MariaDB/MySQL Avancé

Pierre Mavro
<pierre@mavro.fr>

Sommaire

Sommaire

Sommaire

Introduction

MySQL

MySQL est un système de gestion de base de données (SGBD). Il est distribué sous une double licence GPL et propriétaire. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde, autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle, Informix et Microsoft SQL Server.

Son nom vient du prénom de la fille du co-créateur Michael Widenius, My. SQL fait allusion au Structured Query Language, le langage de requête utilisé.

MySQL AB a été acheté le 16 janvier 2008 par Sun Microsystems pour un milliard de dollars américains. En 2009, Sun Microsystems a été acquis par Oracle Corporation, mettant entre les mains d'une même société les deux produits concurrents que sont Oracle Database et MySQL. Ce rachat a été autorisé par la Commission européenne le 21 janvier 2010.

Depuis mai 2009, son créateur Michael Widenius a créé MariaDB pour continuer son développement en tant que projet Open Source.

Source : Wikipedia

Sommaire

Introduction

MariaDB

Suite au rachat de MySQL par Sun Microsystems et des annonces du rachat de Sun Microsystems par Oracle Corporation, Michael Widenius, fondateur de MySQL, quitte cette société pour lancer Maria DB, version concurrente (fork) et 100 % compatible avec MySQL.

Les différentes versions de MariaDB s'articulent sur le code source de MySQL de la version 5.1 aux versions plus récentes (comme la 5.6 fin 2012).

En parallèle, un consortium a été créé et sera chargé du développement de MariaDB : l'Open Database Alliance.

Depuis décembre 2012, Wikipédia utilise MariaDB.

En avril 2013 Mariadb (Monty Program AB) signe et annonce un accord de fusion avec la société SkySQL, l'objectif est de développer MariaDB dans une base de données open source « NewSQL », regroupant le meilleur de SQL et de NoSQL.

Source : Wikipedia

Sommaire

Installation

MariaDB

MariaDB est un logiciel relativement jeune qui évolue vite. Cependant, il n'est pas encore disponible dans toutes les distributions. Il existe des dépôts officiels de MariaDB disponible sur le site permettant d'avoir accès à toutes les releases et de choisir sa version majeure :
`https://downloads.mariadb.org/mariadb/repositories/`

Sur une Debian Wheezy (la distribution utilisée ici), il faut ajouter les clés permettant d'approuver le respository pour éviter les warning d'apt et avoir accès aux packages spécifiques pour Debian :

```
aptitude install python-software-properties
apt-key adv --recv-keys --keyserver keyserver.ubuntu.com \
0xcbcb082a1bb943db
add-apt-repository \'deb http://ftp.igh.cnrs.fr/pub/mariadb/repo
/5.5/debian wheezy main'
```

Installation

MariaDB

Afin que MariaDB puisse s'installer, il faut utiliser un fichier de préférence APT dans Configuration File /etc/apt/preferences.d/ :

/etc/apt/preferences.d/mariadb

```
Package: *
Pin: origin ftp.igh.cnrs.fr
Pin-Priority: 1000
```

Il ne reste plus qu'à procéder à l'installation :

```
aptitude update
aptitude install mariadb-server
```

Sommaire

Migration

MariaDB

Comment faire si un serveur MySQL est déjà existant ? Il suffit de remplacer les binaires :-). Cependant il y a quelques précautions à prendre avant d'effectuer la migration :

Migration

MariaDB

Comment faire si un serveur MySQL est déjà existant ? Il suffit de remplacer les binaires :-). Cependant il y a quelques précautions à prendre avant d'effectuer la migration :

- Ne remplacer MySQL que par une version équivalente de MariaDB (ex : MySQL 5.5 -> MariaDB 5.5)

Migration

MariaDB

Comment faire si un serveur MySQL est déjà existant ? Il suffit de remplacer les binaires :-). Cependant il y a quelques précautions à prendre avant d'effectuer la migration :

- Ne remplacer MySQL que par une version équivalente de MariaDB (ex : MySQL 5.5 -> MariaDB 5.5)
- Ne pas changer la configuration (sauf pour des options obsolète ou nécessaires)

Migration

MariaDB

Comment faire si un serveur MySQL est déjà existant ? Il suffit de remplacer les binaires :-). Cependant il y a quelques précautions à prendre avant d'effectuer la migration :

- Ne remplacer MySQL que par une version équivalente de MariaDB (ex : MySQL 5.5 -> MariaDB 5.5)
- Ne pas changer la configuration (sauf pour des options obsolète ou nécessaires)

Ensuite, vous pouvez effectuer des montées de version de MariaDB ! Vous pouvez même repasser sur MySQL si vous n'êtes pas satisfait de MariaDB en changeant de nouveaux les binaires ! ! ! !

Migration

MariaDB

Voici la procédure pour migrer. Après avoir ajouté le dépôt de MariaDB, désinstallez MySQL :

```
aptitude remove mysql-server mysql-client
```

Puis, installez MariaDB :

```
aptitude update
aptitude install mariadb-server
```

Démarrez le service et ça marche, toujours avec le même nom de binaire !

```
service mysql start
```

Sommaire

Sommaire

Stockage et verrouillage

Type de données

- Les types numériques (http://dev.mysql.com/doc/refman/5.0/fr/numeric-types.html)
- Les types de données temporelles : time/year/timestamp (http://dev.mysql.com/doc/refman/5.0/fr/date-and-time-types.html)
- Types de chaines : char/binary/blob/text (http://dev.mysql.com/doc/refman/5.0/fr/string-types.html)

http://dev.mysql.com/doc/refman/5.0/fr/column-types.html

Sommaire

Stockage et verrouillage

Verrouillage des données

Actuellement, MySQL ne supporte que le verrouillage :

- Par table pour ISAM/MyISAM et MEMORY (HEAP)
- Par ligne de table pour InnoDB
- Par page pour DBD

Rappel : une page est une donnée interne permettant de structurer les données de la base. La concaténation logique de ses pages forment la tablespace (16KB sur InnoDB par défaut). Les pages sont groupées en "extents" de 1M (64 pages consécutives) dans une tablespace appelée "Segment" sur InnoDB.

Stockage et verrouillage

Verrouillage des données

Dans de nombreux cas, vous pouvez faire prévoir le type de verrouillage qui sera le plus efficace pour une application, mais il est très difficile de savoir si un type de verrou est meilleur que l'autre. Tout dépend de l'application, et des différentes composants qui utilisent les verrous.

Toutes les méthodes de verrouillage de MySQL sont exemptes de blocage, sauf pour les tables InnoDB et BDB

Ceci fonctionne en demandant tous les verrous d'un seul coup, au début de la requête, et en verrouillant les tables toujours dans le même ordre.

Stockage et verrouillage

Verrouillage des données

La méthode de verrouillage des tables de MySQL en écriture (WRITE) fonctionne comme ceci :

- Si il n'y a pas de verrou sur la table, pose un verrou en écriture dessus.
- Sinon, soumet une requête de verrouillage dans la queue de verrous d'écriture.

Stockage et verrouillage

Verrouillage des données

La méthode de verrouillage des tables de MySQL en écriture (WRITE) fonctionne comme ceci :

- Si il n'y a pas de verrou sur la table, pose un verrou en écriture dessus.
- Sinon, soumet une requête de verrouillage dans la queue de verrous d'écriture.

La méthode de verrouillage des tables de MySQL en lecture (READ) fonctionne comme ceci :

- Si il n'y a pas de verrou sur la table, pose un verrou en écriture dessus.
- Sinon, soumet une requête de verrouillage dans la queue de verrou de lecture.

Lorsqu'un verrou est libéré, le verrou est donné aux threads de la queue de verrou en écriture, puis à ceux de la queue de verrou en lecture.

Stockage et verrouillage

Verrouillage des données

Cela signifie que si vous avez de nombreuses modifications dans une table, la commande SELECT va attendre qu'il n'y ait plus d'écriture avant de lire.

Vous pouvez analyser le comportement des verrous sur une table avec les variables de statut Table_locks_waited et Table_locks_immediate :

```
mysql> SHOW STATUS LIKE 'Table%';
+-----------------------+---------+
| Variable_name         | Value   |
+-----------------------+---------+
| Table_locks_immediate | 1151552 |
| Table_locks_waited    | 15324   |
+-----------------------+---------+
```

Sommaire

Stockage et verrouillage

Les différents moteurs de stockage

L'une des spécificités de MySQL est de pouvoir gérer plusieurs moteurs au sein d'une seule base. Chaque table peut utiliser un moteur différent au sein d'une base. Ceci afin d'optimiser l'utilisation de chaque table. Voici les moteurs les plus connu de MySQL :

Stockage et verrouillage

Les différents moteurs de stockage

L'une des spécificités de MySQL est de pouvoir gérer plusieurs moteurs au sein d'une seule base. Chaque table peut utiliser un moteur différent au sein d'une base. Ceci afin d'optimiser l'utilisation de chaque table. Voici les moteurs les plus connu de MySQL :

- MyISAM : moteur par défaut de MySQL. Il est le plus simple à utiliser et à mettre en œuvre. Il utilise plusieurs fichiers qui grandissent au fur et à mesure que la base grossit. Il ne supporte pas les transactions, ni les clefs étrangères

Stockage et verrouillage

Les différents moteurs de stockage

L'une des spécificités de MySQL est de pouvoir gérer plusieurs moteurs au sein d'une seule base. Chaque table peut utiliser un moteur différent au sein d'une base. Ceci afin d'optimiser l'utilisation de chaque table. Voici les moteurs les plus connu de MySQL :

- MyISAM : moteur par défaut de MySQL. Il est le plus simple à utiliser et à mettre en œuvre. Il utilise plusieurs fichiers qui grandissent au fur et à mesure que la base grossit. Il ne supporte pas les transactions, ni les clefs étrangères
- InnoDB : moteur créé et maintenu par InnoBase (racheté par Oracle le 7 octobre 2005). Il gère les transactions et les clefs étrangères (et donc l'intégrité de ses tables). En contrepartie, les bases qui l'utilisent occupent bien plus d'espace sur le disque

Stockage et verrouillage

Les différents moteurs de stockage

- ARCHIVE : moteur adapté à l'archivage de données. Les lignes sont compressées au fur et à mesure de leur insertion. Les requêtes de recherches sont alors sensiblement plus lentes

Stockage et verrouillage

Les différents moteurs de stockage

- ARCHIVE : moteur adapté à l'archivage de données. Les lignes sont compressées au fur et à mesure de leur insertion. Les requêtes de recherches sont alors sensiblement plus lentes
- MEMORY (HEAP) : moteur où les tables sont stockées uniquement en mémoire. La structure de la base est stockée sur le disque dur mais les données sont stockées dans la RAM, si la machine serveur redémarre, les données seront perdues. Cependant, étant donné qu'il n'y a plus d'accès disque, une requête de modification (UPDATE, INSERT...) s'exécutera bien plus rapidement et sans charger les bras d'accès ; convient pour les mémorisations temporaires, comme un panier en e-commerce

Stockage et verrouillage

Les différents moteurs de stockage

- BLACKHOLE : moteur réceptionnant les données, les transférant mais ne les stockant pas. Il peut être utilisé comme répéteur ou comme filtre de données

Stockage et verrouillage

Les différents moteurs de stockage

- BLACKHOLE : moteur réceptionnant les données, les transférant mais ne les stockant pas. Il peut être utilisé comme répéteur ou comme filtre de données
- NDB (uniquement dans la version MaxDB) : moteur de base de données réseau gérant les grappes de serveurs

Sommaire

Sommaire

Intégrité des données

Maintenance des tables

Pour réparer des tables de type MyISAM, vous pouvez utiliser myisamchk ou plus simplement la commande universelle REPAIR. Elle vous permettra de réparer n'importe quel type de table.

Pour réparer une base de donnée complète :

```
select concat('repair table ', table_name, ';')
from information_schema.tables
where table_schema='mydatabase';
```

http ://dev.mysql.com/doc/refman/5.0/en/mysqlcheck.html
http ://dev.mysql.com/doc/refman/5.0/fr/table-maintenance.html

Sommaire

Intégrité des données

Sauvegarde et restauration - mysqldump

Par défaut, MySQL/MariaDB viennent avec un outil appelé mysqldump pour effectuer les sauvegardes. Il fonctionne très bien pour les bases de petites taille (inférieur à quelques gigas). Voici comment l'utiliser :

```
mysqldump -uUTILISATEUR -pMOTDEPASSE BASEDEDONNEES > backup.sql
```

Pour restaurer cette base :

```
mysql -uUTILISATEUR -pMOTDEPASSE BASEDEDONNEES < backup.sql
```

Simple d'utilisation, il existe cependant beaucoup d'options pour la commande mysqldump.

Intégrité des données

Sauvegarde et restauration - mysqldump

Voici quelques options intéressantes pour mysqldump :

- –no-data : permet de générer un fichier qui contiendra toutes les instructions de création des tables d'une base (CREATE DATABASE, CREATE TABLE...), mais pas les données elles-mêmes.

Intégrité des données

Sauvegarde et restauration - mysqldump

Voici quelques options intéressantes pour mysqldump :

- –no-data : permet de générer un fichier qui contiendra toutes les instructions de création des tables d'une base (CREATE DATABASE, CREATE TABLE...), mais pas les données elles-mêmes.
- –no-createdb et –no-create-info permettent de commenter automatiquement les instructions de création de schéma. Donc d'insérer des données depuis une sauvegarde dans une base avec des tables déjà existantes.

Intégrité des données

Sauvegarde et restauration - mysqldump

Voici quelques options intéressantes pour mysqldump :

- –no-data : permet de générer un fichier qui contiendra toutes les instructions de création des tables d'une base (CREATE DATABASE, CREATE TABLE...), mais pas les données elles-mêmes.
- –no-createdb et –no-create-info permettent de commenter automatiquement les instructions de création de schéma. Donc d'insérer des données depuis une sauvegarde dans une base avec des tables déjà existantes.
- –lock-tables : permet de poser un LOCK sur toute une base. On obtient alors une cohérence entre les tables.

Intégrité des données

Sauvegarde et restauration - mysqldump

- –lock-all-tables : permet de poser un LOCK sur toutes les tables de toutes les bases de données. Soyez conscient des impacts d'un tel LOCK sur une grosse base de données non transactionnel comme MyISAM. La mise en attente de toutes les requêtes pendant la durée de la sauvegarde pourrait faire croire aux visiteurs que le serveur du site a planté. A la fin du dump, toutes les requêtes en attente seront exécutées (pas de pertes, sauf si la longueur de la queue a été dépassée).

Intégrité des données

Sauvegarde et restauration - mysqldump

- –lock-all-tables : permet de poser un LOCK sur toutes les tables de toutes les bases de données. Soyez conscient des impacts d'un tel LOCK sur une grosse base de données non transactionnel comme MyISAM. La mise en attente de toutes les requêtes pendant la durée de la sauvegarde pourrait faire croire aux visiteurs que le serveur du site a planté. A la fin du dump, toutes les requêtes en attente seront exécutées (pas de pertes, sauf si la longueur de la queue a été dépassée).
- -e : permet de cumuler les création d'éléments de même type (va donc plus vite à la réimportation)

Intégrité des données

Sauvegarde et restauration - mysqldump

- –single-transaction : permet de lancer les sauvegardes dans une transaction. Elle garantit donc que les tables InnoDB (moteur transactionnel) seront dans un état cohérent entre elles. Les tables qui ne supportent pas les transactions seront également sauvegardées, mais la cohérence ne sera pas garantie (l'instruction BEGIN est simplement ignorée pour ces tables). L'avantage de cette option est la non utilisation d'un LOCK. En faite, elles seront verrouillés à l'intérieur de la transaction, sans empêcher les autres clients de faire des modifications de la base.

La meilleure façon d'utiliser mysqldump est donc :

```
mysqldump -uUTILISATEUR -pMOTDEPASSE -e --single-transaction \
--opt BASEDEDONNEES > backup.sql
```

Sommaire

Intégrité des données

Sauvegarde et restauration - xtrabackup

Xtrabackup est une solution libre créer par la société Percona permettant d'optimiser vos backups MySQL/MariaDB. Il est bien plus rapide que mysqldump puisqu'il joue avec les fichiers et non des requêtes SQL. Il est capable de sauvegarder uniquement des tables de type InnoDB, XtraDB et MyISAM.

Les points forts d'Xtrabackup sont :

- Les sauvegardes terminent rapidement et de manière fiable (environ 25x plus rapide)

Intégrité des données

Sauvegarde et restauration - xtrabackup

Xtrabackup est une solution libre créer par la société Percona permettant d'optimiser vos backups MySQL/MariaDB. Il est bien plus rapide que mysqldump puisqu'il joue avec les fichiers et non des requêtes SQL. Il est capable de sauvegarder uniquement des tables de type InnoDB, XtraDB et MyISAM.

Les points forts d'Xtrabackup sont :

- Les sauvegardes terminent rapidement et de manière fiable (environ 25x plus rapide)
- Traitement des transactions sans interruption durant les sauvegardes

Intégrité des données

Sauvegarde et restauration - xtrabackup

Xtrabackup est une solution libre créer par la société Percona permettant d'optimiser vos backups MySQL/MariaDB. Il est bien plus rapide que mysqldump puisqu'il joue avec les fichiers et non des requêtes SQL. Il est capable de sauvegarder uniquement des tables de type InnoDB, XtraDB et MyISAM.

Les points forts d'Xtrabackup sont :

- Les sauvegardes terminent rapidement et de manière fiable (environ 25x plus rapide)
- Traitement des transactions sans interruption durant les sauvegardes
- Économie d'espace disque, de bande passante sur le réseau et de charge sur la machine

Intégrité des données

Sauvegarde et restauration - xtrabackup

Xtrabackup est une solution libre créer par la société Percona permettant d'optimiser vos backups MySQL/MariaDB. Il est bien plus rapide que mysqldump puisqu'il joue avec les fichiers et non des requêtes SQL. Il est capable de sauvegarder uniquement des tables de type InnoDB, XtraDB et MyISAM.

Les points forts d'Xtrabackup sont :

- Les sauvegardes terminent rapidement et de manière fiable (environ 25x plus rapide)
- Traitement des transactions sans interruption durant les sauvegardes
- Économie d'espace disque, de bande passante sur le réseau et de charge sur la machine
- Vérification des sauvegardes automatique

Intégrité des données

Sauvegarde et restauration - xtrabackup

- Temps de restauration extrêmement rapide

Intégrité des données

Sauvegarde et restauration - xtrabackup

- Temps de restauration extrêmement rapide
- Compression à la volée

Intégrité des données

Sauvegarde et restauration - xtrabackup

- Temps de restauration extrêmement rapide
- Compression à la volée
- Sauvegardes incrémentales

Intégrité des données

Sauvegarde et restauration - xtrabackup

- Temps de restauration extrêmement rapide
- Compression à la volée
- Sauvegardes incrémentales
- Chiffrement des sauvegardes

Intégrité des données

Sauvegarde et restauration - xtrabackup

- Temps de restauration extrêmement rapide
- Compression à la volée
- Sauvegardes incrémentales
- Chiffrement des sauvegardes
- Comparaison de sauvegardes

Intégrité des données

Sauvegarde et restauration - xtrabackup

- Temps de restauration extrêmement rapide
- Compression à la volée
- Sauvegardes incrémentales
- Chiffrement des sauvegardes
- Comparaison de sauvegardes
- Déplacement de tables d'un serveur à un autre à chaud

Intégrité des données

Sauvegarde et restauration - xtrabackup

- Temps de restauration extrêmement rapide
- Compression à la volée
- Sauvegardes incrémentales
- Chiffrement des sauvegardes
- Comparaison de sauvegardes
- Déplacement de tables d'un serveur à un autre à chaud
- Facilite la création de slave
- ...

http ://www.percona.com/doc/percona-xtrabackup/2.1/intro.html

Intégrité des données

Sauvegarde et restauration - xtrabackup

Pour installer Xtrabackup, il faut utiliser le dépôt officiel de Percona :

```
apt-key adv --keyserver keys.gnupg.net \
--recv-keys 1C4CBDCDCD2EFD2A
```

/etc/apt/preferences.d/percona

```
deb http://repo.percona.com/apt wheezy main
deb-src http://repo.percona.com/apt wheezy main
```

Puis installer le serveur :

```
aptitude update
aptitude install percona-server-server percona-server-client
```

Intégrité des données

Sauvegarde et restauration - xtrabackup

Pour effectuer des sauvegardes complètes :

```
innobackupex --user=xxxxx --password=xxxx \
--databases=nom_de_la_base repertoire_ou_stocker_le_backup
```

Pour des sauvegardes incrémentales :

```
innobackupex --incremental repertoire_ou_stocker_le_backup \
--incremental-basedir=repertoire_contenant_le_full --user=root \
--password=xxxxx
```

Note : il faut une sauvegarde complète pour pouvoir effectuer une incrémentale

Intégrité des données

Sauvegarde et restauration - xtrabackup

Pour la restauration depuis une sauvegarde complète, l'opération se fait en trois étapes :

1. Utiliser l'argument apply-log :

```
innobackupex --apply-log repertoire_du_backup_full
```

2. Déplacer la base de donnée existante pour éviter tout résidu indésirable :

```
mv /var/lib/mysql/dossier_de_la_base{,.old}
```

3. Lancer cette commande pour restaurer (argument copy-back) :

```
innobackupex --ibbackup=xtrabackup \
--copy-back repertoire_du_backup_full
```

Intégrité des données

Sauvegarde et restauration - xtrabackup

Pour effectuer une restauration par rapport à une sauvegarde incrémentale, il faut à nouveau plusieurs étapes :

1. Préparer le backup full : ici, le use-memory est optionnel, il permet juste d'accélérer le processus :

```
innobackupex --apply-log --redo-only repertoire_du_full_backup \
[--use-memory=1G] --user=root --password=xxxxx
```

2. Appliquer les backups incrémentiels, **dans l'ordre** :

```
innobackupex --apply-log repertoire_du_full_backup \
--incremental-dir=repertoire_du_backup_incrementiel \
[--use-memory=1G] --user=root --password=xxxxx
```

Intégrité des données

Sauvegarde et restauration - xtrabackup

3. Préparer le backup final :

```
innobackupex --apply-log repertoire_du_full_backup \
[--use-memory=1G] --user=root --password=xxxxx
```

4. Restaurer le backup final :

```
mv /var/lib/mysql/dossier_de_la_base{,.old\}
innobackupex --ibbackup=xtrabackup \
--copy-back repertoire_du_backup_full
```

Intégrité des données

Sauvegarde et restauration - xtrabackup

Il existe des interfaces pour commander Xtrabackup et avoir un peu plus de visibilité sur l'état des sauvegardes :

- FenriSQL `http` ://sourceforge.net/projects/fenrisql/ ?source=directory
- XtraBackup-Manager
  `http` ://code.google.com/p/xtrabackup-manager/

Note : FenriSQL peut être forké avec plaisir :-)

Sommaire

Gestion de la sécurité

La sécurité d'une base de données est critique dans n'importe quel SI. Voici quelques bonnes pratiques pour éviter les problèmes :

- N'autoriser l'écoute du serveur que sur le loopback (bind-address)

Gestion de la sécurité

La sécurité d'une base de données est critique dans n'importe quel SI. Voici quelques bonnes pratiques pour éviter les problèmes :

- N'autoriser l'écoute du serveur que sur le loopback (bind-address)
- Après l'installation de MySQL/MariaDB, lancez la commande mysql_secure_installation pour changer le mot de passe root et supprimer la base/utilisateur de test.

Gestion de la sécurité

La sécurité d'une base de données est critique dans n'importe quel SI. Voici quelques bonnes pratiques pour éviter les problèmes :

- N'autoriser l'écoute du serveur que sur le loopback (bind-address)
- Après l'installation de MySQL/MariaDB, lancez la commande mysql_secure_installation pour changer le mot de passe root et supprimer la base/utilisateur de test.
- Les privilèges utilisateurs des bases de données ne doivent avoir que les droits minimum pour leur utilisation.

Gestion de la sécurité

La sécurité d'une base de données est critique dans n'importe quel SI. Voici quelques bonnes pratiques pour éviter les problèmes :

- N'autoriser l'écoute du serveur que sur le loopback (bind-address)
- Après l'installation de MySQL/MariaDB, lancez la commande mysql_secure_installation pour changer le mot de passe root et supprimer la base/utilisateur de test.
- Les privilèges utilisateurs des bases de données ne doivent avoir que les droits minimum pour leur utilisation.
- Limiter le nombre de connections (max_connections/max_user_connections)

Gestion de la sécurité

La sécurité d'une base de données est critique dans n'importe quel SI. Voici quelques bonnes pratiques pour éviter les problèmes :

- N'autoriser l'écoute du serveur que sur le loopback (bind-address)
- Après l'installation de MySQL/MariaDB, lancez la commande mysql_secure_installation pour changer le mot de passe root et supprimer la base/utilisateur de test.
- Les privilèges utilisateurs des bases de données ne doivent avoir que les droits minimum pour leur utilisation.
- Limiter le nombre de connections (max_connections/max_user_connections)
- Connections SSL pour les applications

Gestion de la sécurité

La sécurité d'une base de données est critique dans n'importe quel SI. Voici quelques bonnes pratiques pour éviter les problèmes :

- N'autoriser l'écoute du serveur que sur le loopback (bind-address)
- Après l'installation de MySQL/MariaDB, lancez la commande mysql_secure_installation pour changer le mot de passe root et supprimer la base/utilisateur de test.
- Les privilèges utilisateurs des bases de données ne doivent avoir que les droits minimum pour leur utilisation.
- Limiter le nombre de connections (max_connections/max_user_connections)
- Connections SSL pour les applications
- Chrooter MySQL/MariaDB

Gestion de la sécurité

La sécurité d'une base de données est critique dans n'importe quel SI. Voici quelques bonnes pratiques pour éviter les problèmes :

- N'autoriser l'écoute du serveur que sur le loopback (bind-address)
- Après l'installation de MySQL/MariaDB, lancez la commande mysql_secure_installation pour changer le mot de passe root et supprimer la base/utilisateur de test.
- Les privilèges utilisateurs des bases de données ne doivent avoir que les droits minimum pour leur utilisation.
- Limiter le nombre de connections (max_connections/max_user_connections)
- Connections SSL pour les applications
- Chrooter MySQL/MariaDB
- Ne pas omettre un outil d'administration sur la machine ou tourne MySQL/MariaDB (phpmyadmin & co)

Sommaire

Sommaire

Performances

Configuration

La configuration par défaut est faite pour un Pentium II 400Mhz. Ce n'est donc pas une configuration pour les performances, mais la plus légère. Cependant, en standard, plusieurs exemples de configuration sont fournies vous permettant d'augmenter les performances de vos instances MySQL (my-small.cnf, my-medium.cnf, my-large.cnf et my-huge.cnf).

Certains paramètres sont indispensable pour augmenter les performances de MySQL/MariaDB. Récupérez une configuration optimisée ici :

`https://git.deimos.fr/?p=git_deimosfr.git;a=blob_plain;f=server_config/mariadb/my.cnf`

Performances

Configuration

Avant de remplacer la configuration actuelle par l'optimisée, il faut prendre connaissance des valeurs actuelles :

```
SHOW VARIABLES;
```

Voici la première partie qui peut sembler anodine au premier abord :

/etc/mysql/my.cnf

```
datadir      = /var/lib/mysql
tmpdir       = /tmp
lc_messages = en_US
character_set_server = utf8
collation_server = utf8_general_ci
```

- datadir : l'endroit ou vos bases de données seront stockées
- tmpdir stocke les éléments temporaires. **Il faut utiliser du tmpfs pour réduire les temps d'accès**
- les 3 éléments suivants évitent les problèmes d'encodage (aucuns rapports avec les performances)

Performances

Configuration

/etc/mysql/my.cnf

```
connect_timeout     = 5
wait_timeout        = 600
max_allowed_packet  = 16M
thread_cache_size   = 128
sort_buffer_size    = 16M
bulk_insert_buffer_size = 16M
tmp_table_size      = 32M
max_heap_table_size = 64M
net_buffer_length   = 4k
```

Afin de ne pas surcharger le serveur, on limite les connexions TCP.

- thread_cache_size : Indique combien de threads MySQL/MariaDB doit conserver en cache pour réutilisation. Lorsqu'un client se déconnecte, les threads du client sont mis en cache. S'il n'y en a pas déjà thread_cache_size de conservé. Tous les nouveaux threads sont d'abord prélevé dans le cache, et uniquement lorsque le cache est vide, un nouveau thread est créé. Cette variable peut vous permettre d'améliorer les performances si vous avez de nombreuses connexions.

Performances

Configuration

- sort_buffer_size : Chaque thread qui doit faire un tri, alloue un buffer de cette taille. Augmentez cette taille pour accélérer les clauses ORDER BY ou GROUP BY.
- bulk_insert_buffer_size : MyISAM utilise une cache hiérarchisé pour les insertions de masses (c'est à dire INSERT ... SELECT, INSERT ... VALUES (...), (...), ..., et LOAD DATA INFILE). Cette variable limite la taille du cache en octets, par threads. Utiliser la valeur de 0 va désactiver cette optimisation. Note : ce cache est uniquement utilisé lorsque vous ajoutez des données dans une table non-vide. Par défaut, cette option vaut 8 Mo.
- tmp_table_size : Si une table temporaire en mémoire excède cette taille, MySQL va automatiquement la convertir en une table MyISAM sur le disque. Augmentez la valeur de tmp_table_size si vous faites un usage intensif de la clause GROUP BY et que vous avez la taille de mémoire vive sur votre machine vous le permet.

Performances

Configuration

- max_heap_table_size : Ne pas autoriser la création de tables de type MEMORY (HEAP) plus grande que max_heap_table_size. La valeur de la variable est utilisée pour calculer la valeur maximale de MAX_ROWS pour la table MEMORY. Modifier cette variable n'a pas d'effet sur les tables existantes, à moins que la table ne soit recrée, avec une commande comme CREATE TABLE ou TRUNCATE TABLE, ou encore modifiée avec ALTER TABLE.

`http ://dev.mysql.com/doc/refman/5.0/fr/server-system-variables.html`

Performances

Configuration

/etc/mysql/my.cnf

```
query_cache_limit       = 128K
query_cache_size        = 64M
default_storage_engine  = InnoDB
```

- query_cache_limit : Ne met pas en cache les résultats qui sont plus grands que query_cache_limit. Par défaut, 1 Mo.
- query_cache_size : La mémoire allouée pour stocker les résultats des vieilles requêtes. Si query_cache_size vaut 0, le cache de requête est désactivé (par défaut).
- defaut_storage_engine : permet de spécifier que toute création de nouvelles tables non définies, sera par défaut en InnoDB.

Performances

Configuration

/etc/mysql/my.cnf

```
innodb_buffer_pool_size         = 256M
innodb_log_file_size            = 256M
innodb_log_buffer_size          = 8M
thread_concurrency          = 64
...
```

Attentions aux 2 première lignes. Pour les appliquer, vous devez :

Performances

Configuration

/etc/mysql/my.cnf

```
innodb_buffer_pool_size         = 256M
innodb_log_file_size            = 256M
innodb_log_buffer_size          = 8M
thread_concurrency          = 64
...
```

Attentions aux 2 première lignes. Pour les appliquer, vous devez :

- Arrêter le serveur MySQL
- Faire un backup des fichiers 'ib_log*'
- Supprimer les fichiers 'ib_log*'
- Relancer le serveur

Sommaire

Performances

Choisir son moteur

Une des particularités de MySQL est qu'il supporte plusieurs moteurs de stockage. Chaque moteur a ses avantages et ses inconvénients. Le choix du moteur se fait donc à la création de la base de données, selon l'utilisation que l'on va en faire et selon les performances désirées.

Les moteurs les plus courants sont MyISAM (le moteur proposé par défaut par MySQL) et InnoDB. Mais d'autres moteurs sont également proposés ayant chacun leur points forts et points faibles.

Performances

Choisir son moteur - MyISAM

Ce moteur est simple et rapide, car il ne gère pas les transactions et ne vérifie pas l'intégrité des bases. Chaque table MyISAM est enregistrée sous 3 fichiers :

- table.frm : définition de la table
- table.MYD : données
- table.MYI : index

La recherche en FULLTEXT permet une approche comparable aux moteurs de recherche.
Ce moteur est donc très utile pour des tables sur lesquelles vous devez faire beaucoup de recherche et pour lesquelles l'intégrité des données n'est pas primordiale.

Performances

Choisir son moteur - InnoDB/XtraDB

Le moteur de table InnoDB/XtraDB est un moteur transactionnel avec validation (COMMIT), annulation (ROLLBACK) et restauration après crash. De plus, il utilise un système de verrouillage, ce qui permet une utilisation simultanée accrue.
InnoDB est un moteur de tables relationnel, il utilise les contraintes de clés étrangères (FOREIGN KEY).

La structure d'une table InnoDB est enregistrée dans le fichier table.frm. Les données et le journal des transactions sont eux enregistrés dans des fichiers spécifiques (ib_logfile et ib_data). Ce moteur est à utiliser si l'intégrité de vos données est une priorité. Il est cependant moins performant que MyISAM en terme de vitesse sauf si il est correctement configuré.

Performances

Choisir son moteur - Archive

Ce moteur permet d'enregistrer et de lire des données. En effet, il ne supporte que les requêtes INSERT et SELECT. Il est très performant pour enregistrer des logs, qui n'ont pas besoin d'être modifiés ou supprimés. Les enregistrements sont compressés au moment de leur insertion.

Performances

Choisir son moteur - Memory / HEAP

Le moteur MEMORY est un moteur qui stocke les données en mémoire. Ce qui accélère le traitement des données, mais le contenu sera perdu au redémarrage du serveur.

Sommaire

Performances

Les types de données

Il est important de bien choisir son type de données. Beaucoup de programmes utilisent des colonnes inadaptées, ce qui se ressent par la suite sur les performances.

Par exemple, il n'est pas rare de voir la BDD d'un programme (celui qui a initialiser la structure de la base), dont une colonne de type INT (-2147483648->2147483647) va stocker des éléments de type SMALLINT (-32768->32767), voir BOOLEAN (0/1). Tout comme il est inutile de faire appel à un VARCHAR(255) pour des données qui ne dépassent pas 50 caractères.

Déterminer correctement les colonnes permet de diminuer l'espace requis par la base de données, limiter les accès disque et augmenter le nombre d'information stockées en mémoire et donc améliorer les performances. De même, lors d'une migration, plus la base est légère plus vite se font les dumps et les imports.

`http ://dev.mysql.com/doc/refman/5.0/fr/column-types.html ?ff=nopfpls`

Performances

Les types de données

De part leur nature, les types numériques s'avèrent plus performants (plus légers, ...) que les types chaînes de caractères dans les requêtes. Dans certains cas de figure, il est donc plus pertinent de faire appel à ce type de données. Notamment sur des données dont le nombre de valeurs possibles est limité et fermé. Par exemple, quand on veut stocker la civilité, plutôt que de stocker sous forme de chaîne de caractère ('M.', 'Mme', 'Melle'), il est plus efficace de stocker sous forme numérique (1 pour monsieur, 2 pour madame, 3 pour mademoiselle). Ainsi, les requêtes en lecture sur cette donnée se feront sur un type numérique et s'exécuteront donc plus vite, surtout sur des gros volumes.

Performances

Les types de données

- Un autre paramètre à envisager, lors de l'utilisation des clauses JOIN dans les requêtes, si les colonnes comparées sont de types différents, MySQL va réaliser une opération afin de convertir les données pour pouvoir les comparer, ce qui ralentira l'exécution de la requête. C'est donc également un critère à prendre en compte lors de la structuration des tables.
- La valeur NULL est une valeur différente des autres valeurs. Et de part sa spécificité, elle est plus lourde à stocker et nécessite des traitements particuliers. Elle a donc un impact négatif sur les performances. Il est donc préférable de systématiquement indiquer la clause NOT NULL sur chaque colonne.
- Utilisez également des types numériques plutôt que les types chaînes de caractères

Performances

Les types de données

Il existe une commande permettant de vérifier l'état d'une table, en donnant des statistiques et en proposant la meilleure solution d'optimisation (optimal_field_table) par rapport aux tables existantes (en s'appuyant sur l'analyse des données actuellement en base) :

```
SELECT * FROM wished_table PROCEDURE ANALYSE();
```

Analysez ensuite avec les champs actuels :

```
describe whished_table;
```

Vous connaissez maintenant les colonnes pouvant être optimisées.

Sommaire

Performances

Les indexes

Il existe 3 types d'index en MySQL : INDEX, UNIQUE, FULLTEXT.

- Les index de type INDEX sont les plus basiques dans le sens où ils segmentent la table sans autre contrainte. Pour autant ils permettent de traiter des sous-ensembles plus grands, ce qui peut être plus utiles sur certaines fonctionnalités.
- Les index de type UNIQUE fonctionnent sur le même principe qu'une clé primaire, à la seule différence qu'ils peuvent traiter les valeurs NULL. Les index de ce type vont segmenter la table tout en s'assurant qu'il n'y ait aucun doublon dans la colonne concernée. Exemple, si on place un index de type UNIQUE sur une colonne email, cet index empêchera qu'un même email soit enregistré deux fois dans la table en renvoyant une erreur MySQL.

Performances

Les indexes

- Les index de type FULL TEXT sont assez complexes à appréhender. Ils sont particulièrement adaptés pour les requêtes traitant des recherches sur des chaînes de caractères, permettant des manipulations avancées des textes et des critères de recherche. A noter que ce type d'index n'est pas supporté sur MySQL Cluster.

Il existe même des logiciels dédiés à ce type de requêtes dites "FULL TEXT" tel que Sphinx ou Apache Solar.

`http ://dev.mysql.com/doc/refman/5.0/fr/fulltext-search.html`

Performances

Les indexes

Pour déterminer quelles colonnes indexer, il suffit de répondre aux questions suivantes :

- Quelles sont les colonnes utilisées dans mes clauses WHERE ? (cas particuliers avec LIKE où seuls les filtres LIKE 'qquechose%', LIKE 'qque%chose%' exploitent les index, les filtres LIKE '%qquechose', LIKE '%qquechose%' et LIKE uneautrecolonne n'utilisent pas les index)

Performances

Les indexes

Pour déterminer quelles colonnes indexer, il suffit de répondre aux questions suivantes :

- Quelles sont les colonnes utilisées dans mes clauses WHERE ? (cas particuliers avec LIKE où seuls les filtres LIKE 'qquechose%', LIKE 'qque%chose%' exploitent les index, les filtres LIKE '%qquechose', LIKE '%qquechose%' et LIKE uneautrecolonne n'utilisent pas les index)
- Quelles sont les colonnes utilisées dans des jointures avec d'autres tables ?

Performances

Les indexes

Pour déterminer quelles colonnes indexer, il suffit de répondre aux questions suivantes :

- Quelles sont les colonnes utilisées dans mes clauses WHERE ? (cas particuliers avec LIKE où seuls les filtres LIKE 'qquechose%', LIKE 'qque%chose%' exploitent les index, les filtres LIKE '%qquechose', LIKE '%qquechose%' et LIKE uneautrecolonne n'utilisent pas les index)
- Quelles sont les colonnes utilisées dans des jointures avec d'autres tables ?
- Quelles sont les colonnes utilisées dans mes clauses GROUP BY ou ORDER BY ?

Performances

Les indexes

Pour déterminer quelles colonnes indexer, il suffit de répondre aux questions suivantes :

- Quelles sont les colonnes utilisées dans mes clauses WHERE ? (cas particuliers avec LIKE où seuls les filtres LIKE 'qquechose%', LIKE 'qque%chose%' exploitent les index, les filtres LIKE '%qquechose', LIKE '%qquechose%' et LIKE uneautrecolonne n'utilisent pas les index)
- Quelles sont les colonnes utilisées dans des jointures avec d'autres tables ?
- Quelles sont les colonnes utilisées dans mes clauses GROUP BY ou ORDER BY ?
- Quelles sont les colonnes utilisées par les fonctions MIN() et MAX() ?

Performances

Les indexes

Règle importante : **Trop d'indexes, tue l'index !** Il faut éviter d'en mettre partout, car ils sont stockées en mémoire et la mémoire est limitée.

Lorsqu'une requête est exécutée et que des indexes sont présents, MySQL va utiliser l'index comprenant le moins de résultat pour optimiser le traitement. Il est possible de forcer l'utilisation d'un index en particulier, mais cela n'a aucuns sens avec les moteurs tels qu'InnoDB ou MyISAM car leur gestion des index est optimale. Ce n'est par contre pas le cas pour le moteur MEMORY (MySQL Cluster).

Sommaire

Performances

Les caches

Les caches sont important dans MySQL/MariaDB et il faut apprendre à les utiliser. Voici comment obtenir l'état d'un cache et changer sa valeur (par exemple 'tmp_table_size') :

```
mysql> SHOW global VARIABLES LIKE 'tmp_table_size';
+-----------------+----------+
| Variable_name   | VALUE    |
+-----------------+----------+
| tmp_table_size | 16777216 |
+-----------------+----------+
1 ROW IN SET (0.00 sec)
```

Performances

Les caches

Pour changer sa valeur à chaud :

```
> SET global tmp_table_size=32*1024*1024;
```

Et de manière persistante :

/etc/mysql/my.cnf

```
tmp_table_size = 32M
```

Performances

Les caches

Le log binaire contient toutes les informations du log de modifications, dans un format plus efficace, et compatible avec les transactions. Comme le log de modifications, il contient toutes les requêtes qui modifient les données. Ainsi, une commande UPDATE ou DELETE avec une clause WHERE qui ne trouve aucune ligne ne sera pas écrite dans le log. Il contient aussi des informations sur le temps d'exécution de la requête dans la base. **Il ne contient que des commandes qui modifient des données**.

```
> SHOW global STATUS;
| Binlog_cache_disk_use                  | 0          |
| Binlog_cache_use                       | 16         |
```

- Le cache disk use doit rester à 0 pour être performant (pas d'écriture directe sur disque, on passe d'abord par le cache).
- Utilisation du cache Binlog_cache_use pour les logs binaires. C'est bien quand ce numéro n'est pas à 0.

Performances

Les caches

Les espaces temporaires :

```
SHOW global STATUS;
| Created_tmp_disk_tables                | 8          |
| Created_tmp_files                      | 5          |
| Created_tmp_tables                     | 54         |
```

- Created_tmp_disk_tables : devrait être à 0 le plus souvent possible (sauf si on a des champs de type blob ou texte, on ne pourra pas faire grand chose).
- Created_tmp_files : quand Created_tmp_disk_tables n'est pas suffisant et qu'il faut en plus aller créer d'autres fichiers sur le disque.
- Created_tmp_tables : nombre de fois que des tables temporaires ont été crées.

Performances

Les caches

Pour consulter la taille actuelle de ces tables temporaires :

```
> SHOW global VARIABLES LIKE 'tmp_table_size';
+-----------------+----------+
| Variable_name   | VALUE    |
+-----------------+----------+
| tmp_table_size | 16777216 |
+-----------------+----------+
1 ROW IN SET (0.00 sec)
```

Nous allons donc la doubler :

```
> SET global tmp_table_size=32*1024*1024;
```

N'oubliez pas de mettre ces paramètre permanent.

Performances

Les caches

Afin d'être cohérent avec les valeurs précédemment modifiées, nous allons mettre au moins à la même valeur le paramètre 'max_heap_table_size' :

```
mysql> SHOW global VARIABLES LIKE 'max_heap_table_size';
+---------------------+----------+
| Variable_name       | VALUE    |
+---------------------+----------+
| max_heap_table_size | 16777216 |
+---------------------+----------+
1 ROW IN SET (0.00 sec)
```

Sommaire

Performances

Les caches - Query Cache

Depuis la version 4.0.1, le MySQL server bénéficie d'un cache de requêtes. En fait, le cache sauvegarde le texte d'une requête SELECT avec le résultat qui a été envoyé au client. Si une requête identique est appelée par la suite, le serveur retournera le résultat à partir du cache plutôt que d'analyser puis exécuter la requête à nouveau.

Le cache de requêtes est extrêmement utile dans un environnement où les tables ne changent pas souvent, et que vous avez de nombreuses requêtes identiques. C'est la situation classique des serveurs Web, qui génèrent beaucoup de pages dynamiques à partir du même contenu.

Performances

Les caches - Query Cache

Pour afficher les valeurs des options pour le query cache :

```
mysql> show global status like 'Qc%';
+--------------------------+-----------+
| Variable_name            | Value     |
+--------------------------+-----------+
| Qcache_free_blocks       | 1         |
| Qcache_free_memory       | 134208784 |
| Qcache_hits              | 0         |
| Qcache_inserts           | 207       |
| Qcache_lowmem_prunes     | 0         |
| Qcache_not_cached        | 2825      |
| Qcache_queries_in_cache  | 0         |
| Qcache_total_blocks      | 1         |
+--------------------------+-----------+
8 rows in set (0.00 sec)
```

Performances

Les caches - Query Cache

- Qcache_free_memory : Mémoire libre pour ajouter des requêtes dans le query cache (enregistrement de la requête + résultat (130 Mo ici)).
- Qcache_hits : le nombre de fois que le cache a été atteint
- Qcache_inserts : il faut que le Qcache_hits soit supérieur au Qcache_inserts pour un bon fonctionnement du cache (ici vu qu'il n'y a pas beaucoup de trafic, ceci n'augmente pas).

Pour gagner en performances, on doit voir s'il est nécessaire de garder ou non le query cache. Donc si :

- Qcache_hits < Qcache_inserts : on désactive le cache.
- Qcache_not_cached < (Qcache_hits + Qcache_inserts) : On va essayer d'augmenter la taille du cache et la limite de la taille d'une requête en cache. Si Qcache_not_cached ne cesse d'augmenter, alors on désactive le cache.
- Qcache_lowmem_prunes : si plus de place dans le cache, alors les anciennes requêtes seront remplacées par des nouvelles. Augmentez le cache si vous en avez trop de Qcache_lowmem_prunes.

Performances

Les caches - Query Cache

Pour désactiver le Query cache :

```
SET global query_cache_type = 0
```

De façon permanente :

/etc/mysql/my.cnf

```
query_cache_type = 0
```

Performances

Les caches - Query Cache

Pour récupérer la taille actuelle du cache pour une requête et son résultat :

```
mysql> show global variables like 'query%';
+------------------------------+----------+
| Variable_name                | Value    |
+------------------------------+----------+
| query_alloc_block_size       | 8192     |
| query_cache_limit            | 1048576  |
| query_cache_min_res_unit     | 4096     |
| query_cache_size             | 16777216 |
| query_cache_type             | ON       |
| query_cache_wlock_invalidate | OFF      |
| query_prealloc_size          | 8192     |
+------------------------------+----------+
7 rows in set (0.00 sec)
```

Performances

Les caches - Query Cache

Ici, la taille maximum pour une requête du cache est de 1 Mo, on peut l'augmenter à 2Mo :

```
SET global query_cache_limit=2*1024*1024;
```

On change également la taille complète du cache. Exemple, on le passe à 32 Mo :

```
SET global query_cache_size = 32*1024*1024;
```

N'oubliez pas la persistance dans la configuration de MySQL.

Sommaire

Performances

Les caches - Table cache

Il est important de connaître les tables qui ne peuvent pas être cachées (qui provoquent l'ajout de fichiers sur le disque) :

```
> show global status like 'Open%';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Open_files    | 130   |
| Open_streams  | 0     |
| Open_tables   | 64    |
| Opened_tables | 14581 |
+---------------+-------+
4 rows in set (0.00 sec)
```

Ici on voit que que certaines tables sont actuellement ouvertes et beaucoup d'entre elles qui l'ont été.

Performances

Les caches - Table cache

Pour remédier à ce problème, il faut changer la taille du cache des tables qui est actuellement à 64 Mo :

```
mysql> show global variables like 'table%';
+---------------------------+---------+
| Variable_name             | Value   |
+---------------------------+---------+
| table_cache               | 64      |
| table_lock_wait_timeout   | 50      |
| table_type                | MyISAM  |
+---------------------------+---------+
3 rows in set (0.00 sec)
```

Performances

Les caches - Table cache

Et passer la valeur à 128 Mo. Par contre, **il ne faut surtout pas le faire à la volée** (effets de bords quasi garanti) et il n'est pas du tout recommandé de dépasser 4 Go.

Il modifier le fichier de configuration de MySQL/MariaDB :

/etc/mysql/my.cnf

```
table_cache = 128M
```

Sommaire

Performances

Les caches - Cache InnoDB

Pour vérifier l'état du moteur InnoDB, il existe une commande permettant de récupérer beaucoup d'informations, dont 2 particulièrement importantes pour les performances :

```
> SHOW engine innodb STATUS \G;
Buffer pool size        16383
Free buffers            0
```

Performances

Les caches - Cache InnoDB

"Free buffers" est ici à 0, InnoDB consume déjà tout l'espace qui lui est alloué. Il est conseillé d'augmenter la taille ce de buffer pour augmenter les performances. Pour l'espace mémoire dont il a besoin :

```
> SELECT
TABLE_SCHEMA,ENGINE,SUM(TABLE_ROWS),ENGINE,SUM(DATA_LENGTH),
SUM(INDEX_LENGTH)
FROM INFORMATION_SCHEMA.TABLES GROUP BY ENGINE,TABLE_SCHEMA
ORDER BY TABLE_SCHEMA;
```

Puis faites le calcul suivant pour avoir la taille en Ko :

```
(SUM(DATA_LENGTH) + SUM(INDEX_LENGTH)) / 1024
```

Si par exemple vous tombez sur 24568K (24Mo), passez 'innodb_buffer_pool_size' à 32Mo.

Performances

Les caches - Cache InnoDB

Voici d'autres bonnes options pour InnoDB :

/etc/mysql/my.cnf

```
innodb_buffer_pool_size          = 256M
innodb_log_file_size             = 256M
innodb_log_buffer_size           = 8M
thread_concurrency          = 64
innodb_thread_concurrency        = 64
innodb_read_io_threads           = 16
innodb_write_io_threads          = 16
innodb_flush_log_at_trx_commit      = 2
innodb_file_per_table            = 1
innodb_open_files           = 400
innodb_io_capacity          = 600
innodb_lock_wait_timeout         = 60
innodb_flush_method         = O_DIRECT
innodb_doublewrite          = 0
innodb_additional_mem_pool_size     = 20M
innodb_buffer_pool_restore_at_startup   = 500
innodb_use_native_aio            = 0
innodb_file_per_table
```

Pour conclure sur ces performances, voici un outil non maintenu, mais pratique : MySQLTuner `https ://github.com/major/MySQLTuner-perl`

Sommaire

Performances

Outils de mesure et diagnostique - slow query

Pour traquer les requêtes lentes, il faut activer les slow_query et mettre une valeur maximum avant que les requêtes soient loguées (ici 1 seconde). Pour vérifier si cette option est activé ou non :

```
> show global variables like '%log%';
+---------------------------------+-------------+
| Variable_name                   | Value       |
+---------------------------------+-------------+
...
| log_slow_queries                | OFF         |
...
+---------------------------------+-------------+
26 rows in set (0.00 sec)
```

Performances

Outils de mesure et diagnostique - slow query

A partir de la version 5.1 de MySQL, on peut les activer a chaud, voici la procédure à suivre :

```
SET global slow_query_log=1;
SET global long_query_time=1;
```

Si vous souhaitez activer de façon permanente les slow query :

/etc/mysql/my.cnf

```
slow_query_log=1
slow_query_log_file = /var/lib/mysql/mariadb-slow.log
long_query_time = 1
```

Note : activer les slow query peut légèrement impacter les performances

Performances

Outils de mesure et diagnostique - slow query

Les slow queries peuvent être analysées à tout moment dans le fichier désigné. Percona propose même un outil permettant de faire des statistiques sur ces requêtes lentes appelé pt-query-toolkit :

```
pt-query-digest /var/lib/mysql/mariadb-slow.log
```

```
# Rank Query ID           Response time  Calls  R/Call Apdx V/M   Item
# ==== ================== ============== ====== ====== ==== ===== ========
#    1 0x7678A4638EE87E49 356.6015 39.7%   3482 0.1024 1.00  0.17 SELECT companies company_aliases
#    2 0xE90B52050022B6AE  83.6626  9.3% 469260 0.0002 1.00  0.00 SELECT advertising advertising_positions
#    3 0xCA680D009DEB5855  60.7427  6.8%    477 0.1273 1.00  0.00 SELECT reports companies users
#    4 0x11F701C02F3A10F4  25.8345  2.9%    381 0.0678 1.00  0.27 SELECT reports companies users
#    5 0xF7C29DEB04DB7396  24.6843  2.8%    435 0.0567 1.00  0.26 SELECT reports companies users
#    6 0xF3BE8BA748E7FCB4  20.5375  2.3%   1240 0.0166 1.00  0.05 SELECT reports companies users
#    7 0xB6E8010076B61765  19.3797  2.2%   1440 0.0135 1.00  0.06 SELECT reports companies users
#    8 0x29FF14086FD8F9AB  18.1132  2.0%    199 0.0910 1.00  0.11 SELECT reports users reports companies
#    9 0x9BE0DAFAF30BF1CF  14.5519  1.6%    672 0.0217 1.00  0.04 SELECT reports users industries
#   10 0x732A4D1450F882B1  13.7154  1.5%   1027 0.0134 1.00  0.05 SELECT reports companies users
```

Vous pouvez également avoir une description plus détaillée des requêtes : http ://www.percona.com/doc/percona-toolkit/2.0/pt-query-digest.html

Performances

Outils de mesure et diagnostique - slow query

```
# Query 1: 2.60 QPS, 0.27x concurrency, ID 0x7678A4638EE87E49 at byte 178580675
# This item is included in the report because it matches --limit.
# Scores: Apdex = 1.00 [1.0], V/M = 0.17
# Query_time sparkline: | ^_  ^  |
# Time range: 2013-01-10 14:48:06 to 15:10:27
# Attribute    pct   total     min     max     avg     95%  stddev  median
# ============ === ======= ======= ======= ======= ======= ======= =======
# Count          0    3482
# Exec time     39    357s    11us   619ms   102ms   356ms   133ms    38us
# Lock time      0   225ms       0    64ms    64us   119us     1ms       0
# Rows sent      0   5.84k       0      20    1.72   17.65    4.63       0
# Rows examine  31  63.46M       0  93.39k  18.66k  54.03k  21.64k       0
# Query size     0   2.24M     649     785  673.96  685.39   16.28  652.75
# String:
# Databases    hdb
# Hosts
# Users        hdb_usr
# Query_time distribution
#   1us
#  10us  ################################################################
# 100us  #
#   1ms
#  10ms
# 100ms  ######################################################
#    1s
#  10s+
# Tables
#    SHOW TABLE STATUS FROM `hdb` LIKE 'companies'\G
#    SHOW CREATE TABLE `hdb`.`companies`\G
#    SHOW TABLE STATUS FROM `hdb` LIKE 'company_stats'\G
#    SHOW CREATE TABLE `hdb`.`company_stats`\G
#    SHOW TABLE STATUS FROM `hdb` LIKE 'company_aliases'\G
#    SHOW CREATE TABLE `hdb`.`company_aliases`\G
# EXPLAIN /*!50100 PARTITIONS*/
select c.*, (if(isnull(c.`id`),true,(coalesce(c.`created`,0)>'2012-11-29 00:00:00'
join `company_stats` cst on cst.`company`=c.`id`
where (exists(select * from `company_aliases` where `alias` like 'just lett%' and
'just lett%')
```

Performances

Outils de mesure et diagnostique - explain

Avec l'aide de EXPLAIN, vous pouvez identifier les index à ajouter pour accélérer les commandes SELECT.

Pour les jointures complexes, EXPLAIN retourne une ligne d'information pour chaque table utilisée dans la commande SELECT. Les tables sont listées dans l'ordre dans lequel elles seront lues. MySQL résout toutes les jointures avec une seule passe multi-jointure. Cela signifie que MySQL lit une ligne dans la première table, puis recherche les lignes qui correspondent dans la seconde, puis dans la troisième, etc.

Lorsque toutes les tables ont été traitées, MySQL affiche les colonnes demandées, et il remonte dans les tables jusqu'à la dernière qui avait encore des lignes à traiter. La prochaine ligne est alors traitée de la même façon.

`http ://dev.mysql.com/doc/refman/5.0/fr/explain.html`

Sommaire

Sommaire

Haute disponibilité

Réplication maître / esclaves

Depuis la version 3.23.15, MySQL supporte la réplication unidirectionnelle interne. Un serveur sert de maître, et les autres serveurs servent d'esclaves. Le serveur entretient des logs binaires de toutes les modifications qui surviennent. Il entretient aussi un fichier d'index des fichiers de logs binaires, pour garder la trace de la rotation des logs. Chaque esclave, après connexion réussie au serveur maître, indique au maître le point qu'il avait atteint depuis la fin de la dernière réplication, puis rattrape les dernières modifications qui ont eu lieu, puis se met en attente des prochains événements en provenance du maître.

Haute disponibilité

Réplication maître / esclaves

Afin d'éviter des problèmes de réplication, il est **fortement recommandé** de créer la première réplication avec des instances vierges.

Haute disponibilité

Réplication maître / esclaves

Afin d'éviter des problèmes de réplication, il est **fortement recommandé** de créer la première réplication avec des instances vierges.

Des réplications à travers le WAN sont possible et fonctionnent très bien. Cependant, il faut s'assurer que la bande passante est suffisante pour que la réplication ne soit pas trop en retard par rapport au serveur maître.

Haute disponibilité

Réplication maître / esclaves

Afin d'éviter des problèmes de réplication, il est **fortement recommandé** de créer la première réplication avec des instances vierges.

Des réplications à travers le WAN sont possible et fonctionnent très bien. Cependant, il faut s'assurer que la bande passante est suffisante pour que la réplication ne soit pas trop en retard par rapport au serveur maître.

Avoir plusieurs esclaves pour un seul maître, c'est possible. Cependant, **un esclave ne peut avoir plusieurs maîtres !**

Haute disponibilité

Réplication maître / esclaves

Voici la configuration du master :

/etc/mysql/my.cnf

```
[mysqld]
log-bin=/var/log/mysql/mysql-bin.log
binlog-do-db=<database name>
binlog-ignore-db=test
expire_logs_days=14
# binlog_cache_size = 64K
sync_binlog = 1
slave_compressed_protocol = 1
server-id=1
bind-address = 0.0.0.0
```

- binlog-do-db : nom de la base de donnée à répliquer (optionnel)
- binlog-ignore-db :nom de la base de donnée à ne pas répliquer (optionnel)
- expire_logs_days : délai d'expiration des binlogs avant rotation
- binlog_cache_size : activez ceci si binlog_cache_disk_use augmente
- sync_binlog : réduit les performances, mais est essentiel pour garantir l'intégrité des données
- slave_compressed_protocol : compression des données de réplication à travers le réseau
- server-id : numéro de serveur **unique**

Haute disponibilité

Réplication maître / esclaves

Sur le serveur maître, le server-id doit être positionné à 1 et sur le serveur esclave à 2 (incrémentez ce nombre pour chaque esclave). Redémarrez ensuite tous les services mysql :

```
service mysql restart
```

La prochaine étape consiste à créer un utilisateur de réplication **sur le master** afin que les esclaves puissent s'y connecter :

```
create user 'replication'@'@IP_esclave' identified by 'password';
grant replication slave on *.* to 'replication'@'@IP_esclave';
flush privileges;
```

Haute disponibilité

Réplication maître / esclaves

Il existe 2 méthodes de lancer une réplication :

- Méthode 1 : En créant un backup de la ou des bases de données à répliquer depuis le serveur maître. Puis les importer sur le serveur esclave pour enfin lancer la réplication. Le transfert de données à ce moment là sera minimum. **Ceci est la méthodes recommandée**, mais pas toujours applicable, surtout sur de très grosses bases.

Haute disponibilité

Réplication maître / esclaves

Il existe 2 méthodes de lancer une réplication :

- Méthode 1 : En créant un backup de la ou des bases de données à répliquer depuis le serveur maître. Puis les importer sur le serveur esclave pour enfin lancer la réplication. Le transfert de données à ce moment là sera minimum. **Ceci est la méthodes recommandée**, mais pas toujours applicable, surtout sur de très grosses bases.
- Méthode 2 : Cette méthode consiste à lancer la réplication depuis zéro. Toutes les données vont donc transiter via la réseau. Ceci peut prendre beaucoup de temps suivant la taille de la base et la bande passante réseau entre le maître et l'esclave.

Haute disponibilité

Réplication maître / esclaves

Nous allons voir la méthode 1, car c'est celle recommandée et la plus longue à mettre en place. La différence pour ces 2 méthodes est simplement la sauvegarde et l'import qui doit être fait pour cette première.

Il faut demander la fermeture des verrous présent sur les tables pour avoir une sauvegarde cohérente. Puis afficher la position de la dernière action dans le binlog, ainsi que son fichier contenant cette dernière position :

```
> flush tables with read lock;
> show master status;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000173 |    50937 | wordpress    |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

Note : si votre base de données est entièrement en InnoDB, il est inutile de flusher les tables.

Haute disponibilité

Réplication maître / esclaves

La sauvegarde de la base de donnée complète peut alors être exécutée :

```
mysqldump -uroot -p --opt --add-drop-table --routines --triggers \
--events --single-transaction --master-data=2 \
-B wordpress > wordpress.sql
```

Toutes ces options ne sont pas nécessaire (a voir en fonction de l'utilisation). Cependant, par mesure de sécurité, il est préférable de toutes les déclarer.

Transférez les données sur votre esclave pour préparer l'importation.

Haute disponibilité

Réplication maître / esclaves

Débloquez les verrous des tables :

```
unlock tables;
```

Ce sera tout pour le serveur maître.

Note : LOCK TABLES verrouille une table pour le thread courant. UNLOCK TABLES déverrouillera automatiquement tous les verrous posés par le thread courant. Toutes les tables verrouillées par le thread courant sont automatiquement déverrouillées quand ce thread utilise à nouveau LOCK TABLES, ou quand la connexion au serveur est perdue.

Haute disponibilité

Réplication maître / esclaves

Nous pouvons passer à la configuration de l'esclave. La configuration de l'esclave est identique au maître comme cité plus haut à l'exception de l'identifiant unique (server-id).

Avant de passer à l'activation de l'esclave, il va falloir importer le dump :

```
mysql -uroot -p base < dump.sql
```

Note : ne pas oublier d'importer la base MySQL si on souhaite que les users soient dumpés également.

Haute disponibilité

Réplication maître / esclaves

Puis pour activer un serveur esclave, voici la marche à suivre :

```
stop slave;
reset slave;
change master to master_host='@IP_maitre',
master_user='replication', master_password='password',
master_log_file='mysql-bin.000173',
master_log_pos=50937;
start slave;
```

- stop slave : permet d'arrêter la synchronisation d'un serveur esclave
- reset slave : permet de réinitialiser l'état dans lequel se trouve l'esclave. Ceci est utile dans le cas ou il s'est arréter sur un echec.
- change master : il faut spécifier certains éléments tel que la position des binlogs et le fichier de binlogs du master au moment de la sauvegarde.
- start slave : démarrage de la synchronisation entre le maître et l'esclave

Haute disponibilité

Réplication maître / esclaves

Il est possible à tout moment de vérifier l'état d'un serveur esclave :

```
> show slave status\G;
 *************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
               Slave_IO_Running: Yes
               Slave_SQL_Running: Yes
               Last_Error:
               Seconds_Behind_Master: 0
```

- Slave_IO_State : l'état actuelle de la réplication
- Slave_IO_Running et Slave_SQL_Running : ces 2 éléments doivent être à yes pour confirmer que la réplication est opérationnelle
- Last_Error/Last_IO_Error : la dernière erreur sur laquelle la réplication s'est arrêtée
- Seconds_Behind_Master : le nombre de secondes de retard entre le maître et l'esclave

Sommaire

Haute disponibilité

Réparer une réplication

Réparer une réplication en soit n'est pas très compliqué. Ses impacts peuvent l'être bien plus si ce n'est pas fait correctement.

La manière la plus rapide de détecter si une synchronisation est arrêtée est de regarder les valeurs de Slave_IO_Running et Slave_SQL_Running. Si l'une des 2 n'est pas à 'Yes', il faut regarder la dernière erreur (Last_Error/Last_IO_Error).

Celle ci est montrée sous sa forme de requête, ainsi que des informations supplémentaires pour l'aide au debug :

```
                  Last_Error: Error 'Table 'mydb.taggregate_temp_
                              1212047760' doesn't exist' on query.
                              Default database: 'mydb'.
Query: 'UPDATE thread AS thread,taggregate_temp_1212047760
        AS aggregate
        SET thread.views = thread.views + aggregate.views
        WHERE thread.threadid = aggregate.threadid'
```

Haute disponibilité

Réparer une réplication

A ce moment là, arrêtez la réplication de l'esclave :

```
stop slave;
```

Réparez le problème sur votre base de données. Si la requête en question n'impactera pas l'intégrité de vos données, vous pouvez la sauter :

```
set global sql_slave_skip_counter=1;
```

Puis démarrez l'esclave pour qu'il reprenne la synchronisation après cette erreur :

```
start slave;
```

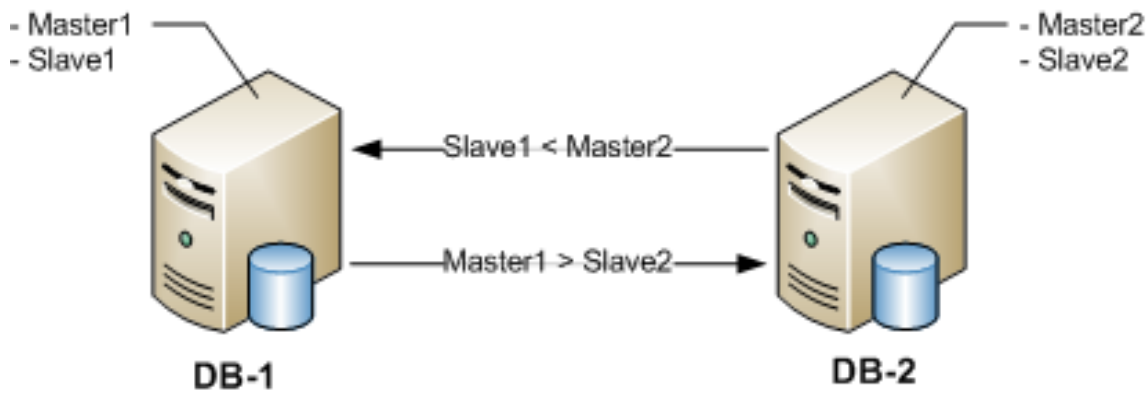
Si tout se passe bien, la réplication rattrapera son retard et tous les indicateurs vu précédemment seront bon.

Sommaire

Haute disponibilité

Maître / Maître

Une synchronisation maître <-> maître n'est en fait qu'une synchronisation maitre -> esclave dans les 2 sens. Si vous avez une synchronisation déjà en place, répétez la manipulation dans le sens inverse (sans faire de dump évidement).



Haute disponibilité

Maître / Maître

Attention à ne pas tomber dans le piège du multi maîtres ! Ce n'est pas parce que vous avez 2 maîtres que vous pouvez effectuer des écritures en simultanés dessus !

Techniquement vous pouvez ! Mais si vous le faite, vous risquez de tomber dans des situations ou les 2 maîtres veulent écrire en même temps sur la même table. Pour peu que vous ayez des clés primaires sur cette table, non seulement vous casserez la réplication, mais vos données seront corrompus. Le travail de récupération dans ce genre de situation est généralement fastidieux.

Haute disponibilité

Maître / Maître

L'avantage du maître <-> maître, est d'avoir une réplication bidirectionnelle. Ce qui permet, dans le cas ou l'un des 2 noeuds viendrait à tomber, d'avoir une reprise de synchronisation automatique, là ou il l'avait laissé avant de tomber.

Dans ce genre de configuration, il est conseillé d'utiliser un load balancer (HA-Proxy/MySQL Proxy) ou un cluster (Heartbeat/Pacemaker) sur lequel on va mettre une IP virtuelle.

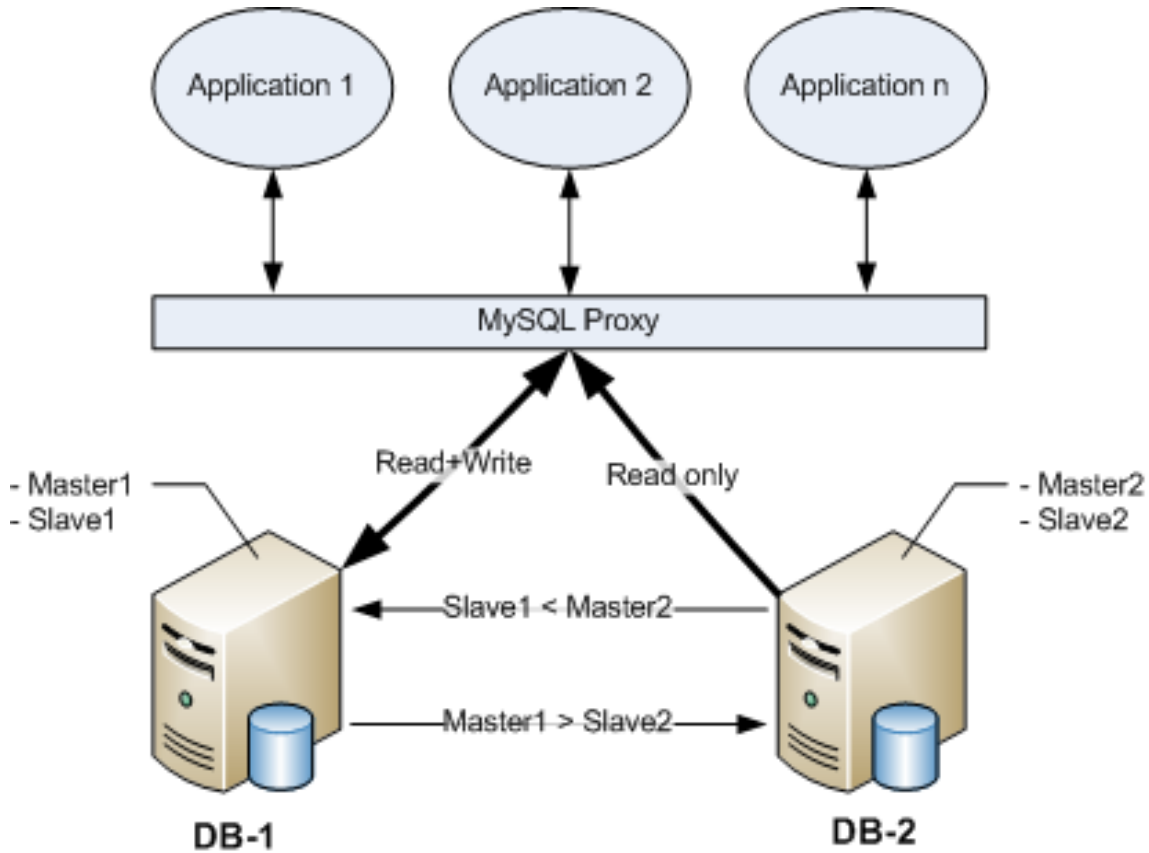
Un seul à la fois seulement doit être utilisé pour l'écriture !

Sommaire

Haute disponibilité

Répartition des données

Pour séparer la lecture des écritures, on peut utilise MySQL Proxy :



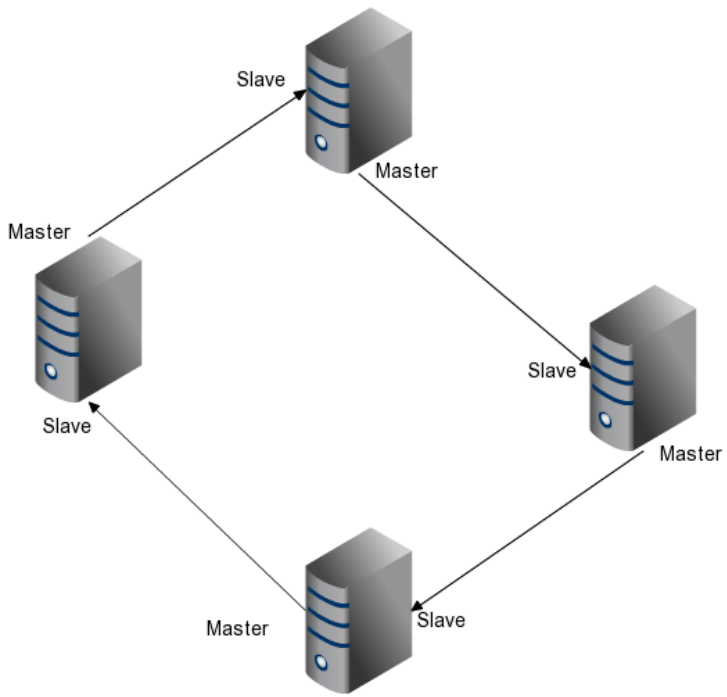
On peut alors multiplier le nombre d'esclaves pour augmenter la capacité de l'infrastructure à absorber les demandes.

Sommaire

Haute disponibilité

Multi maîtres

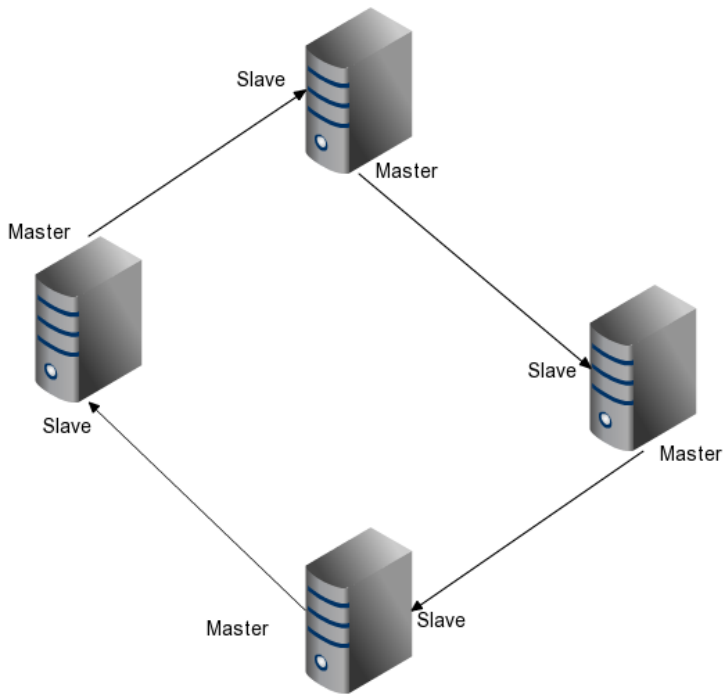
Lorsque l'on a sait utiliser la réplication maître <-> maître, il est tentant d'en rajouter encore plus pour avoir ce type d'infrastructure :



Haute disponibilité

Multi maîtres

Lorsque l'on a sait utiliser la réplication maître <-> maître, il est tentant d'en rajouter encore plus pour avoir ce type d'infrastructure :



Hors, ceci est l'exemple de ce qui ne faut surtout pas faire ! C'est beaucoup trop complexe à maintenir et corriger en cas d'erreurs. Les erreurs humaines arrivent trop vite dans ce type de configuration.

Haute disponibilité

Multi maîtres

Lorsque l'on souhaite avoir plus d'un noeud en écriture simultanées, il existe aujourd'hui 2 solutions :

Haute disponibilité

Multi maîtres

Lorsque l'on souhaite avoir plus d'un noeud en écriture simultanées, il existe aujourd'hui 2 solutions :

- MySQL Cluster : largement utilisé aujourd'hui mais nécessite des personnes à plein temps pour maintenir ce type d'infrastructure complexe.

Haute disponibilité

Multi maîtres

Lorsque l'on souhaite avoir plus d'un noeud en écriture simultanées, il existe aujourd'hui 2 solutions :

- MySQL Cluster : largement utilisé aujourd'hui mais nécessite des personnes à plein temps pour maintenir ce type d'infrastructure complexe.
- Galera Cluster : c'est le cluster officiel de MariaDB et celui ci est également disponible sous forme de patchs pour MySQL. C'est la solution la plus simple à administrer offre des performances très bonne lorsque l'on multiplie les noeuds.

Haute disponibilité

Multi maîtres

Lorsque l'on souhaite avoir plus d'un noeud en écriture simultanées, il existe aujourd'hui 2 solutions :

- MySQL Cluster : largement utilisé aujourd'hui mais nécessite des personnes à plein temps pour maintenir ce type d'infrastructure complexe.
- Galera Cluster : c'est le cluster officiel de MariaDB et celui ci est également disponible sous forme de patchs pour MySQL. C'est la solution la plus simple à administrer offre des performances très bonne lorsque l'on multiplie les noeuds.

Note : Les solutions tel que MMM ou MHA qui savent faire du multi maître sous MySQL/MariaDB. Mais ce ne sont que des scripts qui abusent de l'utilisation du mécanisme de synchronisation déjà présent dans MySQL/MariaDB. Il est donc conseillé d'utiliser une solution dédiée pour faire cela.

Sommaire

Haute disponibilité

Multi maîtres - Galera

Galera est vraiment différent de MySQL Cluster qui sait scaler les écritures. Ici c'est multi threadé uniquement. Et contrairement à MHA qui est une solution asynchrone, Galera est synchrone. Galera fonctionne **uniquement** pour le moteur InnoDB et permet :

Haute disponibilité

Multi maîtres - Galera

Galera est vraiment différent de MySQL Cluster qui sait scaler les écritures. Ici c'est multi threadé uniquement. Et contrairement à MHA qui est une solution asynchrone, Galera est synchrone. Galera fonctionne **uniquement** pour le moteur InnoDB et permet :

- Les réplications synchrones

Haute disponibilité

Multi maîtres - Galera

Galera est vraiment différent de MySQL Cluster qui sait scaler les écritures. Ici c'est multi threadé uniquement. Et contrairement à MHA qui est une solution asynchrone, Galera est synchrone. Galera fonctionne **uniquement** pour le moteur InnoDB et permet :

- Les réplications synchrones
- Des réplications multi maîtres actifs

Haute disponibilité

Multi maîtres - Galera

Galera est vraiment différent de MySQL Cluster qui sait scaler les écritures. Ici c'est multi threadé uniquement. Et contrairement à MHA qui est une solution asynchrone, Galera est synchrone. Galera fonctionne **uniquement** pour le moteur InnoDB et permet :

- Les réplications synchrones
- Des réplications multi maîtres actifs
- Lecture/Écriture sur plusieurs nœuds simultané

Haute disponibilité

Multi maîtres - Galera

Galera est vraiment différent de MySQL Cluster qui sait scaler les écritures. Ici c'est multi threadé uniquement. Et contrairement à MHA qui est une solution asynchrone, Galera est synchrone. Galera fonctionne **uniquement** pour le moteur InnoDB et permet :

- Les réplications synchrones
- Des réplications multi maîtres actifs
- Lecture/Écriture sur plusieurs nœuds simultané
- Détection automatique lorsqu'un nœud tombe

Haute disponibilité

Multi maîtres - Galera

Galera est vraiment différent de MySQL Cluster qui sait scaler les écritures. Ici c'est multi threadé uniquement. Et contrairement à MHA qui est une solution asynchrone, Galera est synchrone. Galera fonctionne **uniquement** pour le moteur InnoDB et permet :

- Les réplications synchrones
- Des réplications multi maîtres actifs
- Lecture/Écriture sur plusieurs nœuds simultané
- Détection automatique lorsqu'un nœud tombe
- Réintégration d'un nœud automatiquement

Haute disponibilité

Multi maîtres - Galera

Galera est vraiment différent de MySQL Cluster qui sait scaler les écritures. Ici c'est multi threadé uniquement. Et contrairement à MHA qui est une solution asynchrone, Galera est synchrone. Galera fonctionne **uniquement** pour le moteur InnoDB et permet :

- Les réplications synchrones
- Des réplications multi maîtres actifs
- Lecture/Écriture sur plusieurs nœuds simultané
- Détection automatique lorsqu'un nœud tombe
- Réintégration d'un nœud automatiquement
- Pas de lag au niveau des slaves

Haute disponibilité

Multi maîtres - Galera

Galera est vraiment différent de MySQL Cluster qui sait scaler les écritures. Ici c'est multi threadé uniquement. Et contrairement à MHA qui est une solution asynchrone, Galera est synchrone. Galera fonctionne **uniquement** pour le moteur InnoDB et permet :

- Les réplications synchrones
- Des réplications multi maîtres actifs
- Lecture/Écriture sur plusieurs nœuds simultané
- Détection automatique lorsqu'un nœud tombe
- Réintégration d'un nœud automatiquement
- Pas de lag au niveau des slaves
- Aucunes transactions perdues

Haute disponibilité

Multi maîtres - Galera

Galera est vraiment différent de MySQL Cluster qui sait scaler les écritures. Ici c'est multi threadé uniquement. Et contrairement à MHA qui est une solution asynchrone, Galera est synchrone. Galera fonctionne **uniquement** pour le moteur InnoDB et permet :

- Les réplications synchrones
- Des réplications multi maîtres actifs
- Lecture/Écriture sur plusieurs nœuds simultané
- Détection automatique lorsqu'un nœud tombe
- Réintégration d'un nœud automatiquement
- Pas de lag au niveau des slaves
- Aucunes transactions perdues
- Latences clientes plus faible

Haute disponibilité

Multi maîtres - Galera

Il y a cependant quelques limitations :

- Supporte uniquement InnoDB
- DELETE ne fonctionne que sur les tables munies de clefs primaires
- LOCK/UNLOCK/GET_LOCK/RELEASE_LOCK ne fonctionne pas en multi maitre
- Les query logs ne peuvent être envoyés sur des tables, mais uniquement sur des fichiers
- Les transactions XA ne sont pas supportées
- 3 noeuds minimum

Haute disponibilité

Multi maîtres - Galera

Il y a cependant quelques limitations :

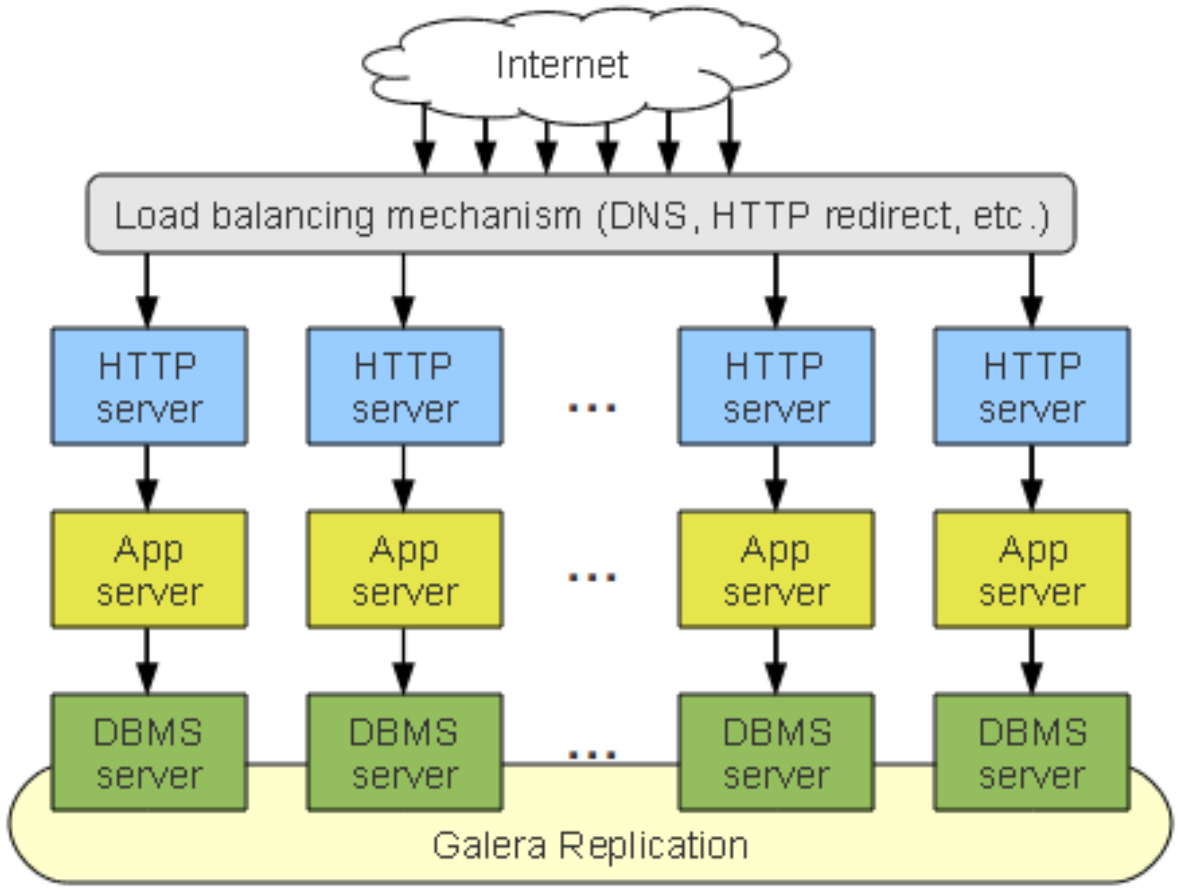
- Supporte uniquement InnoDB
- DELETE ne fonctionne que sur les tables munies de clefs primaires
- LOCK/UNLOCK/GET_LOCK/RELEASE_LOCK ne fonctionne pas en multi maitre
- Les query logs ne peuvent être envoyés sur des tables, mais uniquement sur des fichiers
- Les transactions XA ne sont pas supportées
- 3 noeuds minimum (2 possible en utilisant Garbd)

`http://www.codership.com/wiki/doku.php?id=limitations`

Haute disponibilité

Multi maîtres - Galera

Voici le type d'architecture possible avec Galera Cluster :



Haute disponibilité

Multi maîtres - Galera

Pour installer le cluster Galera, il vous faut le dépôt de MariaDB configuré (comme décrit au début des slides), puis installer comme ceci le cluster :

```
aptitude update
aptitude install mariadb-galera-server galera rsync
```

Puis ajoutez/modifiez ces lignes dans le fichier de configuration de Galera (uniquement sur les esclaves pour le moment) :

/etc/mysql/conf.d/mariadb.cnf

```
wsrep_cluster_name='mariadb_cluster'
wsrep_node_name=node2
wsrep_node_address="10.0.0.2"
wsrep_cluster_address = 'gcomm://10.0.0.1,10.0.0.2,10.0.0.3,10.0.0.4'
wsrep_provider = /usr/lib/galera/libgalera_smm.so
wsrep_provider_options=""
wsrep_retry_autocommit = 0
wsrep_sst_method = rsync
```

L'adressage de mon réseau ici est 10.0.0.X/24.

Haute disponibilité

Multi maîtres - Galera

- wsrep_cluster_name : le nom du cluster Galera. A utiliser, surtout si vous disposez de plusieurs cluster Galera dans le même subnet, afin d'éviter que certains noeuds entrent dans le mauvais cluster.
- wsrep_node_name : le nom de la machine sur laquelle se trouve ce fichier de configuration. Vous l'aurez compris, **il faut a tout prix éviter les doublons (surtout pour le debug)**
- wsrep_node_address : adresse ip du noeud actuel **(même avertissement que la ligne précédente)**
- wsrep_cluster_address : liste des membres du cluster pouvant être maître (séparé par des virgules).
- wsrep_provider_options : permet d'activer des options supplémentaires.
- wsrep_retry_autocommit : permet de définir le nombre de tentatives une requête doit être réexécutée en cas de conflit.
- wsrep_sst_method : la méthode d'échange des données. Rsync est la plus rapide à l'heure actuelle

Haute disponibilité

Multi maîtres - Galera

Pour la configuration du maître, elle est quasiment identique si ce n'est le paramètre 'wsrep_cluster_address' qui doit être vide :

/etc/mysql/conf.d/mariadb-master.cnf

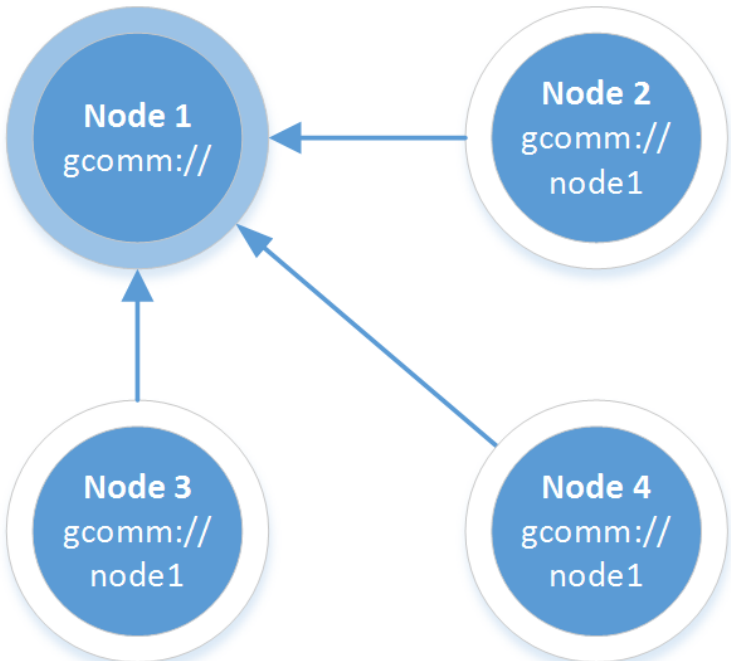
```
wsrep_cluster_name='mariadb_cluster'
wsrep_node_name=node1
wsrep_node_address="10.0.0.1"
wsrep_cluster_address = 'gcomm://'
wsrep_provider = /usr/lib/galera/libgalera_smm.so
wsrep_provider_options=""
wsrep_retry_autocommit = 0
wsrep_sst_method = rsync
```

Il est important qu'une seule machine ai la configuration 'gcomm ://', car c'est l'initialisation du cluster !

Haute disponibilité

Multi maîtres - Galera

Pour bien comprendre le principe, lorsque nous allons démarrer nos instances MariaDB, nous allons nous retrouver dans ce cas de configuration (je n'ai volontairement pas fait toutes les flèches de communication pour éviter que ça devienne trop fouillis, mais tous les nodes discutent entre eux) :



Haute disponibilité

Multi maîtres - Galera

- Le node 1 initialise le cluster avec la valeur du gcomm vide.
- Les autres nodes se connectent sur le node 1 et échangent leurs données pour avoir le même niveau de données partout

Avant de démarrer les services, il va falloir créer pour la première fois le cluster (sur le noeud 1) :

```
mysqld --wsrep_cluster_address=gcomm://
```

Haute disponibilité

Multi maîtres - Galera

Sur le noeud maître lancez cette commande pour voir l'état du cluster :

```
MariaDB [(none)]> SHOW STATUS LIKE 'wsrep_%';
+----------------------------+-----------------------+
| Variable_name              | Value                 |
+----------------------------+-----------------------+
| wsrep_cluster_size         | 0                     |
| wsrep_cluster_status       | Disconnected          |
| wsrep_ready                | ON                    |
+----------------------------+-----------------------+
```

- wsrep_cluster_size : correspond au nombre de noeuds dans le cluster
- wsrep_ready : indique si le cluster est fonctionnel

Haute disponibilité

Multi maîtres - Galera

Puis lorsque les noeuds esclave démarrent :

```
service mysql start
```

On peut constater que ce nombre augmente :

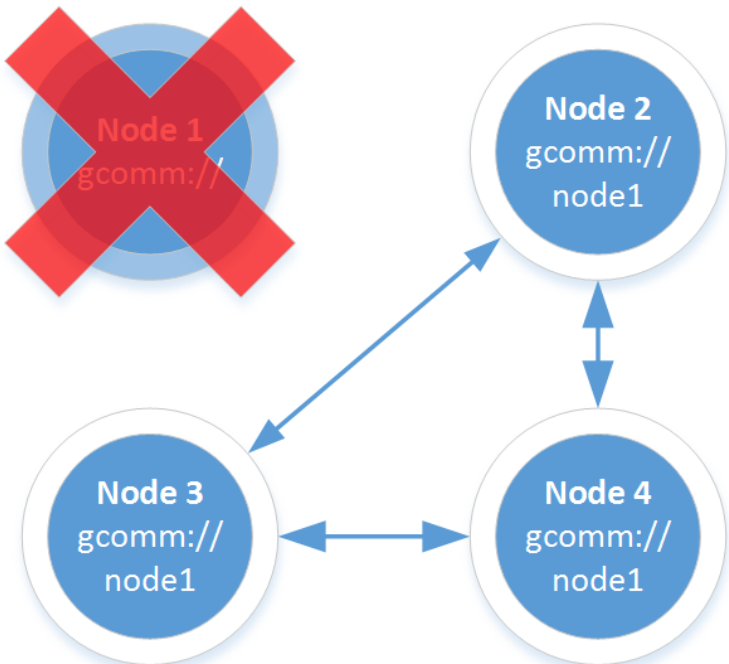
```
> mysql -uroot -p -e "SHOW STATUS LIKE 'wsrep_%';"
+------------------------------+------------+
| Variable_name                | Value      |
+------------------------------+------------+
| wsrep_cluster_size           | 4          |
| wsrep_cluster_status         | Primary    |
| wsrep_ready                  | ON         |
+------------------------------+------------+
```

Sommaire

Haute disponibilité

Multi maîtres - Galera (Récupération et maintenance)

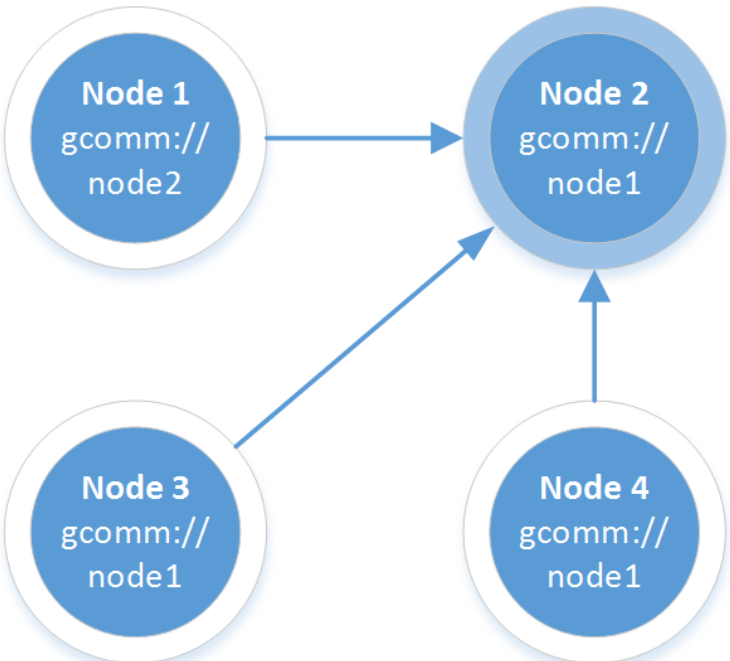
Il n'y a pas à se soucier de la réplication si un nœud tombe (autre que le master (node1)). Une fois réparé et allumé, il se reconnectera automatiquement sur le node 1 et rattrapera son retard. Par contre, en cas de problème sur le noeud 1 :



Haute disponibilité

Multi maîtres - Galera (Récupération et maintenance)

Les autres nodes continueront à communiquer entre eux et attendront le maître revienne. Une fois le maître rallumé, il faut lui indiquer un autre node sur lequel il devra se connecter pour continuer la synchronisation :



Haute disponibilité

Multi maîtres - Galera (Récupération et maintenance)

Que ce soit pour forcer une reconnexion ou pour effectuer une maintenance sur le nœud master, il est conseillé de rediriger les autres serveurs vers un autre master pour éviter les coupures :

```
SET GLOBAL wsrep_cluster_address='gcomm://10.0.0.2';
```

Vous pouvez ensuite vérifier le noeud maître sur vos instances MariaDB :

```
MariaDB [(none)]> SHOW VARIABLES LIKE 'wsrep_cluster_address';
+-------------------------+-------------------------+
| Variable_name           | Value                   |
+-------------------------+-------------------------+
| wsrep_cluster_address   | gcomm://10.0.0.2        |
+-------------------------+-------------------------+
```

Haute disponibilité

Multi maîtres - Galera (Récupération et maintenance)

Il peut arriver que lors de l'extinction du service MariaDB, les fichiers de locks se libèrent mal et que le service rsync tourne encore. Pour réparer (sans éviter de redémarrer complètement la machine), voici les opérations à suivre :

1. On s'assure que MariaDB ne fonctionne plus :

```
service mysql stop
ps aux | grep mysql
```

2. Si il tourne encore, on kill le processus !
3. On vérifie que le process rsync ne tourne plus non plus et on le kill si c'est le cas.

Haute disponibilité

Multi maîtres - Galera

4. On supprime le fichier de lock :

```
rm -f /var/run/mysqld/mysqld.sock
```

5. On vérifie que le dossier pour stocker le pid existe, sinon on le créé :

```
if [ ! -d /var/run/mysqld ] ; then
mkdir /var/run/mysqld
chown mysql. /var/run/mysqld
fi
```

6. Vous pouvez maintenant démarrer le service, ça devrait fonctionner :

```
service mysql start
```

Sinon regardez les logs (/var/log/syslog)

Sommaire

Annexes

Sources

Ces slides reprennent des parties provenant de certains sites dont le site officiel de MySQL. Voici les liens utilisés :

http ://dev.mysql.com/doc/refman/5.0/fr/internal-locking.html ?ff=nopfpls
https ://wiki.deimos.fr/MysqlTuner_ :_Optimiser_votre_serveur_MySQL
http ://blog.alexis-ragot.com/optimisation-mysql-type-donnees
http ://blog.alexis-ragot.com/optimisation-index-mysql
http ://www.percona.com/software/percona-server/feature-comparison

Merci !