Scale MySQL on EC2 with Ansible

Presented by: Miklos ‘Mukka’ Szel

June 16, 2015

Pythian®
love your data®

About : Miklos ‘Mukka’ Szel

- Currently MySQL Consultant, Pythian (Adobe, Edmodo, Ebay, Fitbit, Sendgrid)
- 15+ years experience System and Database world
- Acted as Sr. DBA, Sr. Ops, Developer
- Previously worked
  - Own startup
  - ISP
  - Walt Disney & ESPN - cricinfo.com, running ~1500 smaller and bigger web sites
- How to Contact:

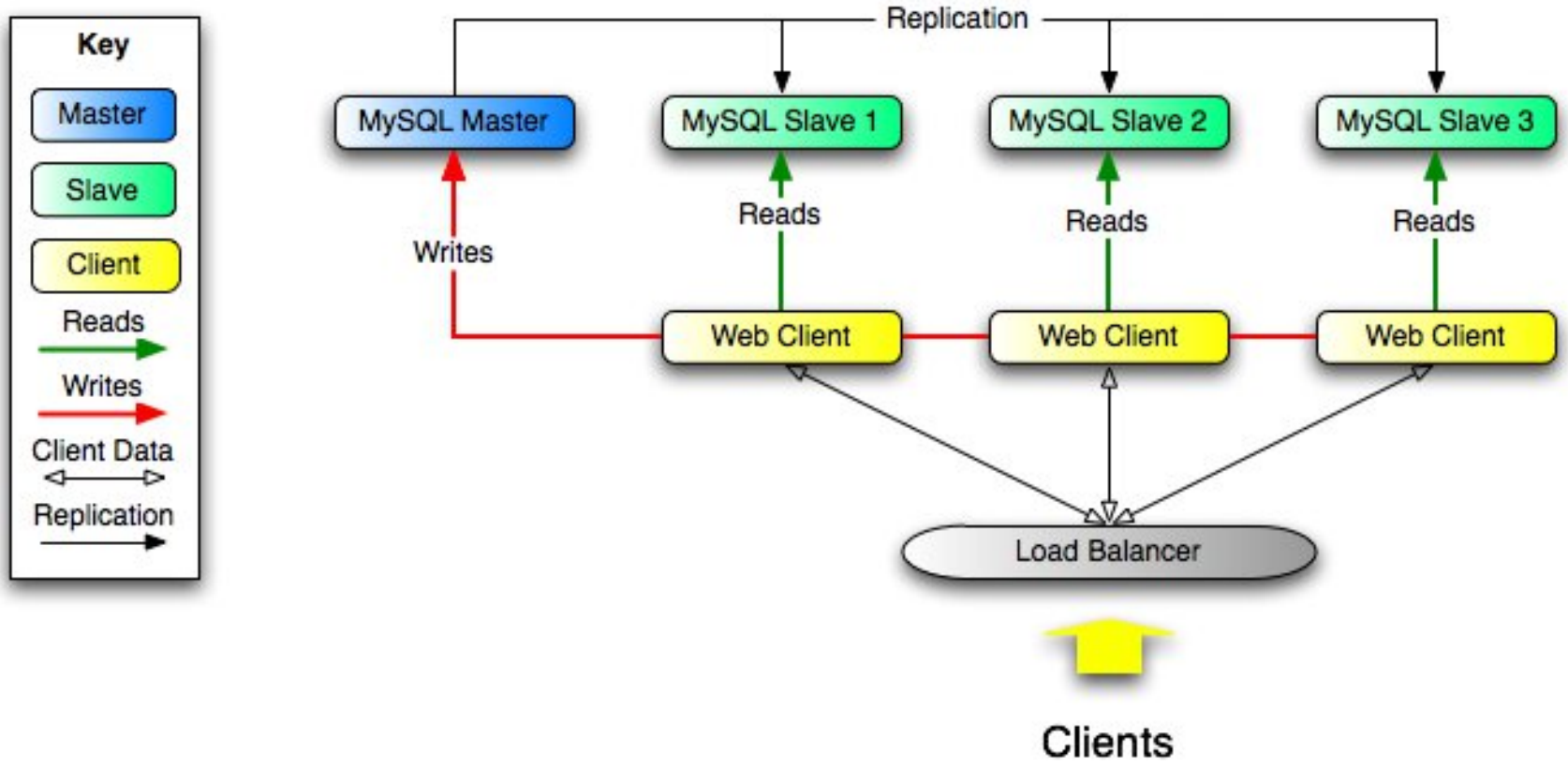
Email: szel@pythian.com

Linkedin: hu.linkedin.com/in/miklosszel/en

AGENDA

- Amazon RDS, EC2/EBS - pros, cons, backup, scale, bottlenecks
- Ansible/MySQL/MHA
- Demo
- ???
- Profit

MySQL Master-Slave overview



MySQL - how to scale for reads

Create a new read replica in a nutshell:

- set up the new instance
- transfer the data
- set up the replication
- wait until it is in sync with the master
- enable traffic

MySQL in the cloud - RDS

pros:

- easy to manage/backup
- ease to scale
- Multi-AZ as HA (99.95% monthly up time SLA)

Cons:

- limited control over the MySQL(no SUPER priv, no shell access, no direct access to your logfiles etc)
- data import/export via logical backup only (large dataset?)
- backup/scale with snapshots (see later)

MySQL in the cloud - EC2

Pros:

- you only pay for what you need (ideally)
- relatively easy to scale (config management)

Cons:

- instances might die
- retirement policy is sometimes unpredictable
- you don’t always know what’s going on in the background
- encryption have costs

MySQL in the cloud - EC2

Backup:

Logical

- mysqldump
- mydumper

Cold

- stop mysql, **archive data files**

Hot or Online

- MySQL Enterprise backup(expensive)
- **Percona XtraBackup**

Snapshots

- LVM(performance overhead)
- **EBS snapshots**

MySQL in the cloud - EC2 Disk types

Ephemeral

- Fast local disks, ready to use
- Comes with many instance types
- Doesn’t support snapshots
- “The data in an instance store persists only during the lifetime of its associated instance.”

EBS

- Magnetic
- Generic SSD
- Provisioned SSD

MySQL in the cloud - EC2/EBS

Why EBS snapshots are great for backups/restore:

- easy to implement
- EC2 uses incremental snapshots
- great tool to provision multiple machines

Cons

- EBSs are network drives, check the throughput limit for your instance
- (n disks * throughput)+other nw activity> bandwidth limit
- volumes are COLD in the beginning, penalty!

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-ec2-config.html

MySQL in the cloud - EBS

Allocating blocks on first access - warming up:

"This preliminary action takes time and can cause a 5 to 50 percent loss of IOPS for your volume the first time each block is accessed."

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-prewarm.html

MySQL in the cloud - EBS - Contd.

Warming up **new** 100GB EBS disks - m3.2xlarge EBS optimized instance

Type	Warmup Time	Warmup Speed
SSD ‘io1’ PIOPS 3000	869s	**124 MB/s**
generic purpose ‘gp2’ SSD 300/3000	864s	**124 MB/s**
SSD ‘io1’ PIOPS 1500	1604s	**66.8 MB/s**
Magnetic ‘standard’ no iops	4884s	**22.0 MB/s**

MySQL in the cloud - Let's scale

Warming up **20GB EBS** volumes created from snapshots

20GB PIOPS 600 4302.73 s, **5.0 MB/s**

500GB => 27 hours

MySQL in the cloud - EBS

It's a download from S3

"If you access a piece of data that hasn't been loaded yet, the volume immediately downloads the requested data from Amazon S3, and then continues loading the rest of the volume's data in the background. "

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSSnapshots.html

MySQL in the cloud - Let's scale

let's use a 'multithreaded download manager' for snapshots - RAID0.

20GB 600 PIOPS single threaded
4302.73 s, 5.0 MB/s

5GB 150 PIOPS in parallel RAID0

5368709120 bytes (5.4 GB) copied, **1001.02** s, 5.4 MB/s
5368709120 bytes (5.4 GB) copied, **985.397** s, 5.4 MB/s
5368709120 bytes (5.4 GB) copied, **941.726** s, 5.7 MB/s
5368709120 bytes (5.4 GB) copied, **935.607** s, 5.7 MB/s

MySQL in the cloud - Let's scale

Chain copy data with XtraBackup streams

last box:
nc -l 1234 | pigz -d | tar xvf -
boxes in the middle:
mkfifo copy_redirect
nc next_host_in_chain 1337 <copy_redirect &
nc -l 1337 | tee copy_redirect | pigz -d | tar -xvf -
'donor' box:
innobackupex --stream=tar /tmp/ --slave-info | pv --size $(du -sh /data/mysql/ | cut -f1) | pigz
| nc first_host_in_chain 1337

http://blog.jericon.net/2012/05/17/chain-copying-to-multiple-hosts/
http://engineering.tumblr.com/post/7658008285/efficiently-copying-files-to-multiple-destinations

MySQL on EC2

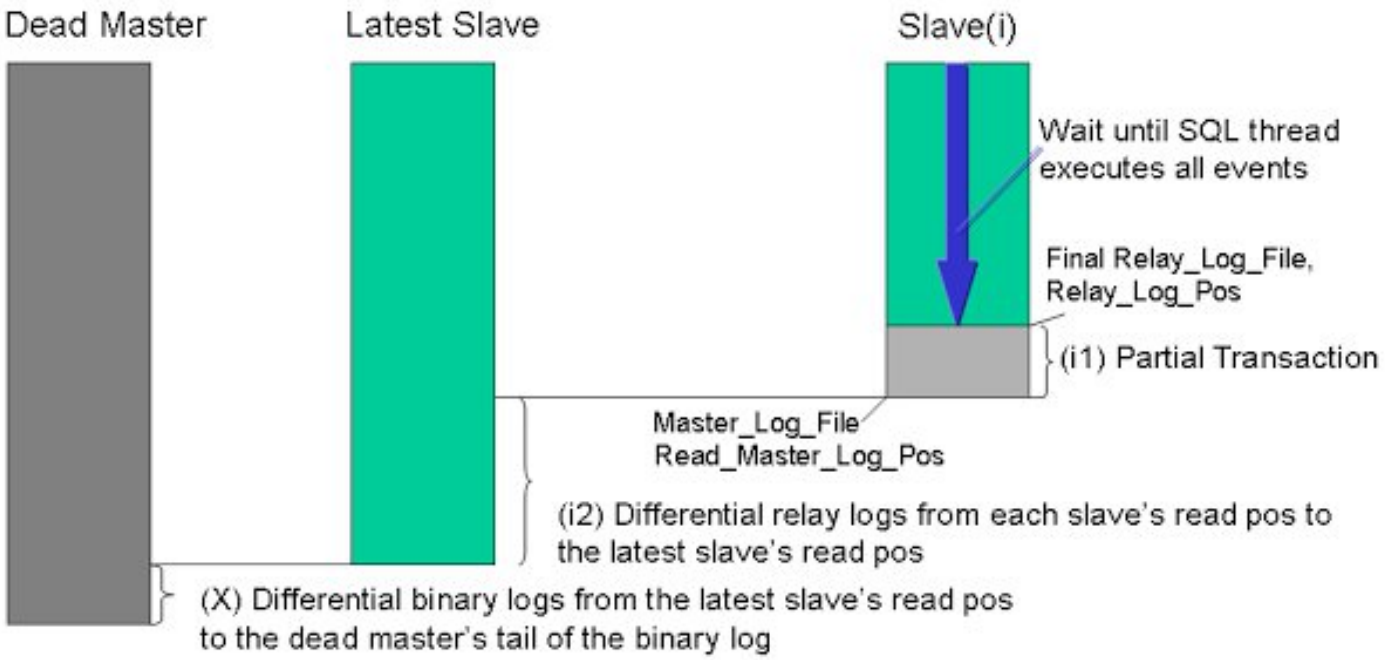
- MHA
- MYSQL
- Ansible

About MHA

- Open Source HA utility written in Perl
- Automated/Interactive Master failover
- Easy to configure

For more: https://code.google.com/p/mysql-master-ha/

About MHA



- On slave(i),
  - Wait until the SQL thread executes events
  - Apply i1 -> i2 -> X
    - On the latest slave, i2 is empty

Scale with CM - Ansible

- Simple
- Agentless
- Easy to manage
- Idempotent
- 100% Python
- Uses ssh to perform changes

Ansible - Modules

MySQL Modules

- mysql_db Add remove databases to remote hosts
- mysql_replication Manage MySQL replication
- mysql_user MySQL user management
- mysql_variables Manage global variables

Others Modules

- Package managers (yum, apt)
- Service
- File
- Template

For more: http://docs.ansible.com/list_of_database_modules.html#mysql

Ansible - Templates

- Use of templates

```
# MHA cluster configuration
# generated by Ansible, do not modify by hand.
# config for cluster: {{cluster_id}}
[server default]
user={{my_mha_user}}
password={{my_mha_pass}}
repl_user={{my_repl_user}}
repl_password={{my_repl_pass}}
ssh_user=root
manager_workdir=/var/log/masterha/{{cluster_id}}
remote_workdir=/var/log/masterha/{{cluster_id}}
...
#my.cnf
innodb_buffer_pool_size                    = {{ (ansible_memtotal_mb * 70 / 100) |round|int }}M
…
```

Demo – https://github.com/mszel-pythian/ansible_mha

- Show configuration files and sample setup
- Demonstrate launching EC2 mysql instances configuring MHA and failing over with Ansible

- SYSTEM SETUP(common role)
  - Fine tune system variables (sysctl - fe.: , swappiness=0)
  - Install system packages and mha
  - Create configuration files based on templates
  - Create users based on public keys under files/users
- MYSQL PART(mysql role)
  - Install mysql binaries
  - Create my.cnf based on template (exception handling)
  - Install mysql users (replica,system users, monitor app users), remove passwordless users
  - Build slave automatically with xtrabackup streaming

Demo

[system]

54.90.37.115 mysql_role=master nickname=dev-meetup-m1

54.237.26.245 mysql_role=slave nickname=dev-meetup-m2

54.157.139.143 mycnf=**custom** mysql_role=slave nickname=dev-meetup-s3

54.237.9.103 mysql_role=mha_manager ephemeral=true

[databases]

54.90.37.115 mysql_role=master server_id=42421

54.237.26.245 mysql_role=slave server_id=42422 backup_src=**54.90.37.115**

54.157.139.143 mysql_role=slave server_id=42423 backup_src=**54.90.37.115**

Demo

TASK: [mysql | copy src=root/mysql_parse_config.sh dest=/root/ owner=root group=root mode=700] ***
skipping: [54.90.37.115]
skipping: [54.237.26.245]
changed: [54.157.139.143]

[...]
roles/mysql/tasks/main.yml:
- include: mysql_slave_xtrabackup.yml
  when: "bootstrap_enabled and mysql_role != 'master'"
  tags: slave
TASK: [mysql | Streaming the backup from {{ backup_src }} to {{ inventory_hostname }}] ***
skipping: [54.90.37.115]
changed: **[54.237.26.245]**
changed: [54.157.139.143]