Tuning MySQL – Gestion et Optimisation de l'utilisation de la mémoire

I► Dans cet article, nous allons voir comment MySQL gère et utilise la mémoire :

Avant de comprendre comment MySQL gère les allocations mémoires, faisons un bref petit rappel des différences existantes entre les mémoires **registres** | **buffers** | **caches** :
Les registres, buffers et caches sont tous les 3 des zones d'allocations en mémoire.
Leurs différences se situent au niveau de leur gestion et de leur manière d'accéder aux informations des ces zones.

? Les **registres** (ou mémoire **tampon**) se situent directement dans le processeur et adressent directement les données dans des zones que l'on pourrait comparer à des boites placées exactement là ou elles doivent se trouver. Le registre est adapté aux calculs des demandes des processus.

? Les **caches** (ou **anté-mémoires**) se situent dans le processeur et/ou en mémoire vive mais la gestion des informations est gérée sous formes de tables de données et les données sont beaucoup plus rapidement accessible et sont permanentes. Le cache pourrait se comparer à un buffer en RAM si ce n'est qu'il conserve les informations.

? Les **buffers** se situent en RAM dans des files d'attente (exemple : la pile FIFO => First IN First OUT) que gère l'ordonnanceur du noyau du système. Les données sont stockées de manière temporaire.

► Voyons maintenant comment MySQL gère ses buffers et ses

caches :

? MySQL gère de manière particulière la dénomination de ses variables caches et buffers.
Le **key_buffer_size** par exemple des tables MyISAM (décrit ci-dessous) est en fait une mémoire de type cache et non un buffer…
Il faudra donc veiller à ne pas confondre les définitions générales et le vocabulaire propre à MySQL ?

Voyons comment se font les allocations mémoire de MySQL :

Au démarrage du serveur plusieurs allocations mémoires sont crées et utilisées par la suite lors des connexions et requêtes jouées par les clients. Voici une liste des plus importants caches :

- Le cache d'index pour MyIsam (le **key_buffer_size**)
- le cache d'index pour innoDB (le **innoDB_buffer_pool_size**)

Ces 2 caches mémoire sont les plus importants et les plus gourmands en ressources. Il gère les index des tables des bases de données.
On peut voir la taille de ces caches grimper jusqu'à plusieurs Go voire plusieurs dizaines de Go pour le buffer_pool innoDB !

Il existe aussi un autre cache très important à personnaliser : **] query cache [**
Les requêtes déjà jouées sur le serveur sont stockées dans ce cache.
Sa taille varie entre 128 et 256 Mo.

Chaque connexion au serveur crée un thread de connexion (allocation par threads). Cette mémoire est allouée et désallouée dynamiquement en fonction des besoins des clients connectés (threads). Lorsque la connexion se termine, soit le

thread est détruit et la mémoire libérée, soit conservé si le cache le permet. Ceci évite de recréer un thread et accélère le processus.

MySQL gère par lui même des tables en mémoire (**tables MEMORY**) => lors de tri par exemple.
Leur taille maximale est fixé par la plus petite valeur entre **] tmp_table_size [** => la taille des tables temporaires et **] max_heap_table_size [** => la taille maximale des tables Memory.

MySQL stocke en mémoire les tables des droits et privilèges des utilisateurs pour des raisons d'optimisation de performances lors de la lecture des requêtes.

MySQL gère également des buffers utlilisés par le sthreads clients lors des requêtes.
Exemple : **] sort_buffer_size [** et **] read_buffer_size [**

Voici un schéma illustrant l'utilisation de la mémoire par MySQL :

? Investir dans la mémoire Vive peut être une solution plus qu'intéressante pour améliorer les performances de votre serveur surtout si les requêtes sont principalement des requêtes en lecture car les accès solliciteront beaucoup plus la RAM que le disque. Qui plus est, le SWAPINESS sera fortement moins sollicité..

? Optimiser les performances d'un serveur sollicitant les accès en écriture se fera par l'acquisition de matériel DISQUE performant de type RAID | SSD]Solid State Drive [| cartes controleurs BBWC] Battery Back Write Cache]

? L'outil mysqlreport permet d'analyser les ratio des lecture/écriture du serveur MySQL.

Le fichier de configuration du serveur MySQL est my.cnf | my.ini permet de personnaliser et de customiser le server MySQL …

Pour optimiser le serveur MySQL, voici les directives les plus importantes à paramétrer

? max_connections (100) : limite le nombre de connexions simultanées établies au serveur MySQL.
Si la variable @@max_user_connections est proche de @@max_connections, il vaut mieux l'augmenter. On peut également indiquer le nombre de connections autorisés depuis Apache (MaxClients / httpd.conf)…

? key_buffer_size : est le cache pour les index des tables MySQL. L'idéal est 25% à 50% de la moyenne de mémoire disponible du serveur.

? thread_cache_size : détermine le nombre maximal de processus ouverts simultanément par le service MySQL.
Quand une connexion au serveur MySQL se termine, le thread associé à cette connexion est placé dans un cache afin de pouvoir être réutilisé, évitant une opération de création d'un nouveau thread.
La commande show status like 'created_threads'; permettra d'ajuster cette directive.

? table_cache_size : stocke des informations sur les tables. Quand le serveur a besoin d'utiliser une table, ce cache en accélère l'accès.
Depuis la version 5.1, le cache est divisé en deux parties :

- cache des tables ouvertes **(table_open_cache)**
- cache de définition de table (**table_definition_cache**).

Les directives **open_tables** et **opened_tables** permettent

d’ajuster la valeur de **table_open_cache**

Les directives **open_table_definitions** et **opened_table_definitions** permettent d’ajuster la valeur de **table_definition_cache**.

Michel BOCCIOLESI