Deploying MySQL in Production

Daniel Kowalewski — Senior Technical Operations Engineer, Percona
daniel.kowalewski@percona.com

PERCONA

Deploying MySQL in Production

- Installation
- Configuration (OS and MySQL)
- Backups
- Monitoring
- Before you go live...

Installation

MySQL Installation

- This is not your job!
- Don't reinvent the wheel
- Use a package manager
  - Oracle, Percona, MariaDB all have repos
- Stay current

Config management

- Ansible, Puppet, Chef, SaltStack, etc.
  - Use what the rest of your company uses if possible
- Infrastructure as code (revision control!)
- Consistency

Use config management for:

- Package installation
- Volume and directory creation
- Permissions
- Firewalls
- Initial MySQL accounts
  - delete default accounts

PERCONA

OS Configuration

Storage

- Can you afford SSDs?
- RAID10 👍, RAID5/6👎, RAID0👎👎 (*)
- LVM for flexibility
- Filesystem – XFS or ext4
  - XFS: inode64,nobarrier,noatime,logbufs=8
  - ext4: noatime,nodiratime,barrier=0
- IO Scheduler – noop or deadline

PERCONA

CPU

- Set CPU governor to "performance"

Life Goal: Don't run out of memory

- Set vm.swappiness = 1
- Don't disable swap, it's your reserve parachute
- Use jemalloc
- Disable Transparent Huge Pages

MySQL Configuration

MySQL Configuration

- Don't use the default files
- Percona Config Wizard isn't a bad start

Basic settings

- innodb_buffer_pool_size (still defaults to 128 MiB!)
- max_connections (still defaults to 151!)
- innodb_log_file_size
- query_cache_size and query_cache_type
- innodb_thread_concurrency
- innodb_buffer_pool_instances

Also consider

- innodb_flush_log_at_transaction_commit
- sync_binlog

Set now, or regret later

- character_set (utf8 or utf8mb4)
- innodb_file_per_table
- log-bin
- expire_logs_days
- innodb_numa_interleave

Backups

Things that are not backups

- RAID
- SAN
    - mirrored SAN
- MySQL replicas
    - even delayed replicas
- Untested backups
- Backups stored on the DB host

Type	Size	Time to Back up	Time to Restore	Partial Restores	Tools
Physical	larger	faster	faster	hard	xtrabackup, MySQL Enterprise Backup
Logical	smaller	slower	slower	easy	mysqldump, mydumper, mysqlpump

PERCONA

Incremental/differential backups

- Store changes since the last backup
- Only possible with physical backups
- More complicated restores
- Don't go too long without a full backup!

Binary log backups

- Essential for point-in-time recovery
- Stream with mysqlbinlog --read-from-remote-server
- Play back by piping to mysql
- You did turn on log_bin, right?

Test your backups

- Automate restores
- Watch for errors (especially with logical backups)
- Replicate from production
    - pt-table-checksum
    - Compare schemas

Monitoring

Alerting

- Nagios, Icinga, Sensu, VividCortex, New Relic, etc
  - Use what you already have
- Alert on:
  - Things that matter to the application
  - Things that are hard to recover from
- Don't over-alert

Metrics that affect the application:

- Service status (connection check)
  - as realistic as possible
  - Update / select from table
- Average execution time
- threads_running
- Maybe replication delay

Disasters waiting to happen:

- Disk space
- History list size
- Auto-increment keys
- threads_connected (percentage of max_connections)
- Checksum differences
- Swap usage

Maybe not so important

- Long transactions (check history list instead)
- threads_connected
- CPU usage
- Deadlocks

Trending

- Don't over-alert, but trend everything you can!
- PMM, VividCortex, Sensu, etc.

Query analyzer

- Removes guesswork
- "This query is responsible for OOM"
- "This is our worst query"
- "3% of queries would benefit from this index"
- PMM, VividCortex, NewRelic

Before you go live…

LEEEEEEROOOOOY JENKINS!

Benchmark everything

- sysbench for the hardware and OS
- sysbench OLTP or tpcc-mysql
- Load test from application
- Save the results to reference later

Data retention

- Don't count on vertical scaling forever
- Every problem becomes harder with more data

References

https://git.io/v9UMe

Questions?

Database Performance Matters