Apache Cassandra sur AWS

Consignes et meilleures pratiques

Janvier 2016

Mentions légales

Ce document est fourni à titre informatif uniquement. Il présente l'offre de produits et les pratiques actuelles d'AWS à la date de publication de ce document, des informations qui sont susceptibles d'être modifiées sans avis préalable. Il incombe aux clients de procéder à leur propre évaluation indépendante des informations contenues dans ce document et chaque client est responsable de son utilisation des produits ou services AWS, chacun étant fourni « en l'état », sans garantie d'aucune sorte, qu'elle soit explicite ou implicite. Ce document ne crée pas de garanties, représentations, engagements contractuels, conditions ou assurances à l'encontre d'AWS, de ses affiliés, fournisseurs ou donneurs de licence. Les responsabilités et obligations d'AWS vis-à-vis de ses clients sont régies par les contrats AWS. Le présent document ne fait partie d'aucun et ne modifie aucun contrat entre AWS et ses clients.

Résumé

Amazon Web Services (AWS) est une plateforme de cloud computing flexible, économique et simple à utiliser. Apache Cassandra est une base de données NoSQL demandée qui est largement déployée dans le cloud AWS. L'exécution de votre propre déploiement Cassandra sur Amazon Elastic Cloud Compute (Amazon EC2) est une excellente solution pour les utilisateurs dont les applications ont des exigences de débit élevé.

Ce livre blanc fournit une présentation de Cassandra et son implémentation sur la plateforme de cloud AWS. Il traite également des meilleures pratiques et des caractéristiques d'implémentation telles que la performance, la durabilité et la sécurité, et se concentre sur les fonctions AWS importantes pour Cassandra qui contribuent à garantir l'évolutivité, la haute disponibilité et la récupération d'urgence de manière rentable.

Introduction

Les bases de données NoSQL sont un type de base de données optimisé pour les opérations haute performance sur de grands ensembles de données. Chaque type de base de données NoSQL fournit sa propre interface pour accéder au système et à ses fonctions. Une manière de choisir des types de base de données NoSQL consiste à consulter le modèle de données sous-jacent, comme présenté ci-après :

- Stockages de valeurs de clé : les données sont organisées comme des relations de valeurs de clé et ouvertes via une clé principale. Ces produits sont généralement des stockages en ligne distribués. Cassandra et Amazon DynamoDB en sont des exemples.
- Bases de données de graphiques : les données sont organisées sous forme de structures de données de graphiques et ouvertes via des requêtes sémantiques. Titan et Neo4J en sont des exemples.
- Bases de données de document : les données sont organisées sous forme de documents (par exemple, des fichiers JSON) et ouvertes par des champs dans les documents. MongoDB et DynamoDB en sont des exemples.
- Bases de données en colonnes : les données sont organisées sous forme de sections de colonnes de données, au lieu de lignes de données. Exemple : HBase.

DynamoDB s'affiche à la fois dans le document et dans les stockages de valeurs de clé dans cette liste car il prend en charge le stockage et les requêtes à la fois des objets et des paires de valeurs de clé dans un format de document tel que JSON, XML ou HTML.

NoSQL on AWS

Amazon Web Services fournit plusieurs options de logiciel de base de données NoSQL pour les clients cherchant une solution entièrement gérée, ou pour les clients qui veulent un contrôle complet de leurs bases de données NoSQL mais ne souhaitent pas gérer l'infrastructure matérielle. Toutes nos solutions offrent une tarification flexible, à l'utilisation qui vous permet une mise à l'échelle rapide et facile, à moindre coût.

Envisagez les options suivantes comme des alternatives possibles de développement de votre propre système avec un logiciel open source (OSS) ou un produit NoSQL commercial.

- Amazon DynamoDB est un service de base de données NoSQL entièrement géré qui fournit des performances rapides et prévisibles avec une évolutivité transparente.[1] Tous les éléments de données dans DynamoDB sont stockés sur des disques SSD et sont automatiquement répliqués sur trois sites dans une région AWS pour fournir une durabilité des données et une disponibilité élevée intégrées. Avec Amazon DynamoDB, vous pouvez vous défaire de la lourde charge administrative que représentent l'exploitation et le dimensionnement d'un cluster de base de données distribuée hautement disponible, tout en ne payant qu'un faible prix variable en fonction des ressources que vous utilisez.
- Amazon Simple Storage Service (Amazon S3) fournit une interface simple de services Web qui peut stocker et récupérer n'importe quel volume de données, à tout moment, depuis n'importe où sur le Web.[2] Amazon S3 permet aux développeurs d'accéder à la même infrastructure hautement évolutive, fiable, sécurisée, rapide et peu coûteuse que celle utilisée par Amazon pour exécuter son propre réseau global de sites Web. Amazon S3 maximise les avantages de l'évolutivité et vous transmet ces bénéfices.

Cassandra : courte présentation

Remarque : il s'agit d'une courte présentation de la manière dont Cassandra fonctionne. Pour plus d'informations, consultez la documentation DataStax.[3]

Apache Cassandra est une base de données NoSQL open source amplement évolutive, qui est idéale pour la gestion de grands volumes de données structurées, semi-structurées et non structurées sur plusieurs sites distribués. Cassandra se base sur une arborescence fusionnée à structure de journaux, une structure de données qui est hautement efficace avec des opérations d'écriture à volume élevé.[4] Le cas d'utilisation le plus connu pour Cassandra est le stockage de données de série chronologique.

Cassandra fournit une simplicité opérationnelle, une évolutivité linéaire et une disponibilité continue sur plusieurs serveurs de commodité sans point de défaillance unique, ainsi qu'un modèle de données dynamique puissant conçu pour offrir des temps de réponse rapides et une flexibilité maximum. Cassandra est un système distribué pair à pair sans maître où les données sont distribuées parmi tous les nœuds du cluster. Chaque nœud a des connaissances sur la topologie du cluster et échange des informations sur le cluster à chaque seconde.

Cassandra: Key Terms and Concepts

Avant que nous discutions des meilleures pratiques et considérations concernant l'utilisation de Cassandra sur AWS, consultons quelques concepts clés.

Un *cluster* est la plus grande unité de déploiement dans Cassandra. Chaque cluster se compose de nœuds d'un ou de plusieurs emplacements distribués (Zones de disponibilité ou AZ en termes AWS).

Un *emplacement distribué* contient un ensemble de nœuds qui font partie d'un cluster. En général, lors de la conception d'un cluster Cassandra sur AWS, nous recommandons l'utilisation de plusieurs zones de disponibilité où stocker vos données dans le cluster. Vous pouvez configurer Cassandra pour répliquer des données sur plusieurs zones de disponibilité, ce qui permet à votre cluster de base de données d'être hautement disponible, même en cas de défaillance d'une zone de disponibilité. Pour garantir une distribution homogène des données, le nombre de zones de disponibilité doit être un multiple du facteur de réplication. Les zones de disponibilité sont également connectées via des liens à faible latence, ce qui contribue à éviter la latence pour la réplication.

Un *nœud* est un élément d'un emplacement distribué unique dans un cluster Cassandra qui stocke des partitions de données en fonction de l'algorithme de partitionnement.

Un *journal de validation* est un journal WAL (write-ahead log) sur chaque nœud du cluster. Toute opération d'écriture effectuée sur Cassandra est tout d'abord écrite dans l'ordre sur cette structure d'ajout uniquement, qui est ensuite vidée depuis le cache en écriture différée sur le système d'exploitation (SE) vers le disque, soit périodiquement, soit par lots. Dans l'éventualité d'une récupération de nœud, les journaux de validation sont rejoués pour réaliser la récupération des données.

Un *memtable* est fondamentalement un cache en écriture différée de lignes de données qui peut être recherché par clé. Il s'agit d'une structure en mémoire. Un memtable unique stocke uniquement des données pour une table unique et est vidé sur le disque ou bien lorsque les seuils de mémoire globale du nœud ont été atteints, ou bien lorsque le journal de validation est plein, ou encore après qu'un intervalle de niveau de table est atteint.

Une *SStable* (table de chaînes triées) est une structure logique constituée de plusieurs fichiers physiques sur disque. Une SStable est créée lorsqu'un memtable est vidé sur le disque. Une SStable est une structure de données non modifiable. Les memtables sont triés par clé puis écrits séquentiellement pour créer une SStable. Ainsi, les opérations d'écriture dans Cassandra sont extrêmement rapides, ne coûtant qu'un ajout dans un journal de validation et une opération d'écriture séquentielle amortie pour le vidage.

Un *filtre bloom* est une structure de données basée sur les probabilités pour tester l'appartenance Set qui ne produit jamais un faux négatif, mais peut être ajustée pour de faux positifs. Les filtres bloom sont des structures hors segment de mémoire. Ainsi, si un filtre bloom répond qu'une clé n'est pas présente dans une SStable, alors la clé n'est pas présente. En revanche, s'il répond que la clé est présente dans la SStable, elle peut être présente ou non. Les filtres bloom peuvent aider à évaluer les demandes en lecture dans Cassandra. Les filtres bloom peuvent également enregistrer d'autres opérations de lecture sur disque lisant la SStable, en indiquant si une clé n'est pas présente dans la SStable.

Un *fichier d'index* entretient le décalage des clés dans le fichier de données principal (SStable). Cassandra contient par défaut un échantillon du fichier d'index en mémoire qui stocke le décalage pour chaque 128ème clé dans le fichier de données principal (cette valeur est configurable). Les fichiers index peuvent également aider à mieux lire des opérations car ils peuvent vous fournir la position aléatoire dans la SStable à partir de laquelle vous pouvez analyser séquentiellement pour obtenir les données. Sans les fichiers d'index, vous devez analyser l'ensemble de la SStable pour récupérer des données.

Un *espace de clés* est un conteneur logique dans un cluster qui contient une ou plusieurs tables. La stratégie de réplication est généralement définie comme le niveau d'espace de clés.

Une *table*, également appelée *famille de colonnes*, est une entité logique dans un espace de clés composée d'un ensemble de colonnes ordonnées récupérées par ligne. La définition de clé principale est requise lors de la définition d'une table.

Write Request Flow

Le schéma suivant affiche un cluster Cassandra avec sept nœuds avec un facteur de réplication de 3. Les clients écrivent sur le cluster en utilisant le niveau de cohérence quorum.[5] Lors de l'utilisation du niveau de cohérence quorum, les opérations d'écriture aboutissent si deux nœuds sur trois reconnaissent la réussite au coordinateur (le nœud auquel le client se connecte).

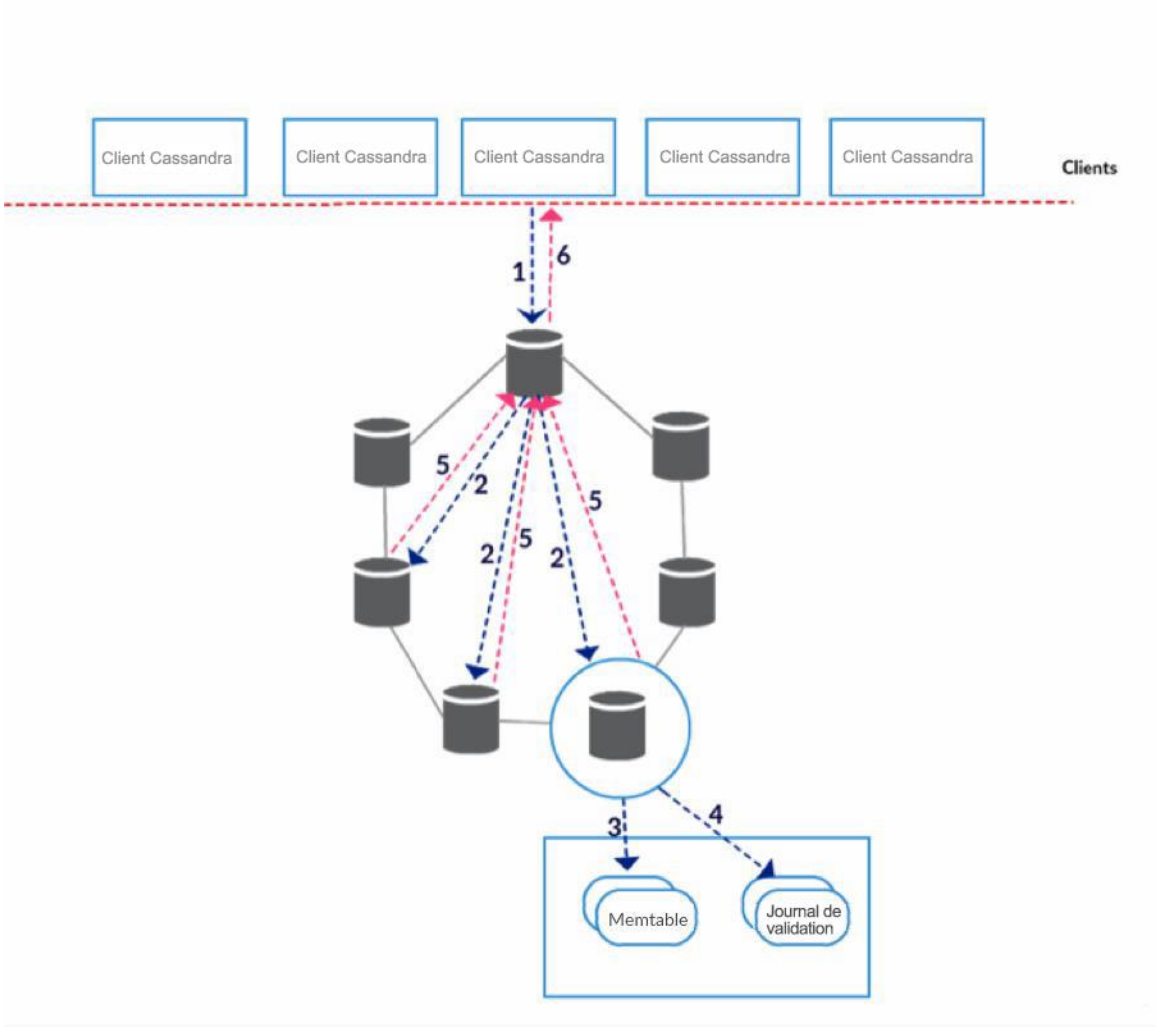

Figure 1 : Flux de demande d'écriture

Le schéma précédent illustre une demande d'écriture type sur Cassandra avec une réplication tridimensionnelle, comme décrit ci-après :

1. Un client envoie une demande à un nœud dans le cluster pour stocker une clé donnée. À ce stade, le nœud peut être ou ne pas être la bonne partition pour stocker la clé. S'il n'est pas la bonne partition, le nœud sert de coordinateur (le cas dans cet exemple). Veuillez noter qu'un nœud peut servir soit de réplica, soit de coordinateur, soit les deux (si le nœud mappe vers les données et parle au client).
2. Le coordinateur détermine les nœuds de réplica qui devraient stocker la clé et transmet la demande à ces nœuds.
3. Chaque nœud qui obtient les clés réalise une opération d'écriture séquentielle des données, ainsi que les métadonnées requises pour recréer les données dans le journal de validation localement.
4. La clé, ainsi que ses données, sont écrites dans le memtable en mémoire, localement.
5. Les nœuds de réplica répondent au coordinateur avec une réussite ou un échec.
6. Selon le niveau de cohérence spécifié dans le cadre de la demande, le coordinateur répondra avec une réussite ou un échec au client. Par exemple, avec un niveau de cohérence de quorum et un facteur de réplication de 3, le coordinateur répond avec une réussite dès que deux nœuds sur trois répondent par une réussite.

Cela dit, pendant l'étape 5 précédente, si certains nœuds ne répondent pas et échouent (par exemple, un nœud sur trois), alors le coordinateur stocke un indicateur localement pour envoyer l'opération d'écriture au nœud ou aux nœuds ayant échoué lorsque le ou les nœuds sont à nouveau disponibles. Ces indicateurs sont stockés avec une durée de vie égale à la valeur de paramètre *gc_grace_seconds* , afin qu'ils ne soient pas relus ultérieurement. Les indicateurs ne seront enregistrés que pour une période égale au paramètre *max_hint_window_in_ms* (défini dans cassandra.yaml), qui est par défaut sur trois heures.

Alors que les clients continuent d'écrire sur le cluster, un thread d'arrière-plan continue de vérifier la taille de tous les memtables actuels. Si le thread détermine que les seuils de mémoire globale de nœud ont été atteints, que le journal de validation est plein, ou qu'un intervalle de niveau de table a été atteint, il crée un memtable pour remplacer l'actuel et marque le memtable remplacé pour vidage. Les memtables marqués pour vidage le sont sur le disque par un autre thread (généralement par plusieurs threads).

Une fois qu'un memtable est vidé sur le disque, toutes les entrées pour les clés correspondant à ce memtable qui résident dans un journal de validation ne sont plus requises, et ces segments de journal de validation sont marqués pour recyclage.

Lorsqu'un memtable est vidé sur le disque, deux autres structures de données sont créées : un filtre bloom et un fichier d'index.

Compactage

Le nombre de SStables peut augmenter sur une période. Pour que les SStables demeurent gérables, Cassandra réalise automatiquement des compactages mineurs par défaut. Le compactage fusionne plusieurs SStables en fonction d'un algorithme que vous spécifiez en utilisant une stratégie de compactage.

Le compactage vous donne la possibilité d'optimiser vos opérations en lecture en vous permettant de lire un plus petit nombre de SStables pour satisfaire la demande de lecture. Le compactage fusionne fondamentalement plusieurs SStables en fonction du seuil configurable pour créer une ou plusieurs SStables non modifiables. Par exemple, la stratégie de compactage par défaut, Size Tiered Compaction, regroupe plusieurs SStables de taille similaire et crée une seule et grande SStable. Elle continue de répéter ce processus sur les SStables de taille similaire.

Le compactage ne modifie pas les SStables existantes (n'oubliez pas, les SStables sont non modifiables) et crée uniquement une SStable à partir de celles existantes. Lorsqu'une SStable est créée, les anciennes sont marquées pour suppression. Ainsi, l'espace utilisé est temporairement supérieur pendant le compactage. Le volume d'espace supplémentaire dû au compactage dépend de la stratégie de compactage utilisée. Cet espace supplémentaire doit être pris en considération pendant le processus de planification. Les SStables qui sont marquées pour suppression le sont à l'aide d'un mécanisme de comptage de références ou pendant un redémarrage.

Read Request Flow

Avant de nous plonger dans le flux de demandes de lecture, nous allons résumer ce que nous savons concernant un cluster Cassandra.

Dans un cluster, chaque ligne est répliquée sur plusieurs nœuds (selon votre facteur de réplication). Il n'y a pas de concept de nœud principal. Cette approche signifie que tout nœud dans le cluster qui contient la ligne peut répondre à des demandes concernant cette ligne. Cassandra utilise le protocole Gossip pour échanger des informations sur la topologie de réseau entre les nœuds. Grâce à Gossip, chaque nœud découvre la topologie du cluster et peut déterminer où doit être envoyée une demande pour une ligne donnée dans le cluster.

Dans le schéma suivant, nous avons un cluster Cassandra avec sept nœuds et un facteur de réplication de 3. Les clients lisent depuis le cluster en utilisant le niveau de cohérence quorum. Tout en utilisant un niveau de cohérence quorum, les opérations de lecture aboutissent si deux nœuds sur trois reconnaissent la réussite.

Avec ce bref contexte, étudions de quelle manière les demandes de lecture sont servies. La figure et la liste ci-après en sont une illustration.

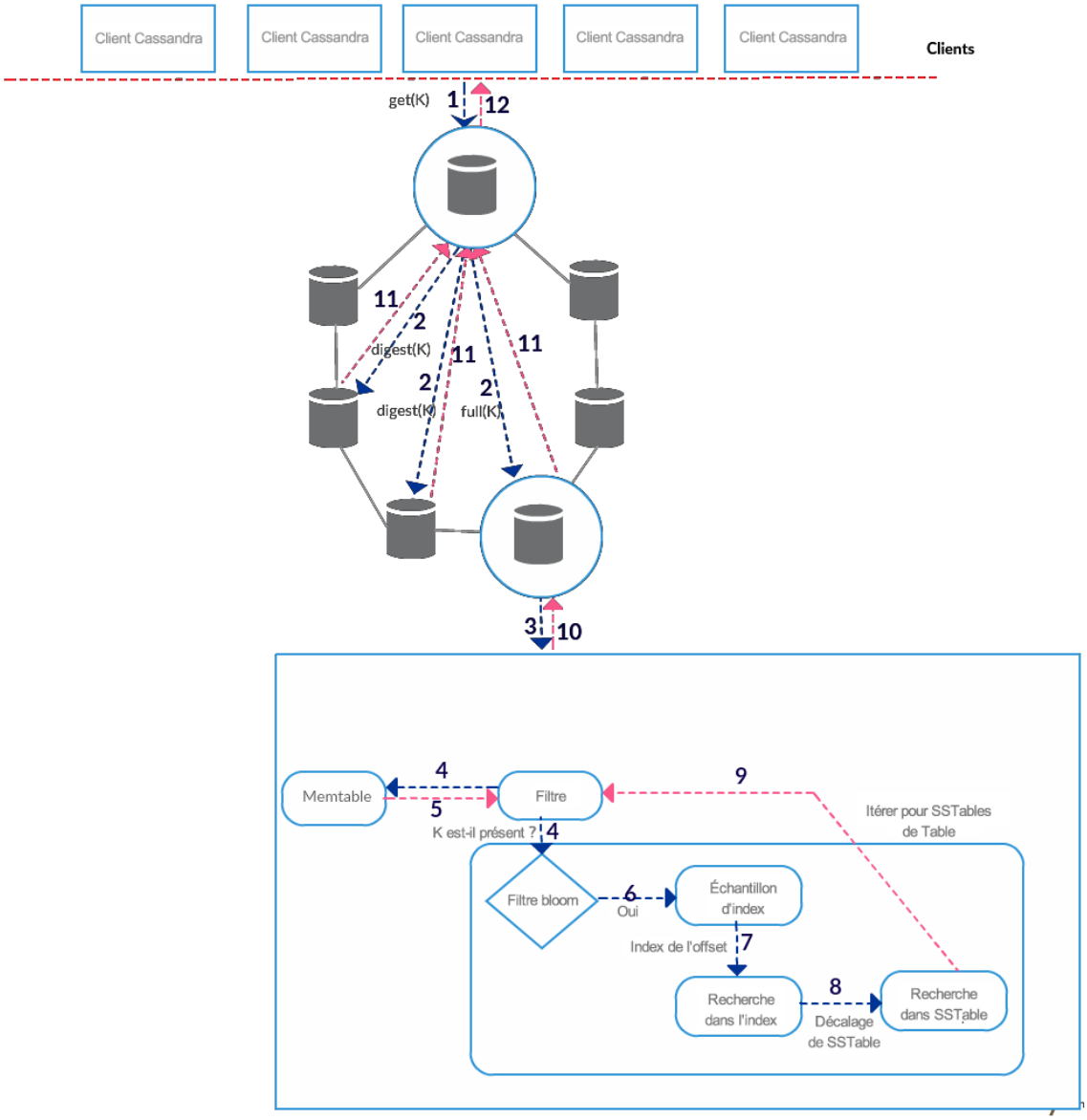

Figure 2 : Flux de demande de lecture

1. Un client envoie une demande à un nœud dans le cluster pour obtenir des données pour une clé donnée, K. À ce stade, si la clé n'est pas mappée sur ce nœud, alors le nœud sert de coordinateur. Note that a node can either act as a replica or a coordinator or both (if the node maps to the data and is talking to the client).

2. Le coordinateur détermine les nœuds de réplica qui peuvent contenir la clé et transmet la demande à ces nœuds. En envoyant la demande aux nœuds de réplica, le coordinateur détermine quel nœud est le plus proche de lui-même (via un snitch) et envoie une demande pour données complètes au nœud le plus proche et une demande pour la valeur de hachage générée avec le code de hachage des données des autres nœuds. (Un *snitch* détermine quel hôte est le plus proche de l'emplacement actuel.)
3. La demande est transmise aux services internes du nœud pour poursuivre le traitement.
4. Une demande de données à la fois du memtable et des SStables est réalisée. La demande se poursuit sur les filtres bloom pour les SStables, demandant si la clé est présente.
5. Du fait que le memtable est en mémoire, les données peuvent être renvoyées plus rapidement depuis le memtable, mais une ou plusieurs SStables doivent encore être consultées pour les données.
6. Si un filtre bloom répond que la clé n'est pas présente, le filtre bloom suivant est vérifié. Si un filtre bloom répond que la clé peut être présente (ce qui est le cas ici), il vérifie alors l'échantillon d'index en mémoire.
7. Une recherche binaire est réalisée sur l'échantillon d'index afin de déterminer un décalage de début dans le fichier d'index effectif. Ce décalage est utilisé pour créer un décalage dans le fichier d'index et procéder à une opération de lecture séquentielle pour obtenir le décalage dans la SStable pour la clé effective.
8. Avec le décalage obtenu depuis l'étape 7, les données effectives de la SStable sont renvoyées via un décalage dans le fichier SStable.
9. Les données pour la clé sont renvoyées depuis la recherche SStable. La commande de filtre regroupe toutes les versions des données clés obtenues depuis les recherches SStable et le memtable.
10. La dernière version consolidée des données clés est renvoyée aux services internes.
11. Le même processus est répété par les services internes sur d'autres nœuds et des résultats sont renvoyés au nœud du coordinateur.

12. Le coordinateur compare la valeur de hachage obtenue de tous les nœuds et détermine s'il y a un conflit dans les données. En cas de conflit, le coordinateur rapproche les données et renvoie la version rapprochée au client. Une réparation en lecture est également initiée pour assurer la cohérence des données.

Veuillez noter que nous n'avons pas mentionné le cache auparavant. Pour plus d'informations sur la mise en cache, reportez-vous à la documentation DataStax.[6]

Les réparations en lecture peuvent résoudre des incohérences de données lorsque les données écrites sont lues. Mais lorsque les données écrites ne sont pas lues, vous pouvez uniquement utiliser soit le transfert avec indicateur[7] , soit le mécanisme anti-entropie[8].

Cassandra : demandes de ressources

Étudions à présent les ressources requises pour exécuter Cassandra. Nous consulterons le stockage et les exigences de mise en réseau, de mémoire, d'UC et d'E/S.

Storage and IO Requirements

La plupart des E/S se produisant dans Cassandra sont séquentielles. Mais dans certains cas vous avez besoin d'E/S aléatoires. La lecture de SStables pendant des opérations de lecture en est un exemple.

SSD est le mécanisme de stockage recommandé pour Cassandra, car il fournit des temps de réponse à faible latence extrême pour des opérations de lecture aléatoires tout en proposant une ample performance d'écriture séquentielle pour des opérations de compactage.

La réplication et le stockage supplémentaire suite au compactage doivent être pris en considération lors de la détermination des exigences de stockage.

Le système de fichiers recommandé pour tous les volumes est XFS. Il est préférable d'utiliser Ext4. Ext3 est considérablement plus lent et nous vous recommandons de l'éviter.

AWS fournit deux types d'options de stockage, à savoir le stockage local et Amazon Elastic Block Store (Amazon EBS). Le stockage local est disponible localement pour l'instance, et EBS est un périphérique de stockage NAS. Nous aborderons plus en détails le choix d'une option de stockage sur AWS pour Cassandra ultérieurement dans ce livre blanc.

Exigences réseau

Cassandra utilise le protocole Gossip pour échanger des informations avec d'autres nœuds sur la topologie de réseau. L'utilisation de Gossip, associée à la nature distribuée de Cassandra, qui implique de parler à plusieurs nœuds pour les opérations de lecture et d'écriture, entraîne un volume important de transfert de données via le réseau.

Cela étant le cas, nous recommandons de toujours choisir des instances avec au moins 1 Gbit/s de bande passante réseau pour tenir compte de la réplication et de Gossip. Lors de l'utilisation d'AWS, nous recommandons de choisir des types d'instance avec la mise en réseau améliorée activée. La mise en réseau améliorée offre de meilleures performances réseau. Nous aborderons plus en détails cette option et ses avantages ultérieurement dans ce livre blanc.

Exigences mémoire

Cassandra s'exécute fondamentalement sur une machine virtuelle Java (JVM). La taille de la JVM doit être adaptée pour la performance. De grands segments de mémoire peuvent introduire des pauses de nettoyage de la mémoire (GC) pouvant entraîner une latence, voire afficher un nœud Cassandra comme étant hors ligne. Un réglage adéquat du segment de mémoire peut minimiser l'impact de GC dans la JVM.

Le paramètre *MAX_HEAP_SIZE* détermine la taille de segment de mémoire de la JVM Cassandra. DataStax recommande de ne pas allouer plus de 8 Go pour le segment de mémoire.

Le paramètre *HEAP_NEW_SIZE* correspond à la taille de la jeune génération dans Java. Une règle de base générale consiste à définir cette valeur sur 100 Mo par vCPU sur Amazon EC2.

Cassandra s'appuie également largement sur le cache du fichier du système d'exploitation pour la performance de lecture. Ainsi, le choix d'une taille optimale de segment de mémoire JVM et le maintien d'un volume de mémoire suffisant pour le cache du fichier du système d'exploitation sont importants. Pour les charges de travail de production, notre recommandation générale consiste à choisir un type d'instance avec au moins 32 Go de DRAM.

Plus d'informations sur le réglage de JVM dans la documentation DataStax.[9]

Exigences UC

Lors de l'analyse des exigences d'UC Cassandra, il est utile de noter que les charges de travail à forte densité d'insertions sont liées à l'UC dans Cassandra avant de devenir liées à l'E/S. En d'autres termes, toutes les opérations en écriture intègrent le journal de validation, mais Cassandra est tellement efficace en écriture, que l'UC devient le facteur de limitation. Cassandra est extrêmement simultanée et utilise autant de cœurs de processeur qu'il y en a de disponibles.

Lors du choix de types d'instance avec Amazon EC2 pour les charges de travail à forte densité d'écritures, vous devez rechercher des types d'instance avec au moins 4 vCPU. Bien que ce soit un bon point de départ, nous vous recommandons de tester une charge de travail représentative avant de définir le type d'instance pour la production.

Planification de clusters Cassandra sur AWS

Cette section discute de la manière dont vous pouvez appliquer des fonctionnalités Cassandra à des services AWS pour déployer Cassandra de la manière la plus optimale et efficace possible.

Planning Regions and Availability Zones

L'infrastructure du cloud AWS est basée sur les régions et les zones de disponibilité (« AZ »). [10]Une *Région* est un emplacement physique dans le monde où nous avons plusieurs zones de disponibilité. Les *zones de disponibilité* se composent d'un ou de plusieurs centres de données distincts, chacun avec une connectivité, une mise en réseau et une alimentation redondantes, hébergées dans des sites séparés. Ces zones de disponibilités vous offrent la capacité d'opérer des bases de données et des applications de production qui sont plus largement disponibles, à tolérance de panne et évolutives que ne pourrait le proposer un centre de données unique *(en savoir plus)*. Cela aide également les clients à implémenter une expansion géographique et une conformité réglementaire.

Vous pouvez utiliser l'infrastructure AWS pour gérer la latence du réseau, ainsi que vos besoins en matière de conformité réglementaire. Par exemple, les données dans une région ne sont pas automatiquement répliquées en dehors de cette région. Si votre activité a besoin de davantage de disponibilité, il vous incombe de répliquer les données dans les différentes régions.

Lorsque vous développez votre cluster Cassandra, sélectionnez la même région pour vos données et applications afin de minimiser la latence des applications. Pour obtenir des détails sur l'emplacement exact des régions AWS, consultez la Region Table. [11]

Contrairement aux architectures héritées maître/subordonné, Cassandra est dotée d'une architecture sans maître dans laquelle tous les nœuds jouent un rôle identique afin qu'il n'y ait aucun point de défaillance unique. Envisagez de diffuser les nœuds Cassandra sur plusieurs zones de disponibilité afin d'activer une haute disponibilité. En diffusant les nœuds entre des zones de disponibilité, en cas de sinistre, vous pouvez maintenir la disponibilité et les temps d'activité.

Les clusters Cassandra peuvent être dotés d'une prise en charge Amazon EC2 et donc prendre en charge la haute disponibilité en définissant un paramètre snitch adapté via le paramètre *endpoint_snitch* dans cassandra.yaml.[12] Ce paramètre peut être défini sur Ec2Snitch[13] ou sur Ec2MultiRegionSnitch.[14] Ec2Snitch vous permet de développer des clusters dans une seule région, et Ec2MultiRegionSnitch vous permet de déployer des clusters dans plusieurs régions AWS.

Conseil de base
Si vous démarrez tout juste avec Cassandra, prévoyez la croissance de cluster depuis le début. Choisissez Ec2MultiRegionSnitch pour éviter des complications lorsque vous décidez de développer votre cluster.

La configuration d'un snitch permet à Cassandra de placer les réplicas pour les partitions de données sur des nœuds qui sont dans différentes zones de disponibilité. Ainsi, si votre espace de clés a un facteur de réplication de 3 dans la région USA Est (Virginie du Nord), us-east-1, alors le paramètre Ec2Snitch permet la réplication de vos données sur les zones de disponibilité dans us- east-1. Par exemple, si vous configurez un cluster dans les zones de disponibilité us-east-1a, us-east-1b et us-east-1c, et que vous définissez un facteur de réplication de 3 au niveau de l'espace de clés, alors chaque opération d'écriture sera répliquée sur les nœuds dans ces trois zones de disponibilité.

Le paramètre snitch fournit alors une meilleure disponibilité en permettant à Cassandra de placer des réplicas de vos données dans différentes zones de disponibilité lorsqu'une opération d'écriture a lieu.

Cassandra a également des *nœuds initiaux,* qui sont à l'origine consultés par un nouveau nœud qui souhaite amorcer et joindre l'anneau Cassandra. En l'absence d'un nœud initial, l'amorçage ne se produit pas et le nœud ne démarre pas. Nous vous recommandons fortement de diffuser vos nœuds initiaux sur plusieurs zones de disponibilité. Dans cette situation, même si une zone de disponibilité connaît une défaillance, les nouveaux nœuds peuvent s'amorcer depuis les nœuds initiaux d'une autre zone de disponibilité.

Les clusters Cassandra peuvent également jouir d'une prise en charge centre de données et rack. « Centre de données » en terminologie Cassandra se traduit par « région » en termes AWS, et « rack » se traduit par « zone de disponibilité » en termes AWS.

Les clusters Cassandra peuvent être conçus pour être hautement résistants aux défaillances en optimisant des régions et des zones de disponibilité dans AWS associées à des snitches EC2 qui sont fournis avec le logiciel. Par exemple, avec une conception trois zones de disponibilité, les demandes peuvent continuer à aboutir en cas de panne d'une zone de disponibilité entière. Vous pouvez créer des sauvegardes en direct de votre cluster en concevant un cluster dans une autre région AWS et laisser Cassandra gérer la réplication asynchrone.

Planning an Amazon Virtual Private Cloud

Amazon Virtual Private Cloud (Amazon VPC) vous permet de mettre en service une section du cloud AWS qui a été isolée de manière logique et dans laquelle vous pouvez lancer des ressources AWS dans un réseau virtuel que vous définissez. Vous conservez la totale maîtrise de votre environnement réseau virtuel, y compris pour la sélection de votre propre plage d'adresses IP, la création de sous-réseaux et la configuration de tables de routage et de passerelles réseau.

Nous vous recommandons fortement de lancer votre cluster Cassandra dans un VPC. Un des principaux avantages que VPC offre pour les charges de travail Cassandra est la fonction de mise en réseau avancée. L'activation de la mise en réseau améliorée sur votre instance se traduit par des performances (plus de paquets par seconde) plus élevées, ainsi qu'une latence et une instabilité réduites. Actuellement, C3, C4, D2, I2, M4 et R3 sont les familles d'instance pour lesquelles la mise en réseau améliorée est prise en charge dans un VPC.[15] Afin d'activer cette fonctionnalité, vous devez également lancer une AMI HVM avec le pilote SR-IOV adéquat.

Meilleures pratiques pour la configuration de Cassandra sur VPC

Commencez par créer un VPC suffisamment grand avec un bloc /16 de routage CIDR (Classless InterDomain Routing) (par exemple, 10.0.0.0/16) pour accueillir un nombre suffisant d'instances dans un seul VPC. Cette approche n'impacte en aucun cas la performance, et elle vous offre suffisamment de place pour mettre à l'échelle, si nécessaire. Si vous créez à la place un VPC plus petit, par exemple 10.0.0.0/28, qui a jusqu'à 14 adresses IP, vous devez alors créer un VPC. Les clusters Cassandra évoluent avec les données en augmentant la capacité, et disposer d'un VPC suffisamment grand facilitera la mise à l'échelle.

Vous pouvez définir des sous-réseaux publics et privés dans un VPC. Nous vous recommandons d'héberger vos clusters Cassandra dans un sous-réseau privé de votre VPC qui n'a pas d'accès Internet. Vous pouvez alors configurer une instance NAT (traduction d'adresses réseau) dans un sous-réseau public pour permettre aux nœuds de cluster de communiquer avec l'Internet concernant les mises à jour logicielles.

Du fait que les sous-réseaux ne s'étendent pas sur les zones de disponibilité, prévoyez de créer plusieurs sous-réseaux, en fonction du facteur de réplication que vous envisagez de configurer pour votre espace de clés. Cassandra [16] Par exemple, si votre facteur de réplication est 3, vous pouvez créer trois sous-réseaux, chacun dans une zone de disponibilité différente, pour héberger votre cluster. Un autre élément à prendre en considération dans la planification des sous-réseaux pour votre cluster Cassandra est le fait qu'Amazon réserve les quatre premières adresses IP et la dernière adresse IP de chaque sous-réseau à des fins de mise en réseau d'IP.

Si vous avez des administrateurs qui ont besoin d'un accès Secure Shell (SSH) aux nœuds dans le cluster à des fins de maintenance et d'administration, la pratique standard du secteur consiste à configurer un hôte bastion.

Si vous avez déjà un cluster Cassandra avec AWS, mais sur la plateforme EC2-Classic , nous vous recommandons de migrer le cluster sur VPC pour optimiser les meilleures fonctions de performance et le niveau de sécurité amélioré le plus élevé proposés par VPC. Pour simplifier cette migration, AWS propose ClassicLink, qui vous permet d'activer la communication entre EC2-Classic et VPC et de migrer de façon incrémentielle votre cluster Cassandra sur VPC sans temps d'arrêt.[17] Nous vous donnerons plus d'informations sur cette fonction dans la section Migration.

Planning Elastic Network Interfaces

Une *Elastic Network Interface (ENI)* est une interface réseau virtuelle que vous pouvez attacher à une instance dans un VPC dans une zone de disponibilité unique. Vous pouvez créer une ENI, l'attacher à une instance, la détacher de cette instance, puis l'attacher à une autre instance dans la même zone de disponibilité. Lorsque vous faites cela, les attributs de l'ENI suivent au fil des détachements et rattachements. Lorsque vous déplacez une ENI d'une instance vers une autre, le trafic réseau est redirigé vers la nouvelle instance. Le nombre maximum d'ENI que vous pouvez attacher par instance dépend du type d'instance, comme expliqué dans Elastic Network Interfaces (ENI) dans le guide *Amazon EC2 User Guide*.[18]

Lors de l'utilisation de Cassandra, les ENI sont généralement utilisées pour prendre en charge la configuration de nœud initial. Les adresses IP de nœud initial sont codées en dur dans le fichier de configuration Cassandra .yaml. En cas de défaillance d'un nœud initial, la configuration yaml sur chaque nœud dans le cluster doit être mise à jour avec l'adresse IP du nouveau nœud initial qui est proposé à la place de celui qui a échoué. Cela pourrait créer des difficultés opérationnelles avec les clusters de grande taille.

Vous pouvez néanmoins éviter ce scénario si vous attachez une ENI à chaque nœud initial et que vous ajoutez l'adresse IP ENI à la liste de nœuds initiaux dans le fichier de configuration .yaml. Si vous procédez de cette manière, en cas de défaillance d'un nœud initial, vous pouvez automatiser de manière à ce que le nouveau nœud initial remplace par programmation l'adresse IP ENI. L'automatisation implique normalement l'utilisation de l'interface AWS CLI et l'exécution de commandes spécifiques ENI pour procéder au détachement et à l'attachement.

Veuillez noter que l'attachement d'une ENI à une instance EC2 n'augmente pas la bande passante réseau disponible pour l'instance.

Planning High-Performance Storage Options

Il est essentiel de comprendre le nombre total d'E/S requis pour sélectionner une configuration de stockage adéquate sur AWS. La plupart des E/S gérées par Cassandra sont séquentielles pour une charge de travail à forte densité d'écritures. Cependant, les charges de travail à forte densité de lectures nécessitent un accès aléatoire et si vos plages de travail ne tiennent pas dans la mémoire, votre configuration aura sûrement une liaison E/S à un moment. Ainsi, il est important de bien peser le choix initial de votre stockage.

AWS offre deux choix principaux pour construire la couche de stockage de votre infrastructure Cassandra : volumes Amazon EBS et stockages d'instances Amazon EC2.

Volumes Amazon EBS (Elastic Block Store)

Amazon EBS fournit des volumes de stockage permanent au niveau bloc à utiliser avec les instances Amazon EC2 dans le cloud AWS. Chaque volume Amazon EBS est automatiquement répliqué au sein de sa Zone de Disponibilité, afin de vous protéger contre toute défaillance de composants, tout en garantissant une disponibilité et une durabilité élevées.

Les volumes Amazon EBS offrent les performances homogènes, à faible latence, nécessaires pour exécuter vos charges de travail. Les volumes Amazon EBS fournissent une excellente conception pour les systèmes qui ont besoin d'une variabilité de la performance de stockage.

Il existe deux type de volumes Amazon EBS que vous devez envisager pour les déploiements Cassandra :

- Les volumes à usage général (SSD) offrent des latences en millisecondes à chiffre unique, fournissent des performances de référence cohérentes de 3 IOPS/Go à un maximum de 10 000 IOPS, et fournissent jusqu'à 160 Mo/s de débit par volume.[19]

- Les volumes IOPS provisionnées (SSD) offrent des latences en millisecondes à chiffre unique, fournissent des performances de référence cohérentes de 3 IOPS/Go à un maximum de 20 000 IOPS, et fournissent jusqu'à 320 Mo/s de débit par volume. Ces fonctions permettent de prévoir bien plus facilement la performance attendue d'une configuration système.[20]

DataStax recommande ces deux types de volume EBS pour les charges de travail Cassandra. Pour plus d'informations, consultez la documentation DataStax.[21] Les types de volume magnétique ne sont généralement pas recommandés pour des raisons de performance.

Au minimum, l'utilisation d'un volume EBS unique sur une instance Amazon EC2 peut atteindre 10 000 IOPS ou 20 000 IOPS depuis le stockage sous-jacent, en fonction du type de volume (GP2 ou PIOPS). Pour optimiser la performance, utilisez des instances optimisées Amazon EBS.[22]

Conseil
DataStax recommande les types de volume EBS GP2 et IOPS provisionnées pour les charges de travail Cassandra. Pour plus d'informations, consultez la documentation DataStax

Une instance optimisée Amazon EBS a recours à une pile de configuration optimisée et fournit une capacité supplémentaire dédiée pour les E/S Amazon EBS. Cette optimisation offre les meilleures performances pour vos volumes EBS en réduisant les conflits entre les E/S Amazon EBS et le trafic restant de votre instance.

Les instances optimisées pour EBS offrent un débit dédié entre Amazon EC2 et Amazon EBS, avec différentes options comprises entre 500 et 4 000 Mbits/s, selon le type d'instance utilisé. Par exemple, une instance optimisée EBS dans une configuration Cassandra peut fournir un volume de données à 4 To et 10 000 IOPS avec un journal de validation à 1 To et 3 000 IOPS.

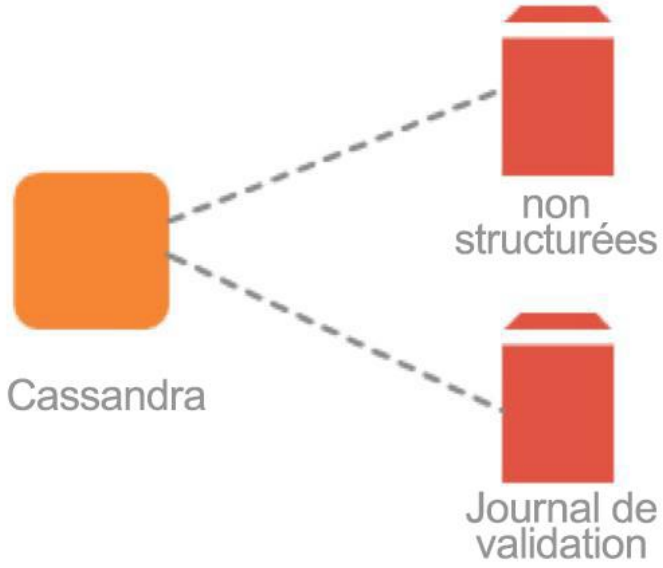

Figure 3 : volume EBS unique

Pour poursuivre la mise à l'échelle de l'IOPS au-delà de ce qu'offre un volume unique, vous pouvez utiliser plusieurs volumes EBS, comme indiqué ci-après.

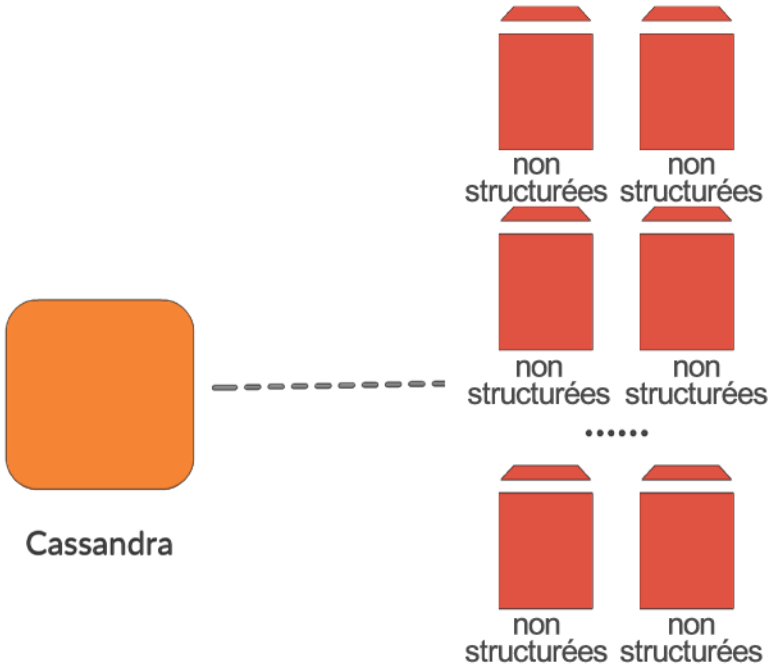

Figure 4 : plusieurs volumes EBS pour données

Vous pouvez choisir entre plusieurs combinaisons de tailles de volumes et d'IOPS, mais n'oubliez pas d'optimiser en fonction du nombre maximum d'IOPS pris en charge par l'instance.

Dans la configuration qui vient d'être présentée, vous voudrez peut-être attacher des volumes avec des IOPS combinées au-delà des IOPS proposées par l'instance EC2 optimisée pour EBS. Par exemple, un volume IOPS provisionnées (SSD) avec 16 000 IOPS ou deux volumes à usage général (SSD) avec 8 000 IOPS agrégé par bandes ensemble atteignent les 16 000 IOPS offertes par une instance c4.4xlarge via sa bande passante dédiée optimisée EBS.

Les instances avec un réseau 10 Gb/s et une mise en réseau améliorée peuvent fournir jusqu'à 48 000 IOPS et 800 Mo/s de débit sur des volumes Amazon EBS. Par exemple, avec ces instances, cinq volumes à usage général (SSD) de 10 000 IOPS chacun peuvent saturer le lien vers Amazon EBS.

Vous devez également noter que les connexions optimisées EBS sont en duplex intégral et peuvent donner lieu à plus de débit et d'IOPS dans une charge de travail 50/50 lecture/écriture où les deux voies de communication sont utilisées.

Conseil de meilleure pratique

La meilleure pratique générale lors de l'utilisation de volumes EBS pour le stockage consiste à utiliser des volumes distincts pour les données et le journal de validation. Cette approche vous permet d'optimiser la mise à l'échelle.

La meilleure pratique lors de l'utilisation de volumes EBS pour Cassandra consiste à utiliser des volumes distincts pour les répertoires de données et le journal de validation. This approach will allow you to scale better.

Amazon EBS fournit également une fonction pour la sauvegarde des données sur vos volumes EBS vers Amazon S3 en prenant des instantanés à un moment donné.

Stockages d'instance Amazon EC2

De nombreux types d'instances Amazon EC2 ont accès au stockage sur disque situé sur des disques qui sont attachés physiquement à l'ordinateur hôte. Ce stockage sur disque est appelé un *stockage d'instance*. Si vous utilisez un stockage d'instance sur des instances qui exposent plus d'un volume, vous pouvez agréger en bandes les volumes de stockage d'instance (en utilisant RAID0) pour améliorer le débit E/S. N'oubliez pas que si l'instance est arrêtée, connaît une défaillance ou est résiliée, vous perdrez toutes vos données. Par conséquent, nous vous recommandons fortement de configurer une réplication sur plusieurs instances sur des zones de disponibilité pour votre cluster Cassandra.

Lors de l'utilisation d'un gestionnaire de volume logique (par exemple, mdadm ou LVM), assurez-vous que toutes les métadonnées et données sont cohérentes lorsque vous réalisez la sauvegarde.

Planning Instance Types Based on Storage Needs

Avant d'aller aux recommandations de type d'instance, permettez-nous d'étudier au préalable un exemple afin de comprendre le stockage et de calculer les exigences pour gérer une charge de travail Cassandra. Supposons, pour faire simple, que nous ayons une seule table (famille de colonnes) pour un cluster.

Hypothèses et formule de stockage

Pour cet exemple, nous émettrons les hypothèses suivantes :

- 100 colonnes par ligne.
- 10 000 opérations d'écriture par seconde et 500 opérations de lecture par seconde (95 pour cent d'écritures et 5 pour cent de lectures).
- Pour faire simple, nous supposerons également que les lectures et les écritures par seconde sont uniformes tout au long de la journée.
- La taille totale de chaque valeur et nom de colonne est de 30 octets.
- Nous stockons des données de série chronologique, et par conséquent les colonnes sont toutes définies pour arriver à expiration. Nous définissons l'expiration sur 1 mois parce que nous n'avons pas besoin de données de plus d'un mois dans notre table.
- La taille moyenne d'une clé primaire est de 10 octets.

- Le facteur de réplication est 3.
- La stratégie de compactage est par niveau de taille (supplément de stockage de 50 pour cent).

Calculons les exigences de stockage pour un mois pour cet exemple :

```
Storage requirement = (((Number_of_columns *
(column_name_size
+ column_value_size + 23)) + 23) * Number_of_rows +
Number_of_rows * (32 + primary_key_size)) *
Replication_Factor
* (1 + Compaction_Overhead)/1024/1024/1024/1024 TB
```

Cette formule peut en général être appliquée pour calculer vos exigences de stockage au niveau d'une famille de colonnes.

Veuillez également noter ces points importants :

- Le supplément de stockage par colonne est de 23 octets pour une colonne qui expire, de la même manière qu'avec les séries chronologiques.
- Le supplément de stockage par ligne est de 23 octets.
- Le supplément de stockage par clé pour la clé primaire est de 32 octets.
- Pour une colonne normale, le supplément de stockage par ligne est de 15 octets.
- Le nombre total de lignes écrites par mois est le suivant :
  `10,000*86400*30 = 25920000000`

Voici nos variables :

- `Number_of_columns = 100`
- `column_name_size = 20`
- `column_value_size = 10`
- `Number_of_rows = 25920000000`
- `Primary_key_size = 10`
- `Replication_Factor = 3`
- `Compaction_Overhead = 50 percent(0.5)`

L'application de la formule précédente avec les valeurs de notre exemple susmentionné aboutit à une exigence de stockage de 569 To. Ainsi, nous aurons besoin d'au moins 569 To d'espace disque pour alimenter ce cluster.

Comparons à présent nos options disponibles sur AWS pour satisfaire cette exigence.

Utilisation d'instances I2

Le type d'instance le plus couramment utilisé par les clients pour Cassandra sur AWS est l'instance I2. Les instances I2 avec niveau E/S élevé sont optimisées pour fournir des dizaines de milliers d'IOPS aléatoires à faible latence aux applications. Le type d'instance I2 offre suffisamment de disque, d'IOPS et de mémoire pour vous permettre d'exécuter une charge de travail faible latence pour les deux opérations : lecture et écriture. I2 prend également en charge la mise en réseau améliorée. Ce type d'instance est généralement plus fréquemment utilisé en présence de volumes de données (en téraoctets) et d'IOPS (multiple de dizaines de milliers) importants.

En étudiant les spécifications I2, en supposant l'utilisation de la région USA Est (Virginie du Nord) us- east-1 et en prévoyant un supplément de 10 pour cent pour le formatage de disque, nous pouvons déterminer que pour le stockage dont nous avons parlé précédemment, nous aurons besoin de 405 instances I2 (`569/((2*800*.9)/1024)`).[23] Cette charge de travail vous coûtera 497 178 $ (`405*1.705*720`) pour un mois d'exécution. Ce chiffre n'inclut pas de frais de transfert de données, qui sont facturés séparément. À des fins de simplicité, nous n'incluons pas non plus tout calcul de journal de validation, et nous supposons que les données et le journal de validation seront colocalisés sur les mêmes disques.

Avons-nous cependant réellement besoin de 405 nœuds pour traiter 10 000 opérations d'écriture et 500 opérations de lecture par seconde sur l'I2 ? La réponse pourrait être négative. En fait, nous suralimentons sans doute chaque ressource de calcul à notre disposition, hors stockage local. Nous pouvons passer à une instance I2 de plus grande taille avec davantage de stockage local. Mais cela pourrait ne pas être bénéfique en termes de coûts. Quelle autre option pourrait fonctionner dans ce cas ?

Si nous pouvons d'une manière ou d'une autre découpler du stockage du calcul, nous disposerons alors d'un scénario viable nous permettant de fournir un stockage optimal et de calculer la puissance en fonction des exigences. Nous pouvons réaliser ce découplage avec EBS. L'idée est ici d'utiliser EBS pour atteindre un niveau de performance identique ou équivalent, à moindre coût.

Découplage avec EBS

Pour réaliser ce découplage, nous pouvons fournir des volumes EBS GP2, ce qui peut nous donner la fourniture de 3 IOPS/Go. Suivons les recommandations DataStax concernant le volume maximum de données par nœud pour Cassandra (3 à 5 To) et affectons 4 To de volumes EBS GP2 par nœud. Cette configuration nous donnera 10 000 IOPS.

Nous devons également allouer du stockage pour nos journaux de validation. En effet, les tenir à jour sur des volumes distincts fournit de meilleures performances. Nous allouerons 500 Go de volume pour le journal de validation sur chaque nœud. Cette allocation fournira la garantie d'environ 1 500 IOPS, avec une capacité de rafale pouvant atteindre 3 000 IOPS pendant 30 minutes, selon le nombre d'IOPS sauvegardées auparavant.

Nous devons à présent choisir un type d'instance qui permette une mise en réseau améliorée et une bande passante optimisée EBS. Il y a des frais supplémentaires pour des instances optimisées EBS sur de nombreux types d'instance, mais AWS a récemment lancé de nouveaux types d'instance qui sont optimisés EBS par défaut, sans coût supplémentaire pour la bande passante optimisée EBS.

Utilisation d'instances C4 avec EBS

L'instance C4 devient un choix répandu pour l'exécution de Cassandra sur EBS. Pour voir un exemple, consultez la présentation de Crowd Strike de re:Invent 2015.[24] C4 n'a pas de stockage local et par conséquent EBS doit être attaché pour le stockage. Dans ce cas spécifique, où nous avons 95 pour cent d'opérations d'écriture, C4 peut tout à fait convenir car votre UC arrive normalement à saturation avant que vous n'ayez plus d'E/S pour ces charges de travail faussées en écriture.

Nous aurons également besoin d'un type d'instance dans C4 qui prenne en charge au moins 3 000 IOPS pour le débit d'écriture normal, plus les IOPS de compactage via sa bande passante optimisée EBS. N'oubliez pas que nous recommandons également une configuration minimale requise pour la mémoire de 32 Go de DRAM pour les charges de travail de production. C4.4X convient aux exigences UC, mémoire et IOPS comme décrit.

Si nous tarifons cette option, nous avons besoin de 143 instances C4.4X (4 To par nœud pour un total de 569 To). Nous obtiendrons 572 To (`143*4` To).

Nous avons à présent besoin de 500 Go de journaux de validation par nœud. Ainsi, nous aurons besoin de 72 To de GP2 simplement pour héberger des journaux de validation. En ajoutant à la fois de données et de journaux de validation (572 To + 72 To), nous devrons fournir 644 To pour EBS GP2.

En supposant que la région USA Est (Virginie du Nord) us-east-1 soit sélectionnée, et en utilisant la tarification actuelle, le calcul de coûts EC2 pour 143 instances C4.4XL fournit le résultat suivant :[25]

```
143 C4.4X = 143*0,838*720 = 86 280,48 USD
```

Le calcul des coûts de volume EBS fournit le résultat suivant :

```
644 To EBS GP2 = 65 946 USD
```

Notre coût total correspond au coût d'EC2 plus le coût d'EBS, soit 152 226 USD.

La comparaison de ce résultat avec le coût d'exécution d'I2, 497 178 USD, nous permet de découvrir que l'option C4.4X plus EBS est environ 3,2 fois moins chère en comparaison de l'option I2, sans compromettre la performance et la fiabilité.

Optimisation de coût pour C4 plus EBS et pour I2

Lorsque vous choisissez des types d'instance pour votre cluster Cassandra, AWS propose des options de tarification telles que des instances réservées afin d'optimiser vos coûts.

Par exemple, imaginons que vous décidiez d'exécuter la configuration C4.4X plus EBS pour votre cluster et que vous soyez satisfait de la performance du cluster sur deux mois. À ce stade, avec un choix d'instance stable pour votre cluster, il peut être judicieux de faire des réservations et d'optimiser les coûts. Si vous décidez de faire une réservation initiale partielle sur trois ans, vous économiserez 59 pour cent sur le prix à la demande.

Voici le calcul revu :

```
143 C4.4X = 143*0,3476*720 = 35 788,89 USD

644 TB EBS GP2 = $65,946
```

Notre coût total est à présent de 101 734,89 USD. Ainsi, vous pouvez faire baisser votre coût de 152 226 USD à 101 734,89 USD avec une réservation sur trois ans.

À présent que nous connaissons le meilleur coût optimisé pour la configuration C4.4X plus EBS GP2, revisitions l'option I2 et appliquons une tarification réservée pour voir dans quelle mesure nous pouvons nous en approcher.

Avec une réservation initiale partielle sur trois ans sur I2, vous pouvez économiser 74 pour cent par rapport au prix à la demande. Cela signifie que vous ne payez que 26 pour cent du coût à la demande. Le coût chute à $129,266 (`$497,178/4`). À présent, la comparaison entre l'option C4 et l'option I2 ne semble pas trop mauvaise, mais l'option C4 est encore environ 1,2 fois moins chère que l'option I2.

Résumé d'exemple de charge de travail

Écritures/s	Lectures/s	Facteur de réplication	Stockage (To)	Taille de ligne moyenne (Ko)	Nombre d'instances/ Type	Coût
10000	500	3	570	3	405/I2	$497,178
10000	500	3	570	3	405/I2/3-Yr RI	$129,266
10000	500	3	644	3	143/C4.4XL	$152,226
10000	500	3	644	3	143/C4.4XL/3-Yr RI	$101,735

Utilisation d'instances R3 ou I2 avec des charges de travail à forte densité de lectures

Si vous exécutez une charge de travail à forte densité de lectures, vous devez alors penser à utiliser la famille d'instances R3 avec ou sans stockage EBS, ou bien la famille d'instances I2. Pour sa performance de lecture, Cassandra dépend largement du cache du système de fichiers du système d'exploitation, et les instances R3 fournissent une bande passante mémoire élevée soutenue avec des instabilités et une latence réseau faibles.

Par défaut, un stockage local est attaché aux instances R3. Si vous découvrez que vos exigences de stockage sont suffisamment faibles après avoir pris en considération la réplication et le supplément de stockage, vous n'avez alors peut-être pas besoin d'EBS pour le stockage. Si vous prévoyez d'utiliser R3 avec EBS, vous devez noter que les types d'instance R3 ne sont pas optimisés EBS par défaut et qu'un coût horaire supplémentaire vient s'ajouter lorsque l'optimisation EBS est activée pour les instances R3.

À titre d'exemple d'utilisation d'instances R3, supposez que vous souhaitiez stocker un total de 1 To de données et que vous ayez une exigence élevée en matière de débit de lecture. D'après ces facteurs, vous pouvez fournir quatre r3.4xl, selon vos tests de charge. Cette approche vous coûte environ 4 000 USD par mois. Dans ce cas, vous ne voulez pas réduire le nombre de nœuds pour des raisons de disponibilité.

Et si vous n'avez pas besoin des niveaux élevés de mémoire et d'UC offerts avec r3.4xl mais que vous avez encore besoin de 1 To pour le stockage et que vous souhaitez encore optimiser ce coût, vous pouvez utiliser quatre nœuds i2.x et réduire le coût à environ 2 500 USD.

Pour fournir un autre exemple, supposez que vous souhaitiez stocker 6 To de données et que vous ayez une exigence élevée en matière de débit de lecture. Dans ce cas, vous pouvez fournir 20 instances r3.4xl, ce qui vous coûte environ 20 000 USD. Supposons cependant que vous avez réalisé des tests de charge et déterminé que du point de vue du calcul, six instances r3.4xl sont plus que suffisantes pour gérer votre charge de travail. Vous déterminez également que la majeure partie de la charge de travail est liée à la mémoire en raison du volume de DRAM disponible sur les instances r3.4x et que vous aurez besoin d'un maximum de 2 000 IOPS par instance. À ce stade, vous pouvez optimiser les EBS GP2 et fournir 1 To de GP2 par instance. Cela fait chuter le nombre de nœuds de 20 à 6 et réduit vos coûts à environ 7 000 USD, ce qui représente environ un tiers du coût d'utilisation du stockage local pour la même charge de travail.

En général, si vous cherchez à exécuter plus de 20 000 IOPS par nœud pour votre cluster Cassandra, les types d'instance I2 peuvent mieux convenir.

Montée en charge horizontale de Cassandra avec EBS

Il est intéressant de noter que si votre charge de travail a besoin de pousser la bande passante du disque sur la bande passante optimisées EBS maximum pour un type d'instance spécifique, vous devez optimiser la nature distribuée pouvant bénéficier d'une montée en charge horizontale de Cassandra et ajouter des nœuds pour étendre la charge. La diffusion de la charge de cette manière permet à votre cluster d'atteindre pratiquement n'importe quel montant d'IOPS tout en utilisant EBS à un coût raisonnable

Déploiement de Cassandra sur AWS

Cassandra fournit des fonctions de réplication native permettant une disponibilité élevée et peut bénéficier d'une montée en charge horizontale, comme le montre l'illustration suivante.

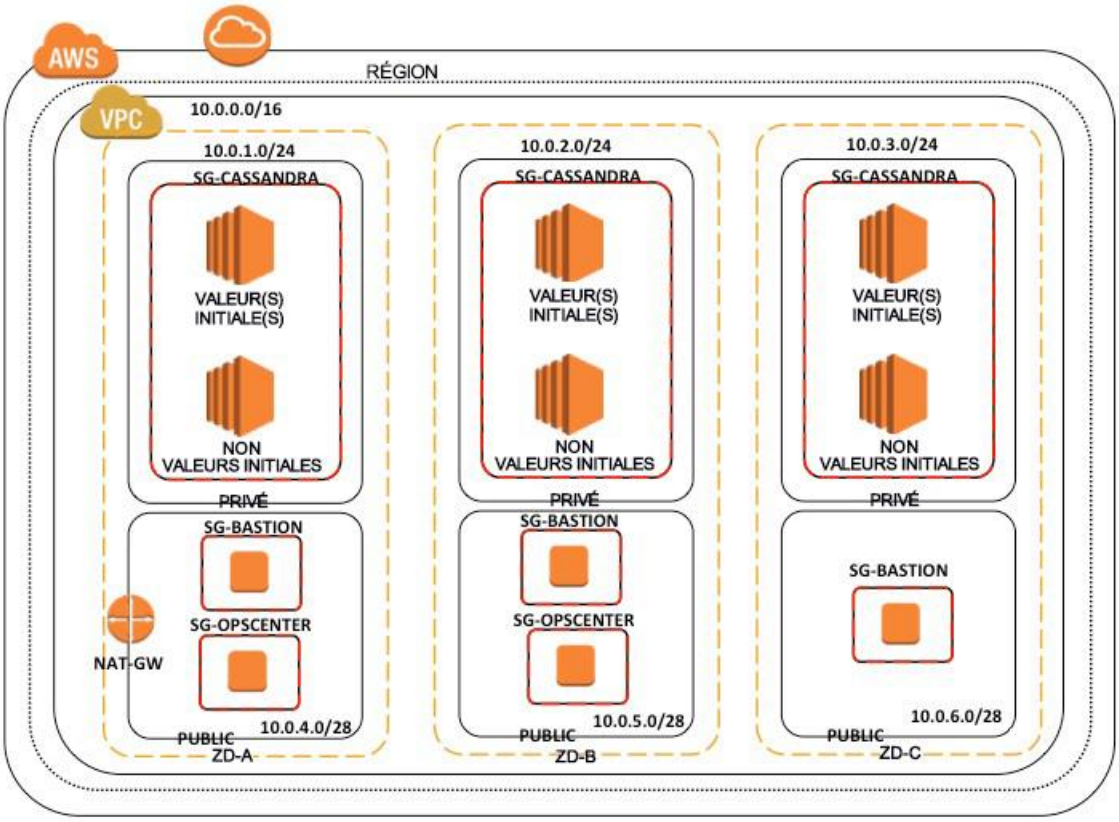

Figure 5 : cluster Cassandra hautement disponible

Bien que vous puissiez bénéficier d'une montée en charge verticale en utilisant des instances haute performance au lieu d'une topologie répliquée et partitionnée, les instances à montée en charge verticale ne fournissent pas les importants avantages de tolérance de pannes offerts par une topologie répliquée. Du fait qu'AWS dispose d'une réserve de ressources quasi illimitée, il est souvent préférable d'opter pour une montée en charge horizontale.

Setting Up High Availability

Le schéma d'architecture précédent présente une configuration haute disponibilité de base pour un cluster Cassandra. Nous recommandons fortement cette configuration pour une disponibilité élevée. Le cluster est dans un VPC de grande taille avec un bloc CIDR /16. Il y a trois sous-réseaux dans différentes zones de disponibilité. Les sous-réseaux privés ont chacun un bloc CIDR suffisamment grand (/24) pour accueillir les nœuds dans le cluster. Nous avons également trois sous-réseaux publics configurés. Ces sous-réseaux publics hébergent la NAT, les hôtes bastion et OpsCenter, qui est l'utilitaire Cassandra qui surveille un cluster Cassandra en cours d'exécution. La NAT est utile lorsque les nœuds du cluster dans le sous-réseau privé ont besoin de communiquer avec le monde extérieur pour des éléments tels que des mises à jour logicielles. Les hôtes bastion permettent un accès SSH à vos nœuds de cluster à des fins administratives. Pour fournir un contrôle d'accès, notre configuration de base associe également chaque couche à un groupe de sécurité.

Le schéma d'architecture montre les nœuds initiaux séparément afin d'insister sur le besoin de disposer d'au moins un nœud par zone de disponibilité. Nous recommandons néanmoins de ne pas rendre initiaux tous les nœuds, ce qui peut avoir un impact négatif sur votre performance en raison de l'utilisation de Gossip. La règle de base générale consiste à disposer d'un maximum de deux nœuds initiaux par zone de disponibilité.

Cassandra a plusieurs modes de cohérence pour les opération de lecture et d'écriture.[26] Dans un cluster avec un facteur de réplication de 3, l'opération LOCAL_QUORUM pour la lecture ou l'écriture permet à une opération de réussir lorsque deux réplicas sur trois dans une seule région signalent une réussite. Du fait que le cluster est diffusé sur trois zones de disponibilité, les opérations de lecture et d'écriture de votre quorum local demeurent disponibles, même pendant la survenue improbable d'une panne de zone de disponibilité. Elles restent également disponibles pendant une panne de nœud.

Dans notre cluster de base, les nœuds OpsCenter sont paramétrés dans une configuration de basculement. Il y a deux nœuds OpsCenter, qui se surveillent mutuellement sur les canaux stomp.[27] L'un d'eux est configuré comme nœud principal. En cas de panne sur le principal, le nœud de sauvegarde endosse le rôle du principal.

Jusqu'à ce stade, l'architecture décrite est manuelle. Si un nœud dans le cluster Cassandra échoue, vous devez le remplacer manuellement et configurer un nouveau nœud à la place. Si le nœud principal OpsCenter échoue, le nœud de sauvegarde prend le relais, mais vous devrez utiliser l'adresse IP ou l'adresse DNS (Domain Name System) du nouveau nœud. Vous devez également remplacer le nœud principal en panne. Si l'hôte bastion échoue, le remplacement doit être réalisé manuellement.

Automating This Setup

Vous pouvez automatiser la plupart de cette configuration dans AWS. Prenons une couche à la fois pour vous montrer les composants AWS qui peuvent être utilisés pour cette automatisation.

Commençons par le cluster Cassandra. Les étapes entreprises pour remplacer un nœud en panne dans Cassandra sont plutôt statiques et des actions manuelles avec étapes statiques peuvent être automatisées. Ainsi, si vous pouvez regrouper les étapes pour le remplacement d'un nœud mort, vous pouvez utiliser Auto Scaling pour automatiser ces étapes pour plusieurs nœuds.[28]

Pour cela, vous pouvez configurer un groupe Auto Scaling avec une taille minimum, maximum et souhaitée définie sur la même valeur.[29] Cela permet à Auto Scaling de mettre un nouveau nœud à la place d'un nœud en panne en cas de défaillance d'un nœud. L'automatisation que vous avez développée peut être ajoutée pour amorcer le nœud avec le logiciel et l'initier dans l'anneau en remplaçant le nœud en panne. Une fois cela réalisé, le logiciel Cassandra s'occupe d'apporter les données du nœud dans l'état actuel en diffusant les données d'autres nœuds dans le cluster. Il vous faudra peut-être définir des groupes Auto Scaling distincts pour les initiaux et les autres. Si vous avez besoin de mieux contrôler le positionnement, vous pouvez choisir de créer un groupe Auto Scaling pour chaque zone de disponibilité et de maintenir la distribution.

De même, vous pouvez ajouter les hôtes bastion à un groupe Auto Scaling pour permettre le remplacement de nœuds en panne en cas de panne de zone de disponibilité ou de nœud.

Pour plus d'informations sur la création d'une passerelle NAT pour votre cluster, reportez-vous à NAT Gateway dans le *Guide de l'utilisateur Amazon VPC*.[30]

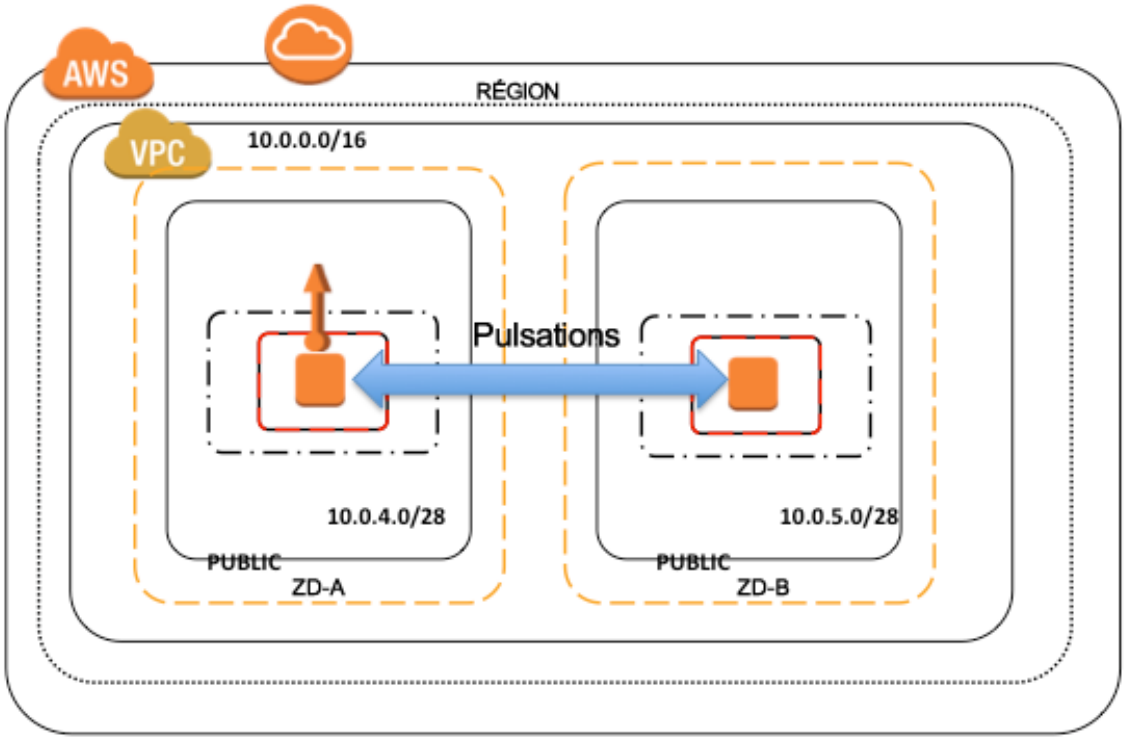

Figure 6 : OpsCenter hautement disponible

Le schéma précédent montre la configuration que nous recommandons pour OpsCenter dans une configuration haute disponibilité. Les nœuds OpsCenter ont une configuration maître-esclave. Pour toujours mapper le maître sur une adresse IP statique, vous pouvez utiliser une adresse IP Elastic.[31] Pour une approche de défaillance, vous pouvez déployer les nœuds maître et esclaves OpsCenter dans des groupes Auto Scaling individuels avec une taille minimum, maximum et souhaitée de 1. Cela permet de garantir que si Auto Scaling résilie un nœud en cas de panne de nœud, Auto Scaling apporte un nouveau nœud pour remplacer celui en panne. L'automatisation peut alors configurer le nouveau nœud pour qu'il devienne le nouveau nœud de sauvegarde. Pendant une panne, l'automatisation doit être en mesure de détecter la panne et de remapper l'adresse IP Elastic sur le nouveau maître. Cette approche vous permet d'utiliser l'adresse IP Elastic statique pour accéder à OpsCenter pour la surveillance.

Étudions à présent l'aspect de l'architecture du cluster avec Auto Scaling pour Cassandra, présentée dans le schéma suivant.

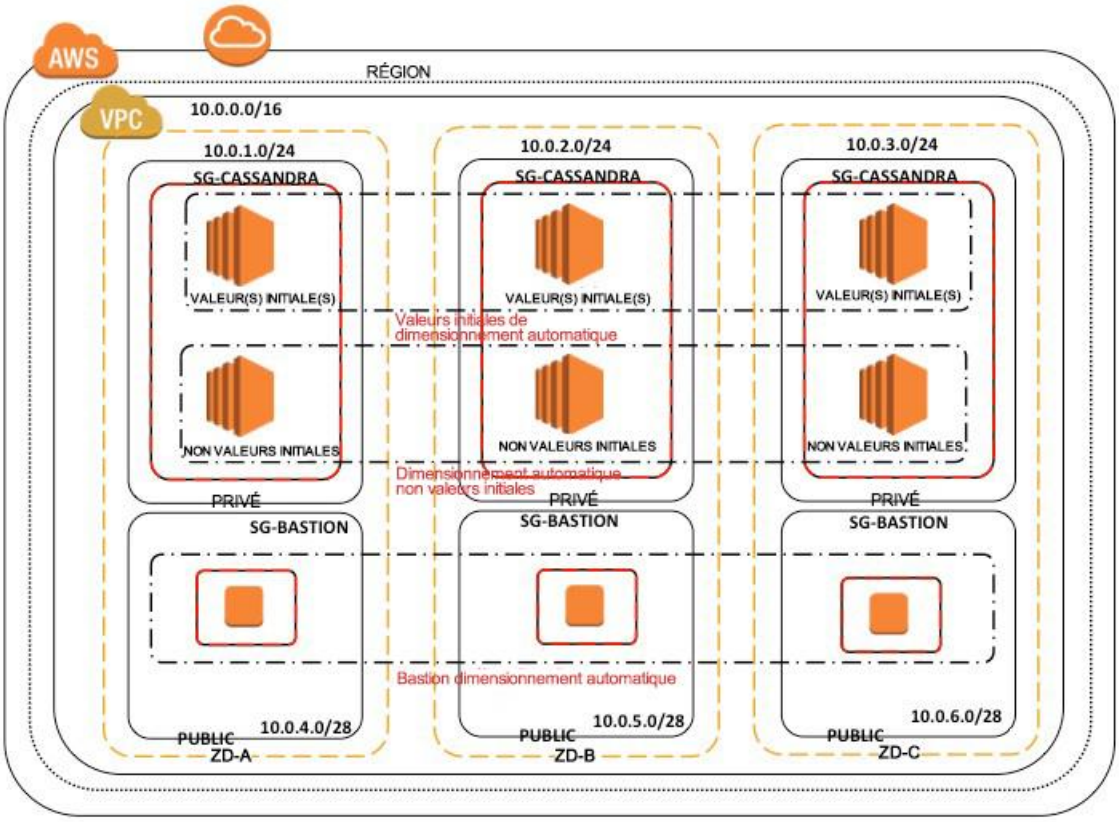

Figure 7 : Cassandra hautement disponible avec Auto Scaling

La meilleure pratique consiste à assurer la cohérence de la liste de nœuds initiaux sur tous les nœuds dans le cluster. Vous serez peut-être en mesure d'obtenir une liste d'adresses IP pour les nœuds initiaux pendant le lancement initial du cluster. Lorsqu'un nœud initial échoue et qu'un nouveau nœud est fourni en remplacement, une nouvelle adresse IP est allouée à ce dernier. Cette fonctionnalité signifie que vous devrez remplacer la liste de nœuds initiaux sur tous les autres nœuds dans le cluster pour refléter la nouvelle adresse IP. Cela peut représenter un défi à plusieurs niveaux.

Vous pouvez gérer cette situation en créant une Elastic Network Interface (ENI).[32] Vous pouvez créer une ENI par nœud initial, comme décrit précédemment dans ce livre blanc. Chaque ENI est associée à une adresse IP privée statique et vous pouvez ajouter l'adresse IP de l'ENI à la liste de nœuds initiaux. L'automatisation des nœuds initiaux peut alors attacher l'ENI à la nouvelle instance exécutée par le groupe Auto Scaling pour remplacer l'instance d'amorçage en panne.

Setting Up for Security

Notre configuration haute disponibilité traite les questions de sécurité. Dans cette configuration, les nœuds de chaque sous-réseau privé ont un groupe de sécurité pour contrôler l'accès réseau entrant. Cependant, des nœuds dans les sous-réseaux privés n'ont pas d'itinéraire vers Internet. Pour les mises à jour logicielles, lorsque les nœuds Cassandra doivent communiquer avec Internet, vous pouvez créer une instance NAT dans un sous-réseau public ou configurer une passerelle NAT pouvant acheminer le trafic depuis le cluster Cassandra dans le sous-réseau privé.

Si vous avez besoin d'utiliser SSH pour accéder à un nœud à des fins de dépannage, un hôte bastion dans le sous-réseau public permettra d'accéder à n'importe quel nœud dans les sous-réseaux privés. Pour obtenir des informations sur la manière de configurer un hôte bastion pour AWS, consultez cet exemple sur le Blog de sécurité AWS.[33]

Dans cette configuration, les hôtes OpsCenter sont configurés avec des groupes de sécurité qui permettent un accès uniquement depuis des instances spécifiques dans votre réseau.

Si vos exigences de conformité nécessitent que vous ayez un chiffrement au repos pour votre cluster Cassandra, vous pouvez optimiser les volumes Amazon EBS avec le chiffrement activé. Le chiffrement Amazon EBS utilise les clés principales du client AWS Key Management Service (AWS KMS) pour le chiffrement.[34] Le chiffrement se produit sur les serveurs qui hébergent des instances EC2, fournissant un chiffrement de données en transit depuis des instances EC2 vers le stockage EBS. Pour en savoir plus, reportez-vous à la documentation sur le chiffrement EBS.[35]

Le chiffrement de données en transit peut être configuré en suivant la documentation DataStax.[36]

La table suivante présente l'ensemble de ports par défaut pour Cassandra. Vous pouvez également choisir de remplacer ces ports par défaut par des ports personnalisés. L'utilisation de ces ports vous permet de développer un modèle de privilèges minimum et améliore encore la sécurité via les groupes de sécurité.

Numéro de port	Description	Type de port	Liste blanche activée
22	Port SSH	Public	Tous les nœuds
8888	Site Web OpsCenter. Le démon opscenterd écoute sur ce port les requêtes HTTP provenant directement du navigateur.	Public	Serveurs OpsCenter
7000	Communication de cluster entre les nœuds Cassandra.	Privé	Nœuds de cluster Cassandra
7001	Communication de cluster entre les nœuds Secure Socket Layer (SSL) Cassandra.	Privé	Cassandra cluster nodes
7199	Port de surveillance JMX Cassandra.	Privé	Cassandra cluster nodes
9042	Port client Cassandra.	Privé	Cassandra cluster nodes
9160	Port client Cassandra (Thrift).	Privé	Cassandra cluster nodes
61620	Port de surveillance OpsCenter. Le démon opscenterd écoute sur ce port le trafic du protocole TCP provenant de l'agent.	Privé	Nœuds de cluster Cassandra, serveurs OpsCenter
61621	Port d'agent OpsCenter. Les agents écoutent sur ce port le trafic SSL initié par OpsCenter.	Privé	Cassandra cluster nodes, OpsCenter servers

Monitoring by Using Amazon CloudWatch

Amazon CloudWatch est un service de surveillance pour les ressources du cloud AWS et les applications que vous exécutez sur AWS. Vous pouvez utiliser Amazon CloudWatch pour collecter et assurer le suivi des métriques, pour recueillir et surveiller les fichiers journaux, et pour régler les alarmes. Amazon CloudWatch peut envoyer une alarme via Amazon Simple Notification Service (Amazon SNS) ou par e-mail lorsque des seuils définis par l'utilisateur sont atteints sur des services AWS individuels. Par exemple, vous pouvez définir une alarme pour prévenir d'une utilisation excessive de l'UC.

Vous pouvez sinon écrire une métrique personnalisée et la soumettre à Amazon CloudWatch pour surveillance. Par exemple, vous pouvez écrire une métrique personnalisée pour vérifier la mémoire actuellement libre sur vos instances et pour définir des alarmes ou déclencher des réponses automatiques lorsque ces mesures dépassent un seuil que vous spécifiez. Vous pouvez également publier des métriques JMX ou NodeTool sur CloudWatch. Vous pouvez ensuite configurer des alarmes pour vous notifier lorsque les métriques dépassent certains seuils définis.

Pour publier des métriques pour Cassandra dans CloudWatch, vous devez utiliser des rôles AWS Identity ans Access Management (IAM) pour accorder des autorisations à vos instances. Vous pouvez ensuite utiliser la CLI (interface ligne de commande) AWS (AWS CLI) pour publier n'importe quelle mesure directement dans CloudWatch. L'exemple suivant démontre comment publier une mesure simple nommée `CompactionsPending` dans CloudWatch avec une valeur de 15.

```
aws cloudwatch put-metric-data --metric-name
CompactionsPending --namespace Cassandra --timestamp 2015-
12-13T19:50:00Z --value 15 --unit
```

Pour plus d'informations, consultez la documentation DataStax sur la surveillance.[37]

Using Multi-Region Clusters

Un exemple de cluster multi-régions dans Cassandra s'affiche comme suit.

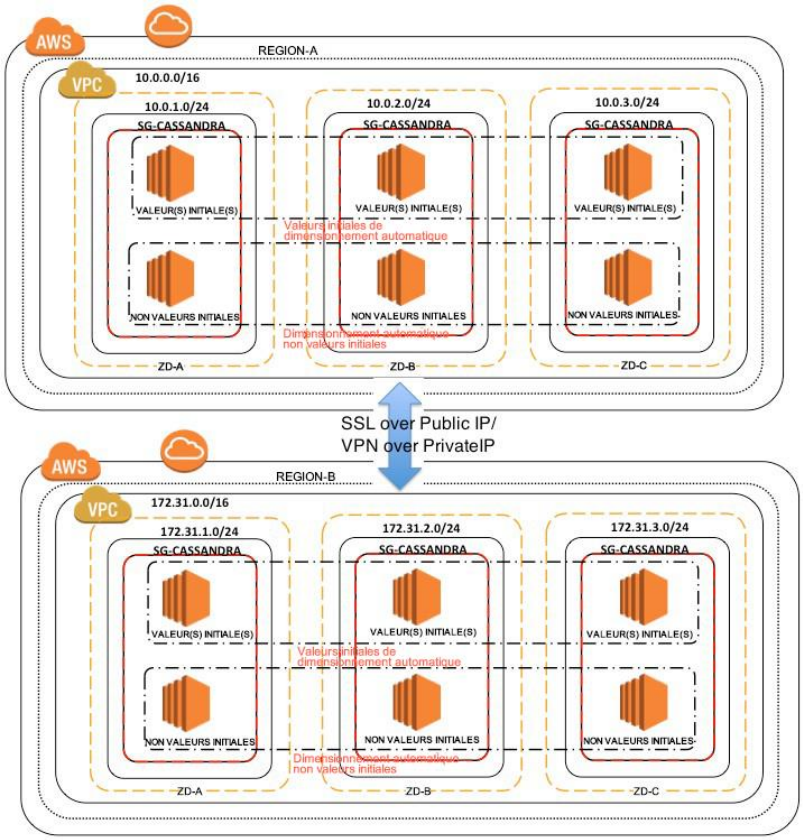

Figure 8 : Cluster Cassandra multi-régions

Pour configurer des clusters multi-régions, vous pouvez activer une communication de cluster multi-régions via des tunnels de sécurité du protocole Internet (IPsec) en configurant un réseau privé virtuel (VPN). Cette approche vous permet d'utiliser l'adresse IP privée de votre cluster pour la communication entre régions. Vous pouvez utiliser au choix Ec2Snitch ou gossippingpropertyfilesnitch pour votre snitch lorsque vous utilisez un VPN pour la communication entre régions. Pour plus d'informations sur cette configuration, consultez l'article AWS Connexion de plusieurs VPC avec des instances EC2 (IPsec).[38]

Vous pouvez également autoriser la communication entre régions en utilisant des adresses IP publiques sur Internet. Vous devez envisager d'utiliser EC2MultiRegionSnitch pour cette configuration. Ce snitch définit automatiquement l'adresse de diffusion depuis l'adresse IP publique obtenue de l'API métadonnées EC2. Vous pouvez sécuriser cette communication en configurant SSL pour votre communication entre les nœuds. Pour plus d'informations, consultez activation du chiffrement nœud à nœud dans la documentation DataStax.[39] Cependant, vous devez définir le paramètre *listen_address* pour utiliser les adresses IP privées. Cette approche permet à la communication dans une région d'utiliser des adresses IP privées à la place d'adresses IP publiques.

Une autre meilleure pratique que nous recommandons consiste à réaliser des opérations de lecture et d'écriture à un niveau de cohérence de LOCAL_QUORUM. Si vous utilisez à la place un quorum global, les opérations de lecture et d'écriture doivent atteindre un quorum sur des régions AWS, ce qui peut entraîner une augmentation de la latence côté client.

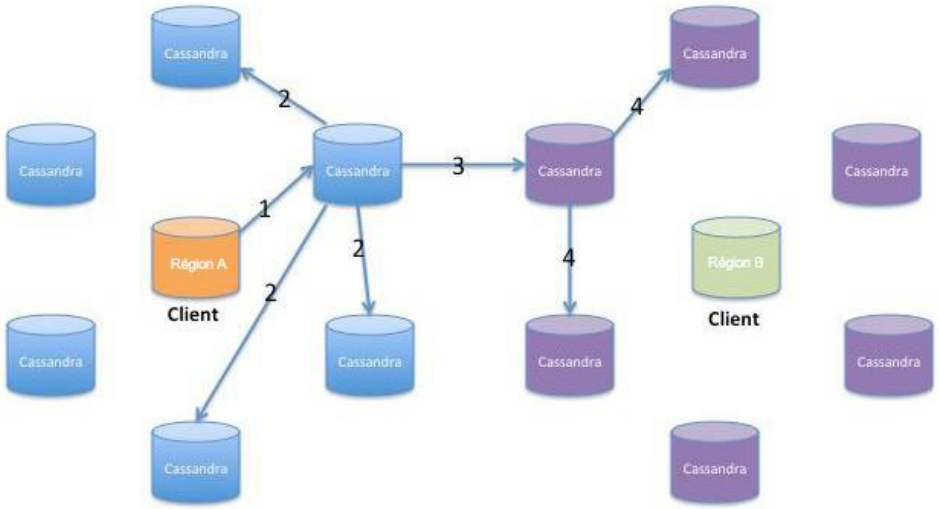

Figure 9 : demande d'écriture sur plusieurs régions

L'illustration précédente montre la manière dont le client réalise une opération d'écriture en utilisant le quorum local dans un scénario d'écriture sur plusieurs régions. La liste suivante illustre le processus :

1. Un client envoie une demande d'écriture à un nœud de coordinateur dans le cluster.
2. Le nœud de coordinateur envoie la demande aux nœuds de données adéquats. À ce stade, si deux nœuds sur trois dans la région locale renvoie un accusé de réception, l'opération d'écriture du client aboutit.
3. Le coordinateur local suit l'opération d'écriture de manière asynchrone sur un coordinateur distant dans l'autre région.
4. Le coordinateur distant envoie à présent les données aux réplicas distants. Les nœuds de réplicas distants accusent réception au coordinateur distant.

Si un nœud ou une région se déconnecte, hinted handoff peut réaliser une opération d'écriture lorsque le nœud revient. Vous pouvez exécuter des tâches de réparation nocturnes pour maintenir la cohérence des régions.[40]

Réalisation de sauvegardes

Vous pouvez utiliser la commande d'instantané NodeTool pour prendre un instantané de votre espace de clés à des fins de sauvegarde. Cette commande va créer des liens physiques vers vos SStables dans un répertoire d'instantanés dans votre répertoire de données d'espace de clés. Veuillez noter que cette commande va créer un instantané uniquement sur le nœud unique sur lequel la commande est exécutée. Pour prendre un instantané d'un cluster en entier, la commande doit être exécutée sur tous les nœuds dans le cluster en utilisant un SSH parallèle ou en spécifiant une adresse IP/un nom d'hôte à la commande NodeTool depuis une instance unique. Les SStables peuvent alors être sauvegardées sur Amazon S3 ou sur Amazon EBS. Avec EBS, vous pouvez prendre un instantané du volume et supprimer le volume pour économiser sur les coûts.

La méthode précédente nécessite un effort important de votre part pour développer, tester et automatiser une sauvegarde de travail et restaurer un processus. Si vous préférez éviter cet effort, envisagez d'utiliser l'outil open source créé par Netflix Priam.[41] Cet outil est excellent pour la sauvegarde et la récupération Cassandra. Priam sauvegarde Cassandra sur S3 progressivement et fournit des API pour restaurer et récupérer le cluster sur S3.

Une autre approche simple pour les sauvegardes qui fonctionne avec Amazon S3 consiste à utiliser l'outil tablesnap. Cet outil recherche simplement de nouveaux fichiers dans le répertoire de l'espace de clés et télécharge ces fichiers sur S3 depuis chaque nœud.[42]

Building Custom AMIs

La création de votre propre Amazon Machine Image (AMI) personnalisée pour vos clusters Cassandra est une bonne pratique. Cependant, vous pourrez ne pas souhaiter partir de zéro lorsque vous amorcerez un nouveau nœud Cassandra.

Pour créer une AMI personnalisée pour vos clusters Cassandra, vous devez commencer par identifier et développer votre AMI de base. Ici, une des étapes peut consister à commencer par une distribution Linux de base. Vous pouvez ensuite développer une AMI de base sur l'AMI initiale, par exemple en installant un JDK stable, les agents requis, etc. Lorsque vous avez une AMI de base, vous pouvez exécuter des personnalisations sur cette dernière et développer une AMI personnalisée. Les AMI personnalisées qui contiennent les packages logiciels Cassandra installés peuvent être utilisées comme image finale (à savoir, un modèle) pendant votre déploiement.

À mesure que votre organisation se développe, la configuration d'AMI peut contribuer à prendre en charge nombre de vos processus. Pour plus d'informations sur la gestion AMI sur AWS, reportez-vous à ce livre blanc.[43]

Migration vers AWS

Les étapes suivantes migrent un cluster Cassandra existant dans EC2 sur AWS sans temps d'arrêt.

1. Décidez d'un bon point de départ pour votre type d'instance et stockage pour ce cluster en fonction des recommandations apportées plus tôt dans ce livre blanc.
2. Testez et comparez votre instance pour une charge de travail représentative en utilisant cassandra-stress ou YCSB (description ci-après), Jmeter ou votre outil préféré.
3. Une fois que vous êtes satisfait du stockage et du type d'instance, fournissez un nouveau cluster Cassandra dans EC2 dans la région de votre choix. Veuillez noter que vous pouvez avoir besoin de changer le paramétrage de votre snitch pour utiliser EC2Snitch ou EC2MultiRegionSnitch, comme décrit plus tôt.
4. Vous devez à présent configurer votre schéma sur le nouveau cluster dans EC2. Ce schéma sera dupliqué depuis votre cluster existant.
5. Pour une expérience réseau plus cohérente qu'Internet, vous pouvez choisir d'utiliser AWS Direct Connect.[44]
6. Assurez-vous que le client utilise LOCAL_QUORUM sur le cluster existant. Cette étape garantit que vous ne subissez pas de latences élevées du fait de l'ajout d'un nouveau centre de données.
7. Mettez à jour l'espace de clés pour répliquer les données vers le cluster EC2 de manière asynchrone. Cette étape permettra au cluster Cassandra sur EC2 d'obtenir les nouvelles opérations d'écriture.
8. Validez vos journaux d'application pour vous assurer qu'il n'y a pas d'erreurs et que les latences d'application sont normales.

9. Exécutez la commande de reconstruction NodeTool sur les nœuds sur le cluster EC2. L'opération de reconstruction peut être exigeante en E/S. Si vous souhaitez limiter l'impact de cette opération sur le cluster existant, vous devez envisager d'exécuter l'opération sur un nœud à la fois. Mais si votre cluster peut assumer des E/S supplémentaires, vous pouvez toujours propager et exécuter l'opération sur plusieurs nœuds simultanément.
10. Une fois la reconstruction terminée sur tous les nœuds EC2 du côté AWS, vous aurez toutes les données synchronisées sur le cluster EC2. Réalisez la validation pour vous assurer que le nouveau cluster fonctionne comme prévu.
11. Vous pouvez sinon choisir d'augmenter le niveau de cohérence pour les opérations d'écriture sur EACH_QUORUM. Cela permet aux opérations d'écriture d'aboutir uniquement si à la fois votre cluster actuel dans votre centre de données et le cluster EC2 accusent réception de l'opération d'écriture. Cependant, cette approche peut augmenter la latence sur votre côté client.
12. Si vous avez activé EACH_QUORUM, surveillez vos journaux d'application pour vérifier que les latences sont telles que prévu, qu'il n'y a pas d'erreurs et que le cluster répond normalement aux requêtes.
13. À ce stade, vous pouvez basculer votre application pour communiquer avec le cluster EC2.
14. Vous devez mettre à jour votre client pour réduire le niveau de cohérence au LOCAL_QUORUM.
15. Là encore, si vous avez activé EACH_QUORUM, surveillez vos journaux d'application pour vérifier que les latences sont telles que prévu, qu'il n'y a pas d'erreurs et que le cluster répond normalement aux requêtes.
16. Si vous avez des problèmes, vous pouvez suivre la procédure décrite dans cette section pour procéder à une restauration.
17. Vous pouvez sinon migrer votre pile client sur AWS pour réduire les latences réseau.
18. Vous pouvez à présent désactiver l'ancien cluster.

Analyses sur Cassandra avec Amazon EMR

Pour contribuer à réaliser des analyses sur Cassandra, DataStax a publié un pilote Cassandra Apache Spark open source.[45] Apache Spark est un moteur de traitement rapide et général compatible avec les données Hadoop et le pilote DataStax vous permet d'utiliser les familles de colonnes Cassandra comme RDD (Resilient Distributed Datasets) Apache Spark. Vous pouvez utiliser ce pilote pour développer des applications Spark qui peuvent lire depuis et écrire vers Cassandra, ce qui vous permet de combiner la puissance des fonctions de traitement analytique en mémoire de traitement massivement parallèles de Spark avec Cassandra.

Pour gérer des clusters Apache Spark, pensez à utiliser Amazon Elastic MapReduce (Amazon EMR), un service Web qui permet de traiter rapidement et de manière rentable d'importants volumes de données.[46] Apache Spark sur Hadoop YARN est pris en charge en mode natif dans Amazon EMR, et vous pouvez créer rapidement et facilement des clusters Apache Spark gérés depuis AWS Management Console, AWS CLI ou l'API Amazon EMR.

Vous pouvez simplement créer un cluster avec Spark installé sur EMR.[47] À présent, avec Spark disponible instantanément sur le cluster EMR, vous pouvez installer le pilote Cassandra Spark et ses dépendances pour créer des applications Spark qui s'adressent à votre cluster Cassandra dans AWS.

Vous trouverez ci-après un exemple simple de code que vous pouvez exécuter sur le Shell Spark sur le nœud maître EMR pour obtenir le nombre d'éléments dans une famille de colonnes Cassandra nommée `kv` dans un test d'espace de clés.

```
import org.apache.spark.SparkConf
val conf = new
SparkConf(true).set("spark.cassandra.connection.host",
"<Cassandra Listen IP Address>")
```

```
import org.apache.spark.SparkContext
import org.apache.spark.SparkContext._
sc.stop
val sc = new SparkContext("local[2]", "test", conf)
import com.datastax.spark.connector._
val rdd = sc.cassandraTable("test", "kv")
println(rdd.count)
```

Amazon EMR vous permet également de lancer Apache Zeppelin comme application sandbox.[48] Zeppelin est une GUI open source qui crée des bloc-notes interactifs et collaboratifs pour l'exploration de données. Apache Zeppelin prend en charge un interprète pour Cassandra. Vous pouvez configurer cet interprète dans Zeppelin sur le cluster Amazon EMR. Cela vous permet de créer des blocs-notes interactifs pour exécuter Cassandra Query Language (CQL) sur Cassandra depuis Zeppelin afin de visualiser les données dans votre cluster Cassandra sur AWS.

Optimisation de coûts des transferts de données

Comme abordé précédemment, nous recommandons que la communication dans une région au sein du cluster se fasse avec des adresses IP privées en définissant les paramètres *listen_address* et *rpc_address* sur des adresses IP privées au lieu d'adresses IP publiques. Cependant, des frais de transfert de données entre des instances EC2 s'appliquent toujours lorsque vous utilisez des adresses IP publiques pour la communication. Avec les adresses IP privées, des frais s'appliquent lorsque la communication s'effectue entre des instances dans différentes zones de disponibilité.

Vous pouvez réaliser un certain nombre d'optimisations pour simplifier le transfert de données et minimiser les coûts. Par exemple, normalement lorsqu'une requête est soumise par un client, cette dernière est envoyée à n'importe quel nœud dans le cluster qui sert de coordinateur. Le coordinateur envoie alors les données aux nœuds qui sont propriétaires de l'espace de clés. Cependant, le coordinateur est une étape supplémentaire dans le flux de requêtes qui peut être évitée dans la plupart des cas en utilisant un client compatible avec les tokens. Cette approche peut potentiellement éviter ce tronçon supplémentaire et accroître la performance. Pour trouver un pilote client pour cette approche, consultez cette liste de pilotes clients pris en charge par DataStax.[49]

Pour optimiser encore en acheminant des requêtes vers la même zone de disponibilité que le client, pensez à Astyanax, un client Java pour Cassandra qui a été mis en open-source par Netflix (parmi plusieurs autres outils).[50] Ce client a été spécifiquement écrit en pensant à EC2 et vous permet d'acheminer des demandes vers la même zone de disponibilité que le client. Par exemple, si votre client est dans la région USA Est (Virginie du Nord), à savoir us- east-1a, le client peut identifier les nœuds responsables de la plage de clés et le nœud qui est dans la même zone de disponibilité que le client (us-east-1a dans cet exemple) et envoyer la demande à ce nœud. Cette approche peut améliorer la performance et optimiser les coûts de transferts de données en évitant des tronçons supplémentaires.

En option, vous pouvez également vérifier si la compression entre les nœuds est activée ou non. Elle est définie sur tous par défaut, ce qui signifie que toutes les communications entre les nœuds sont compressées. Cette approche peut réduire le volume de trafic réseau envoyé sur le canal réseau et peut contribuer à optimiser les coûts de transfert de données. Cependant, l'utilisation de ce paramètre place une faible surcharge sur l'UC. Avant de l'utiliser, vous devez tester pour vous assurer que ce paramètre n'affecte pas la performance de votre cluster.

La désactivation de réparation de lecture, une propriété de la table, réduira encore les coûts entre les zones de disponibilité. Lorsque cette propriété est activée, certaines demandes vont « sur-lire » depuis plus de nœuds que requis par le niveau de cohérence spécifié. Cette fonctionnalité leur permet de réparer les incohérences à l'arrière-plan. La fonction de nouvelle tentative spéculative peut également augmenter le nombre de lectures de données, mais dans ce cas elle peut s'exécuter parce que l'opération de lecture met longtemps à aboutir.

Comparaison Cassandra

Vous voudrez peut-être comparer la performance de Cassandra avec plusieurs types et configurations d'instance avant de décider de la bonne configuration pour votre cluster. Cassandra-stress est un excellent outil pour comparer Cassandra. Nous recommandons fortement l'utilisation de cet outil pour comparer Cassandra sur AWS.[51]

Si vous souhaitez comparer Cassandra à d'autres moteurs NoSQL avec un outil courant, vous pouvez utiliser YCSB.[52] Cet outil bénéficie d'une prise en charge pour plusieurs moteurs NoSQL, dont, mais sans s'y limiter, Amazon DynamoDB, MongoDB et Aerospike.

Si vous voulez comparer des fournisseurs via des tests de performances, il est généralement judicieux d'effectuer une comparaison entre éléments semblables. Pour cela, certains des éléments courants à prendre en considération incluent les suivants :

1. Vous devez vous assurer que le cluster Cassandra et les clients à partir desquels la comparaison est exécutée sont sur le même fournisseur. Par exemple, vous pouvez vous assurer que votre cluster et vos clients Cassandra s'exécutent dans AWS au lieu d'exécuter un cluster Cassandra sur AWS et les clients sur un autre fournisseur.
2. Vous devez fournir un volume équivalent de ressources tels que l'UC, la mémoire, les IOPS et le réseau. Si vous ne pouvez pas atteindre une équivalence du fait de l'indisponibilité de types d'instances comparables entre fournisseurs, assurez-vous que ce manque d'équivalence soit reflété, ajusté ou pris en compte dans les résultats.
3. Vous devez vous assurer que les paramètres de configuration pour Cassandra, tels que les paramètres d'environnement et YAML, sont les mêmes lorsque vous développez des clusters avec plusieurs fournisseurs pour la comparaison.

Vous devez vous assurer que toutes les optimisations (le cas échéant) réalisées sur une plateforme soient répliquées et réalisées sur l'autre plateforme pour obtenir une comparaison équitable.

Utilisation du déploiement Quick Start Cassandra

Les déploiements de référence AWS Quick Start vous aident à déployer un logiciel d'entreprise pleinement fonctionnel sur le cloud AWS, en suivant les meilleures pratiques AWS en termes de sécurité et de disponibilité.[53]

Nous fournissons un Quick Start Cassandra qui lance et exécute automatiquement un cluster Cassandra sur AWS. Cela automatise le déploiement via un modèle CloudFormation AWS et vous permet de lancer le cluster Cassandra sur votre propre VPC Amazon ou dans un nouveau VPC Amazon. Les options de personnalisation incluent la version Cassandra que vous souhaitez déployer (version 2.0 ou 2.1), le nombre de valeurs initiales et non initiales que vous souhaitez lancer pour garantir une disponibilité élevée (une à trois valeurs initiales), les remplacements de nœuds automatiques avec Auto Scaling, OpsCenter pour la surveillance, l'intégration de la passerelle NAT avec la configuration VPC et le choix de la version Java (Java 7 ou Java 8).

Le déploiement du Quick Start Cassandra dure environ 15 minutes. Vous payez uniquement les ressources de stockage et de calcul AWS que vous utilisez. Aucun coût ne vient s'ajouter pour l'exécution de Quick Start. Les modèles sont à votre disposition pour une utilisation dès aujourd'hui. Le modèle CloudFormation pour développer le cluster dans un nouveau VPC est disponible ici, [54]et le modèle CloudFormation pour développer le cluster dans un VPC existant est disponible ici.[55]

Conclusion

Le cloud AWS fournit une plateforme unique pour l'exécution d'applications NoSQL, dont Cassandra. Avec des capacités qui peuvent satisfaire des besoins dynamiques, des coûts basés sur l'utilisation et une intégration facile avec d'autres produits AWS, tels qu'Amazon CloudWatch, AWS Cloud Formation et Amazon EBS, le cloud AWS vous permet d'exécuter diverses applications NoSQL sans devoir gérer le matériel vous-même. Cassandra, en association avec AWS, fournit une plateforme robuste pour le développement d'applications évolutives, haute performance.

Collaborateurs

Les personnes et organisations suivantes ont participé à l'élaboration de ce document :

- Babu Elumalai, Solutions Architect, Amazon Web Services

Suggestions de lecture

Pour obtenir de l'aide, consultez les ressources suivantes :

- Régions et zones de disponibilité
- Documentation VPC Amazon
- Amazon EBS
- FAQ Amazon EC2
- Centre de sécurité AWS
- Documentation DataStax Cassandra 2.1

Remarques

1 https://aws.amazon.com/dynamodb/

2 https://aws.amazon.com/s3/

3 http://docs.datastax.com/en/landing_page/doc/landing_page/current.html

4 https://en.wikipedia.org/wiki/Log-structured_merge-tree

5 https://en.wikipedia.org/wiki/Quorum_(distributed_computing)

6 https://docs.datastax.com/en/cassandra/2.0/cassandra/operations/ops_how_cache_works_c.html

7 https://docs.datastax.com/en/cassandra/2.1/cassandra/dml/dml_about_hh_c.html

8 http://docs.datastax.com/en/cassandra/2.1/cassandra/tools/toolsRepair.html

9 https://docs.datastax.com/en/cassandra/3.0/cassandra/operations/opsTuneJVM.html

10 https://aws.amazon.com/about-aws/global-infrastructure/

11 https://aws.amazon.com/about-aws/global-infrastructure/

12 https://docs.datastax.com/en/cassandra/2.1/cassandra/architecture/architectureSnitchesAbout_c.html

13 https://docs.datastax.com/en/cassandra/2.1/cassandra/architecture/architectureSnitchEC2_t.html

14 https://docs.datastax.com/en/cassandra/2.1/cassandra/architecture/architectureSnitchEC2MultiRegion_c.html

15 http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking.html#enhanced_networking_instance_types

16 http://docs.datastax.com/en/cql/3.1/cql/cql_reference/create_keyspace_r.html

17 http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/vpc-classiclink.html

18 http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html#AvailableIpPerENI

19 http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSVolumeTypes.html#EBSVolumeTypes_gp2

[20] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSVolumeTypes.html#EBSVolumeTypes_piops

[21] http://docs.datastax.com/en/cassandra/2.1/cassandra/planning/architecturePlanningEC2_c.html

[22] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSOptimized.html

[23] https://aws.amazon.com/ec2/pricing/

[24] https://www.youtube.com/watch?v=1R-mgOcOSd4

[25] https://aws.amazon.com/ec2/pricing/

[26] http://docs.datastax.com/en/cassandra/2.0/cassandra/dml/dml_config_consistency_c.html

[27] http://stomp.github.com/

[28] http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/WhatIsAutoScaling.html

[29] http://docs.aws.amazon.com/AutoScaling/latest/DeveloperGuide/AutoScalingGroup.html

[30] https://aws.amazon.com/articles/2781451301784570

[31] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html

[32] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-eni.html

[33] https://blogs.aws.amazon.com/security/post/Tx3N8GFK85UN1G6/Securely-connect-to-Linux-instances-running-in-a-private-Amazon-VPC

[34] https://aws.amazon.com/kms/

[35] http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html

[36] https://docs.datastax.com/en/cassandra/2.1/cassandra/security/secureSslEncryptionTOC.html

[37] https://docs.datastax.com/en/cassandra/2.1/cassandra/operations/ops_monitoring_c.html

[38] http://aws.amazon.com/articles/5472675506466066

[39] http://docs.datastax.com/en/cassandra/2.1/cassandra/security/secureSSLNodeToNode_t.html

[40] https://docs.datastax.com/en/cassandra/2.1/cassandra/dml/dml_about_hh_c.html

[41] https://github.com/Netflix/Priam

[42] https://pypi.python.org/pypi/tablesnap

[43] https://d0.awsstatic.com/whitepapers/managing-your-aws-infrastructure-at-scale.pdf

[44] https://aws.amazon.com/directconnect/

[45] https://github.com/datastax/spark-cassandra-connector

[46] https://aws.amazon.com/elasticmapreduce/

[47] https://docs.aws.amazon.com/ElasticMapReduce/latest/ReleaseGuide/emr-spark-launch.html

[48] http://docs.aws.amazon.com/ElasticMapReduce/latest/ReleaseGuide/emr-sandbox.html

[49] http://www.planetcassandra.org/client-drivers-tools/#%20DataStax%20Supported%20Drivers

[50] https://github.com/Netflix/astyanax/wiki/Getting-Started

[51] http://docs.datastax.com/en/cassandra/2.1/cassandra/tools/toolsCStress_t.html

[52] https://github.com/brianfrankcooper/YCSB

[53] http://aws.amazon.com/quickstart/

[54] https://s3.amazonaws.com/quickstart-reference/cassandra/latest/templates/Cassandra_EBS_VPC.json

[55] https://s3.amazonaws.com/quickstart-reference/cassandra/latest/templates/Cassandra_EBS_NoVPC.json