Hyperscaling Aurora MySQL-Compatible pour faire face à la croissance soudaine du trafic

AWS Conseils prescriptifs

AWS Conseils prescriptifs: Hyperscaling Aurora MySQL-Compatible pour faire face à la croissance soudaine du trafic

Table of Contents

Hyperéchelle Aurora compatible MySQL pour gérer les hausses de trafic soudaines

Oliver Francis, Shyam Sunder Rakhecha et Vikram Singh Rai, Amazon Web Services (AWS)

Octobre 2022

Votre activité sur Internet pourrait faire face à une situation d'hyperéchelle sans précédent, que votre infrastructure d'applications existante n'est pas capable de gérer. Cela peut se produire en réponse à divers facteurs, tels qu'une publicité concernant votre produit qui suscite un intérêt plus important que prévu ou un changement soudain des habitudes d'achat des clients, qui passent de l'achat en personne à l'achat en ligne.

Pour faire face à ces charges inattendues, votre infrastructure doit passer par un processus d'hyperéchelle. La mise à l'échelle du niveau applicatif implique l'ajout de serveurs ou de pods. Toutefois, l'élément le plus complexe de l'hyperéchelle est la base de données. Vous pouvez mettre à l'échelle votre instance de base de données à la plus grande taille d'instance disponible, mais cela ne résoudra peut-être pas votre problème.

Si vous exécutez la charge de travail de votre base de données sur Édition compatible avec Amazon Aurora MySQL, ce guide fournit des recommandations que vous pouvez mettre en œuvre pour mettre à l'hyperéchelle votre base de données afin de faire face à une hypercroissance soudaine. Toutes ces recommandations ne constituent pas de bonnes pratiques à long terme. Si vous vous attendez à une hypercroissance et que vous avez le temps de planifier pour y faire face, veuillez consulter les billets de blog répertoriés dans la section Ressources. Ils peuvent vous aider à implémenter des bonnes pratiques à long terme. D'autre part, si vous êtes confronté à une hypercroissance inattendue, ce guide vous aidera à gérer votre charge de travail et à atteindre une stabilité raisonnable tout en planifiant et implémentant des solutions à long terme pour votre entreprise.

Notez que les recommandations décrites dans ce guide nécessiteront l'aide de votre équipe de développement en matière d'implémentation.

Résultats commerciaux ciblés

Les approches décrites dans ce guide vous aideront à effectuer les opérations suivantes :

- Stabiliser votre activité en cas d'hypercroissance inattendue. Atteindre une stabilité suffisante pour mettre en œuvre les bonnes pratiques à long terme en matière d'hypercroissance.

- Prévenir les pertes financières. Des interruptions soudaines dans un environnement mis à l'hyperéchelle peuvent entraîner une baisse du nombre de transactions commerciales effectuées sur votre application par vos clients. Cela peut entraîner des pertes financières importantes dans certains cas. Un environnement à hyperéchelle stabilisé est essentiel pour éviter les interruptions prolongées qui entraînent une perte d'activité.

Gestion des connexions

À mesure que la demande en faveur de votre application augmente, le trafic frontend augmente. Dans un scénario classique, vous configurez la mise à l'échelle automatique au niveau de l'application pour gérer une telle hausse du trafic entrant. Par conséquent, le niveau d'application commence à se mettre à l'échelle automatiquement et de nouveaux serveurs d'applications (instances) sont ajoutés pour répondre à l'augmentation du trafic. Comme tous les serveurs d'applications ont des paramètres de groupe de connexions à la base de données préconfigurés, le nombre de connexions entrantes vers la base de données augmente proportionnellement au nombre d'instances récemment déployées.

Par exemple, 20 serveurs d'applications configurés avec 200 connexions à la base de données ouvriraient chacun un total de 4 000 connexions à la base de données. Si le groupe d'applications augmente pour atteindre 200 instances (par exemple, pendant les heures de pointe), le nombre total de connexions atteindra 40 000. Dans le cadre d'une charge de travail type, la plupart de ces connexions sont probablement inactives. Toutefois, l'augmentation du nombre de connexions peut limiter la capacité de mise à l'échelle de votre base de données Amazon Aurora Édition compatible avec MySQL. Cela s'explique par le fait que même les connexions inactives consomment de la mémoire et d'autres ressources du serveur, telles que les descripteurs de fichiers. Aurora compatible MySQL utilise généralement moins de mémoire que MySQL Community Edition pour maintenir le même nombre de connexions. Cependant, l'utilisation de la mémoire pour les connexions inactives n'est toujours pas nulle.

Variables de configuration

Vous pouvez contrôler le nombre de connexions entrantes autorisées pour votre base de données à l'aide de deux principales variables de configuration du serveur : `max_connections` et `max_connect_errors`.

Variable de configuration max_connections

La variable de configuration `max_connections` limite le nombre de connexions à la base de données pour chaque instance MySQL. Il est recommandé de la définir sur une valeur légèrement supérieure au nombre maximal de connexions que vous comptez ouvrir sur chaque instance de base de données.

Si vous avez également activé `performance_schema`, soyez particulièrement prudent avec le paramètre `max_connections`. Les structures de mémoire du schéma de performance sont dimensionnées automatiquement en fonction des variables de configuration du serveur, notamment `max_connections`. Plus la variable est élevée, plus le schéma de performance utilise de mémoire. Dans les cas extrêmes, cela peut entraîner des out-of-memory problèmes sur des types d'instances plus petits. Notez que l'activation de l'Analyse des performances activera automatiquement le schéma de performance.

Variable de configuration max_connect_errors

La variable de configuration `max_connect_errors` détermine le nombre de demandes de connexion interrompues successives autorisées par un hôte client donné. Si l'hôte client dépasse le nombre spécifié d'échecs de tentatives de connexion successives, le serveur le bloque. D'autres tentatives de connexion de la part de ce client génèrent une erreur.

```
Host 'host_name' is blocked because of many connection errors. Unblock with
'mysqladmin flush-hosts'
```

Si vous rencontrez des erreurs « l'hôte est bloqué », évitez d'augmenter la valeur de la `max_connect_errors` variable. Examinez plutôt les compteurs de diagnostic du serveur dans la variable d'état `aborted_connects` et la table `host_cach`. Utilisez les informations collectées pour identifier et corriger les clients qui rencontrent des problèmes de connexion. Notez également que ce paramètre n'a aucun effet si `skip_name_resolve` est défini sur `1` (valeur par défaut).

Consultez le Manuel de référence de MySQL pour en savoir plus sur les points suivants :

- Variable Max_connect_errors
- Erreur « L'hôte est bloqué »
- Variable d'état Aborted_connects
- Table Host_cache

Implémenter un groupe de connexions

Un événement de mise à l'échelle peut ajouter d'autres serveurs d'applications, ce qui peut entraîner le dépassement du nombre de connexions actives entièrement chargées par le serveur de base de données. L'ajout d'un groupe de connexions ou d'une couche proxy entre les serveurs d'applications et la base de données agit comme un entonnoir, en réduisant le nombre total de connexions sur

la base de données. L'objectif principal d'un proxy est de réutiliser les connexions aux bases de données par le biais du multiplexage.

D'un côté, le proxy se connecte à la base de données avec un nombre de connexions contrôlé. De l'autre côté, le proxy accepte les connexions aux applications. Il propose également des fonctionnalités supplémentaires, telles que la mise en cache des requêtes, la mise en mémoire tampon des connexions, la réécriture et le routage des requêtes, ainsi que l'équilibrage de charge. La couche du groupe de connexions doit être configurée pour maintenir le nombre maximal de connexions à la base de données en dessous du nombre de connexions complètement chargées. Proxy Amazon RDS est un proxy entièrement géré que vous pouvez implémenter à cette fin. Proxy Amazon RDS ne nécessite aucune modification de code pour la plupart des applications et vous n'avez pas besoin de gérer d'infrastructure supplémentaire pour implémenter la solution.

Vous pouvez également explorer les proxys tiers suivants qui peuvent être utilisés avec Aurora compatible MySQL :

- ProxySQL
- MariaDB MaxScale
- ScaleARC
- Heimdall Data

Éviter les tempêtes de connexion

Réfléchissez au comportement de votre groupe de connexions en cas de surcharge de la base de données ou de réplica trop en retard par rapport au nœud primaire. Lorsque vous configurez votre serveur proxy ou vos groupes de connexions, veillez à ne pas réinitialiser l'ensemble du groupe de connexions en raison de la lenteur des réponses de la base de données (en raison de problèmes matériels ou de stockage sous-jacents ou à des contraintes de ressources de base de données).

Le démarrage soudain de centaines de connexions génère une tempête de connexion, car un grand nombre de demandes de nouvelles connexions à la base de données sont toutes lancées en même temps. La tempête consomme énormément de ressources. La création d'une connexion à une base de données dans MySQL est une opération coûteuse, car le backend échange plusieurs paquets réseau pour la première prise de contact, lance un nouveau processus, alloue de la mémoire, gère l'authentification, etc. Si un grand nombre de demandes sont reçues en peu de temps, la base de données peut donner l'impression de ne pas répondre.

MySQL dispose d'un mécanisme pour se protéger contre un tel pic de demandes de connexion. La variable `back_log` peut être définie sur le nombre de demandes qui peuvent être empilées pendant une courte période avant que MySQL ne cesse momentanément de répondre aux nouvelles demandes. La valeur est appliquée par un thread de gestion des connexions, qui peut lui-même être submergé par une tempête de connexion. Pour plus d'informations, consultez le manuel de référence MySQL.

Si votre connexion est configurée pour être réinitialisée lorsque la base de données est lente, vous relancerez le cycle encore et encore. De même, si vous anticipez une augmentation soudaine du trafic de base de données à certains moments de la journée (par exemple, à l'ouverture de la bourse), préchauffez votre groupe de connexions afin de ne pas essayer d'ouvrir de nombreuses connexions au même moment où une charge de trafic importante commence.

Optimisation de la charge de travail des requêtes

Une charge de travail bien ajustée vous permettra de trouver une solution stable à l'hypercroissance. Si votre charge de travail n'est pas bien ajustée, quelle que soit la puissance du cluster Amazon Aurora que vous utilisez, vous rencontrerez des goulots d'étranglement qui dégraderont les performances et qui auront un impact sur l'expérience utilisateur de votre application. Il est recommandé de mettre en place dès le départ un processus qui vous aide à identifier et à régler les requêtes problématiques dans votre système.

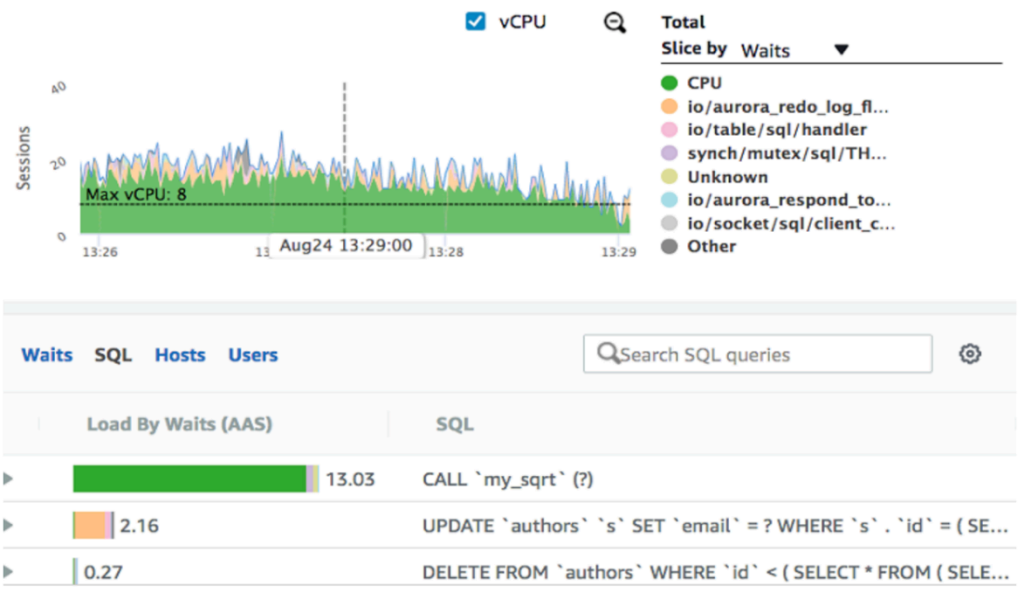
Suivi et réglage des requêtes problématiques liées à votre charge de travail

En cas d'hypercroissance, il suffit d'avoir une charge de travail bien réglée pour faire la moitié du chemin. Pour comprendre la nature des charges de travail en temps réel et des problèmes de performances auxquels votre cluster Aurora est confronté, assurez-vous que votre équipe recueille les requêtes problématiques provenant à la fois de votre instance d'enregistreur et de votre instance de lecteur. Ces requêtes problématiques doivent être réglées pour que votre charge de travail s'exécute de manière optimale. Amazon Aurora MySQL-Compatible Edition vous propose deux méthodes pour y parvenir :

- Performance Insights
- Journaux de requêtes lentes

Utilisation de l'Analyse des performances

L'Analyse des performances suit la charge sur l'instance d'enregistreur ou de lecteur Aurora en fonction de la moyenne des sessions actives (AAS). La valeur AAS est calculée à l'aide de l'échantillonnage et du nombre de sessions actives qui attendent qu'un processeur prenne en charge la charge de travail de ses requêtes et la traite. L'Analyse des performances propose une interface graphique dans laquelle vous pouvez vérifier les instructions SQL qui sont à l'origine de la charge la plus élevée en termes d'attente pour les sessions actives.



Dans la capture d'écran précédente, l'appel à la procédure stockée `my_sqrt` fait attendre le traitement de la charge de 13,03 sessions en moyenne. La prochaine étape logique consiste à ajuster cette procédure. Vous devez identifier les instructions SQL de vos lecteurs et enregistreurs qui sont à l'origine d'une charge sur leur instance respective et les régler pour améliorer les performances jusqu'à ce que les valeurs AAS chutent et restent en dessous de la ligne pointillée Max vCPU dans Analyse des performances. Si vos tentatives de réglage ont atteint un plafond et que vous constatez toujours que l'AAS est au-dessus de la ligne Max vCPU, vous pouvez opter pour une classe d'instance plus importante pour gérer votre charge de travail. N'optez pas pour une instance plus importante sans avoir d'abord essayé d'ajuster la charge de travail de vos requêtes, car l'augmentation du trafic commencera à révéler les lignes erronées créées par de mauvaises requêtes dans votre charge de travail.

Utiliser des journaux de requêtes lents et les publier sur CloudWatch

Le journal des requêtes lentes est une fonctionnalité native de MySQL qui vient compléter l'Analyse des performances. Il est recommandé d'utiliser ces deux méthodes pour garder une longueur d'avance sur les requêtes problématiques susceptibles de perturber vos instances. Le journal des requêtes lentes journalise toutes les requêtes plus lentes que la variable dynamique `long_query_time`. Cette variable peut être configurée sans aucun redémarrage de vos instances de cluster.

Pour garantir flexibilité et isolation lors de l'exercice de réglage, utilisez des groupes de paramètres distincts pour vos instances d'enregistreur et de lecteur. Ceci est particulièrement important si vous utilisez la répartition lecture-écriture. Définissez une limite confortable pour `long_query_time` dans vos instances de cluster en fonction de vos besoins. À mesure que vous réglez votre charge, vous pouvez viser des valeurs agressives dans la variable `long_query_time`, car vous pouvez définir le seuil à la milliseconde près. Avec une simultanéité élevée et une charge de travail bien ajustée, presque toutes vos requêtes devraient s'exécuter en quelques millisecondes.

Lorsque vous configurez Amazon Aurora MySQL-Compatible Edition pour enregistrer les requêtes lentes dans un fichier, Aurora MySQL-Compatible écrit les journaux des requêtes lentes dans le système de MySQL-Compatible fichiers Aurora et les conserve pendant 24 heures. Pour conserver les journaux de requêtes lents pendant une période plus longue à des fins d'analyse, publiez-les sur Amazon CloudWatch. Vous pouvez également créer un CloudWatch tableau de bord pour surveiller la lenteur de vos requêtes. Pour plus d'informations, consultez le billet de blog Creating an Amazon CloudWatch Dashboard to monitor Amazon RDS et Amazon Aurora MySQL. En plus d'analyser vos requêtes lentes sur Amazon CloudWatch, vous pouvez profiler les journaux de requêtes lents à l'aide de pt-query-digest, un outil du Percona Toolkit.

Vous pouvez également choisir d'automatiser ce processus de téléchargement et de profilage des requêtes pour améliorer l'efficacité au sein de votre équipe. Votre équipe doit rechercher les requêtes qui s'exécutent fréquemment et à des intervalles plus longs, et les régler en priorité. Visez un état dans lequel très peu de requêtes sont journalisées dans votre journal des requêtes lentes. Aussi, vous pouvez réduire le `long_query_time` pour devenir plus agressif en comprenant et en ajustant votre charge de travail.

Conseils pour le réglage des requêtes

Une fois que vous avez identifié les requêtes problématiques dans votre charge de travail, chaque requête doit être réglée. Suivez les instructions ci-dessous concernant le réglage pour optimiser l'efficacité de votre charge de travail.

Réduire le nombre de lignes analysées

Aussi simple que cela puisse paraître, c'est un excellent conseil à suivre lorsque vous réglez des requêtes. Utilisez l'instruction EXPLAIN et examinez la colonne des lignes pour voir combien de lignes l'optimiseur analyse à chaque jointure. Essayez de réduire le nombre de lignes analysées en créant un index optimal, puis expliquez à nouveau votre requête pour confirmer votre travail. Pour plus d’informations, consultez la documentation MySQL.

Si vous utilisez des tables partitionnées, interrogez-les toujours avec la clause WHERE qui permet l'élagage de partition afin que l'optimiseur n'ait pas à analyser chacune d'elles. Si votre clause WHERE contient une constante pour la colonne partitionnée, l'optimiseur sait quelle partition rechercher, ce qui améliore l'efficacité de votre requête.

Un autre aspect de ce conseil est la conception de votre base de données. Moins il y a de tables dans votre requête, plus elle sera rapide. Si vous pouvez dénormaliser la conception de votre base de données, vous pouvez faire en sorte que l'optimiseur analyse moins de lignes, et ainsi améliorer les performances des requêtes.

Réduire l'utilisation des tables temporaires et les tables temporaires sur disque

L' MySQL-Compatible optimiseur Aurora crée des tables temporaires à la fois sur la RAM et sur le disque s'il ne peut pas obtenir les résultats souhaités pour votre requête directement à partir des index. Par conséquent, une grande partie du réglage consiste à disposer des bons index desservant votre charge de travail. Toutefois, il se peut que certaines requêtes de votre charge de travail ne puissent pas reposer uniquement sur des index. Certaines opérations peuvent donc être effectuées dans un fichier temporaire. Cela ne pose pas de problème tant que vous les limitez au minimum et que vous veillez à ce que très peu de tables soient créées sur le disque. MySQL crée des tables de disque lorsque la taille de la table temporaire est trop grande pour être stockée en mémoire. La logique utilisée par MySQL pour vérifier la taille de la table temporaire interne est la plus petite des deux valeurs de variables tmp_table_size et max-heap-table-size. Vous pouvez régler ces variables pour obtenir une valeur optimale en fonction de votre charge de travail. Ainsi, dans les cas où vous ne pouvez pas empêcher l'installation de tables temporaires, vous ne les transférez sur disque qu'en de rares occasions.

Éviter de trier des fichiers

Si votre charge de travail comporte beaucoup de requêtes `ORDER BY`, le meilleur moyen de les résoudre est d'utiliser les bons index sur vos tables. Assurez-vous que vos index à colonnes multiples sont bien conçus pour éviter le tri dans des fichiers. Le tri ne peut pas être effectué sur une colonne si les colonnes précédentes ne sont pas analysées avec des constantes (`in`, `>`, `<`, `!=` et `BETWEEN` n'autoriseront pas le tri dans la colonne suivante à droite). La méthode de tri optimale dans MySQL consiste à placer un index à colonnes multiples qui positionne les colonnes contenant les valeurs constantes fournies dans la requête à gauche de la colonne de tri dans une structure contiguë. Si,

en dernier ressort, votre requête ne peut pas renvoyer de résultats sans tri de fichiers, déplacez le tri vers l'application.

Éviter d'exécuter des requêtes d'agrégation en cas de forte simultanéité

Votre charge de travail peut comporter un petit nombre de requêtes d'agrégation pour répondre à certaines fonctionnalités de votre application. Ce cas d'utilisation exige une grande prudence. Le moteur InnoDB est conçu pour des charges de traitement des transactions en ligne (OLTP) appropriées, mais même quelques requêtes groupées en cas de simultanéité élevée peuvent être très lourdes pour le processeur et peuvent rapidement dégrader les performances de votre cluster. Pour résoudre les cas d'utilisation où vous avez besoin d'ensembles de résultats agrégés, agrégez préalablement les données dans des tableaux prêts à être lus afin d'éviter tout regroupement par requêtes.

Tester la simultanéité de vos requêtes

Lorsque vous réglez des requêtes individuelles, n'oubliez pas que celles-ci s'exécutent simultanément sur plusieurs vCPU dans Aurora. MySQL-Compatible Votre requête peut s'exécuter en quelques millisecondes dans votre environnement de test en une seule exécution. Mais ce n'est pas tout. Assurez-vous de tester votre requête avec le niveau de simultanéité attendu sur votre cluster de production et de comparer ses performances. Ne mettez la requête en production que lorsqu'elle répond à vos objectifs de simultanéité. Assurez-vous d'utiliser l'optimiseur `hint sql_no_cache` dans vos scripts de test afin d'éviter de récupérer les résultats depuis le cache. Vous pouvez utiliser des outils tels que mysqlslap pour effectuer le test simultanément et comparer les résultats.

Réglage des charges de travail d'écriture

L'implémentation de l'équilibrage de charge et la libération de l'instance d'enregistreur aideront votre charge de travail d'écriture à fournir de meilleures performances en cas de pics élevés. Pour obtenir de meilleures performances d'écriture en cas de simultanéité élevée, suivez ces étapes supplémentaires.

Déplacer l'intégrité référentielle dans la couche applicative

Bien que les contrôles de l'intégrité référentielle soient importants, l'hyperéchelle peut nuire à votre charge. Pour chaque écriture, des analyses supplémentaires doivent être effectuées avant que l'écriture elle-même ne soit exécutée, ce qui donne lieu à de mauvaises performances. Si votre

application nécessite des contrôles d'intégrité stricts, intégrez-les dans la couche d'application pour éviter qu'ils ne limitent vos écritures.

Éviter d'utiliser des clés primaires lourdes

Faites en sorte que vos clés primaires restent légères. Le moteur de stockage InnoDB ajoute la clé primaire à tous les autres index que vous créez dans votre table. Lorsque votre clé primaire est volumineuse, la taille de l'index en sera affectée. Le stockage et la récupération des pages de données seront ralentis si la clé primaire est assez volumineuse. L'utilisation d'identifiants uniques universels comme clés primaires en est un exemple courant. Il ne s'agit pas d'une bonne approche si vous ciblez les performances dans un environnement à l'hyperéchelle.

Utiliser l'échange de partitions pour charger des données dans des tables partitionnées

Si vous écrivez de grands jeux de données dans des tables partitionnées, la combinaison de LOAD DATA FROM S3 et d'échange de partitions peut améliorer les performances, car la table principale n'est pas accessible pour les insertions. L'échange de partitions utilise un langage de définition de données (DDL) qui verrouille les métadonnées de votre table. Assurez-vous que cette opération est effectuée lorsque peu ou pas de requêtes sont exécutées sur la table afin que le DDL d'échange de partitions puisse obtenir le verrouillage des métadonnées sans attendre. L'échange lui-même ne prend que quelques millisecondes.

Supprimez les index inutilisés

InnoDB optimise ses plans de requêtes en fonction de l'évolution de vos données, et il est conseillé de vérifier les index inutilisés dans votre base de données et de les supprimer. Les index non utilisés consomment des E/S lorsque l'application écrit des données dans une table. Consultez la liste des index non utilisés et vérifiez qu'il ne s'agit pas d'index utilisés dans de rares situations, telles que les rapports trimestriels. Notez également que certains index sont utilisés pour appliquer l'unicité ou l'ordre et doivent également être pris en compte.

S'assurer que les anciennes versions de ligne sont correctement purgées

Dans l'implémentation du contrôle de simultanéité multiversion (MVCC) d'InnoDB, lorsqu'un enregistrement est modifié, la version actuelle (ancienne) des données modifiées est d'abord enregistrée sous la forme d'un enregistrement d'annulation dans un journal d'annulation. Une valeur de longueur de la liste d'historique (HLL) croissante indique que les threads de récupérateur de

mémoire d'InnoDB (threads de purge) ne suivent pas le rythme de la charge de travail d'écriture ou que la purge est bloquée par une requête ou une transaction de longue durée. Lorsque le récupérateur de mémoire est bloqué ou retardé, la base de données peut développer un retard de purge important qui peut affecter négativement les performances des requêtes. Vous pouvez utiliser les recommandations suivantes pour optimiser le processus de purge.

- Limitez le volume de transactions.
- Pour les requêtes de lecture, utilisez le niveau d'isolation READ COMMITTED.
- Augmentez le nombre de threads de purge (innodb_purge_threads et innodb_purge_batch_size). Notez que le réglage de ces paramètres nécessite un redémarrage.
- Surveillez régulièrement la HLL et résolvez les problèmes de charge de travail qui empêchent la progression du récupérateur de mémoire.

S'assurer que la journalisation n'entraîne pas de conflits supplémentaires

Le journal des requêtes générales journalise les connexions et les déconnexions des clients, ainsi que toutes les instructions reçues par le serveur dans l'ordre de leur réception. Lorsqu'elle est activée, la journalisation est synchrone, ce qui peut entraîner une réduction importante des performances sur un système occupé. À moins que cela ne soit nécessaire, nous vous recommandons de désactiver le journal général.

Le journal des requêtes lentes enregistre les instructions qui ont pris plus de temps que le nombre de secondes d'exécution long_query_time, le réglage par défaut étant de 10 secondes. Lorsque le paramètre est défini sur 0, toutes les instructions sont journalisées de manière synchrone, ce qui peut affecter les performances des bases de données occupées.

Déchargement

L'amélioration des temps de réponse et l'augmentation des ressources disponibles pour les flux de travail critiques peuvent nécessiter de se débarrasser de la charge superflue. La plupart des solutions abordées dans cette section sont des décisions de compromis. Elles ont des conséquences sur l'application et doivent être soigneusement évaluées. Vous pouvez envisager ces solutions dans les cas suivants :

- Vous utilisez déjà les instances les plus volumineuses, en particulier pour l'instance de base de données principale accessible en écriture.

- En dernier recours, pour fournir une marge de manœuvre suffisante à court terme pour implémenter d'autres modifications.

Les modifications immédiates sont les suivantes :

- Déplacez le trafic de lecture non critique à l'écart de l'instance de base de données principale.
- Archivez ou purgez les anciennes données.
- Supprimez l'intégrité référentielle.
- Désactivez le journal binaire (s'il est utilisé).
- Différez les processus d'extraction, de transformation et de chargement (ETL) non critiques.
- Suspendez ou dégradez les fonctionnalités non essentielles de l'application.

Avant d'entreprendre ces actions, évaluez-les dans le contexte des objectifs et des risques métier à long terme.

Charges de travail de lecture et d'écriture distinctes

Une technique fréquemment utilisée lors de l'exécution d'applications à technologie MySQL consiste à décharger les opérations de lecture de l'instance de base de données (principale) d'enregistreur vers un ou plusieurs réplicas de base de données en lecture seule. En déchargeant les lectures, vous pouvez réduire la charge globale sur l'instance de base de données principale et libérer de la place pour les écritures. Veillez à ne cibler que les lectures qui ne dépendent pas de la cohérence immédiate de lecture après écriture pour les réplicas. Une approche plus efficace consiste à déplacer tout le trafic de lecture vers le réplica et à prévoir une nouvelle tentative de lecture après écriture en cas de retard de la réplication. Certaines charges de travail de lecture indépendantes peuvent être déchargées, telles que les services de génération de rapports. D'autres lectures nécessiteront des modifications au niveau de l'application, où le contexte dans lequel la lecture a été émise est bien connu.

Une autre approche consiste à implémenter une solution de proxy de base de données en tant qu'intermédiaire entre l'application et la base de données, qui peut fournir la fonction de répartition de lecture/écriture et de routage des requêtes, sans que l'application en soit consciente. ProxySQL, MariaDB MaxScale, ScaleArc et Heimdall Data font partie des solutions de proxy compatibles MySQL disponibles. Ces produits offrent de nombreuses fonctionnalités supplémentaires, telles que la mise en cache ou le multiplexage des connexions. Lorsque vous êtes confronté à une augmentation soudaine du trafic et que vous implémentez une nouvelle technologie dans votre stack

d'applications, nous vous recommandons de commencer par les fonctionnalités de base permettant de fractionner les read/write requêtes et de tester le comportement et les performances avant d'utiliser des fonctionnalités supplémentaires susceptibles d'avoir des effets secondaires imprévus.

Archiver ou purger les anciennes données

Une autre technique visant à améliorer les performances de la base de données est de décharger des données historiques sur une autre table, une autre base de données ou Amazon Simple Storage Service (Amazon S3). De nombreuses bases de données conservent toutes les données en ligne pendant tout l'historique de l'application. Dans des circonstances normales, dans une application standard destinée aux utilisateurs, cela permet aux utilisateurs de consulter l'historique de toutes leurs commandes. Lorsque la demande augmente soudainement, de nombreux utilisateurs actifs sur l'application sont probablement nouveaux ou sont en train de passer une nouvelle commande. Si les données historiques se trouvent en ligne, dans une seule table contenant des milliards de lignes, cela se complique. La table comporte probablement également des index volumineux. Les données de table et les index sont stockés dans une structure arborescente. La présence d'un plus grand nombre d'entrées dans une table nécessite un plus grand nombre de niveaux dans l'arborescence, ce qui nécessite davantage d' I/O opérations pour accéder aux lignes. Cela augmente le temps d'accès requis pour trouver des enregistrements individuels. Plus important encore, de grandes parties inutiles de l'index résident alors dans le cache des pages de la base de données (groupe de mémoires tampons InnoDB).

L'archivage des anciennes données dans une table séparée, une base de données distincte ou Amazon S3 peut réduire les temps d'accès pour les utilisateurs finaux, libérer un précieux cache et améliorer les performances globales des applications. L'archivage des anciennes données, avant l'événement (par exemple, en ne les conservant que 30 ou 90 jours), limite la taille des tables et des index des tables critiques. Cette modification nécessite que l'application soit modifiée pour interroger les anciennes données à partir d'un emplacement secondaire uniquement pour le petit sous-ensemble d'utilisateurs qui demandent explicitement à consulter les données historiques.

Différer les processus ETL non critiques

Les extractions du système de base de données principal pour les processus ETL peuvent présenter un risque de stabilité pour les charges de travail hautement transactionnelles et simultanées dans des conditions d'hyperéchelle. Les processus ETL sont connus pour les caractéristiques suivantes :

- Long-running transactions
- Larges verrous

- Processeur, mémoire et I/O consommation
- Conflit avec des charges de travail transactionnelles servant un chemin critique pour l'utilisateur final.

Processus ETL variables ou imprévisibles (par exemple, les requêtes telles que `INSERT INTO ... SELECT FROM ...;`) s'ajoutent à la variabilité globale en matière de charge et de conflit, augmentant ainsi le risque de stabilité).

Dans la mesure du possible, différez ou réduisez la fréquence d'exécution des processus ETL, en particulier s'ils ne fournissent pas de fonctions critiques. Pour les processus ETL critiques, modifiez-les de manière à ce qu'ils puissent fonctionner dans des unités de travail limitées, comme le traitement de lots d'un nombre fixe de lignes (par exemple, en utilisant des clauses `LIMIT`), l'utilisation de transactions distinctes ou l'extraction de plus petites quantités de données sur une période prolongée afin de réduire les pics de demande de ressources de l'instance de base de données.

Désactiver les fonctionnalités moins critiques des applications

La dégradation normale de l'expérience utilisateur ou la suppression des fonctionnalités secondaires lors de la gestion d'un afflux de demandes permet de préserver les ressources de la base de données pour les fonctions critiques. Cela peut nécessiter une légère modification de l'expérience client, mais un flux différent est préférable à une interruption du site. Peut-être que la personnalisation sur la page d'accueil du site qui adresse toujours un appel à la base de données peut être temporairement désactivée. Vous pouvez également arrêter de présenter des offres personnalisées aux clients qui reviennent lorsqu'ils sélectionnent des produits. La désactivation temporaire des fonctionnalités peut permettre la mise en cache ou la diffusion de la page sans nécessiter l'accès à la base de données.

D'autres exemples incluent un frontend doté d'un mécanisme d'interrogation qui vérifie les modifications des données nécessitant un appel de base de données. La réduction de la fréquence d'interrogation se traduira immédiatement par une diminution du nombre d'appels de base de données. Les interfaces utilisateur qui nécessitent une pagination ou qui offrent aux utilisateurs la possibilité de récupérer des ensembles de résultats triés plus éloignés du premier résultat nécessitent des appels de base de données de plus en plus coûteux. Limitez le nombre de pages d'un ensemble de résultats afin de protéger la base de données contre les appels de base de données les plus coûteux. Utilisez des indicateurs de fonctionnalités dans la couche d'application pour permettre à

l'équipe des opérations de désactiver ou de dégrader des fonctionnalités à mesure que la charge de l'application ou de la base de données augmente.

Ajustement des paramètres

Outre les paramètres liés à la connexion, il existe une poignée de paramètres que vous pouvez régler pour améliorer les performances de votre cluster Amazon Aurora MySQL Edition en cas d'hypercroissance. Lorsque vous modifiez un paramètre de base de données, il est important de suivre les meilleures pratiques suivantes :

- Modifiez un paramètre à la fois afin de pouvoir mesurer l'impact de la modification.
- Certains paramètres peuvent nécessiter une période de préchauffage pour que leur effet se manifeste après leur modification. Tenez-en compte lorsque vous observez et mesurez les performances de votre cluster Aurora compatible avec MySQL
- Évitez les valeurs de paramètres incorrectes, qui pourraient empêcher le démarrage de l'instance de base de données.

Ces paramètres incluent :

- `sync_binlog`
- `table_open_cache`
- `innodb_sync_array_size`
- `innodb_flush_log_at_trx_commit`
- `autocommit`
- `tmp_table_size`
- `max_heap_table_size`

Pour obtenir des instructions sur la configuration de ces paramètres, consultez les sections Paramètres de configuration d'Aurora MySQL, Recommandations relatives aux fonctionnalités MySQL dans Aurora MySQL et Comportement des tables temporaires dans Aurora MySQL dans la AWS documentation.

Ressources

- Journey to Adopt Cloud-Native Architecture Series: #1 – Preparing your Applications for Hypergrowth (billet de blog)
- Journey to Adopt Cloud-Native Architecture Series: #2 – Maximizing System Throughput (billet de blog)
- Utilisation d'Amazon RDS Proxy
- Stratégies de mise en cache
- Activation ou désactivation de l'Analyse des performances
- Le moteur de stockage InnoDB
- Réplicas Aurora
- MariaDB Connector/J
- MariaDB MaxScale
- ProxySQL
- Heimdall Data

Historique du document

Le tableau suivant décrit les modifications importantes apportées à ce guide. Pour être averti des mises à jour à venir, abonnez-vous à un fil RSS.

Modification	Description	Date
Publication initiale	—	5 octobre 2022

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.