Comparaison de l'utilisation d'Amazon DynamoDB et d'Apache HBase pour NoSQL

Wangechi Doble

Septembre 2014

Table des matières

Table des matières 2
Résumé 3
Introduction 3
Présentation 4
- Présentation d'Amazon DynamoDB 4
- Présentation d'Apache HBase 5
- Résumé des fonctions 8
- Cas d'utilisation 10

Modèles de données 11
- Types de données 16
- Indexation 17

Traitement des données 21
- Modèle de débit 21
- Modèle de cohérence 23
- Modèle de transaction 23
- Opérations de table 24

Architecture 25
- Présentation de l'architecture d'Amazon DynamoDB 25
- Présentation de l'architecture d'Apache HBase 26
- Présentation de l'architecture d'Apache HBase sur Amazon EMR 27
- Partitioning 28

Optimisation des performances 29
- Considérations sur les performances d'Amazon DynamoDB 29
- Considérations sur les performances d'Apache HBase 32
- Apache HBase sur Amazon EMR 34

Conclusion 36
Suggestions de lecture 36

Résumé

L'un des défis auxquels les architectes et les développeurs sont confrontés aujourd'hui concerne le traitement d'importants volumes de données d'une manière respectueuse des délais, économique et fiable. Il existe plusieurs solutions NoSQL sur le marché et le choix de celle qui correspond à votre cas d'utilisation peut se révéler difficile. Le livre blanc compare deux magasins de données NoSQL répandus, Amazon DynamoDB[1], un service de base de données NoSQL entièrement géré dans le cloud, et Apache HBase[2], un magasin de Big Data réparti, open source et orienté colonnes. Amazon DynamoDB et Apache HBase sont tous deux disponibles dans le cloud Amazon Web Services (AWS) [3].

Introduction

Le cloud Amazon Web Services (AWS) accélère l'analyse du Big Data. Grâce à l'accès à l'évolutivité et à l'élasticité instantanées sur AWS, vous pouvez vous concentrer sur l'analyse plutôt que sur l'infrastructure. Que vous indexiez d'importants ensembles de données, analysiez des quantités volumineuses de données scientifiques ou traitiez les journaux des flux de clics, AWS fournit un large éventail de produits de Big Data que vous pouvez mettre à profit pour pratiquement n'importe quel projet gourmand en données.

Il existe une large adoption de bases de données NoSQL dans l'industrie croissante des applications de Big Data et des applications web en temps réel. Amazon DynamoDB et Apache HBase sont deux exemples de bases de données NoSQL, qui sont hautement optimisées pour produire des avantages significatifs en termes de performances par rapport à un système de gestion de base de données relationnelle (SGBDR) traditionnel. Amazon DynamoDB et Apache HBase peuvent tous deux traiter d'importants volumes de données avec des performances et un débit élevés.

Amazon DynamoDB offre un service de base de données NoSQL rapide et entièrement géré. Il permet de décharger l'exploitation et le dimensionnement d'un cluster de base de données distribué et hautement disponible. Apache HBase est un magasin de Big Data réparti, open source et orienté colonnes qui s'exécute sur l'infrastructure Apache Hadoop[4].

---

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

2 http://hadoop.apache.org/

3 http://aws.amazon.com/

4 http://hadoop.apache.org/

Dans le cloud AWS, vous pouvez choisir de déployer Apache HBase sur Amazon Elastic Compute Cloud (Amazon EC2) [5] et de le gérer vous-même. Autre solution, vous pouvez exploiter Apache HBase comme service géré sur Amazon Elastic MapReduce (Amazon EMR) [6], infrastructure Hadoop entièrement gérée et hébergée par-dessus Amazon EC2. La figure suivante illustre les relations entre AWS, Amazon DynamoDB, Amazon EC2, Amazon EMR et Apache HBase.

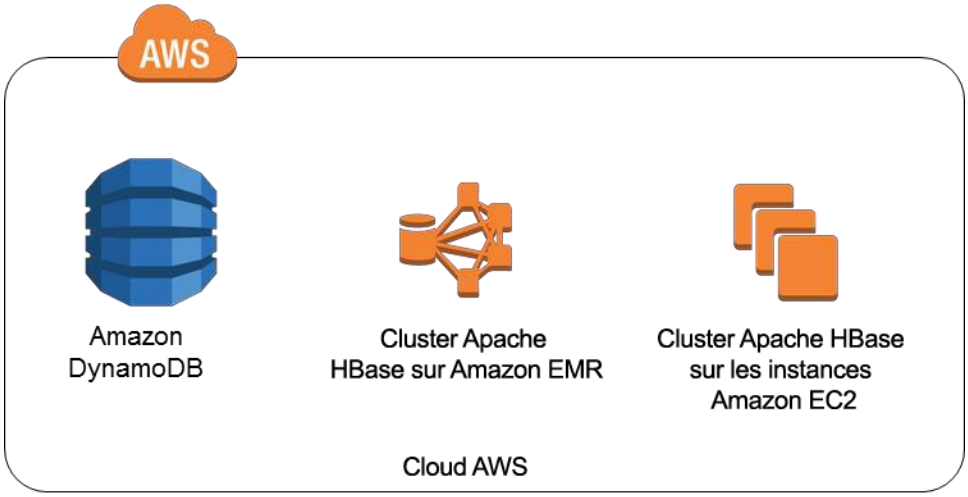

Figure 1. Relations entre AWS, Amazon DynamoDB, Amazon EC2, Amazon EMR et Apache HBase.

Présentation

Cette section résume les principaux avantages, fonctions et cas d'utilisation d'Amazon DynamoDB et d'Apache HBase.

Présentation d'Amazon DynamoDB

Amazon DynamoDB est un service de base de données NoSQL entièrement géré, offrant des performances exceptionnelles et prévisibles en termes de rapidité et d'évolutivité. Amazon DynamoDB offre les avantages suivants :

- **Pas de traitement administratif.** Amazon DynamoDB se charge de gérer l'allocation du matériel, l'installation et la configuration, la réplication, le dimensionnement du cluster, les mises à jour matérielles et logicielles, et la supervision et gestion des défaillances du matériel.

[5] http://aws.amazon.com/ec2/

[6] http://aws.amazon.com/elasticmapreduce/

- **Débit et dimensionnement pratiquement illimités.** Le modèle de débit provisionné d'Amazon DynamoDB vous permet de spécifier la capacité du débit de façon à servir pratiquement tout niveau de trafic de demandes. Avec Amazon DynamoDB, il n'existe pratiquement aucune limite à la quantité de données qui peut être stockée et extraite.
- **Elasticité et flexibilité.** Amazon DynamoDB peut gérer des charges de travail imprévisibles avec des performances prévisibles, tout en continuant à maintenir un profil de latence stable n'affichant aucune augmentation de la latence ou diminution du débit lorsque le volume de données croît suite à une hausse de l'utilisation. Amazon DynamoDB vous permet d'augmenter ou de diminuer la capacité en fonction des besoins afin de gérer les charges de travail variables. Ces qualités font d'Amazon DynamoDB un choix adapté pour les applications en ligne avec la possibilité de se répandre à tout moment.
- **Intégration aux autres services AWS.** Amazon DynamoDB s'intègre harmonieusement aux autres services AWS tels qu'Amazon Identity and Access Management (Amazon IAM) [7] pour contrôler l'accès aux ressources Amazon DynamoDB, Amazon CloudWatch[8] pour surveiller une grande variété de métriques de performances Amazon DynamoDB Amazon Kinesis[9] pour l'intégration des données en temps réel, Amazon Simple Storage Service (Amazon S3)[10] pour le stockage Internet, Amazon EMR pour fournir des capacités d'analyse avancée améliorées, Amazon Redshift[11] pour offrir les fonctionnalités de l'aide à la décision et AWS Data Pipeline[12] pour automatiser les flux de travail pilotés par les données.

Présentation d'Apache HBase

Apache HBase, base de données Hadoop, offre les avantages suivants :

- **Stockage efficace des données éparses.** Apache HBase fournit un stockage tolérant aux pannes pour les grandes quantités de données éparses à l'aide de la compression basée sur les colonnes. Apache HBase peut stocker et traiter des milliards de lignes et des millions de colonnes par ligne.
- **Stockage des compteurs à haute fréquence.** Apache HBase convient aux tâches tels que le regroupement des compteurs à grande vitesse, en raison de ses lectures et écritures cohérentes.
- **Débit élevé d'écriture et fréquences de mise à jour.** Apache HBase prend en charge les analyses de plage et les recherches à faible latence, les mises à jour et les suppressions efficaces d'enregistrements individuels, et le débit élevé d'écriture.

---

7 http://aws.amazon.com/iam/

8 http://aws.amazon.com/cloudwatch/

9 http://aws.amazon.com/kinesis/

10 http://aws.amazon.com/s3/

11 http://aws.amazon.com/redshift/

12 http://aws.amazon.com/datapipeline/

- **Prise en charge de plusieurs tâches Hadoop.** Le magasin de données Apache HBase permet que les données soient utilisées par une ou plusieurs tâches Hadoop sur un seul cluster ou sur plusieurs clusters Hadoop.

Deux des modèles de déploiement les plus courants pour Apache HBase sont le modèle auto-géré et la solution Apache HBase gérée et hébergée sur Amazon EMR. Les sections suivantes fournissent une description de ces options de déploiement.

Modèle de déploiement Apache HBase auto-géré

Le modèle Apache HBase auto-géré offre la plus grande flexibilité en termes de gestion de cluster, mais introduit aussi les défis suivants :

- **Traitement administratif.** Vous devez vous charger de l'administration de l'allocation et de la gestion de vos clusters Apache HBase.
- **Planification des capacités.** Comme pour toute infrastructure traditionnelle, la planification des capacités est difficile et souvent source d'erreur importante coûteuse. Par exemple, vous pouvez sur-investir et vous retrouver à payer une capacité que vous n'utilisez pas, ou sous-investir et prendre le risque de problèmes de performances ou de disponibilité.
- **Gestion de la mémoire.** Apache HBase est essentiellement orienté mémoire. La mémoire peut devenir un critère restrictif, tandis que le cluster croît. Il importe de déterminer la quantité de mémoire nécessaire pour exécuter diverses applications sur votre cluster Apache HBase afin d'empêcher que les nœuds n'échangent les données trop souvent sur le disque. Le nombre de nœuds Apache HBase et la mémoire requise doivent être planifiés bien à l'avance.
- **Planification du calcul, du stockage et du réseau.** Les autres considérations clés relatives au fonctionnement efficace d'un cluster Apache HBase incluent le calcul, le stockage et le réseau. Ces composants de l'infrastructure requièrent souvent des administrateurs Apache Hadoop/Apache HBase dédiés, aux compétences spécialisées.

Apache HBase géré sur Amazon EMR

Apache HBase sur Amazon EMR est optimisé pour s'exécuter sur AWS et offre les avantages suivants :

- **Traitement administratif minimal.** Amazon EMR gère l'allocation des instances Amazon EC2, les paramètres de sécurité, la configuration Apache HBase, la collection des journaux, la surveillance de l'état et le remplacement des instances défectueuses. Vous disposez toujours de la souplesse d'accéder à l'infrastructure sous-jacente et de personnaliser Apache HBase plus avant, si nécessaire.
- **Options de déploiement simple et flexible.** Vous pouvez déployer Apache HBase sur Amazon EMR avec AWS Management Console ou mettre à profit les outils de l'interface de ligne de commande (CLI) AWS[13]. Une fois lancé, un cluster Apache HBase peut être aisément redimensionné à l'aide d'un seul appel d'API. Les activités telles que la modification de la configuration Apache HBase au moment du lancement ou l'installation d'outils tiers tels que Ganglia[14] pour la surveillance des métriques de performance sont réalisables à l'aide de scripts personnalisés ou prédéfinis.
- **Dimensionnement illimité.** Avec Apache HBase s'exécutant sur Amazon EMR, vous pouvez bénéficier d'avantages significatifs du cloud, tels que le dimensionnement aisé, le coût bas, le paiement à l'utilisation uniquement et la facilité d'emploi par opposition au modèle de déploiement auto-géré sur Amazon EC2.
- **Intégration à d'autres services AWS.** Amazon EMR est conçu pour s'intégrer de façon transparente à d'autres services AWS comme Amazon S3, Amazon DynamoDB, Amazon EC2 et Amazon CloudWatch.
- **Fonction de sauvegarde intégrée.** L'un des avantages clés d'Apache HBase s'exécutant sur Amazon EMR est le mécanisme intégré disponible pour sauvegarder les données Apache HBase durablement dans Amazon S3. Avec cette fonction, vous pouvez planifier des sauvegardes complètes ou incrémentielles, et rétrogradez ou même restaurer les sauvegardes vers des clusters existants ou nouvellement lancés à tout moment.

13 http://docs.aws.amazon.com/general/latest/gr/GetTheTools.html

14 http://ganglia.sourceforge.net/

Résumé des fonctions

Amazon DynamoDB et Apache HBase possèdent tous deux des caractéristiques qui sont essentielles pour traiter avec succès d'importantes quantités de données. Le tableau suivant fournit un résumé des fonctions clés d'Amazon DynamoDB et d'Apache HBase qui peuvent vous aider à comprendre les principales similitudes et différences entre les deux bases de données. Ces fonctions sont abordées en détail dans les prochaines sections.

Fonctionnalité	Amazon DynamoDB	Apache HBase
Description	Service de base de données Amazon hébergé et évolutif	Stockage orienté colonnes basé sur Apache Hadoop et sur les concepts de BigTable
Site Web	aws.amazon.com/dynamodb	hbase.apache.org
Documentation	aws.amazon.com/documentation/dynamodb	hbase.apache.org
Développeur	Amazon	Apache Software Foundation
Première version	2012	2008
Licence	N/A	Open Source
Langage d'implémentation	-	Java
Systèmes d'exploitation serveur	Hébergé	Linux, Unix, Windows
Modèle de base de données	Stockage clé-valeur	Stockage à colonne large
Schéma de données	Schéma libre	Schéma libre
Typage	Oui	Non
API et autres méthodes d'accès	RESTful HTTP API, .NET, ColdFusion, Erlang, Groovy	Java API, RESTful HTTP API, Thrift, C, C#, C++
Langages de programmation pris en charge	Java, JavaScript, Perl, PHP, Python, Ruby	Groovy, Java, PHP, Python, Scala
Scripts côté serveur	Non	Oui
Déclencheurs	Non	Oui
Méthodes de partitionnement	Partitionnement	Partitionnement

Fonctionnalité	Amazon DynamoDB	Apache HBase
Modèle de débit	L'utilisateur alloue le débit	Limité à la configuration matérielle
Partitioning	Partitionnement automatique	Partitionnement automatique
Réplication	Plusieurs fois au sein d'une région	Plusieurs fois entre plusieurs serveurs
Durabilité	Oui	Oui
Administration	Pas de surcharge administrative	Surcharge administrative élevée en auto-géré, et minimale sur Amazon EMR
Concepts utilisateur	Les droits d'accès pour les utilisateurs et les rôles peuvent être définis via AWS Identity and Access Management (IAM)	Listes de contrôle d'accès (ACL)
Modèle de données		
Ligne	Un élément peut avoir un nombre quelconque d'attributs	Une ligne peut avoir un nombre quelconque de colonnes qui peuvent ensuite être regroupées par familles
Taille de ligne	Restriction de la taille d'élément	Aucune restriction de taille de ligne
Clé primaire	Clé de hachage ou clé composite hachage-plage	Clé de ligne
Clé étrangère	Non	Non
Index	Les index secondaires locaux facultatifs et les index secondaires globaux peuvent être créés pour les tables avec clés primaires hachage-plage	Aucun modèle d'index intégré, mais les index peuvent être implémentés comme tables secondaires ou coprocesseurs
Transactions		
Transactions de ligne	Transactions de niveau élément	Transactions de ligne unique
Transactions multiligne	Non	Non
Transactions inter-table	Non	Non
Modèle de cohérence	Lectures cohérentes à terme et lectures à cohérence forte	Lectures et écritures à cohérence forte
Simultanéité	Oui	Oui
Mises à jour	Mises à jour conditionnelles	Opérations d'écriture, modification et lecture atomiques

Tableau 1 : Résumé des fonctions d'Amazon DynamoDB et d'Apache HBase

Cas d'utilisation

Amazon DynamoDB et Apache HBase sont optimisés pour traiter d'importantes quantités de données. Les cas d'utilisation les plus répandus d'Amazon DynamoDB et d'Apache HBase sont les suivants :

- **Evénements particuliers à haut volume.** Les événements populaires tels que le Super Bowl, les Jeux olympiques et la Coupe du monde, ou même les événements ponctuels, comme les campagnes électorales, sont d'une durée relativement brève et présentent des charges de travail variables à même d'utiliser d'importantes quantités de ressources. Amazon DynamoDB vous permet d'augmenter ou de diminuer la capacité en fonction des besoins afin de gérer des charges de travail variables. Cette capacité fait d'Amazon DynamoDB un choix parfaitement adapté pour les événements spéciaux à volume élevé.
- **Applications des réseaux sociaux.** Les applications basées sur la communauté, comme les jeux en ligne, le partage de photos ou les applications sensibles à la localisation, entre autres, disposent de modèles d'utilisation imprévisibles et sont capables de se répandre à tout instant. L'élasticité et la flexibilité d'Amazon DynamoDB en font un excellent choix pour de telles charges de travail variables, à volume élevé.
- **Traitement par lots.** Pour les ensembles de données volumineux, comme les journaux de données, les données météorologiques ou les catalogues produit, par exemple, il se peut que vous ayez déjà d'importantes quantités de données historiques que vous souhaitez conserver à des fins d'analyse historique des tendances, mais deviez intégrer et traiter par lots les données actuelles à des fins de prédiction. Pour ces types de charges de travail, Apache HBase constitue un bon choix en raison de son débit élevé en lecture et écriture, et d'un stockage efficace des données éparses.
- **Reporting.** Pour traiter les données transactionnelles à volume élevé, comme les opérations boursières, et en créer des rapports, Apache HBase constitue un bon choix. La raison en est qu'Apache HBase prend en charge les fréquences de mise à jour et les écritures à haut débit, ce qui en fait un outil adapté au stockage des compteurs à haute fréquence et des regroupements complexes.

- **Analyse en temps réel.** La taille de la charge utile ou du message dans les données d'événement, telles que les tweets, l'E-commerce, etc., est relativement petite, comparée aux journaux d'application. Si vous souhaitez intégrer les données d'événement de diffusion en temps réel pour l'analyse des sentiments, la diffusion de publicités, l'analyse des tendances, etc., Amazon DynamoDB vous permet d'augmenter la capacité du débit si nécessaire et de la réduire lorsque vous avez terminé, et ce sans interruption. Apache HBase peut gérer facilement l'intégration en temps réel des données, telles que les journaux d'applications, grâce à son débit élevé en écriture et son stockage efficace des données éparses. La combinaison de ces fonctions et de la capacité d'Hadoop à gérer des analyses et des lectures séquentielles de façon hautement optimisée fait d'Apache HBase un puissant outil pour l'analyse des données en temps réel.

Modèles de données

Les magasins clé/valeur Amazon DynamoDB et Apache HBase ont pour but d'offrir d'importants avantages en matière de performances, avec une latence faible et un débit élevé. Pour atteindre cet objectif, les magasins clé/valeur sont conçus avec des modèles de données plus simples et moins restrictifs que les bases de données relationnelles traditionnelles. Même si les blocs de construction essentiels du modèle de données sont similaires dans Amazon DynamoDB et Apache HBase, chaque base de données utilise une terminologie distincte pour décrire son modèle de données spécifique.

A un haut niveau, une base de données est un ensemble de tables et chaque table est un ensemble de lignes. Une ligne peut comporter une ou plusieurs colonnes. Dans la plupart des cas, les tables de base de données NoSQL ne nécessitent généralement pas un schéma formel, à l'exception d'une clé primaire obligatoire qui identifie chaque ligne de façon unique. Le tableau suivant illustre le concept de haut niveau d'une base de données NoSQL.

Tableau					
Ligne	Clé primaire	Colonne 1	Colonne 2	Colonne 3	Colonne n

Tableau 2 : Représentation d'une table de base de données NoSQL de haut niveau

Les bases de données en colonnes sont conçues pour stocker chaque colonne séparément, de telle sorte que les opérations d'agrégation d'une colonne de l'ensemble de la table soient significativement plus rapides que le modèle de stockage en ligne traditionnel.

D'un point de vue comparatif, une ligne dans Amazon DynamoDB est désignée comme étant un *élément* et chaque *élément* peut avoir un certain nombre d'*attributs.* Un attribut comporte une clé et une valeur, lesquelles sont généralement regroupées sous l'appellation « paire nom-valeur ». Une table Amazon DynamoDB peut avoir un nombre illimité d'éléments indexés sur la clé primaire, comme illustré dans l'exemple suivant.

Tableau					
Elément 1	Clé primaire	Attribut 1	Attribut 2	Attribut 3	Attribut …n
Elément 2	Clé primaire	Attribut 1		Attribut 3	
Elément n	Clé primaire		Attribut 2	Attribut 3	

Tableau 3 : Représentation de haut niveau d'une table Amazon DynamoDB

Amazon DynamoDB définit deux types de clés primaires : une clé primaire unique de *hachage* (Tableau 4) et une clé primaire composite *hachage-plage* avec deux attributs (Tableau 5).

Tableau					
Elément	Clé de hachage	Attribut 1	Attribut 2	Attribut 3	Attribut …n

Tableau 4 : Clé primaire de hachage Amazon DynamoDB

Tableau						
Elément	Clé de hachage	Clé de plage	Attribut 1	Attribut 2	Attribut 3	Attribut …n

Tableau 5 : Clé primaire hachage-plage Amazon DynamoDB

Dans Amazon DynamoDB, une clé primaire de hachage à attribut unique est utile pour les écritures et lectures rapides de données. Par exemple, *IDPersonne* fait office de clé primaire de hachage dans la table *Personne* suivante.

Table Personne					
Elément	IDPersonne	Prénom	Nom	Code postal	Sexe
Elément 1	1001	Prénom-1	Nom-1	00000	
Elément 2	1002	Prénom-2	Nom-2		M
Elément 3	2002	Prénom-3	Nom-3	10000	F

Tableau 6 : *Personne* (Table Amazon DynamoDB)

Une clé composite hachage-plage dans Amazon DynamoDB est indexée comme élément de clé de hachage et élément de clé de plage. Cette clé à plusieurs parties maintient une hiérarchie entre les valeurs du premier et du second élément. En conservant constant l'élément de clé de hachage, vous facilitez les recherches à travers l'élément de clé de plage et permettez d'extraire rapidement les éléments pour une clé de hachage donnée. Dans la table *Scores* suivante, la clé composite hachage-plage est une combinaison d'*IDPersonne* (hachage) et d'*IDJeu* (plage).

Table Scores						
	IDPersonne (clé de hachage)	ID Jeu (clé de plage)	MeilleurScore	DateMeilleurScore	Gagnés	Perdus
élément1	1001	Jeu01	67453	2013-12-09:17:24:31	73	21
élément2	1001	Jeu02	98567	2013-12-11:14:14:37	98	27
élément3	1002	Jeu01	43876	2013-12-15:19:24:39	12	23
élément4	2002	Jeu02	65689	2013-10-01:17:14:41	23	54

Tableau 7 : *Scores* (Table DynamoDB)

Même s'il n'y a pas de limite explicite sur le nombre d'attributs associés à un élément individuel dans une table Amazon DynamoDB, il existe d'autres restrictions sur la taille de regroupement d'un élément ou d'une charge utile, y compris l'ensemble des noms et valeurs d'attributs. Une petite charge utile peut potentiellement améliorer les performances et réduire les coûts, car elle nécessite moins de ressources pour le traitement. Pour plus d'informations sur la gestion des éléments qui dépassent la taille d'élément maximale, consultez Instructions relatives à l'utilisation des éléments[15] dans le *Guide du développeur Amazon DynamoDB*[16].

Dans Apache HBase, l'unité la plus élémentaire est une colonne. Une ou plusieurs colonnes forment une *ligne.* Chaque ligne est adressée de façon unique par une clé primaire appelée *clé de ligne*. Dans Apache HBase, une ligne peut avoir des millions de colonnes. Chaque colonne peut avoir plusieurs versions avec chaque valeur distincte contenue dans une *cellule* séparée.

Un concept fondamental de modélisation dans Apache HBase est celui de *famille de colonnes*. Une famille de colonnes est un conteneur qui regroupe des ensembles de données connexes au sein d'une même table, comme illustré dans l'exemple suivant.

15 http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GuidelinesForItems.html

16 http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/

Tableau							
		Famille de colonnes 1		Famille de colonnes 2		Famille de colonnes 3	
ligne	clé de ligne	Colonne 1	Colonne 2	Colonne 3	Colonne 4	Colonne 5	Colonne 6

Tableau 8 : Représentation de ligne Apache HBase

Apache HBase regroupe les colonnes ayant les mêmes modèles généraux d'accès et les mêmes caractéristiques de taille en familles de colonnes pour former une unité élémentaire de séparation. Par exemple, dans la table *Personne* suivante, vous regroupez les données personnelles en une famille de colonnes appelée *infos_personnelles* et les données statistiques en une famille de colonnes *statistiques*. Toutes les autres colonnes de la table seront aussi regroupées en conséquence, comme illustré dans l'exemple suivant.

Table Personne					
		infos_personnelles		statistiques	
	clé de ligne	prénom	nom	code postal	sexe
ligne 1	1001	Prénom-1	Nom-1	00000	
ligne 2	1002	Prénom-2	Nom-2		M
ligne 3	2002	Prénom-3	Nom-3	10000	F

Tableau 9 : *Personne* (Table Apache HBase)

Les colonnes sont traitées comme une combinaison du nom de la famille de colonnes et le qualificateur de colonne exprimé sous la forme *famille:qualificateur*. Tous les membres d'une famille de colonnes ont le même préfixe. Dans l'exemple précédent, les qualificateurs de colonne *prénom* et *nom* peuvent être référencés comme *infos_personnelles:prénom* et *infos_personnelles:nom*, respectivement.

Les familles de colonnes vous permettent de n'extraire que les colonnes requises par une requête. Tous les membres d'une famille de colonnes sont physiquement stockés ensemble sur un disque. Cela signifie que la portée des fonctions d'optimisation, telles que le réglage des performances, les codages de compression, etc., peut se trouver au niveau des familles de colonnes.

La clé de ligne est une combinaison des identificateurs utilisateur et jeu de la table Apache HBase *Scores* suivante. Une clé de ligne peut se composer de plusieurs parties concaténées afin de fournir une solution immuable pour se référer aux entités. Du point de vue de la modélisation Apache HBase, la table obtenue est *haute et étroite.* La raison en est que la table a peu de colonnes par rapport au nombre de lignes, comme illustré dans l'exemple suivant.

Table Scores					
		meilleurs_scores		mesures	
	clé de ligne	score	date	gagnés	perdus
ligne 1	1001-jeu01	67453	2013-12-09:17:24:31	73	21
ligne 2	1001-jeu02	98567	2013-12-11:14:14:37	98	27
ligne 3	1002-jeu01	43876	2013-12-15:19:24:39	12	23
ligne 4	2002-jeu02	65689	2013-10-01:17:14:41	23	54

Tableau 10 : Table Apache HBase *Scores* haute et étroite

Vous pouvez aussi modéliser l'identificateur jeu comme qualificateur de colonne dans Apache HBase. Cette approche simplifie les recherches sur des colonnes précises et prend en charge l'utilisation de filtres pour lire les données. Le résultat est une table *plate et large* avec peu de lignes par rapport au nombre de colonnes. Ce concept d'une table Apache HBase plate et étroite est illustré dans le tableau suivant.

Table Scores							
			meilleurs_scores		mesures		
	clé de ligne	IDJeu	score	date_meilleur_score	IDJeu	gagnés	perdus
ligne 1	1001	jeu01	98567	2013-12-11:14:14:37	jeu01	98	27
		jeu02	43876	2013-12-15:19:24:39	jeu02	12	23
ligne 2	1002	jeu01	67453	2013-12-09:17:24:31	jeu01	73	21
ligne 3	2002	jeu02	65689	2013-10-01:17:14:41	jeu02	23	54

Tableau 11 : Table Apache HBase *Scores* plate et large

Pour des raisons de performance, il importe que le nombre de familles de colonnes de votre schéma Apache HBase reste bas. Toute quantité au-delà de trois familles de colonnes peut potentiellement dégrader les performances. La bonne pratique recommandée consiste à maintenir une famille de colonnes dans vos schémas et à introduire une deuxième famille et une troisième famille uniquement si l'accès aux données est limité à une seule famille de colonnes à la fois. Notez qu'Apache HBase n'impose aucune restriction sur la taille des lignes.

Types de données

Amazon DynamoDB et Apache HBase prennent tous deux en charge les ensembles de données non structurés avec un large éventail de types de données.

Amazon DynamoDB prend en charge les types de données du tableau suivant.

	Type	Description	Exemple (format JSON)
Scalaire			
Scalaire			
	Chaîne	Unicode avec codage binaire UTF8	`{"S": "Jeu01"}`
	Numéro	Entiers et décimaux à valeur exacte positifs ou négatifs	{ N : 67453 }
	Binaire	Séquence codée d'octets	{ B : dGhpcyB0ZXh0IGIzIGJhc2U2NC1lbmNvZGVk }
Valeurs multiples	Ensemble de chaînes	Ensemble unique de chaînes	{ SS : [Noir , Vert] }
	Ensemble de nombres	Ensemble unique de nombres	{ NS : [42.2 , -19.87] }
	Ensemble binaire	Ensemble unique de valeurs binaires	{ BS : [U3Vubnk= , UmFpbnk=] }

Tableau 12 : Types de données Amazon DynamoDB

Chaque attribut Amazon DynamoDB est une paire nom-valeur qui est un ensemble à valeur unique, un type de données scalaire ou un ensemble à valeurs multiples. Les attributs de clé primaire peuvent être de n'importe quel type scalaire, mais non de type à valeurs multiples. Les éléments individuels d'une table Amazon DynamoDB peuvent avoir un nombre quelconque d'attributs. Les attributs de type binaire peuvent stocker n'importe quelle donnée binaire : par exemple, les données compressées, les données chiffrées ou même les images.

En résumé, Apache HBase définit les concepts suivants :

- **Ligne :** tableau d'octets atomiques ou conteneur clé/valeur.
- **Colonne :** clé au sein du conteneur clé/valeur à l'intérieur d'une ligne.
- **Famille de colonnes :** scinde les colonnes en sous-ensembles de données associés et stockés ensemble sur le disque.
- **Horodatage :** Apache HBase ajoute le concept d'une quatrième colonne de dimension exprimée comme horodatage explicite ou implicite. Un horodate est généralement représenté sous forme d'un entier *long* en *millisecondes*.

- **Valeur :** valeur à plusieurs versions du conteneur clé/valeur. Cela signifie qu'une cellule peut contenir plusieurs versions d'une valeur susceptibles de changer au fil du temps. Les versions sont stockées par horodatage décroissant, le plus récent en premier.

Apache HBase définit ses paires clé/valeur comme tableaux arbitraires d'octets. Comme les clés de ligne et les qualificateurs de colonne sont aussi des tableaux arbitraires d'octets, pratiquement tout peut servir de clé de ligne ou de qualificateur de colonne, des chaînes aux représentations binaires d'entiers longs ou même de structures de données sérialisées.

Les noms de familles de colonnes *doivent* être composés de caractères affichables dans un format contrôlable de visu. La raison en est que les noms de familles de colonnes sont utilisés comme partie du nom du répertoire dans le système de fichiers. De plus, les familles de colonnes doivent être déclarées initialement, au moment de la définition du schéma. Les qualificateurs de colonne ne sont pas soumis à cette restriction et peuvent être composés de caractères binaires arbitraires et créés à l'exécution.

Indexation

En règle générale, les données sont indexées par clé primaire à des fins d'extraction rapide dans Amazon DynamoDB et Apache HBase. Les index secondaires étendent les fonctionnalités d'indexation de base et fournissent un autre chemin d'accès de la requête en plus des requêtes sur la clé primaire.

Amazon DynamoDB prend en charge deux types d'index secondaires sur une table qui implémente déjà une clé hachage-plage :

- **Index secondaire local :** index ayant la même clé de hachage que la table, mais une clé de plage différente.
- **Index secondaire global :** index avec une clé de hachage et une clé de plage qui peuvent être différentes de celles de la table.

Vous pouvez définir un ou plusieurs index secondaires locaux, et un ou plusieurs index secondaires globaux par table.

Un index secondaire local organise les données par clé de plage de l'index, plutôt que par clé de plage de la table source. Chaque index secondaire local contient automatiquement les attributs de hachage et de plage de sa table parent. Un index secondaire local peut copier ou projeter tout ou partie des attributs de la table source et fournir un autre chemin d'accès de la requête pour un accès efficace aux données.

Dans l'exemple de la table *Scores* de la section précédente, vous pouvez définir *IndexMeilleursScoresPersonne* comme index secondaire local de cette table. Cet index contient la même clé de hachage, *IDPersonne,* que la table source et définit aussi *DateMeilleurScore* comme sa clé de plage. La valeur de clé de plage de la table source (dans cet exemple, *IDJeu*) est automatiquement projetée ou copiée dans l'index, mais ne fait pas partie de la clé d'index, comme illustré dans le tableau suivant.

IndexMeilleursScoresPersonne			
Clé d'index		Attribut1	Attribut2
IDPersonne (clé de hachage)	DateMeilleurScore (clé de plage)	IDJeu	MeilleurScore
1001	2013-12-09:17:24:31	Jeu01	67453
1001	2013-12-11:14:14:37	Jeu02	98567

IndexMeilleursScoresPersonne			
1002	2013-12-15:19:24:39	Jeu01	43876
2002	2013-10-01:17:14:41	Jeu02	65689

Tableau 13 : Index secondaire local dans Amazon DynamoDB

Vous pouvez décider de copier ou de projeter des attributs supplémentaires non-clés depuis la table source vers l'index. Par exemple, *IndexMeilleursScoresPersonne* inclut une copie de l'attribut non-clé *MeilleurScore* comme partie de sa définition, tandis qu'*IndexMeilleursScoresPersonne_1* exclut le même attribut de sa définition.

IndexMeilleursScoresPersonne_1		
Clé d'index		Attribut1
IDPersonne (clé de hachage)	DateMeilleurScore (clé de plage)	IDJeu
1001	2013-12-09:17:24:31	Jeu01
1001	2013-12-11:14:14:37	Jeu02
1002	2013-12-15:19:24:39	Jeu01
2002	2013-10-01:17:14:41	Jeu02

Tableau 14 : Index secondaire local dans Amazon DynamoDB sans projection d'attribut non-clé

La principale différence entre un index secondaire global et un index secondaire local est qu'un index secondaire global définit un index hachage/plage entièrement nouveau sur une table. Vous pouvez définir n'importe quel attribut comme clé de hachage pour l'index secondaire global aussi longtemps que son type de données est scalaire plutôt que comme ensemble à valeurs multiples.

Lorsque vous utilisez les index secondaires globaux, gardez à l'esprit les points suivants :

- Contrairement aux tables associées, les valeurs de clé d'un index secondaire global ne doivent pas être uniques.
- Tout attribut d'une table peut être une clé, y compris les attributs qui ne sont pas présents dans tous les éléments, aussi longtemps que les types de données sont des ensembles de scalaires, plutôt que des ensembles à valeurs multiples.
- Chaque index secondaire global doit avoir une clé de hachage et une clé de plage facultative.
- Les attributs de clé primaire de la table source sont automatiquement projetés sur l'index, comme c'est le cas avec les index secondaires locaux.
- Contrairement aux index secondaires locaux qui utilisent les unités de capacité en lecture et en écriture de la table, chaque index secondaire global possède ses propres paramètres de débit provisionné pour les activités de lecture et d'écriture. Pour plus d'informations sur le débit provisionné, consultez la section Modèle de débit du présent livre blanc.

L'exemple *IndexClassement* affiche un index secondaire global pour la table *Scores*, comme expliqué dans les sections précédentes.

IndexClassement		
Clé d'index		Attribut1
IDJeu (clé de hachage)	MeilleurScore (clé de plage)	IDPersonne
Jeu01	98567	1001
Jeu02	43876	1001
Jeu01	65689	1002
Jeu02	67453	2002

Tableau 15 : Index secondaire global de la table Scores dans Amazon DynamoDB

L'index secondaire global *IndexClassement* précédent définit *IDJeu* comme clé de hachage et *MeilleurScore* comme clé de plage. La clé d'index n'a pas besoin d'avoir l'un des attributs de clé de la table source. Cependant, les attributs de clé primaire de la table sont toujours présents dans l'index secondaire global. Dans cet exemple, *IDPersonne* est automatiquement projeté ou copié sur l'index.

L'index *IndexClassement,* dans cet exemple, vous permet d'obtenir aisément une liste des meilleurs scores pour un jeu spécifique en interrogeant l'index. Les résultats sont triés sur la clé de plage *IndexClassement*. Vous pouvez choisir de projeter des attributs supplémentaires depuis la table source vers l'index.

Toutes les lignes dans Apache HBase sont toujours triées lexicographiquement par clé de ligne. Le tri est ordonné par octet. Cela signifie que chaque clé de ligne est comparée sur un niveau binaire, octet par octet, de la gauche vers la droite. Les clés de ligne sont toujours uniques et font office de clé primaire dans Apache HBase.

Même si Apache HBase ne propose pas le support natif pour les modèles d'indexation intégrés tels qu'Amazon DynamoDB, vous pouvez implémenter des index secondaires personnalisés pour servir de chemins d'accès à la requête à l'aide des techniques suivantes :

- **Créer un index dans une autre table.** Vous pouvez maintenir une table secondaire périodiquement mise à jour. Cependant, en fonction de la stratégie de chargement, le risque de cette méthode est que l'index secondaire puisse potentiellement ne plus être synchronisé avec la table principale. Vous pouvez atténuer ce risque si vous créez l'index secondaire tout en publiant les données sur le cluster, et exécutez des écritures simultanées dans la table d'index.
- **Utiliser l'infrastructure de coprocesseur.** Vous pouvez mettre à profit l'infrastructure de coprocesseur pour implémenter les index secondaires personnalisés. Les coprocesseurs se comportent comme des déclencheurs similaires aux procédures stockées dans les SGBDR.

En résumé, Amazon DynamoDB et Apache HBase définissent tous deux des modèles de données qui permettent un stockage efficace des données pour optimiser les performances des requêtes. Amazon DynamoDB impose une restriction sur sa taille d'élément pour autoriser un traitement efficace et réduire les coûts. Apache HBase utilise le concept de familles de colonnes pour fournir la localité des données et permettre des opérations de lecture plus efficaces.

Amazon DynamoDB prend en charge les ensembles de scalaires et les ensembles à valeurs multiples pour accueillir un large éventail d'ensembles de données non structurés. De même, Apache HBase stocke ses paires clé/valeur comme tableaux d'octets arbitraires, ce qui lui confère la souplesse de stocker n'importe quel type de données.

Amazon DynamoDB prend en charge les index secondaires intégrés, et met à jour et synchronise automatiquement tous les index avec leurs tables parents. Avec Apache HBase, vous pouvez implémenter et gérer vous-même les index secondaires personnalisés.

Du point de vue d'un modèle de données, vous pouvez choisir Amazon DynamoDB si votre taille d'élément est relativement petite. Même si Amazon DynamoDB fournit un certain nombre d'options pour surmonter les restrictions de taille de ligne, Apache HBase est mieux équipé pour gérer d'importantes charges utiles complexes avec des restrictions minimales.

Traitement des données

Cette section présente les éléments essentiels de traitement et d'interrogation des données au sein d'Amazon DynamoDB et d'Apache HBase.

Modèle de débit

Amazon DynamoDB utilise un modèle de débit provisionné pour traiter les données. Avec ce modèle, vous pouvez spécifier vos besoins de capacité de lecture et d'écriture par seconde, auxquels une table est censée répondre. Pendant la création de la table, Amazon DynamoDB partitionne et réserve automatiquement la quantité appropriée de ressources afin de satisfaire vos exigences en termes de débit.

Pour décider des valeurs de débit requises en lecture et en écriture pour une table, considérez les critères suivants :

- **Taille d'élément.** Les unités de capacité en lecture et en écriture que vous spécifiez reposent sur une taille d'élément de données prédéfinie par opération de lecture ou d'écriture. Pour plus d'informations sur les restrictions de taille d'élément de données pour le débit provisionné, consultez Débit provisionné dans Amazon DynamoDB[17] dans le *Guide du développeur Amazon DynamoDB*[16].
- **Taux attendu de demandes de lecture et d'écriture.** Vous devez aussi déterminer le nombre attendu d'opérations de lecture et d'écriture par seconde que votre application effectuera sur la table.
- **Cohérence.** Le fait que votre application requiert des lectures à cohérence forte ou des lectures cohérentes à terme est un critère qui contribue à déterminer le nombre d'unités de capacité en lecture que vous avez besoin de provisionner pour votre table. Pour plus d'informations sur la cohérence et Amazon DynamoDB, consultez la section Modèle de cohérence du présent livre blanc.

[17] http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ProvisionedThroughputIntro.html

- **Index secondaires locaux.** Les requêtes sur les index utilisent le débit en lecture provisionné. Pour plus d'informations, consultez Considérations relatives au débit provisionné pour les index secondaires locaux[18] dans le *Guide du développeur Amazon DynamoDB*[16].
- **Index secondaires globaux.** Les paramètres de débit provisionné d'un index secondaire global sont distincts de ceux de sa table parent. Par conséquent, la charge de travail attendue sur l'index secondaire global doit aussi être prise en considération lors de la spécification de la capacité de lecture et d'écriture au moment de la création de l'index.

Bien que les exigences de lecture et d'écriture soient spécifiées à la création de la table, Amazon DynamoDB vous permet d'augmenter ou de diminuer le débit provisionné pour accueillir le chargement sans interruption.

Dans Apache HBase, le nombre de nœuds d'un cluster peut être lié au débit requis pour les lectures et/ou écritures. Le débit disponible sur un nœud donné peut varier en fonction des données, et plus particulièrement des critères suivants :

- Tailles clé/valeur
- Modèles d'accès aux données
- Taux d'accès au cache
- Configuration des nœuds et du système

Vous devez planifier les charges de pointe si la charge est susceptible d'être le principal facteur d'augmentation du nombre de nœuds au sein d'un cluster Apache HBase.

18 http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/LSI.html#LSI.Throughp Considérations

Modèle de cohérence

Un modèle de cohérence de base de données détermine la façon et le timing selon lesquels une mise à jour ou une écriture réussie est reflétée dans une opération de lecture suivante de cette même valeur.

Amazon DynamoDB vous permet de spécifier les caractéristiques de cohérence souhaitées pour chaque demande de lecture au sein d'une application. Vous pouvez spécifier si une lecture est cohérente à terme ou une lecture à cohérence forte.

L'option de cohérence à terme est celle par défaut dans Amazon DynamoDB et optimise le débit en lecture. Cependant, une lecture cohérente à terme peut ne pas toujours refléter les résultats d'une écriture récemment terminée. La cohérence à travers toutes les copies de données est généralement atteinte en une seconde.

Une lecture à cohérence forte dans Amazon DynamoDB retourne un résultat qui reflète toutes les écritures ayant reçu *avec succès* une réponse avant la lecture. Pour obtenir un résultat de lecture à cohérence forte, vous pouvez spécifier des paramètres facultatifs dans une demande. Le traitement d'une lecture à cohérence forte nécessite plus de ressources qu'une lecture cohérente à terme. Pour plus d'informations sur la cohérence des lectures, consultez Considérations relatives à la lecture de données et à la cohérence[19] dans le *Guide du développeur Amazon DynamoDB*[16].

Les lectures et écritures Apache HBase sont à cohérence forte. Cela signifie que toutes les lectures et écritures d'une seule ligne dans Apache HBase sont atomiques. Chaque auteur simultané d'une opération de lecture ou d'écriture peut formuler une hypothèse sécurisée sur l'état d'une ligne. La gestion des versions multiples et l'horodatage dans Apache HBase contribuent à son modèle à cohérence forte.

Modèle de transaction

Contrairement aux SGBDR, les bases de données NoSQL n'ont généralement aucun langage spécifique à un domaine, tel que SQL, pour interroger les données. Amazon DynamoDB et Apache HBase fournissent des API simples pour exécuter les opérations standard de création, lecture, mise à jour et suppression.

Ni Amazon DynamoDB ni Apache HBase ne prennent en charge les transactions à plusieurs éléments/inter-ligne ou inter-table pour des raisons de performance. Cependant, les deux bases de données fournissent des opérations de traitement par lots pour la lecture et l'écriture de plusieurs éléments/lignes sur plusieurs tables sans garantie de transaction.

19 http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/APISummary.html#DataReadConsistency

Amazon DynamoDB fournit les opérations d'attributs et d'éléments atomiques pour l'ajout, la mise à jour ou la suppression de données. De plus, les transactions de niveau élément peuvent spécifier une condition qui doit être satisfaite avant que la transaction ne soit exécutée. Par exemple, vous pouvez choisir de ne mettre à jour un élément que s'il possède déjà une certaine valeur.

Les opérations conditionnelles vous permettent d'implémenter les systèmes de contrôle d'accès concurrentiel optimiste [20] sur Amazon DynamoDB. Pour les mises à jour conditionnelles, Amazon DynamoDB autorise les opérations d'incrémentation et de décrémentation atomiques sur les valeurs scalaires existantes sans interférer avec d'autres demandes d'écriture.

Apache HBase prend aussi en charge les fréquences de mise à jour élevées atomiques (les opérations classiques de lecture, modification et écriture) au sein d'une seule clé de ligne, activant ainsi le stockage des compteurs à fréquence élevée. Contrairement à Amazon DynamoDB, Apache HBase utilise le contrôle d'accès concurrentiel de plusieurs versions pour implémenter les mises à jour. Cela signifie qu'une donnée existante n'est pas remplacée par une nouvelle ; elle devient, à la place, obsolète quand une version plus récente est ajoutée.

L'accès aux données de ligne dans Apache HBase est atomique et inclut un nombre quelconque de colonnes, mais il n'existe aucune garantie ou fonction transactionnelle couvrant plusieurs lignes. De même qu'Amazon DynamoDB, Apache HBase ne prend en charge que les transactions à une seule ligne.

Opérations de table

Amazon DynamoDB et Apache HBase fournissent des opérations d'analyse pour prendre en charge le traitement analytique à grande échelle. Une opération d'analyse est similaire aux *curseurs*[21] dans les SGBDR. En tirant profit du stockage trié et séquentiel sous-jacent, une opération d'analyse peut facilement itérer sur de vastes ensembles d'enregistrements ou des tables entières. L'application de filtres aux opérations d'analyse peut affiner efficacement l'ensemble des résultats et optimiser les performances.

Amazon DynamoDB utilise l'analyse parallèle pour améliorer les performances d'une opération d'analyse. Une analyse parallèle sous-divise logiquement une table Amazon DynamoDB en plusieurs segments, puis traite chaque segment en parallèle. Plutôt que d'utiliser l'opération d'analyse par défaut dans Apache HBase, vous pouvez implémenter une analyse parallèle personnalisée au moyen de l'API pour lire les lignes en parallèle.

---

20 http://en.wikipedia.org/wiki/Optimistic_concurrency_control

21 http://en.wikipedia.org/wiki/Cursor_%28databases%29

Amazon DynamoDB fournit une API Query pour le traitement des requêtes complexes, en plus de son opération d'analyse. L'API Query n'est accessible que dans les tables qui définissent une clé primaire composite. Une demande de requête extrait les éléments d'une table Amazon DynamoDB ou d'un index à l'aide de la clé primaire de hachage-plage.

En résumé, Amazon DynamoDB et Apache HBase possèdent des modèles de traitement des données similaires, en ce sens que tous deux ne prennent en charge que les transactions atomiques à ligne unique. Les deux bases de données fournissent aussi des opérations de traitement par lots pour le traitement en bloc de données sur plusieurs lignes et tables.

Une différence clé entre les deux bases de données est le modèle de débit provisionné flexible d'Amazon DynamoDB. La possibilité d'augmenter la capacité quand vous en avez besoin et de la réduire une fois que vous avez terminé est utile pour traiter les charges de travail variables aux pics imprévisibles.

Pour les charges de travail qui nécessitent des fréquences de mise à jour élevées pour effectuer des regroupements de données ou maintenir des compteurs, Apache HBase constitue un excellent choix. La raison en est qu'Apache HBase prend en charge un mécanisme de contrôle d'accès concurrentiel multi-version, qui contribue à ses lectures et écritures à cohérence forte. Amazon DynamoDB vous offre la possibilité de spécifier si vous voulez que votre demande de lecture soit une lecture cohérente à terme ou une lecture à cohérence forte en fonction de votre charge de travail spécifique.

Architecture

Cette section résume les principaux composants architecturaux d'Amazon DynamoDB et Apache HBase.

Présentation de l'architecture d'Amazon DynamoDB

Les origines de l'architecture d'Amazon DynamoDB remontent à la naissance de Dynamo, base de données NoSQL décrite par DeCandia *et autres*[22]. A un haut niveau, Amazon DynamoDB est conçu pour la haute disponibilité, la durabilité et la latence uniformément basse (généralement inférieure à 10 millisecondes).

Amazon DynamoDB s'exécute sur une flotte de serveurs AWS gérés qui exploitent les disques SSD pour créer une plateforme de stockage optimisée et à haute densité. Cette plateforme sépare les performances de la taille de la table et élimine la nécessité que l'ensemble de données actif s'intègre à la mémoire tout en continuant à retourner aux requêtes des réponses cohérentes, à latence faible. En tant que service géré, Amazon DynamoDB extrait de l'utilisateur ses détails architecturaux sous-jacents.

Présentation de l'architecture d'Apache HBase

Apache HBase est généralement déployée par-dessus lesystème de fichiers distribué Hadoop (HDFS) [23], qui fournit une couche de stockage évolutive et persistante. Apache ZooKeeper[24] est un composant essentiel pour mettre à jour les informations de configuration et gérer la totalité du cluster Apache HBase.

Les trois principaux composants d'Apache HBase sont les suivants :

- API Client
- Serveur maître
- Serveurs de région

Apache HBase stocke les données dans des fichiers de magasin indexés, appelés fichiers HFile, sur HDFS. Les fichiers de magasin sont des séquences de blocs avec un index de bloc stocké à la fin pour les recherches rapides. Ils fournissent une API pour accéder aux valeurs spécifiques ainsi que pour analyser des plages de valeurs, à partir d'une clé de début et d'une clé de fin.

Pendant une opération d'écriture, les données sont d'abord écrites dans un journal de validation appelé journal WAL (write-ahead-log), puis déplacées en mémoire dans une structure appelée Memstore. Quand la taille de la structure Memstore dépasse une valeur maximale donnée, elle est vidée comme fichier HFile sur le disque. Chaque fois que les données sont vidées depuis une structure Memstore vers le disque, de nouveaux fichiers HFile doivent être créés. Quand le nombre de fichiers HFile augmente, un processus de *compactage* fusionne les fichiers en fichiers plus grands et moins nombreux.

Une opération de lecture est essentiellement une fusion de données stockées dans les structures Memstore et dans les fichiers HFile. Le fichier WAL n'est jamais utilisé dans l'opération de lecture. Il est destiné uniquement à des fins de récupération en cas d'incident d'un serveur avant l'écriture sur disque des données en mémoire.

---

22 Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall et Werner Vogels. 2007. Dynamo: Amazon's Highly Available Key-value Store (« Dynamo : le stockage clé-valeur hautement disponible d'Amazon »).

23 http://wiki.apache.org/hadoop/HDFS

24 http://zookeeper.apache.org/

Une région dans Apache HBase fait office de magasin par famille de colonnes. Chaque région contient des plages contiguës de lignes stockées ensemble. Les régions peuvent être fusionnées pour réduire le nombre de fichiers de magasin. Un fichier de magasin volumineux qui dépasse la taille de fichier magasin maximale configurée peut déclencher un fractionnement de région.

Un serveur de région peut desservir plusieurs régions. Chaque région est mappée à un serveur de région exactement. Les serveurs de région gèrent les lectures et les écritures, et conservent les données en mémoire jusqu'à ce qu'un nombre suffisant de données ait été collecté pour justifier un vidage. Les clients communiquent directement avec les serveurs de région pour gérer toutes les opérations associées aux données.

Le serveur maître est responsable de la supervision et de l'affectation des régions aux serveurs de région, et utilise Apache ZooKeeper pour faciliter cette tâche. Apache ZooKeeper fait également office de registre pour les serveurs de région et d'emplacement d'amorçage pour la découverte des régions.

Le serveur maître est également responsable de la gestion des fonctions critiques telles que l'équilibrage de charge des régions entre les serveurs de région, le basculement des serveurs de région et l'exécution des fractionnements de région, mais il ne fait pas partie du stockage réel des données ou du chemin d'extraction.

Vous pouvez exécuter Apache HBase dans un environnement à multiples maîtres. Tous les maîtres concourent à exécuter le cluster en mode à multiples maîtres. Cependant, si le maître actif s'arrête, les maîtres restants s'affrontent pour s'emparer du rôle de maître.

Présentation de l'architecture d'Apache HBase sur Amazon EMR

Amazon EMR définit le concept de groupes d'instance, qui sont des collections d'instances Amazon EC2. Les serveurs virtuels Amazon EC2 exécutent des rôles analogues aux nœuds maître et esclave d'Hadoop. Pour de meilleures performances, les clusters Apache HBase doivent s'exécuter sur au moins deux instances Amazon EC2. Il existe trois types de groupes d'instance dans un cluster Amazon EMR :

- **Maître :** contient un nœud maître qui gère le cluster. Vous pouvez utiliser le protocole Secure Shell (SSH) pour accéder au nœud maître si vous voulez afficher les journaux ou administrer le cluster vous-même. Le nœud maître exécute le serveur maître Apache HBase et Apache Zookeeper.
- **Cœur :** contient un ou plusieurs nœuds principaux qui exécutent HDFS et stockent les données. Les nœuds principaux exécutent les serveurs de région Apache HBase.
- **Tâche :** (facultatif). Contient un nombre quelconque de nœuds de tâche.

Partitioning

Amazon DynamoDB stocke trois réplicas géographiquement distribués de chaque table pour permettre une haute disponibilité et une durabilité des données au sein d'une région. Les données sont partitionnées automatiquement, principalement à l'aide de la clé de hachage. Au fur et à mesure que le débit et la taille des données augmentent, Amazon DynamoDB repartitionne et réalloue automatiquement les données entre plusieurs nœuds.

Les partitions d'Amazon DynamoDB sont entièrement indépendantes, ce qui se traduit par un cluster sans partage. Cependant, le débit provisionné est réparti également entre les partitions.

Une région est l'unité de base de l'évolutivité et de l'équilibrage de charge dans Apache HBase. Le fractionnement des régions et l'équilibrage de charge consécutif obéissent à cette séquence d'événements :

1. Initialement, il n'existe qu'une seule région pour une table, et tandis que de nouvelles données sont ajoutées, le système surveille la charge pour s'assurer que la taille maximale configurée n'est pas dépassée.
2. Si la taille de la région dépasse la limite configurée, le système fractionne dynamiquement la région en deux à la clé de ligne au milieu de la région, créant ainsi deux moitiés approximativement égales.
3. Le maître planifie alors le déplacement des nouvelles régions vers d'autres serveurs à des fins d'équilibrage de charge, si nécessaire.

En coulisses, Apache Zookeeper suit toutes les activités qui prennent place pendant un fractionnement de région et gère l'état de la région en cas de défaillance du serveur. Les régions Apache HBase sont équivalentes aux [partitions de plage][25] utilisées dans le partitionnement des SGBDR. Les régions peuvent être réparties entre plusieurs serveurs physiques qui distribuent ensuite la charge, ce qui se traduit par une évolutivité.

En résumé, en tant que service géré, les détails architecturaux d'Amazon DynamoDB sont extraits pour que vous puissiez vous concentrer sur ceux de votre application. Avec le modèle de déploiement Apache HBase auto-géré, il est essentiel de comprendre les détails architecturaux sous-jacents pour optimiser l'évolutivité et les performances. AWS vous offre la possibilité de vous décharger de l'administration d'Apache HBase, si vous choisissez de lancer votre cluster sur Amazon EMR.

[25] http://en.wikipedia.org/wiki/Partition_%28database%29

Optimisation des performances

Amazon DynamoDB et Apache HBase sont intrinsèquement optimisés pour traiter d'importants volumes de données avec de hautes performances. Les bases de données NoSQL utilisent généralement un format de stockage sur disque et orienté colonnes pour un accès rapide aux données et des E/S réduites lors de l'exécution des requêtes. Cette caractéristique de performance est évidente dans Amazon DynamoDB et Apache HBase.

Amazon DynamoDB stocke les éléments ayant la même clé de hachage de façon contiguë sur disque pour optimiser l'extraction rapide des données. De même, les régions Apache HBase contiennent des plages contiguës de lignes stockées ensemble pour améliorer les opérations de lecture. Vous pouvez améliorer encore plus les performances si vous appliquez des techniques qui optimisent le débit à des coûts réduits, aux niveaux infrastructure et application.

Conseil : une bonne pratique recommandée consiste à surveiller les métriques de performances Amazon DynamoDB et Apache HBase pour détecter et diagnostiquer de façon proactive les goulots d'étranglement des performances.

La section suivante se concentre sur plusieurs optimisations courantes des performances et spécifiques à chaque base de données ou modèle de déploiement.

Considérations sur les performances d'Amazon DynamoDB

Les considérations sur les performances d'Amazon DynamoDB se concentrent sur la façon de définir un débit de lecture et d'écriture approprié, et sur la façon de concevoir un schéma adapté pour une application. Ces considérations couvrent à la fois le niveau infrastructure et le niveau application.

Considérations sur le débit provisionné

Les critères qui doivent être pris en compte lors de la détermination des exigences de débit approprié d'une application sont la taille de l'élément, les taux de lecture et d'écriture attendus, la cohérence et les index secondaires, comme expliqué dans la section Modèle de débit de ce livre blanc.

Si une application effectue plus de lectures par seconde ou plus d'écritures par seconde que la capacité de débit provisionné d'une table ne le permet, les demandes au-dessus de la capacité allouée seront accélérés. Par exemple, si la capacité d'écriture d'une table est 1 000 unités et qu'une application effectue 1 500 écritures par seconde pour la taille d'élément de données maximale, Amazon DynamoDB autorise uniquement un débit de 1 000 écritures par seconde et les demandes supplémentaires sont limitées.

Conseil : une bonne pratique recommandée consiste à provisionner la capacité de débit suffisamment à l'avance pour s'assurer de sa disponibilité en cas de nécessité, et à surveiller les performances avec les alarmes de notification Amazon CloudWatch à des fins d'alerte quand un certain seuil d'unités de capacité consommée est atteint. De plus, Amazon DynamoDB vous permet d'augmenter ou de diminuer la capacité de débit sans interruption.

Considérations relatives à la conception de la clé primaire

La conception de la clé primaire est essentielle pour les performances d'Amazon DynamoDB. Lors du stockage des données, Amazon DynamoDB scinde les éléments d'une table en plusieurs partitions et répartit les données principalement en fonction de la clé de hachage. Le débit provisionné associé à une table est également réparti de façon égale entre les partitions sans aucun partage de débit provisionné entre les partitions.

Conseil : pour utiliser efficacement le débit provisionné global, répartissez la charge de travail entre les valeurs de clé de hachage.

Par exemple, si une table possède un très petit nombre d'éléments de clé de hachage massivement accédés, et même peut-être un seul élément de clé de hachage extrêmement utilisé, le trafic peut se concentrer sur une partition et créer des zones sensibles d'activité de lecture et d'écriture au sein d'une même collection d'éléments. Amazon DynamoDB limite les charges de travail non équilibrées dans de tels cas.

Pour tirer le meilleur parti du débit Amazon DynamoDB, vous pouvez créer des tables où l'élément de clé de hachage possède un grand nombre de valeurs distinctes. Assurez-vous que les valeurs sont demandées de façon assez uniforme et de manière aussi aléatoire que possible. La même instruction s'applique aux index secondaires globaux. Choisissez des clés de hachage et de plage qui fournissent des charges de travail uniformes pour atteindre le débit provisionné global.

Considérations relatives aux index secondaires locaux

Lors de l'interrogation d'un index secondaire local, le nombre d'unités de capacité en lecture consommées dépend de la façon dont les données sont accédées. Par exemple, quand vous créez un index secondaire local et projetez des attributs non-clés sur l'index de la table parent, Amazon DynamoDB peut extraire ces attributs projetés de manière efficace.

De plus, quand vous interrogez un index secondaire local, la requête peut également extraire les attributs qui *ne sont pas* projetés sur l'index. Evitez ces types de requêtes d'index qui lisent des attributs non projetés sur l'index secondaire local. Les attributs d'extraction de la table parent qui ne sont pas spécifiés dans l'index secondaire local entraînent une latence supplémentaire dans les réponses aux requêtes et s'exposent à un coût de débit provisionné supérieur.

Conseil : projetez les attributs non-clés fréquemment accédés sur un index secondaire local pour éviter les extractions et améliorer les performances des requêtes.

Maintenez plusieurs index secondaires locaux dans les tables qui sont mises à jour irrégulièrement, mais interrogées à l'aide de nombreux critères différents pour améliorer les performances d'interrogation. Ces instructions ne s'appliquent pas aux tables confrontées à une activité d'écriture élevée.

S'il est attendu une activité d'écriture très élevée sur la table, une option consiste à minimiser les interférences des lectures en ne procédant à aucune lecture depuis la table. Créez à la place un index secondaire global avec une structure identique à celle de la table, puis dirigez toutes les requêtes vers l'index plutôt que vers la table.

Considérations relatives aux index secondaires globaux

Si une requête dépasse la capacité lue provisionnée d'un index secondaire global, cette demande est limitée. De même, si une demande exécute une activité massive d'écriture sur le table, mais qu'un index secondaire global de cette table a une capacité d'écriture insuffisante, l'activité d'écriture sur la table est limitée.

Conseil : pour qu'une écriture de table réussisse, les paramètres de débit provisionnés de la table et des index secondaires globaux doivent avoir une capacité d'écriture suffisante pour accueillir l'écriture ; sinon, l'écriture est limitée.

Les index secondaires globaux prennent en charge les lectures cohérentes à terme, dont chacune utilise la moitié d'une unité de capacité de lecture. Le nombre d'unités de capacité de lecture est la somme de toutes les tailles des attributs projetés de l'ensemble des éléments retournés dans les résultats des requêtes d'index. Avec les activités d'écriture, le coût total du débit provisionné pour une écriture se compose de la somme des unités de capacité d'écriture utilisées par l'écriture sur la table et de celles utilisées par la mise à jour des index secondaires globaux.

Considérations sur les performances d'Apache HBase

Le réglage des performances d'Apache HBase couvre les configurations matériel, réseau, Apache HBase et Hadoop, ainsi que les paramètres de nettoyage de la mémoire de la machine virtuelle Java. Il inclut aussi l'application des meilleures pratiques lors de l'utilisation de l'API client. Pour optimiser les performances, il est précieux de surveiller les charges de travail Apache HBase avec des outils tels que Ganglia pour identifier les problèmes de performances suffisamment tôt et appliquer les bonnes pratiques recommandées en fonction des métriques de performance observées.

Considérations relatives à la mémoire

La mémoire est l'élément le plus restrictif dans Apache HBase. Les techniques de réglage des performances sont concentrées sur l'optimisation de la consommation mémoire.

Du point de vue de la conception du schéma, il importe de garder à l'esprit que chaque cellule stocke sa valeur comme entièrement qualifiée, avec sa clé de ligne complète, sa famille de colonne, son nom de colonne et son horodatage sur disque. Si les noms de ligne et de colonne sont longs, les coordonnées des valeurs des cellules peuvent devenir très grandes et consommer une plus grande partie de la mémoire allouée à Apache HBase. Il peut en résulter de graves implications de performance, particulièrement si le jeu de données est volumineux.

Conseil : maintenez bas le nombre de familles de colonnes afin d'améliorer les performances et de réduire les coûts associés à la maintenance des fichiers HFile sur disque.

Configurations Apache HBase

Apache HBase prend en charge les mécanismes intégrés pour gérer les compactages et les fractionnements de région. Les graves incidents de fractionnement/compactage peuvent se produire lorsque plusieurs régions se développent approximativement au même rythme et finissent par se scinder à peu près au même moment. Il peut en résulter un pic important dans les E/S disque en raison des compactages nécessaires pour réécrire les régions fractionnées.

Conseil : plutôt que de vous fier à Apache HBase pour fractionner et compacter automatiquement les régions croissantes, vous pouvez exécuter ces tâches manuellement.

Si vous gérez les fractionnements et les compactages manuellement, vous pouvez les exécuter de manière contrôlée dans le temps et les échelonner à travers toutes les régions pour répartir la charge d'E/S autant que possible et éviter ainsi les incidents éventuels de fractionnement/compactage. Avec l'option manuelle, vous pouvez en outre atténuer tout incident problématique de fractionnement/compactage, et bénéficier de performances supplémentaires.

Conception de schéma

Une région peut devenir sensible lorsqu'elle utilise un modèle d'écriture qui ne répartit pas la charge entre tous les serveurs de façon égale. Il s'agit d'un scénario courant quand il s'agit d'utiliser des flux traitant des événements avec des données *en séries chronologiques*. La nature progressivement croissante des données en séries chronologiques peut entraîner l'écriture de toutes les données entrantes dans la même région.

Cette activité d'écriture concentrée sur un seul serveur peut ralentir les performances globales du cluster. La raison en est que l'insertion des données est maintenant liée aux performances d'une seule machine. Ce problème est facilement surmonté en utilisant des stratégies d'index telles que les suivantes :

- Application de préfixes aux clés ; autrement dit, préfixez une ligne par un nombre aléatoire.
- Introduction d'un aspect aléatoire dans la clé grâce à une fonction de hachage.
- Promotion d'un autre champ pour préfixer la clé de ligne.

Ces techniques permettent d'obtenir une charge plus également répartie entre tous les serveurs.

Considérations relatives aux API client

Il existe un certain nombre d'optimisations à prendre en compte lors de la lecture ou de l'écriture de données d'un client à l'aide de l'API Apache HBase. Par exemple, lors de l'exécution d'un grand nombre d'opérations PUT, vous pouvez désactiver la fonction de vidage automatique. Sinon, les opérations PUT sont envoyées l'une après l'autre au serveur de la région.

Chaque fois que vous utilisez une opération d'analyse pour traiter de grands nombres de lignes, utilisez des filtres pour limiter l'étendue de l'analyse. L'utilisation de filtres peut potentiellement améliorer les performances. La raison en est que la sur-sélection de colonnes peut entraîner des pertes de performance significatives, notamment sur les jeux de données volumineux.

Conseil : comme bonne pratique recommandée, définissez la mise en cache de l'analyseur avec une valeur supérieure à la valeur de 1 par défaut, notamment si Apache HBase fait office de source d'entrée d'une tâche MapReduce[26].

La définition de la mise en cache de l'analyseur avec la valeur 500, par exemple, transfère 500 lignes à la fois au client pour qu'elles soient traitées, mais ce transfert peut potentiellement être plus onéreux en consommation mémoire.

[26] http://en.wikipedia.org/wiki/MapReduce

Techniques de compression

La compression des données est un aspect important des charges de travail de production Apache HBase. Apache HBase prend en charge en mode natif un certain nombre d'algorithmes de compression que vous pouvez activer au niveau des familles de colonnes.

Conseil : l'activation de la compression favorise de meilleures performances.

En général, les ressources de calcul pour l'exécution des tâches de compression et de décompression sont généralement inférieures à la surcharge qu'entraîne la lecture d'un plus grand nombre de données sur le disque.

Apache HBase sur Amazon EMR

Apache HBase sur Amazon EMR est optimisé pour s'exécuter sur AWS avec un traitement administratif minimal. Vous pouvez toujours accéder à l'infrastructure sous-jacente et configurer manuellement les paramètres Apache HBase, si nécessaire.

Considérations relatives aux clusters

Vous pouvez redimensionner un cluster Amazon EMR à l'aide des nœuds principaux et des nœuds de tâche. Vous pouvez ajouter plus de nœuds principaux, si nécessaire. Les nœuds de tâche sont utiles pour gérer la capacité des instances Amazon EC2 d'un cluster. Vous pouvez accroître la capacité de façon à gérer les pics de charge et à la réduire ultérieurement lors des baisses de demande.

Conseil : comme bonne pratique recommandée, dans les charges de travail en production, vous pouvez lancer Apache HBase sur un cluster et n'importe quels outils d'analyse, comme Apache Hive, sur un cluster distinct pour améliorer les performances. La gestion de deux clusters distincts garantit qu'Apache HBase a déjà accès aux ressources d'infrastructure qu'il requiert.

Amazon EMR fournit une fonction pour sauvegarder les données Apache HBase sur Amazon S3. Vous pouvez effectuer des sauvegardes manuelles ou automatiques avec la possibilité d'exécuter des sauvegardes complètes ou incrémentielles selon les besoins.

Conseil : comme bonne pratique, chaque cluster de production doit toujours tirer profit de la fonction de sauvegarde disponible sur Amazon EMR.

Configurations Hadoop et Apache HBase

Vous pouvez utiliser une action d'amorçage[27] pour installer des logiciels supplémentaires ou modifier les paramètres de configuration Apache HBase ou Apache Hadoop sur Amazon EMR. Les actions de démarrage sont des scripts exécutés sur les nœuds du cluster quand Amazon EMR lance le cluster. Les scripts s'exécutent avant le démarrage d'Hadoop et avant que le nœud ne commence à traiter les données.

Vous pouvez écrire des actions d'amorçage personnalisés ou utiliser des actions d'amorçage prédéfinies fournies par Amazon EMR. Par exemple, vous pouvez installer Ganglia pour surveiller les métriques de performance Apache HBase à l'aide d'une action d'amorçage prédéfinie sur Amazon EMR.

En résumé, quand vous exécutez une base de données NoSQL gérée telle qu'Amazon DynamoDB ou Apache HBase sur Amazon EMR, ou gérez votre cluster Apache HBase vous-même sur Amazon EC2 ou localement, vous devez prendre en compte les optimisations des performances si vous souhaitez optimiser les performances avec un coût réduit.

La principale différence entre une solution NoSQL hébergée et sa gestion par vous-même est qu'une solution gérée comme Amazon DynamoDB ou Apache HBase sur Amazon EMR vous permet de vous délester des charges administratives et de vous concentrer sur l'optimisation de votre application.

Si vous êtes un développeur qui démarrez avec NoSQL, les solutions Amazon DynamoDB ou Apache HBase sur Amazon EMR sont des options appropriées, en fonction de votre cas d'utilisation. Pour les développeurs ayant des connaissances Apache Hadoop/Apache HBase approfondies et qui ont besoin d'un contrôle total de leurs clusters Apache HBase, le modèle de déploiement Apache HBase auto-géré offre la plus grande souplesse du point de vue de la gestion des clusters.

---

27 http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-plan-bootstrap.html

Conclusion

Amazon DynamoDB vous permet de vous décharger de la gestion et du dimensionnement d'un cluster de base de données distribuée hautement disponible, ce qui en fait un choix approprié pour les applications web en temps réel. En tant que service géré, Apache HBase sur Amazon EMR est optimisé pour s'exécuter sur AWS avec un traitement administratif minimal. Pour les utilisateurs avancés qui veulent garder le contrôle total de leurs clusters Apache HBase, le modèle de déploiement Apache HBase auto-géré est un bon choix.

Amazon DynamoDB et Apache HBase possèdent tous deux des caractéristiques inhérentes, essentielles pour traiter avec succès d'importantes quantités de données. Avec un éventail de scénarios d'utilisation allant du traitement par lots au traitement des données en temps réel, Amazon DynamoDB et Apache HBase sont tous deux optimisés pour gérer des jeux de données volumineux. Cependant, la connaissance de votre jeu de données et des modèles d'accès est essentielle pour choisir la base de données NoSQL appropriée à votre charge de travail.

Suggestions de lecture

Pour plus d'informations, consultez les ressources suivantes :

- Amazon DynamoDB[28]
- Apache HBase[29]
- Apache Hadoop[30]
- Amazon Elastic Compute Cloud (Amazon EC2)[31]
- Amazon Elastic MapReduce (Amazon EMR)[32]
- Amazon Kinesis[33]
- Amazon Redshift[34]
- AWS Data Pipeline[35]
- Amazon CloudWatch[36]

28 http://aws.amazon.com/dynamodb/
29 http://hbase.apache.org/
30 http://hadoop.apache.org/
31 http://aws.amazon.com/ec2/
32 http://aws.amazon.com/elasticmapreduce/
33 http://aws.amazon.com/kinesis/
34 http://aws.amazon.com/redshift/
35 http://aws.amazon.com/datapipeline/
36 http://aws.amazon.com/cloudwatch/

- Amazon Identity and Access Management (Amazon IAM)[37]
- Amazon Simple Storage Service (Amazon S3)[38]
- Interface de ligne de commande AWS (Amazon CLI)[39]
- Ganglia[40]
- Dynamo: Amazon's Highly Available Key-value Store (« Dynamo : le stockage clé-valeur hautement disponible d'Amazon »)[41]
- Guide du développeur Amazon DynamoDB[42]
- Guide de l'utilisateur Amazon EC2[43]
- Guide du développeur Amazon Elastic MapReduce[44]
- Guide du développeur Amazon S3[45]
- HBase: The Definitive Guide, by Lars George[46]
- The Apache HBase™ Reference Guide[47]

37 http://aws.amazon.com/iam/

38 http://aws.amazon.com/s3/

39 http://docs.aws.amazon.com/general/latest/gr/GetTheTools.html

40 http://ganglia.sourceforge.net/

41 http://www.read.seas.harvard.edu/~kohler/class/cs239-w08/decandia07dynamo.pdf

42 http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html

43 http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

44 http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-what-is-emr.html

45 http://docs.aws.amazon.com/AmazonS3/latest/dev/Welcome.html

46 http://www.hbasebook.com/

47 http://hbase.apache.org/book.html