MySQL Performance Tuning

Seminario de Desarrollo de Software - Casa Central.

Maximiliano Osorio
mosorio@inf.utfsm.cl

Universidad Técnica Federico Santa María

11 de octubre de 2017

Mistakes

Change one setting at a time!

This is the only way to estimate if a change is beneficial.

Mistakes

SET GLOBAL.

Most settings can be changed at runtime with SET GLOBAL. It is very handy and it allows you to quickly revert the change if it creates any problem

Caution

But in the end, you want the setting to be adjusted permanently in the configuration file.

Mistakes

A change in the configuration is not visible even after a MySQL restart

Did you use the correct configuration file? Did you put the setting in the right section? (all settings in this post belong to the [mysqld] section)

Mistakes

Do not allow duplicate settings in the configuration file.

If you want to keep track of the changes, use version control.

innodb_buffer_pool_size:

- this is the #1 setting to look at for any installation using InnoDB.

Definition

The buffer pool is where data and indexes are cached: having it as large as possible will ensure you use memory and not disks for most read operations.

Typical values

Typical values are:

- 5-6GB (8GB RAM)
- 20-25GB (32GB RAM)
- 100-120GB (128GB RAM)

innodb_log_file_size

Definition

This is the size of the redo logs. The redo logs are used to make sure writes are fast and durable and also during crash recovery.

Typical values

- Starting with innodb_log_file_size = 512M (giving 1GB of redo logs) should give you plenty of room for writes.
- If you know your application is write-intensive and you are using MySQL 5.6, you can start with innodb_log_file_size = 4G.

max_connections

- if you are often facing the Too many connections error, max_connections is too low.
- It is very frequent that because the application does not close connections to the database correctly, you need much more than the default 151 connections.
- The main drawback of high values for max_connections (like 1000 or more) is that the server will become unresponsive if for any reason it has to run 1000 or more active transactions.

InnoDB

InnoDB has been the default storage engine since MySQL 5.5 and it is much more frequently used than any other storage engine. That's why it should be configured carefully.

innodb_file_per_table

Definition

This setting will tell InnoDB if it should store data and indexes in the shared tablespace (innodb_file_per_table = OFF) or in a separate .ibd file for each table (innodb_file_per_table= ON)

innodb_file_per_table

- Having a file per table allows you to reclaim space when dropping, truncating or rebuilding a table. It is also needed for some advanced features such as compression. However it does not provide any performance benefit
- The main scenario when you do NOT want file per table is when you have a very high number of tables (say 10k+).

innodb_flush_log_at_trx_commit

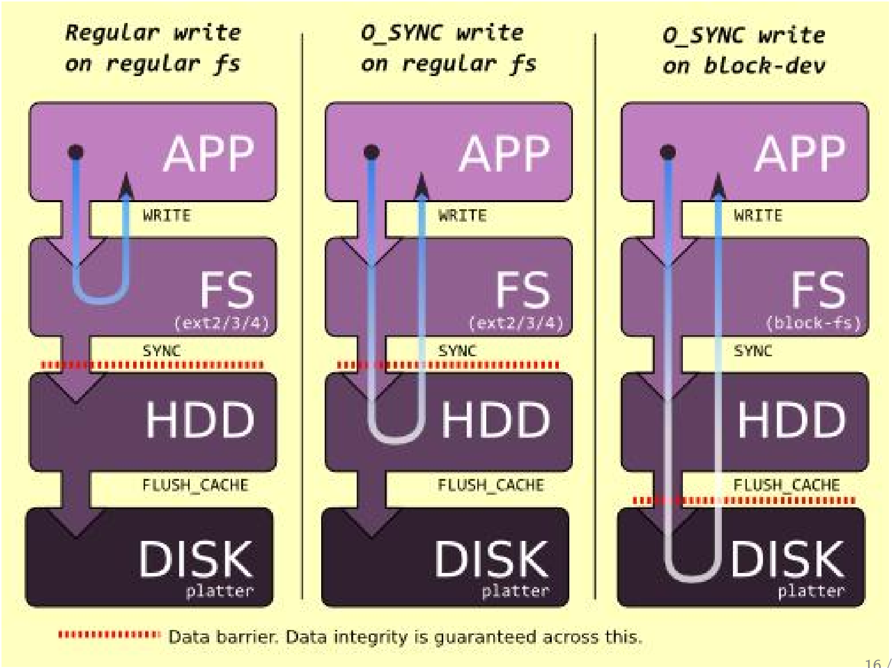
- the default setting of 1 means that InnoDB is fully ACID compliant. It is the best value when your primary concern is data safety
- However it can have a significant overhead on systems with slow disks because of the extra fsyncs
- Setting it to 2 is a bit less reliable because committed transactions will be flushed to the redo logs only once a second
- 0 is even faster but you are more likely to lose some data in case of a crash: it is only a good value for a replica.

innodb_flush_method

This setting controls how data and logs are flushed to disk, popular values are

- O_DIRECT when you have a hardware RAID controller with a battery-protected write-back cache
- fdatasync (default value) for most other scenarios.

sysbench is a good tool to help you choose between the 2 values


Regular write on regular fs
O_SYNC write on regular fs
O_SYNC write on block-dev
APP
WRITE
FS
(ext2/3/4)
(block-fs)
SYNC
HDD
FLUSH_CACHE
DISK
platter
Data barrier. Data integrity is guaranteed across this.

innodb_log_buffer_size

this is the size of the buffer for transactions that have not been committed yet.

- The default value (1MB) is usually fine but as soon as you have transactions with large blob/text fields, the buffer can fill up very quickly and trigger extra I/O load

Look at the Innodb_log_waits status variable and if it is not 0, increase innodb_log_buffer_size.

query_cache_size

the query cache is a well known bottleneck that can be seen even when concurrency is moderate

- The best option is to disable it from day 1 by setting query_cache_size = 0 (now the default on MySQL 5.6) and to use other ways to speed up read queries: good indexing, adding replicas to spread the read load or using an external cache

log_bin

- enabling binary logging is mandatory if you want the server to act as a replication master.
- If so, don't forget to also set server_id to a unique value
- Once created, binary log files are kept forever. So if you do not want to run out of disk space, you should either purge old files with or set expire_logs_days to specify after how many days the logs will be automatically purged.
- Binary logging however is not free, so if you do not need for instance on a replica that is not a master, it is recommended to keep it disabled.

skip_name_resolve

- when a client connects, the server will perform hostname resolution, and when DNS is slow
- It is therefore recommended to start the server with skip-name-resolve to disable all DNS lookups. The only limitation is that the GRANT statements must then use IP addresses only.

References

"Mysql performance tuning, volume 1."
https://www.percona.com/resources/mysql-ebooks/mysql-performance-tuning-volume-1.
(Accessed on 10/10/2017).