Linux Conf AU 2008

High Availability, Replicating, Load Balanced MySQL Databases

Oliver Hookins

Systems Administrator

Network Fixinator

Script Hacker

All round super-fun guy

Background

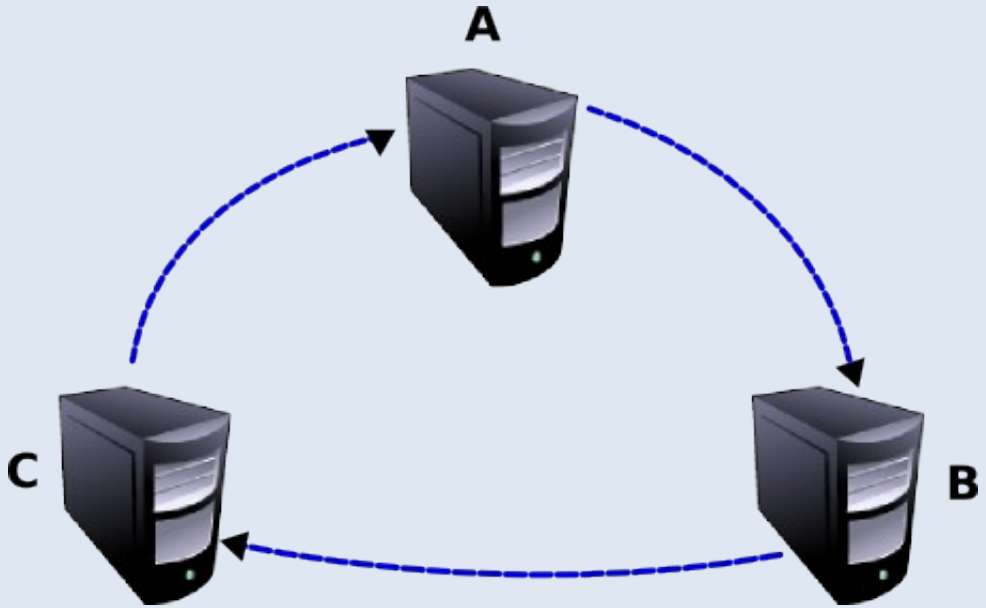
- Who Am I, And How Did I Get Here
- Assumptions -
  - Familiar with single server database systems
  - We aren't dealing with the Application side

Questions

- Why aren't single DB systems good enough?
  - Risk of Failure
  - Performance
  - Time-to-repair from failure
  - Not enough of a geek challenge any more
- What is the ratio of reads to writes?
- Can the app separate reads and writes?

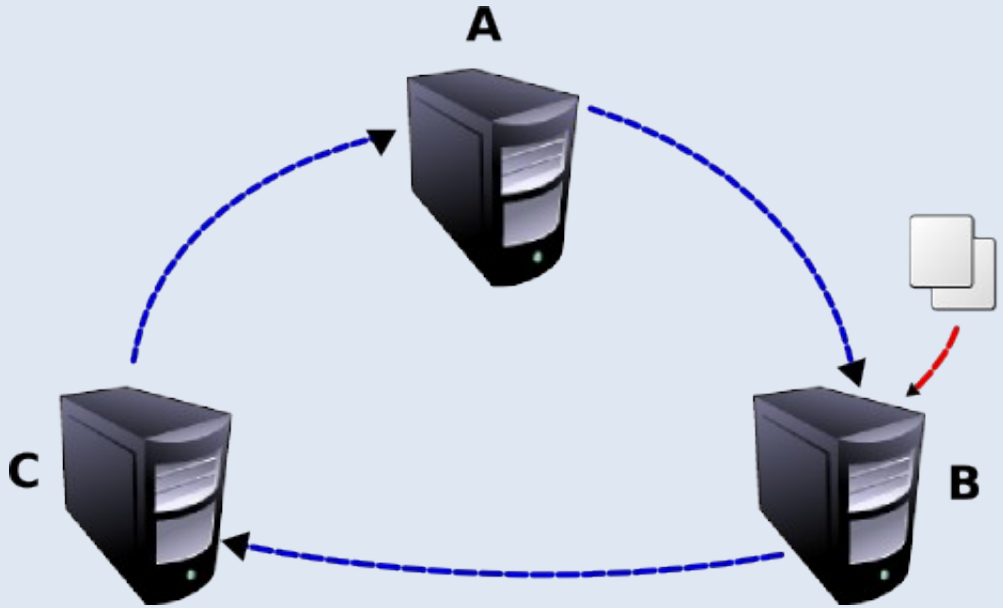
Alternatives

- **Circular/multi-master**
  - Difficult to add more masters – breaks the chain
  - More nodes, more delay. How to verify data?
  - Broken node = broken chain. How to repair?



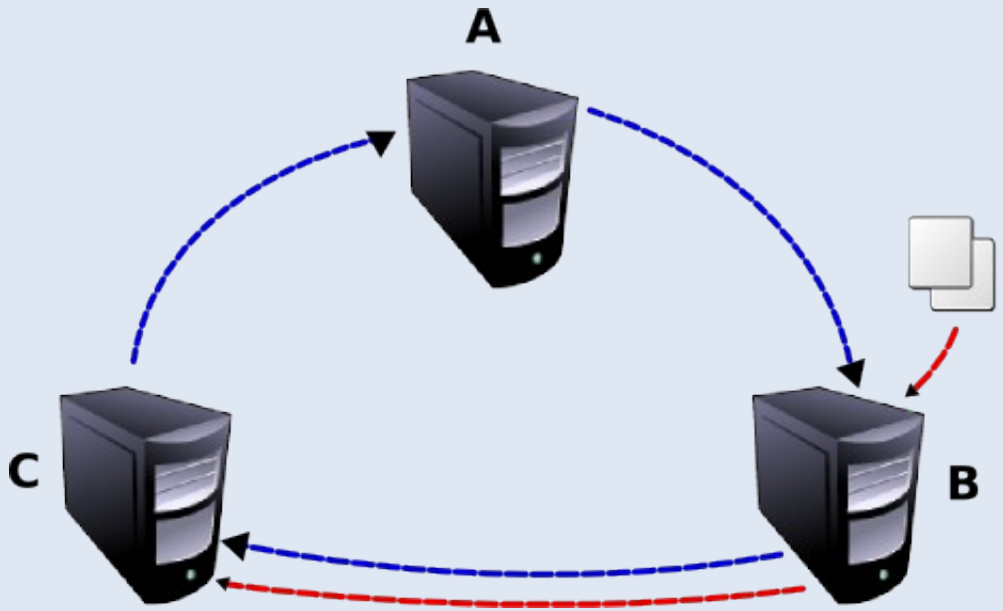
Alternatives

- **Circular/multi-master**
  - Difficult to add more masters – breaks the chain
  - More nodes, more delay. How to verify data?
  - Broken node = broken chain. How to repair?



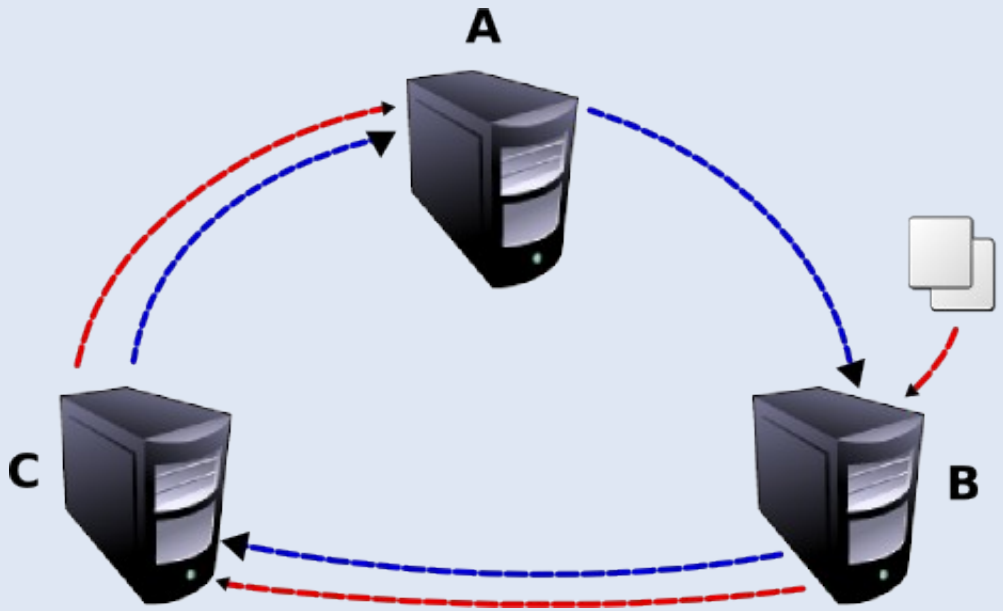
Alternatives

- **Circular/multi-master**
  - Difficult to add more masters – breaks the chain
  - More nodes, more delay. How to verify data?
  - Broken node = broken chain. How to repair?



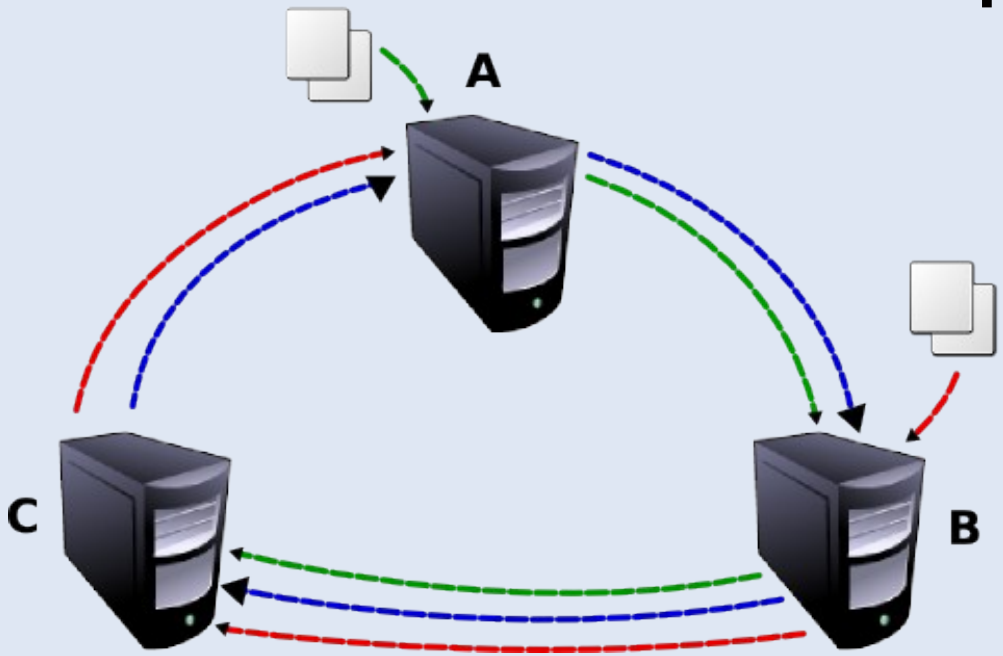
Alternatives

- **Circular/multi-master**
  - Difficult to add more masters – breaks the chain
  - More nodes, more delay. How to verify data?
  - Broken node = broken chain. How to repair?



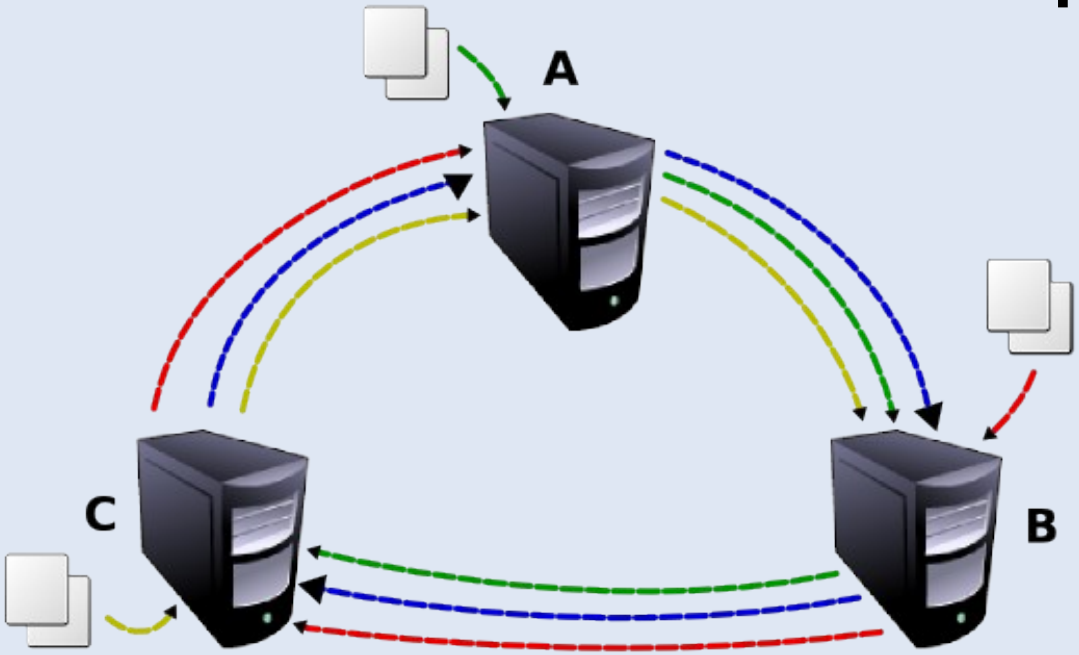
Alternatives

- **Circular/multi-master**
  - Difficult to add more masters – breaks the chain
  - More nodes, more delay. How to verify data?
  - Broken node = broken chain. How to repair?



Alternatives

- **Circular/multi-master**
  - Difficult to add more masters – breaks the chain
  - More nodes, more delay. How to verify data?
  - Broken node = broken chain. How to repair?



Alternatives

- **MySQL Cluster**
  - Prior to 5.1, all data is kept in memory
  - Minimum 3 nodes requirement – Storage Node, Management Node, SQL Node
  - Synchronous
  - Relatively Complex
  - Replication & High Availability built in

Alternatives

- **PostgreSQL and Slony** ... just kidding!

Getting Started

- **Read the documentation!**
  - **MySQL docs now have section on DRBD, Heartbeat, Replication... (but not quite complete)**
  - **Answers many useful questions such as:**

Can I use DRBD on top of LVM?

Getting Started

- **Read the documentation!**
  - **MySQL docs now have section on DRBD, Heartbeat, Replication... (but not quite complete)**
  - **Answers many useful questions such as:**

Can I use DRBD on top of LVM?

Can I use LVM on top of DRBD?

Getting Started

- **Read the documentation!**
  - **MySQL docs now have section on DRBD, Heartbeat, Replication... (but not quite complete)**
  - **Answers many useful questions such as:**

Can I use DRBD on top of LVM?

Can I use LVM on top of DRBD?

Can I use DRBD on top of LVM while at the same time running LVM on top of that DRBD???

Getting Started

- **Decide on a MySQL version**
  - **3.x ? No way**
  - **4.x? Still a lot of replication bugs**
  - **5.0 at least. Latest has a lot of replication bugfixes.**
  - **5.1 adds partitioning, row-based & mixed-format replication & logging, logging to a table, event scheduler, plugins**
  - **6.0 adds Falcon backend, better locking, NDB replication conflict resolution, built-in replication heartbeats, cpu scheduling, backup status reporting**
  - **USE THE SAME VERSION ON ALL HOSTS!**

Getting Started

- DRBD – we use 0.7, but 8.0 is stable
  - 8.0 adds peer authentication with shared secret
  - better split-brain recovery methods
  - more straightforward commands
  - two primaries, and a third server with async replication
  - better integration with Heartbeat, and other improvements
  - 8.2 adds data integrity checking
  - some support for DRBD-proxy
  - some support for cache warming
  - support for online verify of on-disk data

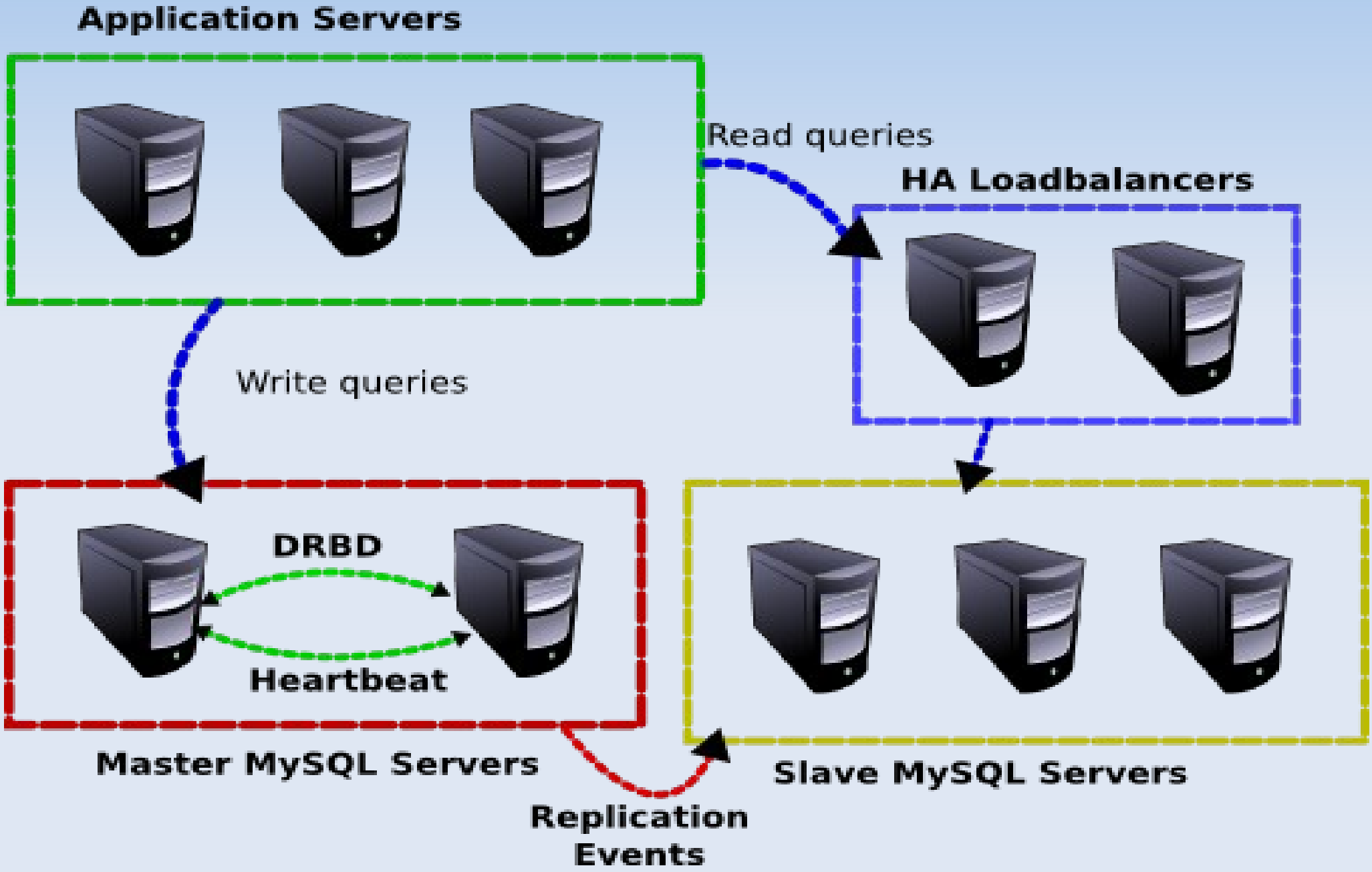
Getting Started

- Heartbeat – use Release 2!
- Use CRM mode
- MySQL docs use non-CRM mode
- Hard to understand at first, but it is better for you in the long run.

Getting Started

- Examples use:
  - 2 masters to avoid SPOF
  - 3 slaves
  - LVS & ldirectord on (HA) loadbalancer
  - DRBD 0.7 (8.0 is the current stable)
  - Heartbeat R2
  - MySQL 5.0
  - A bunch of scripts

Overall Structure



Setting Up The Masters

- Try to use identical hardware
- Allocate a decent bit of diskspace for DRBD (although you can resize later... can be tricky)
- Build DRBD for your kernel & OS
- Set up your config file

DRBD Configuration File

```
1  resource mysqlmaster {
2    protocol C;
3    incon-degr-cmd "echo '!DRBD! pri on incon-degr' | wall ; sleep
   60 ; halt -f";
4    startup {
5      wfc-timeout  60;
6      degr-wfc-timeout 120;    # 2 minutes.
7    }
8    disk {
9      on-io-error   pass_on;
10   }
11   net {
12     sndbuf-size 256K;
13     max-buffers 2048;
14     max-epoch-size 2048;
15   }
16   syncer {
17     rate 50M;
18     group 1;
```

DRBD Configuration File

```
21   on shark {
22     device     /dev/drbd0;
23     disk       /dev/shark/drbd;
24     address    192.168.5.1:7788;
25     meta-disk  internal;
26   }
27   on octopus {
28     device    /dev/drbd0;
29     disk      /dev/octopus/drbd;
30     address   192.168.5.2:7788;
31     meta-disk internal;
32   }
33 }
```

A Side Note About Buffers & Caching

- Use DRBD Protocol C
  - Protocol A – returns after write reaches local net buffer
  - Protocol B – returns after write reaches remove disk buffer
- Turn off disk write-back caching!
- OK to use battery backed write cache
- Can use write-barrier support, but not supported in DM or DRBD currently

DRBD Setup

- Start DRBD on both nodes:

```
/etc/init.d/drbd start
```

- Start syncing:

```
drbdadm -- --do-what-I-say
primary all
```

- Check `/proc/drbd` for DRBD sync completion
- Start using DRBD on the primary!

Heartbeat

- Sends data between nodes to check if they are alive or dead
- Makes decisions for resouces based on status of nodes (and other things)
- Release 2 > Release 1
- Three main config files:
  - `/etc/ha.d/ha.cf`
  - `/etc/ha.d/authkeys`
  - `/var/lib/heartbeat/crm/cib.xml`

/etc/ha.d/ha.cf

```
1  keepalive 2
2  deadtime 30
3  warntime 5
4  initdead 80
5  mcast eth0 225.0.0.4 694 1 0
6  node shark
7  node octopus
8  crm yes
9  autojoin none
10 use_logd on
```

/etc/ha.d/authkeys

```
1  auth 1
2  1 sha1 yeeL3AikaiGoothubee9eiwa2v
```

/var/lib/heartbeat/crm/cib.xml

```
1  <cib>
2   <configuration>
3    <crm_config/>
4    <nodes/>
5
6    <resources>
7     <group id="master" description="MySQL Master Role">
8       <primitive id="master_ip" class="heartbeat"
         type="IPaddr2_vlan" provider="heartbeat">
9        <operations>
10        <op id="master_ip_monitor" name="monitor" interval="60s"
            timeout="15s"/>
11       </operations>
12       <instance_attributes id="master_ip_inst">
13        <attributes>
14         <nvpair id="master_ip_inst_attr" name="1"
             value="192.168.5.10/24/eth0"/>
```

/var/lib/heartbeat/crm/cib.xml

```
17 <primitive id="master_drbd" class="heartbeat" type="drbddisk"
      provider="heartbeat">
18  <operations>
19   <op id="master_drbd_monitor" name="monitor" interval="60s"
      timeout="15s"/>
20  </operations>
21  <instance_attributes id="master_drbd_inst">
22   <attributes>
23    <nvpair id="master_drbd_inst_attr1" name="1"
          value="mysqlmaster"/>
24   </attributes>
25  </instance_attributes>
26 </primitive>
```

/var/lib/heartbeat/crm/cib.xml

```
27 <primitive id="master_filesystem" class="ocf" type="Filesystem"
     provider="heartbeat">
28  <operations>
29   <op id="master_filesystem_monitor" name="monitor" interval="60s"
     timeout="15s"/>
30  </operations>
31  <instance_attributes id="master_filesystem_inst">
32   <attributes>
33    <nvpair id="master_filesystem_inst_attr0" name="device"
        value="/dev/drbd0"/>
34    <nvpair id="master_filesystem_inst_attr1" name="directory"
        value="/hamysql"/>
35    <nvpair id="master_filesystem_inst_attr2" name="fstype"
        value="ext3"/>
36    <nvpair id="master_filesystem_inst_attr3" name="options"
        value="rw,noatime"/>
37   </attributes>
```

/var/lib/heartbeat/crm/cib.xml

```
40 <primitive id="master_mysql" class="lsb" type="mysqld"
      provider="heartbeat">
41  <operations>
42   <op id="master_mysql_monitor" name="monitor" interval="60s"
         timeout="15s"/>
43  </operations>
44 </primitive>
45
46 <primitive id="master_mailto" class="ocf" type="MailTo"
      provider="heartbeat">
47  <instance_attributes id="master_mailto_inst">
48   <attributes>
49    <nvpair id="master_mailto_inst_attr0" name="email"
         value="root@yourcompany.com"/>
50    <nvpair id="master_mailto_inst_attr1" name="subject"
         value="Master_MySQL_Resource"/>
51   </attributes>
```

Notes About the CIB

- Dynamically maintained, so configuration management is difficult
  - Release 1 had a static resource file :(
- Designated Coordinator distributes latest version to nodes
- Annotated DTD is on the Linux HA website and very useful.

Master MySQL Configuration

```
1  [mysqld]
2  safe-user-create
3  safe-show-database
4  set-variable=max_user_connections=1024
5  set-variable=max_connections=2048
6  bind-address=0.0.0.0
7  port=3306
8  datadir=/hamysql/mysql
9  socket=/var/lib/mysql/mysql.sock
10 table_cache=256
11 sync_binlog=1
12 log-bin=mysql-bin
13 server-id=1
14 tmpdir=/hamysql/tmp
15 net_write_timeout=180
16
17 [mysql.server]
18 user=mysql
19 basedir=/hamysql
```

Slave MySQL Configuration

```
1  [mysqld]
2  safe-user-create
3  safe-show-database
4  set-variable=max_user_connections=1024
5  set-variable=max_connections=2048
6  bind-address=0.0.0.0
7  port=3306
8  datadir=/var/lib/mysql
9  socket=/var/lib/mysql/mysql.sock
10 table_cache=256
11 sync_binlog=1
12 server-id=XXXXX
13 report-host=XXXXX
14 master-host=dbmaster
15 master-user=slave
16 master-password=XXXXX
17 read-only
18 master-connect-retry=10
19 master-retry-count=60480
```

Linux Virtual Server

- Used for load-balancing the slaves
- “Layer-4 switch”
- Core LVS code in kernel = fast
- ipvsadm is a fairly low-level userspace program to admin LVS
- ldirectord is a high-level wrapper around ipvsadm and LVS

MySQL Proxy

- Basic functionality includes “load balancing” ...
- Can be extended with LUA scripts to support read/write splitting, partitioning/sharding etc.
- Operates on MySQL queries & not in kernel so not as fast as LVS.
- Still alpha code, but promising.

/etc/ha.d/conf/mysqlslave.cf

```
1  checktimeout = 10
2  negotiatetimeout = 10
3  autoreload = yes
4  quiescent = no
5
6  virtual=192.168.5.30:3306
7          scheduler=wlc
8          # slave 1
9          real=192.168.5.5:3306 masq 65535
10         # slave 2
11         real=192.168.5.6:3306 masq 65535
12         # slave 3
13         real=192.168.5.7:3306 masq 65535
14         # DB master as a last resort
15         real=192.168.5.10:3306 masq 1
16         protocol=tcp
17         service=none
18         checktype=connect
```

Getting Data Onto The Slaves

- If starting from a “clean slate” just set up replication then enter data into the master.
- MySQL replication has no COPY_SET like Slony on PostgreSQL :(
- LOAD DATA|TABLE FROM MASTER... only works on MyISAM and still locks databases
- Could use LVM snapshot on master
- My perl script locks tables then dumps

init_mysql_slave

```
$ init_mysql_slave --source 192.168.1.10 --sourcetype master --user
repl --pass secure --database mysql,clusterdb --repluser slave --
replpass secure --replsource 192.168.1.10
```

- Stops local slave
- Connects to source
- FLUSH TABLES WITH READ LOCK
- Reads binary logfile name and position
- Pipes mysqldump from source into mysql on slave
- Unlocks remote tables
- Starts local replication using binary logfile name and position, and specified master details
- Once you've set up one slave, you can set up new slaves without locking the master

You're Done! Or Are You?

- You aren't providing a service until it is being monitored.
- Monitoring service levels provides better justification for upgrades later.
- We use Nagios & NRPE
  - Remote MySQL TCP connection on slaves, master and loadbalanced slave
  - Local MySQL TCP connection on all servers
  - Replication status on slaves
  - DRBD/Heartbeat on masters

DRBD/Heartbeat Monitoring

- check_proc for Heartbeat processes
- check-drbd for DRBD.
  - Checks state of connection and resynchronisation
- DOPD is available for DRBD automatic monitoring and response
  - You need to know if DRBD is in a critical state anyway
  - You will most likely need to respond
  - Danger of automatically outdating both copies and ending up with no DRBD running at all.

What To Do About Replication Monitoring

- Slave SQL and I/O Threads Running
- Seconds_behind_master
  - Really seconds difference between SQL and I/O threads
  - A transaction can take a very long time to complete on a master, and then the slave has to run it again
- Master can do many writes in parallel
- Slaves have a single I/O thread
- Replication is always asynchronous
- Levels of Replication Paranoia

Replication Paranoia

- Level 1: SQL and I/O thread running and seconds_behind_master “acceptable”
- Level 2: Create replication “metadata” table, insert timestamp on the master. Check the timestamp on slaves.
- Level “2.1”: Not really checking delay behind master. Use CHECKSUM DATABASE|TABLE to see if data is the same on master as it is on slaves.
- Level 3: Specific checks dependent on application data. E.g. number of orders in an order table. Requires in depth knowledge of the nature of the data, and can be as complicated as you have time to spend on it.

write_heartbeat.pl (fragment)

```
1  eval {
2   if ($::rowcount == 0) {
3     debug "INSERT INTO heartbeat (unix_time, db_time) VALUES
          (unix_timestamp(), NULL)\n";
4     $::dbh->do("INSERT INTO heartbeat (unix_time, db_time) VALUES
          (unix_timestamp(), NULL)");
5   }
6   elsif ($::rowcount > 1) {
7     $::rowcount--;
8     debug "DELETE FROM heartbeat ORDER BY unix_time ASC LIMIT
          $::rowcount\n";
9     $::dbh->do("DELETE FROM heartbeat ORDER BY unix_time ASC LIMIT
          $::rowcount");
10  }
11
12  debug "UPDATE heartbeat SET unix_time = UNIX_TIMESTAMP(), db_time =
       NULL\n";
13  $::dbh->do("UPDATE heartbeat SET unix_time = UNIX_TIMESTAMP(), db_time =
       NULL");
14  };
```

Some really useful stuff...

- MySQL toolkit (now called Maatkit) has scripts for:
  - efficient checking of table checksums
  - generating row changesets between servers that have different data
  - slave error detector and resolver
  - heartbeat generator and watcher
  - ...much more

Monitoring And Loadbalancing

- What happens when a slave is out of sync but it is still serving queries?
- We have the monitoring smarts, but need to automate.
- ldirectord has a mysql check ...

lvs-helper

```
1  #!/bin/sh
2  CHAIN="LVS_HELPER"
3  OUTPUT=`/usr/local/sbin/check_replication.pl --slave localhost
   --slave-user replcheck --slave-pass XXXX --master-user replcheck
   --master-pass XXXX`
4  retval=$?
5
6  if [ "$retval" -gt 1 ]
7  then
8    if [ "`/sbin/iptables -L $CHAIN -n | /usr/bin/wc -l`" -le 2 ]
9    then
10     /sbin/iptables -I $CHAIN -j REJECT
11   fi
12   else
13   if [ "`/sbin/iptables -L $CHAIN -n | /usr/bin/wc -l`" -gt 2 ]
14   then
15     /sbin/iptables -F $CHAIN
```

Performance

- You got replication, you don't need any more performance, right?
- Add more slaves = can do more reads
- Bottleneck is the master
  - Use circular replication / MySQL cluster ... argh
  - Tune hardware e.g. hardware RAID, battery backed cache
  - Sharding and/or creative use of MySQL Proxy
    - Later versions of MySQL may support multi-master to single slave.

Backups

- Multiple copies of database, but may all be destroyed by administrative error
- Have to take “consistent” backups
  - Usually means locking

Backups

- Multiple copies of database, but may all be destroyed by administrative error
- Have to take “consistent” backups
  - Usually means locking
- So why not use a slave server?

Backups

- Multiple copies of database, but may all be destroyed by administrative error
- Have to take “consistent” backups
  - Usually means locking
- So why not use a slave server?
- Standard Slave / Dedicated Backups Slave
- OR...

Backups From Standby Master

- Give Masters more disk(s)
- Create a second DRBD resource, active only on standby
- Run a slave on the standby DRBD
- Take backups using the standby's slave

- NOTE: Do not impact standby's write performance, thus degrading the master!

Additional Heartbeat Configuration Part 1

```
1  <group id="slave" description="MySQL Slave Role">
2   <primitive id="slave_ip" class="heartbeat" type="IPaddr2_vlan"
      provider="heartbeat">
3    <operations>
4     <op id="slave_ip_monitor" name="monitor" interval="60s"
        timeout="15s"/>
5    </operations>
6    <instance_attributes id="slave_ip_inst">
7     <attributes>
8      <nvpair id="slave_ip_inst_attr" name="1"
        value="192.168.5.60/24/eth0"/>
9     </attributes>
10   </instance_attributes>
11  </primitive>
12  <primitive id="slave_mysql" class="lsb" type="mysqlslaved"
      provider="heartbeat">
13   <operations>
14    <op id="slave_mysql_monitor" name="monitor" interval="60s"
        timeout="15s"/>
```

Additional Heartbeat Configuration Part 2

```
17 <primitive id="slave_mailto" class="ocf" type="MailTo"
      provider="heartbeat">
18  <instance_attributes id="slave_mailto_inst">
19   <attributes>
20    <nvpair id="slave_mailto_inst_attr0" name="email"
         value="root@yourcompany.com"/>
21    <nvpair id="slave_mailto_inst_attr1" name="subject"
         value="Slave_MySQL_Resource"/>
22   </attributes>
23  </instance_attributes>
24 </primitive>
25 </group>
26 </resources>
27 <constraints>
28  <rsc_colocation id="separate_master_and_slave" from="master"
         to="slave" score="-INFINITY"/>
29 </constraints>
30
```

Even Better Backups

- Backup every 10 mins, use xdelta to generate binary diffs = can store many differentials with little disk space.
- Another handy perl script
  - “Daily” backup mode - full gzipped backup once a day
  - Lock tables and record binlog name & position of where we're up to on the master
  - Full dump and xdelta between it and the daily
  - Record binlog name & position in a metadata file
- Maatkit to generate row diff for rollback?

Restoring

- Slave - just blow it away and re-init
- Master - as long as everything is in sync, can import backup and the slaves will reload the data from the master.
- Or blow it all away and start again :(
-

Finally

- Lots of room for improvement
- New features to make things easier for sysadmins being added to MySQL in later versions.
- Share your “wealth” with the community

Linkage

- http://shipyard.com.au - Resources for this talk
- http://maatkit.sourceforge.net -Replication script Nirvana
- http://mysqldump.azundris.com/archives/71-Replication-now-and-then.html

- http://linux-ha.org - Linux Heartbeat
- http://www.linuxvirtualserver.org/ - LVS
- http://www.vergenet.net/linux/ldirectord/ - ldirectord
- http://drbd.org - DRBD
- http://xdelta.org - xdelta
- http://forge.mysql.com/wiki/MySQL_Proxy - MySQL Proxy
- http://nagios.org - Nagios

Anchor Is Hiring!

- PostgreSQL / Python Developer
- Working on internal systems
- Flexible, fun working environment
- Sydney-based