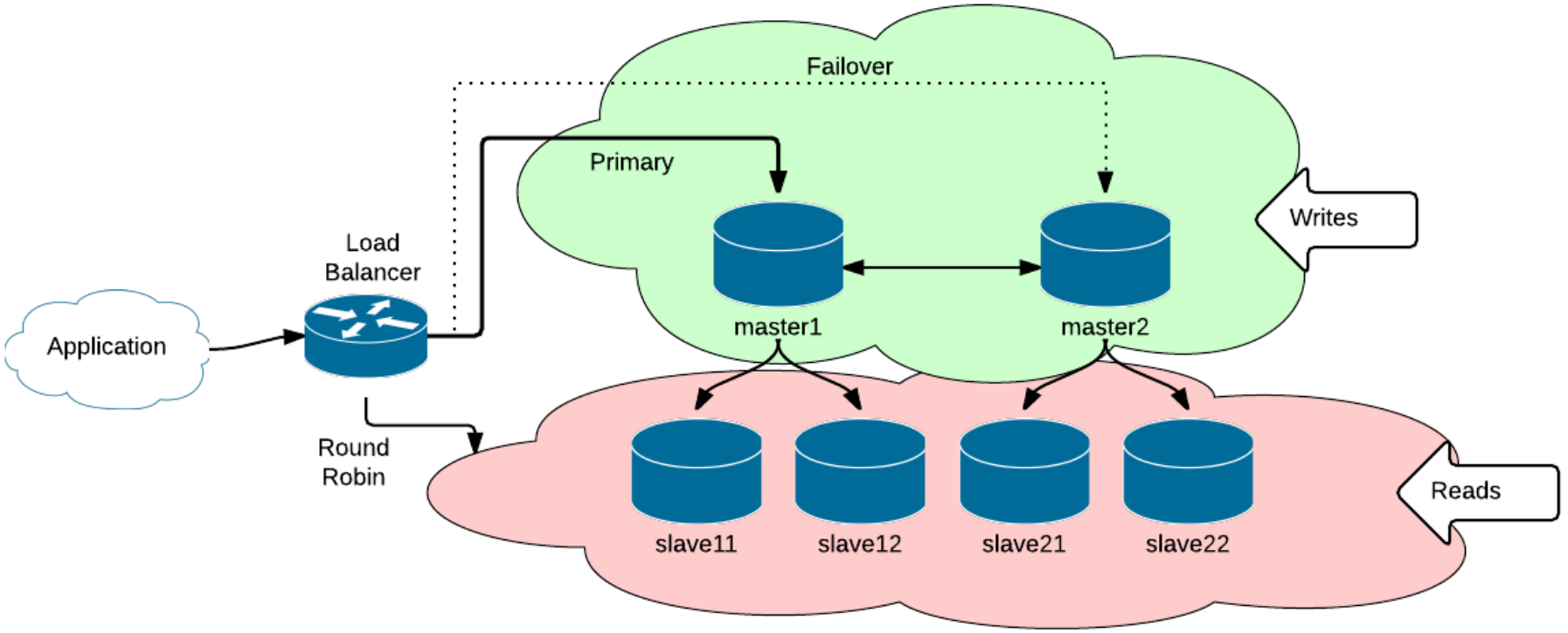
- “I should add more tiers of slaves for increased redundancy”

A "Sane" Approach...

- Master / Master in Active / Passive (i.e. only write to one node)
- Slaves chained off each master
- Load balancer in front of slaves and masters
    - Master pool uses failover only
    - Slave pool uses round robin
    - Incorporate replication health check
- All nodes are identical!

- So what does this look like?

A "Sane" Approach - Diagram



Questions?

(I know you have them, that's why you came today...
so don't be shy)

Thanks for coming!

Website: http://okcmysql.org

Twitter: @okcmysql

Email: mike@okcmysql.org