# Conseils pour optimiser les performances de MySQL/MariaDB


MySQL est un système puissant et open source de gestion de base de données relationnelles (SGBDR).
Il a été libéré en mai 1995.
En Avril 2009, le projet MySQL a été acheté par Oracle.
En conséquence un fork de la communauté MySQL appelé MariaDB a été créé.
La principale raison de la création du fork était de garder le projet libre sous licence publique générale (GPL).
MySQL est distribué sous une double licence GPL et propriétaire.
Il utilise le language Structured Query Language (SQL) qui est probablement le choix le plus populaire pour la gestion de contenu au sein d'une base de données.

Aujourd'hui, MySQL et MariaDB sont l'un des SGBDR les plus (sinon le plus) fréquemment utilisés pour les applications Web telles que WordPress, Joomla, Magento et d'autres.

Cet article va vous montrer quelques conseils utiles pour optimiser les performances de MySQL/MariaDB.

Avant de commencer, ne pas accepter ces suggestions aveuglément.
Chaque configuration de MySQL est unique et nécessite une réflexion.
Ce que vous devez savoir :

- Le fichier de configuration de MySQL/MariaDB est `my.conf`.
  Chaque modification nécessitera un redémarrage du service MySQL pour que les changements de configuration soit pris en compte.
- Dans cet article, j'ai utilisé Mysql 5.6.

## Activation de InnoDB : un fichier par table

D'abord, il est important d'expliquer que InnoDB est un moteur de stockage.
MySQL et MariaDB utilisent InnoDB comme moteur de stockage par défaut.

Son principal avantage par rapport aux autres moteurs de stockage de MySQL est qu'il permet des transactions ACID (atomiques, cohérentes, isolées et durables), ainsi que la gestion des clés étrangères (avec vérification de la cohérence).
Toutes les bases de données sont stockées au même endroit.
Par défaut dans le fichier ibdata1 qui, sous les systèmes de type unix, se trouve généralement dans /var/lib/mysql/.
Il est également possible d'utiliser plusieurs fichiers ou même d'utiliser directement une ou plusieurs partitions sur le disque en mode RAW.

En séparant chaque table d'une base de données, un fichier de données .ibd sera créé.

Les opérations telles qu'un “TRUNCATE” de base de données, peuvent être complétés plus rapidement et vous pouvez également récupérer l'espace inutilisé lors de la suppression ou lorsque vous tronquez une table de base de données.

Un autre avantage de cette configuration est le fait que vous pouvez garder quelques-unes des tables de base de données dans un dispositif de stockage séparé.
Cela peut grandement améliorer la charge des entrées/sorties sur vos disques.

Le paramètre `innodb_file_per_table` est activé par défaut dans MySQL 5.6 et plus.
Vous pouvez voir que dans `my.cnf` fichier.
La directive ressemble à ceci:

```
innodb_file_per_table=1
```

## Taille du cache InnoDB

Plus le cache est grand, moins MySQL devra aller chercher de données sur le disque lors de requêtes et donc plus la base sera rapide.

Sous InnoDB, la taille du cache est gérée par le paramètre `innodb_buffer_pool_size`.
Par défaut, il est initialisé à 128 Mo.
Sur un serveur de base de données dédié, on peut monter jusqu'à 80% de la RAM du serveur.

Cependant, MySQL aura besoin de RAM également pour stocker le programme, la structure de la base, etc.
De plus, si vous utilisez MyISAM en parallèle d'InnoDB, il faudra également laisser de la place en RAM pour le cache MyISAM

La directive ressemble à ceci:

```
innodb_buffer_pool_size = 128M
```

## Évitez swappiness dans MySQL

Le swapping est un processus qui se produit lorsque le système déplace une partie de la mémoire à un espace disque spécial appelé «swap».
L'événement apparaît généralement lorsque votre système manque de mémoire physique RAM et au lieu de libérer un peu de RAM, le système a poussé les informations dans le disque.
Comme vous pourriez avoir deviner, le disque est beaucoup plus lent que votre RAM.

Par défaut, l'option est activée:

```
vm.swappiness = 60
```

Pour désactiver swappiness, exécutez la commande suivante:

```
sysctl -w vm.swappiness=0
```

## Nombre de connexions simultanées

Le nombre de connexions simultanées à la base est limité par le paramètre `max_connections`.

Si le nombre de connexions maximal est atteint, la base de données retournera une erreur (Too many connections).

Il est donc très important que ce paramètre ne soit pas trop bas.
Chaque thread PHP pouvant ouvrir une connexion à la base de données, le nombre de connexions maximum demandées à la base de données devrait être environ égal au nombre de threads PHP autorisés dans Apache par exemple.

La commande « SHOW STATUS » dans MySQL permet de voir le nombre maximal de connections simultanées ouvertes sur le serveur, ce qui permet de voir en production si on se rapproche ou non du maximum autorisé.

Ne modifiez pas le paramètre `max_connections` sans modifier le paramètre `table_cache` qui lui est associé.
`table_cache` représente le nombre de fichiers ouverts simultanément par MySQL.
En MyISAM, pour chaque connexion, lorsque MySQL accède à une table, MySQL ouvre un fichier.
Si MySQL fait une requête avec un join sur 2 tables, il ouvre 2 fichiers, etc...

Dans l'idéal, on devrait donc avoir **table_cache = max_connections * nombre de join maximum**.

Dans le cas d'InnoDB, ce nombre peut être différent puisque selon la configuration d'InnoDB, celui-ci écrit toutes les tables dans le même fichier, ou écrit une table par fichier (suivant la valeur du paramètre innodb_file_per_table).

## Le nombre de thread à mettre en cache

La directive `thread_cache_size` définit le nombre de threads que votre serveur doit mettre en cache.
Comme le client se déconnecte, ses treads sont mis en cache si elles sont inférieures à la `thread_cache_size`.
D'autres demandes sont traitées en utilisant les threads stockés dans le cache.

Pour améliorer votre performance, vous pouvez définir le `thread_cache_size` à un nombre relativement élevé.
Pour trouver le taux de succès du thread cache, vous pouvez utiliser la technique suivante:

```
mysql> show status like 'Threads_created';
```

```
mysql> show status like 'Connections';
```

Maintenant, utilisez la formule suivante pour calculer le pourcentage :

```
100 - ((Threads_created / Connections) * 100)
```

Si vous obtenez un nombre faible, cela signifie que la plupart des nouvelles connexions mysql commencent par un nouveau thread au lieu du chargement de cache.
Vous voudrez sûrement augmenter la thread_cache_size dans de tels cas.

La bonne chose ici est que le thread_cache_size peut être modifié dynamiquement sans avoir à redémarrer le service MySQL.
Vous pouvez réaliser cela en exécutant:

```
mysql> set global thread_cache_size = 16;
```

## Désactiver la recherche inversée DNS

Par défaut MySQL/MariaDB effectuer une recherche DNS de l'adresse IP/nom d'hôte de l'utilisateur à partir de laquelle la connexion est à venir.
Pour chaque connexion client, l'adresse IP est vérifiée par la résolution à un nom d'hôte.
Après que le nom d'hôte est résolu vers une adresse IP pour vérifier que les deux correspondent.

Cela peut malheureusement entraîner des retards dans le cas du DNS ou des problèmes avec le serveur DNS mal configuré.
Ceci est la raison pour laquelle vous pouvez désactiver la recherche DNS inversée en ajoutant ce qui suit dans votre fichier de configuration:

```
[mysqld]
skip-name-resolve
```

Vous devrez redémarrer le service MySQL après l'application de ces changements.

## L'utilisation du cache de requête

Si vous avez de nombreuses requêtes répétitives et vos données ne changent pas souvent, l'utilisation du cache de requêtes est pour vous.
Souvent, les gens ne comprennent pas le concept derrière le `query_cache_size` et mettre cette valeur au gigaoctets, peut effectivement entraîner une dégradation des performances.

La raison de cela réside dans le fait que les threads ont besoin pour verrouiller le cache lors des mises à jour.
Habituellement la valeur de 200-300 Mo devrait être plus que suffisant.
Si votre site est relativement faible, vous pouvez essayer de donner la valeur de 64M et d'augmenter dans le temps.

Vous devrez ajouter les paramètres suivants dans le fichier de configuration MySQL:

```
query_cache_type = 1
query_cache_limit = 256K
query_cache_min_res_unit = 2k
query_cache_size = 80M
```

## Prévenir les écritures sur le disque

Les deux directives devraient avoir la même taille et vous aideront à prévenir les écritures sur disque.
Le `tmp_table_size` est le montant maximum de la taille des tables internes en mémoire.
Dans le cas où la limite en question est dépassée, la table sera convertie en table MyISAM sur le disque.

Cela aura une incidence sur la performance de base de données.
Les administrateurs recommandent généralement de donner 64M pour les deux valeurs pour chaque Go de RAM sur le serveur.

```
[mysqld]
tmp_table_size= 64M
max_heap_table_size= 64M
```

## Activer les Journaux de requêtes lentes de MySQL

Loguer les requêtes lentes peut vous aider à déterminer les problèmes avec votre base de données et vous aider à les déboguer.
Ceci peut être facilement activée en ajoutant les valeurs suivantes dans votre fichier de configuration MySQL:

```
slow-query-log = 1
slow-query-log-file = /var/lib/mysql/mysql-slow.log
long_query_time = 1
```

La première directive permet l'enregistrement des requêtes lentes, tandis que le second indique à MySQL où stocker le fichier journal réel.
Utilisez `long_query_time` pour définir la quantité de temps qui est considéré comme long pour une requête.

## Vérifiez les Connexions inactives de MySQL

Les connexions inactives consomment des ressources et doivent être interrompus ou rafraîchies lorsque cela est possible.
Ces connexions sont en état “sleep” et restent habituellement de cette façon pour une longue période de temps.
Pour rechercher des connexions au ralenti, vous pouvez exécuter la commande suivante:

```
mysqladmin processlist -u root -p | grep "Sleep"
```

Cela vous montrera la liste des processus qui sont en état de sommeil `sleep state`.

Lorsque vous utilisez PHP cet événement peut apparaître lors de l'utilisation `mysql_pconnect` qui ouvre la connexion, après qui exécute des requêtes, supprime l'authentification et laisse la connexion ouverte.
Cela entraînera des tampons par thread à conserver en mémoire jusqu'à ce que le thread meurt. La première chose que vous ferez ici est de vérifier le code et de le corriger.
Si vous ne disposez pas d'accès au code, vous pouvez modifier la directive `wait_timeout`.
La valeur par défaut est 28800 secondes, alors que vous pouvez réduire en toute sécurité à quelque chose comme 60:

```
wait_timeout=60
```

## Choisir un bon système de fichiers pour MySQL

Choisir le bon système de fichiers est vital pour vos bases de données. La plupart des choses importantes que vous devez considérer ici sont - l'intégrité des données, la performance et la facilité d'administration.

Selon les recommandations de MariaDB, les meilleurs systèmes de fichiers sont XFS, Ext4 et Btrfs.
Ils sont tous des systèmes de fichiers journalisés qui peuvent être utilisés avec de très gros fichiers et de grands volumes de stockage.

Ci-dessous vous trouverez des informations utiles sur les trois systèmes de fichiers:

| **Filesystems** | **XFS** | **Ext4** | **Btrfs** |
|---|---|---|---|
| Taille maximum du Filesystem | 8Eo | 1Eo | 16Eo |
| Taille maximum d'un fichier | 8Eo | 16To | 16Eo |

## Définir la taille maximale des paquets

MySQL divise les données en paquets.
Habituellement, un seul paquet est considéré comme une ligne qui est envoyé à un client.
La directive `max_allowed_packet` définit la taille maximale des paquets qui peuvent être envoyés.

Si, la définition de cette valeur est trop basse, elle peut provoquer une requête de décrochage et vous recevrez une erreur dans votre journal des erreurs MySQL.
Il est recommandé de définir la valeur de la taille de votre plus grand paquet.

## Optimiser et réparer les bases de données MySQL

Parfois, les tables de base de données MySQL/MariaDB s’écrase assez facilement, surtout lorsque le serveur se ferme de façon inattendu, une soudaine corruption du système de fichiers ou lors de l'opération de copie, lorsque la base de données est toujours accessible.
Étonnamment, il y a un outil gratuit et open source appelé «mysqlcheck», qui vérifie automatiquement, la réparation et l'optimisation des bases de données de toutes les tables de Linux.

```
mysqlcheck -u root -p --auto-repair --check --optimize --analyze --all-
databases
```

```
mysqlcheck -u root -p --auto-repair --check --optimize --analyze
<databasename>
```

## Stockage des base de données MySQL dans une partition séparé

Cela, ne fonctionne qu'avec MySQL, pas avec MariaDB

Parfois, les lecture/écriture de l'OS peuvent ralentir les performances de votre serveur MySQL, surtout si cela est situé sur un même disque dur.
Au lieu de cela, je vous conseille d'utiliser des disques dur séparé (de préférence SSD) pour le service MySQL.

Pour terminer, il vous faudra associer le nouveau disque à votre ordinateur/serveur.

L'étape suivante consiste à préparer le nouveau disque:

```
 fdisk /dev/sdb
```

Maintenant, appuyez sur **“n”** pour créer une nouvelle partition.
Ensuite, appuyez sur **“p”** pour rendre la nouvelle partition primaire.
Après cela, définissez le numéro de la partition 1-4.
Après cela, vous sélectionnez la taille de la partition.
Appuyez sur Entrée.
A cette étape, vous devrez configurer la taille de la partition.

Si vous souhaitez utiliser l'ensemble du disque appuyez une fois de plus sur Entrée.
Sinon, vous pouvez régler manuellement la taille de la nouvelle partition.
Lorsque vous êtes prêt appuyez sur **“w”** pour écrire les modifications.
Maintenant, nous allons avoir besoin de créer un système de fichiers pour notre nouvelle partition.
Ceci peut être facilement fait avec:

```
mkfs.ext4 /dev/sdb1
```

Maintenant, nous allons monter notre nouvelle partition dans un dossier.
J'ai appelé mon dossier **“ssd”** et créé dans le répertoire racine:

```
mkdir /ssd/
```

Nous sommes prêts à monter la nouvelle partition que nous venons de faire :

```
mount /dev/sdb1  /ssd/
```

Vous pouvez effectuer le montage au démarrage en ajoutant la ligne suivante dans le fichier `/etc/fstab`.

```
/dev/sdb1 /ssd ext3 defaults 0 0
```

Maintenant, vous êtes prêt à passer MySQL sur le nouveau disque.

Tout d'abord arrêter le service MySQL avec:

```
service mysqld stop
```

Je vous recommande d'arrêter les services qui pourrai faire des tentatives d'écritures/lectures dans les bases de données (Apache, nginx, etc).

Maintenant copier le répertoire MySQL entier dans le nouveau lecteur:

```
cp /var/lib/mysql /ssd/ -Rp
```

Cela peut prendre un certain temps selon vos bases de données MySQL.
Une fois ce processus terminé renommer le répertoire MySQL:

```
mv /var/lib/mysql /var/lib/mysql-backup
```

Ensuite, nous allons créer un lien symbolique :

```
ln -s /ssd/mysql /var/lib/mysql
```

Maintenant, vous pouvez relancer MySQL et vos services web. À ce stade, vos bases de données MySQL seront accessibles à partir de la nouvelle unité de stockage.

Voila, si vous avez d'autres moyen d'optimiser MySQL/MariaDB les commentaires sont là. source : developpez.com et tecmin.com