Surveillance du serveur MySQL

Innodb status

Executer

```
SHOW ENGINE INNODB STATUS\G
```

Surveillez notamment le ***Buffer pool hit rate*** qui, s'il est élevé (par exemple 999/1000), indique que mysql va majoritairement chercher ses données en mémoire plutôt que sur le disque.

Pour plus de détails, voir

- https://www.rathishkumar.in/2017/01/how-to-allocate-innodb-buffer-pool-size-in-mysql.html
- http://mysqlopt.blogspot.com/2012/01/mysql-server-tuning.html

Regarder les requêtes en cours d'exécution

Pour demander à MySQL une liste des requêtes actuellement en cours d'exécution, on peut utiliser la requête :

```
SHOW FULL processlist;
```

Inconvénient : On récupère un tableau de la situation à l'instant de la demande. On ne sait rien de ce qui s'est passé avant, ni de ce qui se passe après.

EN CAS DE CRISE, pour garder un historique complet des requêtes SQL en cours sur le serveur (sur bddmoodle pour l'instant), lancer :

```
. /root/check_serveur_bdd.sh
```

Ce script execute notamment les requêtes ci-dessous et stocke ces sorties dans un fichier texte daté (dans /root/) pour analyse ultérieure :

```
SHOW FULL processlist
SHOW ENGINE INNODB STATUS\G
```

Regarder l'historique des requêtes les plus longues

(documenter la manière d'activer cette fonctionnalité...)

Le traçage exhaustif de ces requêtes suspectes se trouve dans le fichier de log :

```
/var/log/mysql/mysql-slow.log
```

Il concerne le jour courant. *logrotate* archive les logs des jours précédents dans les fichier *mysql-slow.log.n.gz* ('n' = itération)

Traçage d'indicateurs

Pour suivre l'évolution dans le temps de certains indicateurs clés, on peut ajouter une tache cron au serveur, avec *crontab -e* :

```
*/1 * * * * /root/check_serveur_bdd.sh
```

Dans le script *check_serveur_bdd.sh*, placer :

```
echo `date +"%d/%m/%Y %H:%M:%S"` = `mysql -u root -p<pw...> -h
bddmoodle.unicaen.fr information_schema -BNe 'select count(*) from
processlist'` >> nb_bdd_queries.log
echo `date +"%d/%m/%Y %H:%M:%S"` = `mysql -u root -p<pw...> -h
bddmoodle.unicaen.fr information_schema -BNe "show status like
'opened_files'"` >> nb_bdd_opened_files.log
```

*(avec **<pw…>** pour le mot de passe root du serveur mysql)*

Ensuite les 2 fichiers générés (*nb_bdd_....log*) peuvent être

- soit regardé sur place, par exemple avec

```
tail nb_bdd_*
```

On attend un nombre de requêtes aux environ de 50. Le nombre de fichiers ouverts, lui, est cumulatif, donc il augmente sans cesse. Ce qui nous intéresse est la vitesse à laquelle il augmente (actuellement est sur un rythme d'environ 14000 en 5 minutes)

- soit récupérés (par exemple sur un poste de travail via un *scp*) et ouverts dans Libre Office Calc (ajouter '**=**' comme séparateur de champs au moment de l'ouverture) pour faire l'objet de graphiques.

Regarder en temps réel les requêtes s'exécuter

Pour surveiller en temps réel l'activité du serveur MySQL, et voir chaque nouvelle requête qui lui est demandée, il faut configurer le serveur MySQL pour avoir ces informations dans une sortie de log verbeuse.

- Avantage : temps réel

- Inconvénients :
  - c'est très verbeux et peu vite devenir illisible
  - génère potentiellement de très gros fichier de log
  - impact sur les performances

(voir la doc MySQL officielle)

Ce qui est expliqué ici concerne MySQL version > 5.1.29 (avant c'était d'autres options de config.)

Ne pas le faire sur la base de données d'une plateforme en production. le fichier de log serait trop gros !

Se connecter à la base de données avec le client SQL de son choix et taper

```
SET GLOBAL general_log=1;
```

1. On peut également limiter la sortie de log aux grosses requêtes (plus de 10 sec.) (voir les options de configuration *slow_query_log*, *slow_query_log_file*, *long_query_time*, et la doc). Je n'ai pas essayé.

Pour voir en direct les requêtes exécutées, sur le serveur mariadb, taper :

```
tail -f /var/lib/mysql/$(hostname).log
```

Performances MySQL

chapitre à l'état embryonnaire...

(synthèse d'informations déduites du ticket 116024)

Parmi les paramètres importants concernant les performances d'ecampus, il y a :

- ***innodb_buffer_pool_size*** qui définit l'espace mémoire alloué à MySQL (à son moteur innodb en particulier)
- ***join_buffer_size*** taille du buffer disponible pour effectuer des jointures
- ***table_definition_cache*** (lié à ***innodb_open_files***, voir https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar_table_definition_cache)

Pour **consulter** la valeur d'une variable de configuration, par exemple sur ecampus :

```
mysql -u root -p ecampus -e "show variables like 'innodb_buffer_pool_size';"
```

(qui retourne ici *2147483648*, ce qui signifie que MySQL dispose de 2Go de RAM pour travailler).

Pour modifier une variable de façon durable, il faut le faire dans le fichier de configuration MySQL.

Après migration vers mariadb, c'est maintenant ***/etc/mysql/my.cnf.migrated*** qui est utilisé.

Pour que les modifications soient prises en compte, il faut lancer un

```
service mysql restart
```

(un *reload* ne suffit malheureusement pas)

RAPPEL ! Des risques de déconnexions utilisateurs existe lors de l'opération de redémarrage.