Reprise sur incident

Récupération sur incident
Contrairement à la réplication MySQL standard, un cluster Galera agit comme une entité logique, qui contrôle le statut et la cohérence de chaque nœud ainsi que le statut de l'ensemble du cluster. Cela permet de maintenir l'intégrité des données plus efficacement qu'avec la réplication asynchrone traditionnelle sans perdre les écritures sécurisées sur plusieurs nœuds en même temps.

Cependant, il existe des scénarios où le service de base de données peut s'arrêter sans qu'aucun nœud ne puisse répondre aux demandes.

Scénario : le nœud A est correctement arrêté

Dans un cluster à trois nœuds (nœud A, nœud B, nœud C), un nœud (nœud A, par exemple) est gracieusement arrêté : à des fins de maintenance, de changement de configuration, etc.

Dans ce cas, les autres nœuds reçoivent un message « au revoir » du nœud arrêté et la taille du cluster est réduite ; certaines propriétés comme le calcul du quorum ou l'incrémentation automatique sont automatiquement modifiées. Dès que le nœud A est redémarré, il rejoint le cluster en fonction de sa wsrep_cluster_address dans my.cnf.

Si le cache d'écriture (gcache.size) sur les nœuds B et/ou C a encore toutes les transactions exécutées pendant que le nœud A était en panne, la jointure est possible via IST . Si IST est impossible en raison de transactions manquantes dans le gcache du donneur, la décision de secours est prise par le donneur et SST est démarré automatiquement.

Scénario : deux nœuds sont correctement arrêtés

Similaire au scénario : le nœud A est arrêté en douceur , la taille du cluster est réduite à 1 — même le seul nœud restant C forme le composant principal et est capable de répondre aux demandes des clients. Pour remettre les nœuds dans le cluster, il vous suffit de les démarrer.

Cependant, lorsqu'un nouveau nœud rejoint le cluster, le nœud C passera à l'état « Donateur/Désynchronisé » car il doit fournir le transfert d'état au moins au premier nœud qui se joint. Il est toujours possible d'y lire/écrire pendant ce processus, mais cela peut être beaucoup plus lent, ce qui dépend de la quantité de données à envoyer pendant le transfert d'état. En outre, certains équilibreurs de charge peuvent considérer le nœud donneur comme non opérationnel et le supprimer du pool. Il est donc préférable d'éviter la situation où un seul nœud est actif.

Si vous redémarrez le nœud A puis le nœud B, vous voudrez peut-être vous assurer que la note B n'utilise pas le nœud A comme donneur de transfert d'état : le nœud A peut ne pas avoir tous les jeux d'écriture nécessaires dans son gcache. Spécifiez node C node comme donneur dans votre fichier de configuration et démarrez le service mysql :

$ systemctl démarrer mysql
Voir également

Documentation Galera : option wsrep_sst_donor
https://galeracluster.com/library/documentation/mysql-wsrep-options.html#wsrep-sst-donor
Scénario : Les trois nœuds sont correctement arrêtés
Le cluster est complètement arrêté et le problème est de l'initialiser à nouveau. Il est important qu'un nœud PXC écrive sa dernière position exécutée dans le grastate.datfichier.

En comparant le numéro de seqno dans ce fichier, vous pouvez voir quel est le nœud le plus avancé (probablement le dernier arrêté). Le cluster doit être amorcé à l'aide de ce nœud, sinon les nœuds qui avaient une position plus avancée devront effectuer le SST complet pour rejoindre le cluster initialisé à partir du moins avancé. En conséquence, certaines transactions seront perdues). Pour amorcer le premier nœud, appelez le script de démarrage comme ceci :

$ systemctl démarrer mysql@bootstrap.service
Noter

Même si vous démarrez à partir du nœud le plus avancé, les autres nœuds ont un numéro de séquence inférieur. Ils devront toujours se joindre via le SST complet car le cache Galera n'est pas conservé au redémarrage.

Pour cette raison, il est recommandé d'arrêter les écritures sur le cluster avant son arrêt complet, afin que tous les nœuds puissent s'arrêter à la même position. Voir aussi pc.recovery.

Scénario : Un nœud disparaît du cluster
C'est le cas lorsqu'un nœud devient indisponible en raison d'une panne de courant, d'une panne matérielle, d'une panique du noyau, d'un crash mysqld, sur mysqld pid, etc.kill -9

Deux nœuds restants remarquent que la connexion au nœud A est interrompue et commencent à essayer de s'y reconnecter. Après plusieurs délais d'expiration, le nœud A est supprimé du cluster. Le quorum est enregistré (2 nœuds sur 3 sont actifs), donc aucune interruption de service ne se produit. Après son redémarrage, le nœud A se joint automatiquement (comme décrit dans Scénario : le nœud A est normalement arrêté).

Scénario : Deux nœuds disparaissent du cluster
Deux nœuds ne sont pas disponibles et le nœud restant (nœud C) n'est pas en mesure de former seul le quorum. Le cluster doit basculer vers un mode non principal, où MySQL refuse de servir les requêtes SQL. Dans cet état, le mysqldprocessus sur le nœud C est toujours en cours d'exécution et peut être connecté, mais toute instruction liée aux données échoue avec une erreur

mysql > sélectionnez * à partir de test . sbtest1 ;
ERREUR 1047 (08 S01): WSREP n'a pas encore préparé le nœud pour l' utilisation de l' application
Les lectures sont possibles jusqu'à ce que le nœud C décide qu'il ne peut pas accéder aux nœuds A et B. Les nouvelles écritures sont interdites.

Dès que les autres nœuds deviennent disponibles, le cluster se reforme automatiquement. Si le nœud B et le nœud C étaient simplement séparés par le réseau du nœud A, mais qu'ils peuvent toujours se joindre, ils continueront à fonctionner car ils forment toujours le quorum.

Si les nœuds A et B tombent en panne, vous devez activer manuellement le composant principal sur le nœud C avant de pouvoir afficher les nœuds A et B. La commande pour ce faire est :

mysql > SET GLOBAL wsrep_provider_options = 'pc.bootstrap=true' ;
Cette approche ne fonctionne que si les autres nœuds sont en panne avant de le faire ! Sinon, vous vous retrouvez avec deux clusters ayant des données différentes.

Références croisées

Ajout de nœuds au cluster

Scénario : tous les nœuds sont tombés en panne sans procédure d'arrêt appropriée
Ce scénario est possible en cas de panne de courant du datacenter ou en cas de bug MySQL ou Galera. En outre, cela peut se produire en raison de la cohérence des données compromise lorsque le cluster détecte que chaque nœud a des données différentes. Le grastate.datfichier n'est pas mis à jour et ne contient pas de numéro de séquence valide (seqno). Cela peut ressembler à ceci :

$ cat /var/lib/mysql/grastate.dat
# État enregistré de GALERA
version : 2.1
uuid : 220dcdcb-1629-11e4-add3-aec059ad3734
numéro de séquence : -1
safe_to_bootstrap : 0
Dans ce cas, vous ne pouvez pas être sûr que tous les nœuds sont cohérents les uns avec les autres. Nous ne pouvons pas utiliser la variable safe_to_bootstrap pour déterminer le nœud qui a la dernière transaction validée car elle est définie sur 0 pour chaque nœud. Une tentative d'amorçage à partir d'un tel nœud échouera à moins que vous ne commenciez mysqldavec le --wsrep-recoverparamètre :

$ mysqld --wsrep-recover
Recherchez dans la sortie la ligne qui signale la position récupérée après l'UUID du nœud (1122 dans ce cas) :

...
... [Remarque] WSREP : Position récupérée : 220dcdcb-1629-11e4-add3-aec059ad3734:1122
...
Le nœud où la position récupérée est marquée par le plus grand nombre est le meilleur candidat bootstrap. Dans son grastate.datfichier, définissez la variable safe_to_bootstrap sur 1 . Ensuite,

amorcez à partir de ce nœud.

Noter

Après un arrêt, vous pouvez boosterrap à partir du nœud qui est marqué comme sûr dans le grastate.datfichier.

...
safe_to_bootstrap : 1
...
Voir également

Documentation Galera
Présentation de la fonctionnalité "Safe-To-Bootstrap" dans Galera Cluster
Dans les versions récentes de Galera, l'option pc.recovery(activée par défaut) enregistre l'état du cluster dans un fichier nommé gvwstate.datsur chaque nœud membre. Comme le nom de cette option l'indique (pc - composant principal), elle n'enregistre qu'un cluster étant dans l'état PRIMAIRE. Un exemple de contenu de : fichier peut ressembler à ceci :

cat /var/lib/mysql/gvwstate.dat
my_uuid : 76de8ad9-2aac-11e4-8089-d27fd06893b9
#vwbeg
view_id : 3 6c821ecc-2aac-11e4-85a5-56fe513c651f 3
bootstrap : 0
membre : 6c821ecc-2aac-11e4-85a5-56fe513c651f 0
membre : 6d80ec1b-2aac-11e4-8d1e-b2b2f6a5018ad 0
membre : 8089-d27fd06893b9 0
#vwend
Nous pouvons voir un cluster à trois nœuds avec tous les membres actifs. Grâce à cette nouvelle fonctionnalité, les nœuds tenteront de restaurer le composant principal une fois que tous les membres commenceront à se voir. Cela permet au cluster PXC de récupérer automatiquement après une mise hors tension sans aucune intervention manuelle ! Dans les logs nous verrons :

Scénario : Le cluster perd son état principal en raison d'un cerveau divisé
Pour les besoins de cet exemple, supposons que nous ayons un cluster composé d'un nombre pair de nœuds : six, par exemple. Trois d'entre eux se trouvent à un endroit tandis que les trois autres se trouvent à un autre endroit et ils perdent la connectivité réseau. Il est recommandé d'éviter une telle topologie : si vous ne pouvez pas avoir un nombre impair de nœuds réels, vous pouvez utiliser un nœud arbitre supplémentaire (garbd) ou définir un poids pc plus élevé pour certains nœuds. Mais lorsque le split brain se produit de quelque manière que ce soit, aucun des groupes séparés ne peut maintenir le quorum : tous les nœuds doivent cesser de répondre aux demandes et les deux parties du cluster essaieront en permanence de se reconnecter.

Si vous souhaitez restaurer le service avant même que le lien réseau ne soit restauré, vous pouvez à nouveau rendre l'un des groupes principal à l'aide de la même commande que celle décrite dans Scénario : deux nœuds disparaissent du cluster

SET GLOBAL wsrep_provider_options = 'pc.bootstrap=true' ;
Après cela, vous êtes en mesure de travailler sur la partie restaurée manuellement du cluster, et l'autre moitié devrait pouvoir se reconnecter automatiquement à l'aide d' IST dès que le lien réseau est restauré.

Avertissement

Si vous définissez l'option d'amorçage sur les deux parties séparées, vous vous retrouverez avec deux instances de cluster vivantes, avec des données susceptibles de diverger les unes des autres. La restauration d'un lien réseau dans ce cas ne les fera pas se rejoindre tant que les nœuds ne seront pas redémarrés et que les membres spécifiés dans le fichier de configuration ne seront pas à nouveau connectés.

Ensuite, comme le modèle de réplication Galera se soucie vraiment de la cohérence des données : une fois l'incohérence détectée, les nœuds qui ne peuvent pas exécuter l'instruction de changement de ligne en raison d'une différence de données - un arrêt d'urgence sera effectué et le seul moyen de ramener les nœuds dans le cluster est via le SST complet

---

Revision #2
Created 11 November 2021 08:31:00 by ggpilou2
Updated 11 November 2021 08:39:17 by ggpilou2